Automating Microsoft Excel
With ooRexx

16th International Rexx Symposium
for Developers and Users
April 18th – April 21st, 2005
Palos Verdes Inn
Redondo Beach, CA
Lee Peedin
VP Research & Development
Safe Data, Inc.
Wallace, NC
lee@safedatausa.com
Automating Microsoft Excel With ooRexx

Retrieve Data From Or Change Data In Existing Workbooks

For

Create New Workbooks
ActiveX/OLE (Object Linking and Embedding) enables the programmer to integrate and use objects from application programs. An ActiveX/OLE server is the application that creates ActiveX/OLE objects. The Microsoft Office Products (Excel, Word, PowerPoint, FrontPage, Outlook, etc.) are examples of ActiveX/OLE server applications. Microsoft Internet Explorer (IE) is also an ActiveX/OLE server application. ooRexx allows us to use the ActiveX/OLE objects created by the application in such a way as to automate the application. The generic term for this process is “Application Automation”.

ActiveX/OLE “beta” was first made available to Rexx in Object Rexx version 1.0.3. It became part of Object Rexx in version 2.1 and is included in ooRexx.
Automating Microsoft Excel
With ooRexx

Disclaimer
All examples in this presentation were done in Microsoft Excel 2002 (10.2614.2625).

Actual methods and/or attributes may vary with different versions of Excel, especially in regards to printing and saving workbooks.

Use of the Excel macro recorder (which produces VBA code), is the best place to determine what methods and attributes are necessary to accomplish the required task.
Object Creation
xlObj=.OleObject~New(‘Excel.Application’)

Setting The Visible Attribute
xlObj~Visible=.true
xlObj~Visible=.false

Defining Constants
xlCenter=xlObj~GetConstant('xlCenter')
Automating Microsoft Excel With ooRexx

Opening Existing WorkBook
OpenIt=xlObj~WorkBooks~Open(infile)

Creating A New WorkBook
xlObj~Application~SheetsInNewWorkbook=1
AddIt=xlObj~WorkBooks~Add

Setting The Alert Attribute
xlObj~Application~DisplayAlerts=.true
xlObj~Application~DisplayAlerts=.false
Automating Microsoft Excel With ooRexx

Saving A WorkBook
xlObj~ActiveSheet~SaveAs(outfile)

Closing All Workbooks
CloseIt=xlObj~Workbooks(1)~Close(SaveAll)

Quitting Excel
QuitIt=xlObj~Quit
Assigning Value To A Cell
xlObj~Cells(row,column)~Value = foo

Retrieving Value From A Cell
bar = xlObj~Cells(row,column)~Value

Determining Last Cell Used
xlLastCell=xlObj~GetConstant('xlLastCell')
lc=xlObj~ActiveCell~SpecialCells(xlLastCell)
parse var lc '$'max_c'$'max_r
Automating Microsoft Excel With ooRexx

Selecting A Range Of Cells
```plaintext
xlObj~Range("A1:"max_c||max_r)~Select
s=xlObj~Selection
```

Changing Selected Cell Attributes
```plaintext
s~HorizontalAlignment=xlCenter
s~Interior~ColorIndex=3
s~Font~Name='Arial Black'
```
Automating Microsoft Excel With ooRexx

Autofit Range of Column Widths
xlObj~Columns('A:U')~Select
xlObj~Selection~Columns~AutoFit

Drawing Boxes Around Cells
(Given that s is a selected range of cells)
s~Borders~LineStyle=xlContinuous
s~Borders~Weight=xlThin
s~Borders~ColorIndex=xlAutomatic
Automating Microsoft Excel With ooRexx

Merge Data Across Cells

xlObj~Range('A1:U1')~Select
  s=xlObj~Selection
  xlCenter = xlObj~GetConstant('xlCenter')
  xlBottom = xlObj~GetConstant('xlBottom')
  s~HorizontalAlignment=xlCenter
  s~VerticalAlignment=xlBottom
  s~Merge
Automating Microsoft Excel With ooRexx

Setup Print Criteria & Print

```bash
xap=xlObj~ActiveSheet~PageSetUp
xap~PrintTitleRows='$1:$2'
xap~PrintTitleColumns=''
xap~PrintArea=''
xap~LeftHeader=''
xap~CenterHeader=''
xap~RightHeader=''
xap~LeftFooter=''
xap~CenterFooter=''
xap~RightFooter=''
```
Automating Microsoft Excel With ooRexx

Setup Print Criteria & Print

xap~LeftMargin=xlObj~Application~InchesToPoints(0.75)
xap~RightMargin=xlObj~Application~InchesToPoints(0.75)
xap~TopMargin=xlObj~Application~InchesToPoints(1)
xap~BottomMargin=xlObj~Application~InchesToPoints(1)
xap~HeaderMargin=xlObj~Application~InchesToPoints(0.5)
xap~FooterMargin=xlObj~Application~InchesToPoints(0.5)
xap~PrintHeadings=.False
xap~PrintGridlines=.False
Automating Microsoft Excel With ooRexx

**Setup Print Criteria & Print**

```plaintext
xap~PrintComments=xlPrintNoComments
xap~PrintQuality=1200
xap~CenterHorizontally=.False
xap~CenterVertically=.False
xap~Orientation=xlLandscape
xap~Draft=.False
xap~PaperSize=xlPaperLetter
xap~FirstPageNumber=xlAutomatic
xap~Order=xlDownThenOver
xap~BlackAndWhite=.False
xap~Zoom=.False
xap~FitToPagesWide=1
xap~FitToPagesTall=100
xap~PrintErrors=xlPrintErrorsDisplayed
```
Automating Microsoft Excel With ooRexx

Print The WorkBook

xlObj~ActiveWindow~SelectedSheets~PrintOut
Converting A VBA Macro To ooRexx
(Autofit Columns A & B)

VBA
Sub Macro1()
    Columns("A:B").Select
    Selection.Columns.AutoFit
End Sub

ooRexx
xlObj~Columns(“A:B”)~Select
xlObj~Selection~Columns~AutoFit
Some Final Points To Remember

Make full use of the macro recorder.

Record your macros in the current workbook.

When converting VBA to ooRexx watch the VBA code for numerous attributes that are set to their default – many of them can be omitted.

Be prepared to do a lot of “trial & error” analysis.

Watch your “Task Manager” for orphaned Excel processes.

When using one ooRexx application to call another ooRexx application that performs Excel automation, make “arrangements” in your calling program to use WMI to kill orphaned Excel processes – this is an issue with Excel, not ooRexx.
Automating Microsoft Excel
With ooRexx

A Real World Demonstration

Purpose: Analyze payroll time data and create an exception report for each office that will include any employee that matches defined criteria.

Desired Output: Color PDF, but “exception” items highlighted such that they are clearly visible if printed in B/W.

# of Offices: 17
# of Employees: 517
Time Required: 3.47 Minutes
Automating Microsoft Excel With ooRexx

A Real World Demonstration

Actual demonstration will be for two (2) offices and includes infoMessage boxes for the first office only to explain the next step.