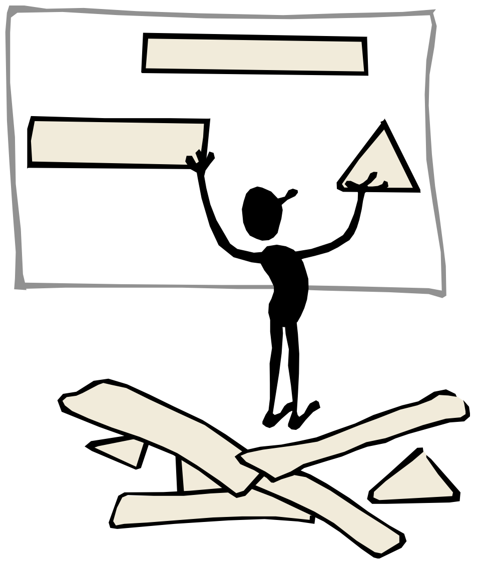
Sage Summit 2011

July 10-15, 2011 | Washington, DC | Gaylord National Hotel and Convention Center

Sage ERP MAS 90 and 200: Customizer Basics and Beyond



Sage Summit 2011

Introduction

- Elliott Pritchard – Principal Architect, Sage MAS90/200
- Steve Malmgren – Sr. Director of Development, Common Components
- This presentation will be available online after the conference. You will receive an email for the Summit session website approximately 1-2 weeks after Summit.
- Follow Sage on Twitter: @Sage_Summit
  - Use the official Summit hashtag: #SageSummit

CPE Credit

- In order to receive CPE credit for this session, you must be present for the entire session.
  - Session Code: **C-ERP35**
  - Recommended CPE Credit = **1.5**
  - Delivery Method = Group Live
  - Field of Study = Specialized Knowledge and Applications
- Visit the Sage Summit Connect kiosks to enter CPE credit during the conference.

Key Objectives

- After participating in this session you will be able to:
  - Be refreshed on some of the Customizer basics
  - Understand the basics of 4.50 Customizer new features, and refresher on existing features
  - Be able to print reports, registers, and forms through the Paperless Office system via User-defined Scripts
  - Get a glimpse into some areas of User-defined scripting that you may not have been aware you could use.
  - Use a combination of user-defined scripts, along with script buttons, and UDF flow, to tailor Sage ERP MAS 90 and MAS 200 to your needs.

The reasons why we customize.

- Add features to Sage's core product that work the way we do
- Differing business needs
- User preferences to improve productivity

A Review of Key Concepts and Terms

- UDF – User Defined Field. New fields that can be added to existing MAS 90 tables.
- UDT – User Defined Table. New tables that can be added to either the MAS_XXX company or MAS_SYSTEM database.
- UDS – User Defined Scripts. Scripts that can be attached to buttons, or database events.
- UI – User Interface. This refers to Sage ERP MAS 90 entry screens
- API – Application Programming Interface.
- Business Object Interface – API to programmatically “use” Sage ERP MAS 90 objects to read and write data, generate and print reports, invoke register/update routines.
- Business Rules – validations that are enforced to ensure proper data is written to a table.

Customizer Selection

- Access through Customizer Menu or <ctrl>+F9 while in the screen you want to customize
- Change tab order
- Hide non-essential fields
- Add new fields to the form
- Add buttons
- Add new panels

Review of User-Defined Fields and User-Defined Tables

- Add your custom fields to almost any table
  - Example Lot Expiry Date in IM_ItemCost
- Flow your new data through the system
  - Move Lot Expiry date from Data entry through the update and back into IM_ItemCost
- Create your own tables

Scripting Review

- 4.40 Introduced user-defined scripts
- Appendix – To have a look at some background info

Taking advantage of new features in Sage ERP MAS 90 Version 4.50

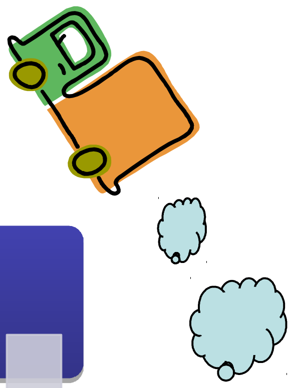
- Scripting can now take advantage of the Paperless Office System
- New class to create Purchase Orders from Sales Orders
- Invoking UI Changes, as if the user had typed it in.
- Using Data from List boxes

Warehouse Transfer Example

- REQUIREMENT: Add ability to treat a Warehouse transfer like a sale and a receipt. My warehouse transfers need an audit trail that Inventory Transaction Entry doesn't fulfill. We need picking sheets, shipping documents, and receipts.
- Detail Requirements:
  - Warehouse manager needs to replenish stock from Main Warehouse, by submitting their own order, and receiving it.
  - Via Inventory Maintenance satellite warehouse orders items. Create Sales order at cost, as they "Shop"
  - While viewing the satellite warehouse orders, auto-create a purchase order for warehouse to use to receive goods.
  - Then ship and receive like normal purchase and sales orders

Warehouse Transfer Workflow

1. Create and Add Items to Sales Order
2. Create Purchase Order from Sales Order
3. Ship Sales Order
4. Receive Goods from Purchase Order
5. Clear created Invoices



Sage Summit 2011

Analyzing Work flow

- Create customer and vendor records for each warehouse, along with paperless records.
- Create sales order while browsing items in Item Maintenance
  - Validate warehouse
- Create purchase order from sales order, when ready to ship.
- Ship/Invoice Order and email warehouse manager that the order has shipped. (Inventory relieved from Main warehouse)
- Receive Goods into inventory (Inventory received into new warehouse)
- Clear invoices

How do we do it?

- Warehouse Maintenance
  - Add fields to store associated customer and vendor numbers, also for current open sales order
- Item Maintenance
  - Use procedure to order items from the Quantity panel, to auto-create sales order, when selecting warehouse and quantity
- Customer Maintenance
  - From Sales Orders folder tab, view open orders and create Purchase Order from selected orders

Ship it! … or at least Transfer it!

What did we just see and what did we use?

- UDF's within existing MAS 90/200 tables
  - Warehouse code in Vendor and Customer tables
  - Customer, Vendor and Sales Order Number in Warehouse table
- UDT's to validate our Transfer To Warehouse
- Data flow through the framework
  - Moving UDF_Warehouse to the Sales Order
- Standard Screen Customization, to add new fields
- Scripting… Scripting… Scripting…
  - Everything from button Scripts to create orders, to event scripts to show and hide fields and to print and email reports

Sage Summit 2011

Script Events

Event	Type	Script Executes…
Pre-Validate	Column	After dictionary validation, prior to Sage/Master Developer
Post-Validate	Column	After the column value has been validated
Script-Initialization	Table	Runs once per business object on first script run
Set-Default-Values	Table	When a new record is established in the business object
Pre-Write	Table	Before a record is written
Post-Write	Table	After a record is written
Pre-Delete	Table	Before a record is deleted
Post-Delete	Table	After a record is deleted
Post-Read	Table	After a record is read
Pre-Totals	Table	Line Entry only, before totals are calculated

What did we just see and what did we use? (con’t)

- New 4.50 features incorporated
  - Printing report through paperless office and emailing
  - Use of purchase order from sales order creation object
- InvokeChange(“QuantityOrdered”, 3, <“GD_Lines”>)
  - Using this will actually change the value of the field as if the user typed it in themselves.
  - Retains the messaging that would normally occur if the user entered the value in the field.
  - Updates other related fields on the screen automatically
  - Works for grid cells as well as other data fields

What do we do when things go wrong?

- Panic? Or Debug?
- Test before implementing
  - Use on a test company and system first while developing
- oScript.DebugPrint()
- Msgbox() for debugging, not for showing messages.
- Common Scripting “Doh”’s!
  - Not including a retval = for function calls
  - Always remember…spelling counts!

How to Find Examples for VBScript

- Google is your friend!!
  - How to read a CSV file? Google “VBScript how to read a csv file”
  - Interested in creating an Excel spreadsheet? Google “VBScript how to create an Excel file”
  - Want to know how to interface with outlook? Google “vbscript how to invoke outlook”

Questions?

Your Feedback is Important to Us!

- Please visit a Sage SummitSurvey kiosks to complete the evaluation form for this session.
- Remember each completed survey form is another entry for one of three iPad drawings.
- Your feedback helps us improve future sessions and presentation techniques.
- Please include your session code on the evaluation form: C-ERP35

Contact Information

- Presenter Contact Information:

  Elliott Pritchard

  Sage

  elliott.pritchard@sage.com

- Follow Sage on Twitter: @Sage_Summit
  - Use the official Summit hashtag: #SageSummit
- **Thank you for your participation.**

Appendix

- How to Enable oScript.DebugPrint()
- Creating Objects
- Scripting.doc
- Script Events
- Link to 4.40 Customizer Recorded Sessions
- Link to 4.30 Customizer Recorded Sessions
- Microsoft's On-line VBScript Reference

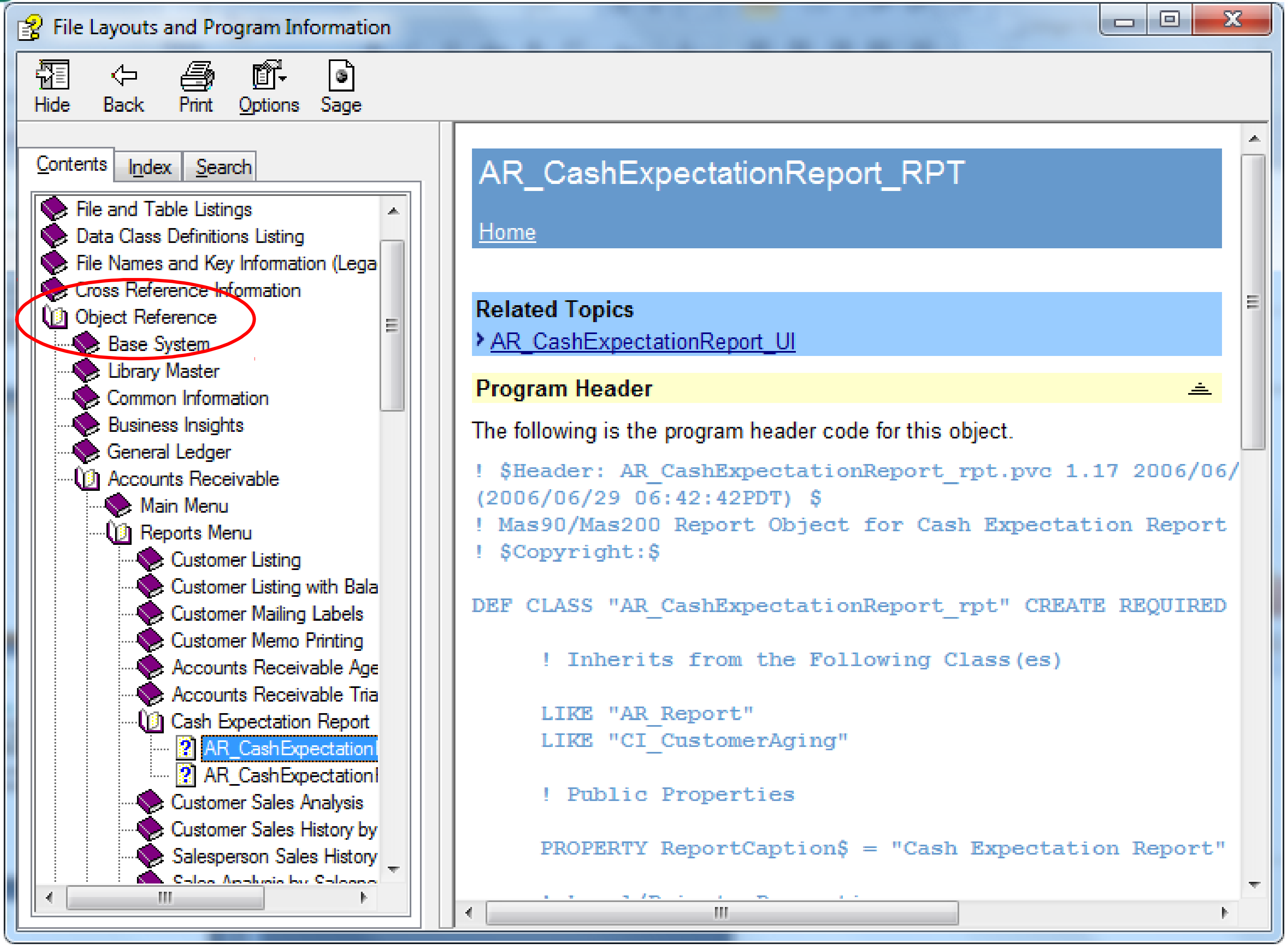
Back

Sage ERP MAS 90 and 200 Object Types

Type	Suffix	Description
Service	_SVC	Read only access to tables
Business	_BUS	Read/Write/Delete enforces business rules when writing to tables
Reports and Forms	_RPT	Print reports and forms with no UI
Update	_UPD	Audit trail and update processes
User Interface	_UI	Sage ERP MAS 90 and 200 screens

Naming Conventions

- Service and Business
  - Table name + _SVC or _BUS
  - e.g. AR_Customer_SVC; AP_Vendor_BUS
  - Exception - Line entry business objects (_BUS):
    - Table names follow convention of XX_XxxxxXxxxHeader and XX_XxxxxXxxxDetail
    - Object name disregards Header and Detail naming is the object is responsible for both tables
    - Name: SO_SalesOrder_BUS; AR_CashReceipts_BUS
  - Reports, Forms, Updates and UI, consult Object Reference (see next slide) to be sure



AR_CashExpectationReport_RPT

Home

Related Topics

AR_CashExpectationReport_UI

Program Header

The following is the program header code for this object.

```
! $Header: AR_CashExpectationReport_rpt.pvc 1.17 2006/06/
(2006/06/29 06:42:42PDT) $
! Mas90/Mas200 Report Object for Cash Expectation Report
! $Copyright:$

DEF CLASS "AR_CashExpectationReport_rpt" CREATE REQUIRED

    ! Inherits from the Following Class(es)

    LIKE "AR_Report"
    LIKE "CI_CustomerAging"

    ! Public Properties

    PROPERTY ReportCaption$ = "Cash Expectation Report"
```

Sage Summit 2011

Service Objects (Key Methods() and Properties)

- MoveFirst(); MoveNext(); MoveLast(); MovePrevious()
- GetKeyColumns() As String – Helper Method
- GetColumns() As String – Helper Method
- SetKeyValue(<keyColumn As String>, <keyValue As String)
- Find() - no arguments, must use SetKeyValue() for each key column
  Find(<keyValue as String>) - only used on single column keys

Service Objects (Key Methods() and Properties)

- GetValue(<column As String>, <val As String OR Numeric)
- GetRecord(<rec As String>, <pvx IOL - not useful in scripting instead use GetColumns()>)
- SetBrowseFilter(<first n characters to filter As String>)
- BOF; EOF – indicates whether or not the row pointer is at the beginning or end of file respectively
- LastErrorNum, LastErrorMsg As String – contains error code and message of last error that occurred. ***TIP*** - May not be based on the last operation if successful. (NOTE: These properties are in ALL object types and should be checked on failed method calls)

Business Objects (_BUS) - Summary

- Business Objects enforce all of the business rules for adding data into the system.
- ***Behavior can be changed via User-Defined Scripting!!***
- Always use Business Objects rather than writing directly to the database to ensure the integrity of the system
- VI uses the Business Objects to import data as does the User Interface (UI) of Sage ERP MAS90 and MAS200
- Some history tables have business objects defined, *BUT* are for migrating a customer from another system to Sage ERP MAS90 and MAS200. Best practice is to use data entry tables and process through the updates (which can all be automated) to get data into history tables.

Business Objects - Key Methods() and Properties

- Inherits all Service Object methods and properties
- SetKey() - Same as Find (with or without arguments, but used to place a row into an EditState for new rows)
- SetValue(<column As String>, <val As String or Numeric>)
- Write() - saves changes
- Delete() - deletes current row
- Clear() - takes row out of edit state and discards any changes

Business Objects - Key Methods() and Properties

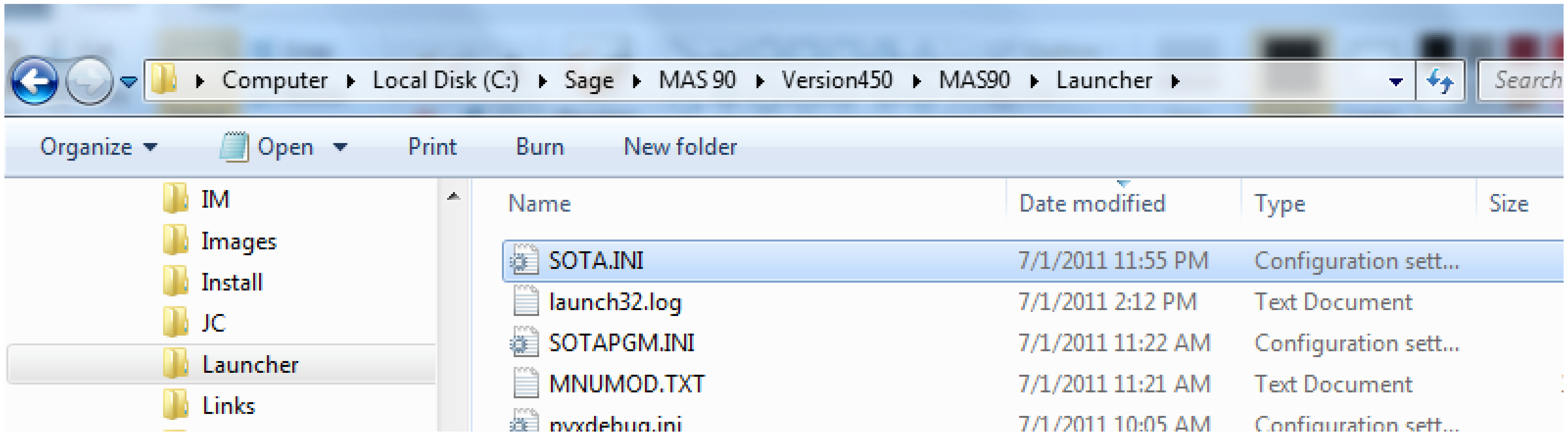
- GetDataSources() As String - returns list of columns that validate against a Service object
- GetChildHandle(<data source As String>) As Object - returns an Object handle to a Service Object
- ReadAdditonal() - reads ALL child data sources for current row ReadAdditional(<data source As String>) - read specified data source
- SetCopyKeyValue(<keyColumn As String>, <keyValue As String>)
- CopyFrom()

Business Objects - Key Methods() and Properties

- EditState – 0 if no record in memory; 1 if existing record; 2 if new record
- RecordChanged – 0 if no changes, 1 if record has changed

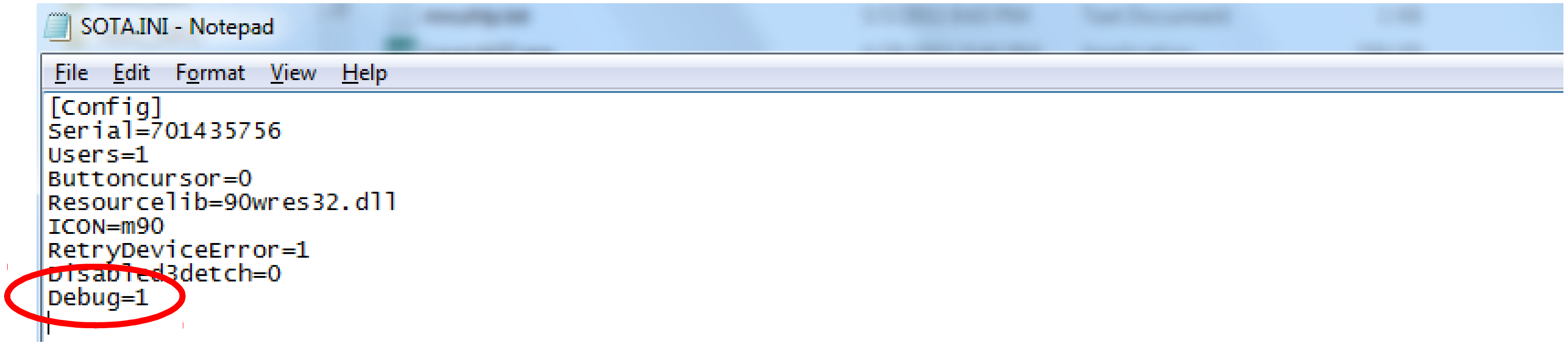
Sage Summit 2011

How to Enable oScript.DebugPrint()



Computer ▸ Local Disk (C:) ▸ Sage ▸ MAS 90 ▸ Version450 ▸ MAS90 ▸ Launcher ▸
Search
Organize
Open
Print
Burn
New folder
IM
Images
Install
JC
Launcher
Links

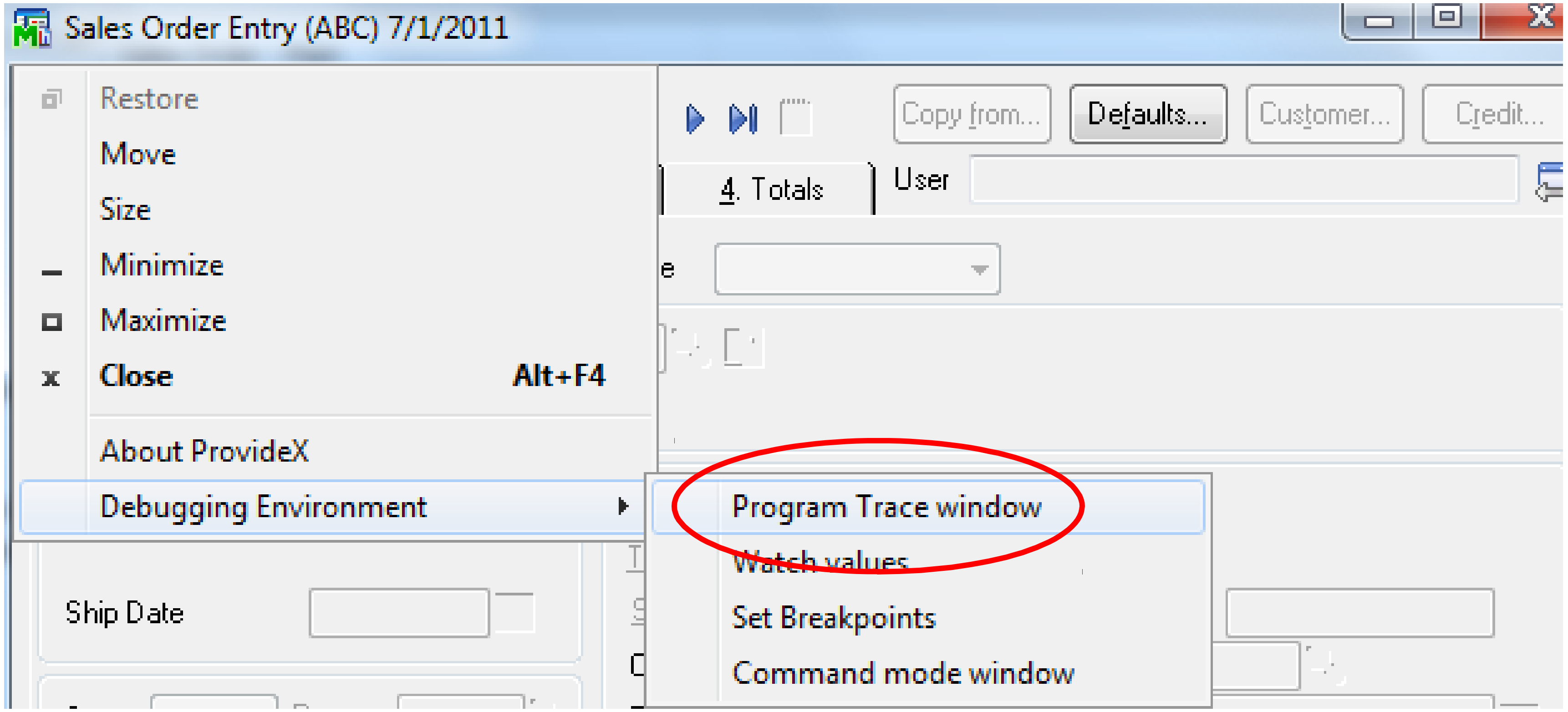
Name	Date modified	Type	Size
SOTA.INI	7/1/2011 11:55 PM	Configuration sett...	
launch32.log	7/1/2011 2:12 PM	Text Document	
SOTAPGM.INI	7/1/2011 11:22 AM	Configuration sett...	
MNUMOD.TXT	7/1/2011 11:21 AM	Text Document	
pvxdebug.ini	7/1/2011 10:05 AM	Configuration sett...	



```
[Config]
Serial=701435756
Users=1
Buttoncursor=0
Resourcelib=90wres32.dll
ICON=m90
RetryDeviceError=1
Disabled3detch=0
Debug=1
```

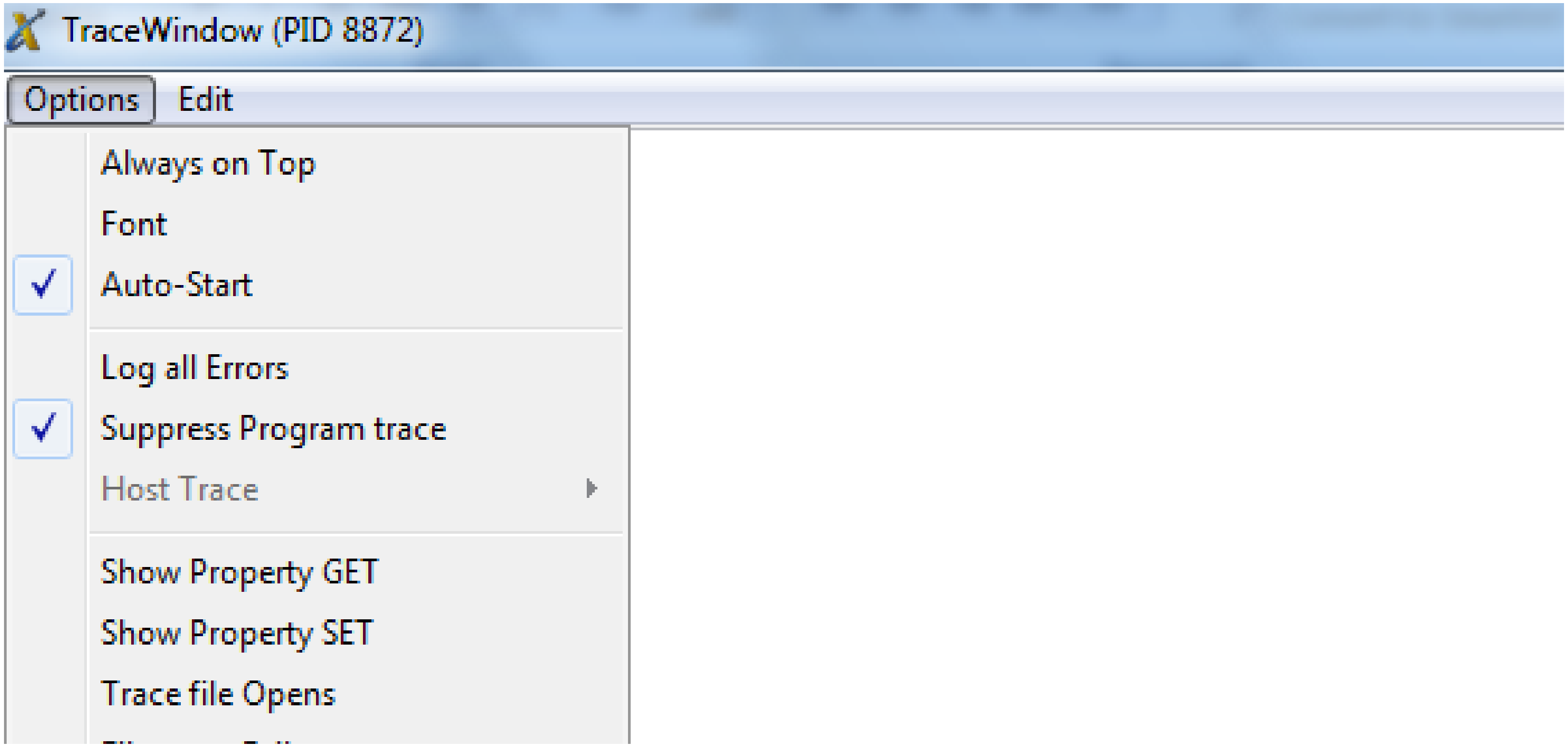
Sage Summit 2011

Right-Click on title bar – Select Debugging Environment..Program Trace window



Sage Summit 2011

Displays this window – Select Suppress Program Trace and optional Auto-Start during script debugging



TraceWindow (PID 8872)

Options Edit

- Always on Top
- Font
- ✓ Auto-Start
- Log all Errors
- ✓ Suppress Program trace
- Host Trace
- Show Property GET
- Show Property SET
- Trace file Opens

Creating Objects

- User-Defined Scripting (4.40 and above)

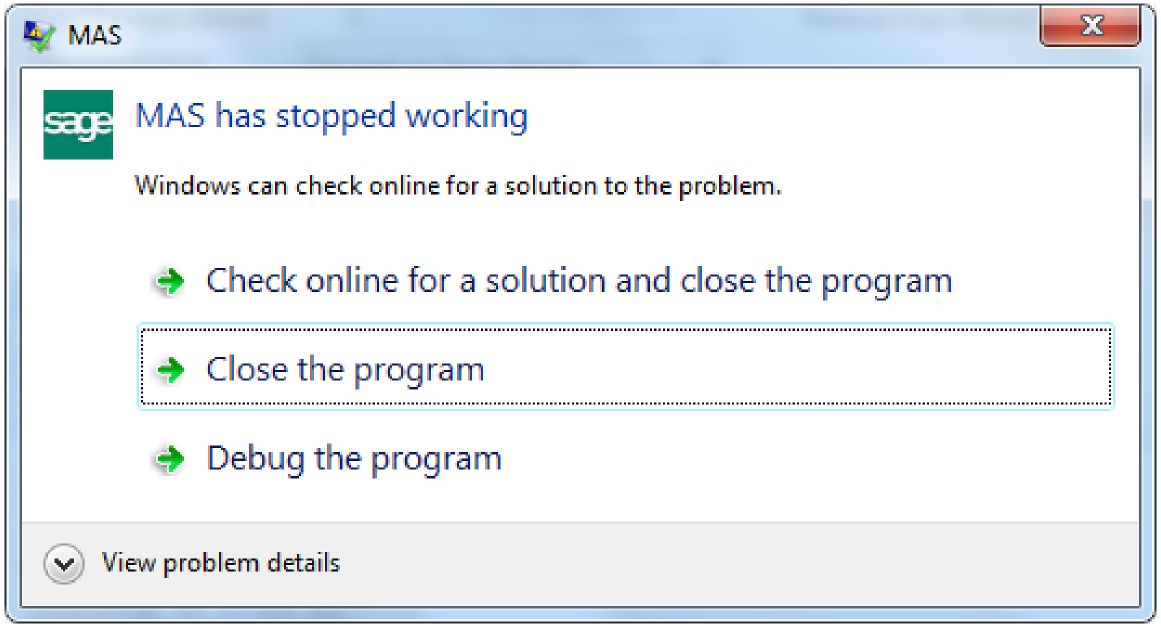
  -
    ```
    oCustSvc = oSession.GetObject(“AR_Customer_svc”)
    If oCustSvc <> 0 then
         SET oCustSvc = oSession.AsObject(oCustSvc)
    End If
    ```
  - Security of current session is used to establish authorization
- COM (Providex.Script)
  - Set oSession = oPVXScript.NewObject(“SY_Session”)
  - … (establish user, password for login, set module and module date; check security, etc.) …
  - Set oCustSvc = oPVXScript.NewObject(“AR_Customer_svc, oSession)

Referencing Existing Objects

- User-Defined Scripting (4.40 and above)
  - ***SET*** `oCustSvc = oBusObj.AsObject(GetChildHandle(“CustomerNo”))`
  - ***SET*** `oLines = oBusObj.AsObject(oBusObj.Lines)`
  - `NOTE: User-Defined Scripting requires the .AsObject() wrapper to indicate return type is an object handle`
- COM (Providex.Script)
  - ***SET*** `oCustSvc = oBusObj.oGetChildHandle(“CustomerNo”)`
  - ***SET*** `oLines = oBusObj.oLines`
  - `NOTE: o prefix in oBusObj.oLines to indicate variable return type is an object handle`

What Can Go Wrong?

- Wrong method or property name for a valid object handle will result in a GPF crash. Usually a typo on the method name OR the right method name using the wrong object handle variable



MAS

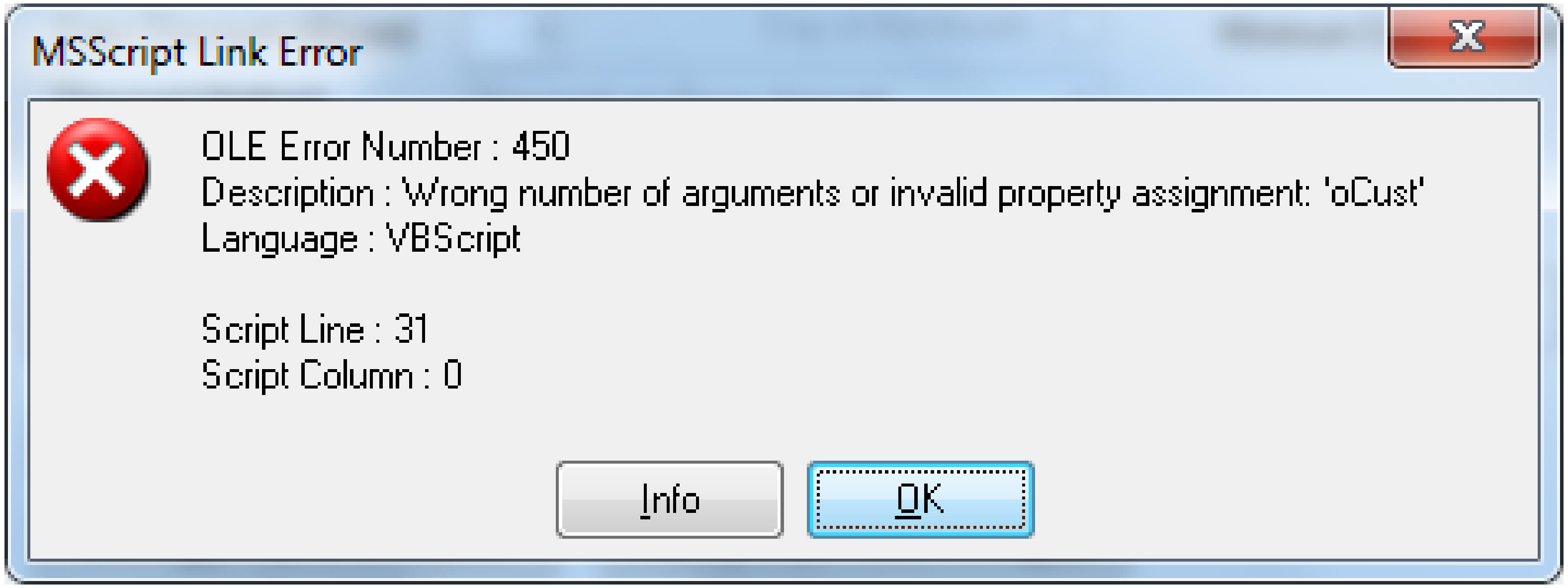
MAS has stopped working

Windows can check online for a solution to the problem.

Sage Summit 2011

What Can Go Wrong

- Forgetting to do a SET on an object handle then trying to use a method or property



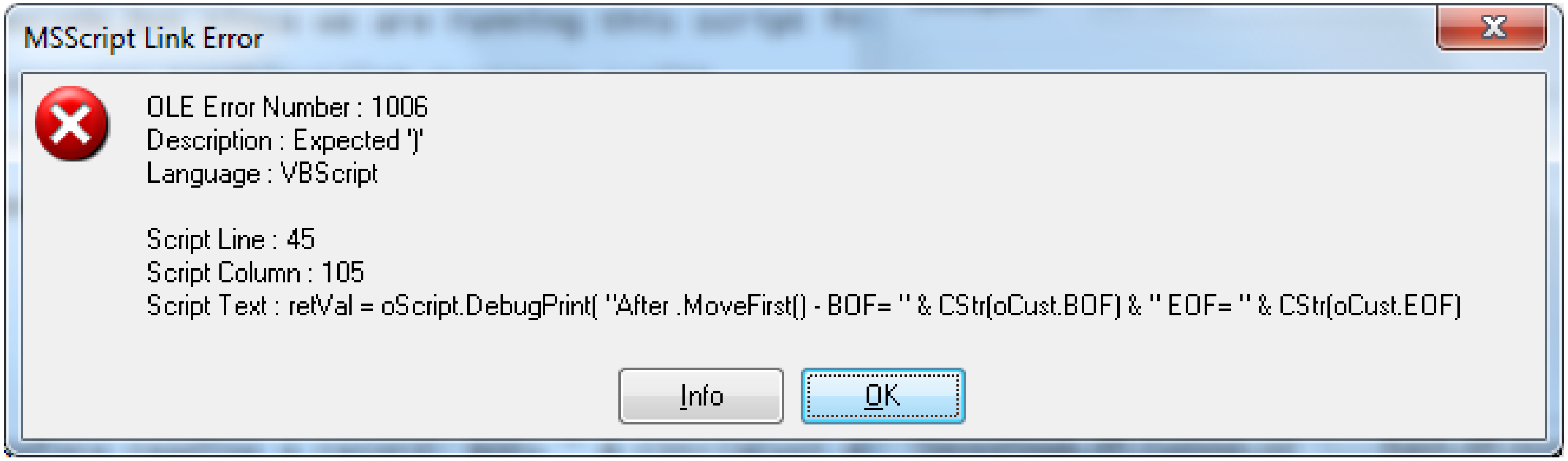
MSScript Link Error

```
OLE Error Number : 450
Description : Wrong number of arguments or invalid property assignment: 'oCust'
Language : VBScript

Script Line : 31
Script Column : 0
```

What Can Go Wrong?

- Missing Parenthesis in a Method Call (typically only in BT_Link scripts and User-Defined Script will catch this syntax error)



```
OLE Error Number : 1006
Description : Expected ')'
Language : VBScript

Script Line : 45
Script Column : 105
Script Text : retVal = oScript.DebugPrint( "After .MoveFirst() - BOF= " & CStr(oCust.BOF) & " EOF= " & CStr(oCust.EOF)
```

What Can Go Wrong?

- Forgetting to initialize return values as part of an argument to either a “” for string or 0 for numeric can cause problems.
- Typo in a argument (such as GetValue(“CutomerNo$”, val))
  - Missing the “s” in CustomerNo$
  - Val will return blank (and you will wonder for hours how that can be until you see the typo mistake)
  - Also if val was initialized to a number it will return 0
- Forgetting to check retVal on a SetValue(), Write() or Delete() and checking LastErrorMsg to see why it failed
  - If you don’t check, it will continue on and you will wonder why you did not get expected results

What Can Go Wrong?

- Using a business object from another module
  - There can be unexpected results, even strange errors that you won’t understand
  - Best practice is store off current module and do a SetDate() and SetModule() for the module code the object belongs to
- SetKey() can fail if another user has a row locked (usually only in line entry objects)
- GetObject() can fail if a user is in a single user task (such as A/R Setup Options)