Sage Summit 2011

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

Sage ERP MAS 90 and 200: Using Business Object Interface - Beginner

Part 1 of 2 – Course Number P-ERP22 and P-ERP22B

sage

CPE Credit

- In order to receive CPE credit for this session, you must be present for the entire session.
  - Session Code: **P-ERP22B (Monday) OR P-ERP22 (Tuesday)**
  - Recommended CPE Credit = **1**
  - Delivery Method = Group Live
  - Field of Study = Specialized Knowledge and Applications
- Visit the Sage SummitConnect kiosks to enter CPE credit during the conference.

Introduction

- Steve Malmgren – Sr. Director of Development
- Elliott Pritchard – Principal Software Architect
- 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. (Or sooner if you brought your laptop or a jump drive)
- Follow us on Twitter: @Sage_Summit, @swmalm
  - Use the official Summit hashtag: #SageSummit

Multipart Sessions

- This session is part of a focused, multipart session series.
- Attendees that register for a multipart session series must register for all parts in the series.

Learning Objectives

- After participating in this session, you will be able to:
  - Understand the different Object Types available for use in Sage ERP MAS 90 and MAS 200
  - Distinguish between “using” objects and “changing the behavior” of objects
  - Use the key methods and properties of each object type

Key Concepts and Terms

- 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.
- COM – Component Object Model – using properties and methods of objects

Sage Summit 2011

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

sage

Approach

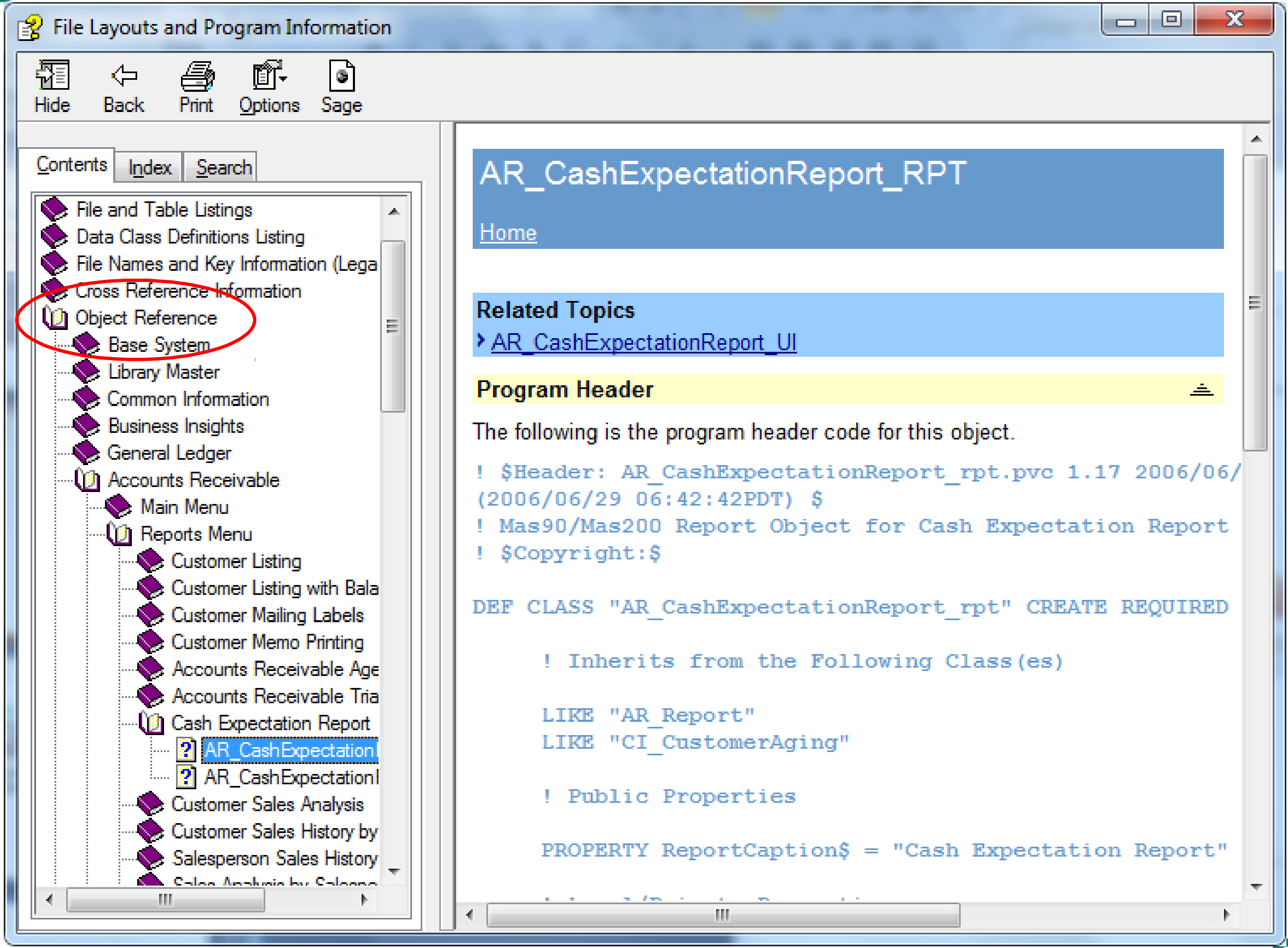
- Overview of object type and how they are used
- Cover Key Methods and Properties of each object type
- Examine code in example scripts
- Run sample scripts with debug trace window visible

- A lot to cover, all examples and presentation will be available to review during conference and/or when you get back to your office

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

Sage Summit 2011



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

Service Objects (_SVC) - Summary

- Service Objects are Read Only access to tables in the system.
- Primarily used by the Business Objects (_BUS) to validate fields against other tables.
  - Sales Order validates the customer being added to the SO_SalesOrderHeader table using the AR_Customer_SVC object
  - Supports flow of UDFs
- Some Service Objects provide other services such as:
  - calculating an aging for a customer
  - calculating an invoice due date based on a terms code
    - SO Sales Order, SO Invoice and AR Invoice all use the AR_TermsCode_SVC.CalcDates() method to calculate due date and discount dates
  - View Public Methods in Object Reference for desired object and inherited classes

Service Objects (_SVC) - Summary

- ***TIP*** – When obtaining an object handle to a Service Object, use oBusObj.GetChildHandle(<dataSource> As String) rather than oSession.GetObject(<serviceObject As String) to minimize memory usage.
  - First Option – service object is already in memory. May need to do a oSvc.ReadAdditional(<dataSource As String) OR .Find() to populate data
  - Second Option – creates a new copy of the object in memory

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)

Let’s Take a Look – Service Object (AR_Customer_svc)

- Appendix – To have a look at some background info
- Service_AR_Customer_svc.txt – Service Object Script

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

Let’s Take a Look – Business Object (AR_Customer_bus)

- Business_AR_Customer_bus.txt – Business Object Script – Existing Customer
- NewRecord_AR_Customer_bus.txt – Business Object – Create New Customer (GetNextCustomerNo(); CopyFrom())

Summary - Service and Business Objects

- Service Objects – Read Only
  - SetKeyValue()
  - Find()
  - GetValue()
- Business Objects – Read/Write/Delete
  - SetKeyValue()
  - SetKey()
  - GetValue()
  - SetValue()
  - Write()
  - Delete()

Summary - Service and Business Objects

- Service Objects – What we've done
  - Browsed rows with and without a browse filter
  - Examined BOF and EOF
  - Found a specific row and retrieved a value
  - Used a While/Wend loop to calculate over 90 day balance for division 01
- Business Objects – What we've done
  - Edited an existing record; examined a failed SetValue() and Delete() method call
  - Created a new customer; used GetNextCustomerNo; used SetCopyKeyValue() and CopyFrom() methods
  - Saw how to use GetChildHandle() to get a service object handle

Business Objects (Line Entry) - Key Methods and Properties

- AddLine() - creates a new, empty line with default values
- Lines – object handle to the detail business object
  - Set oLines = oBusObj.AsObject(oBusObj.Lines)
  - Just like any other business object (GetValue(); SetValue(); Write(); Delete())
- GetEditKey(<lineKey As String>) – Used to get the edit key of a line based on the 6 character LineKey column (will cover these in Part 2)
- EditLine(<editKey As String>)
- DeleteLine(<editKey As String>)

Let’s Take a Look – Business Object (SO_SalesOrder_bus)

- Create_SO_SalesOrder_bus.txt – Business Object Script – Create SalesOrder

Summary – Line Entry Business Objects

- What we’ve done
  - Used a SO object from the AR module – Set the Module and Date to SO prior to “getting” the SO_SalesOrder_bus object
  - GetNextSalesOrderNo
  - SET oLines to the oSO.AsObject(oSO.Lines) property to access the detail business object
  - Used the oScript.DebugPrint() to display any error messages
  - Used the oScript.SetStorageVar() to store off a Sales Order number as well as detail line to be used in the edit line and delete line scripts for Part 2.

Report Objects (_RPT) - Summary

- Report Objects are used to print, preview, defer or deliver (using paperless office) any Report or Form via the Business Object Interface.
- Forms need to be run once from the UI so that the customer can choose which report template (Marbled, Plain Paper, etc.) to use.
- NOTE: Audit Journals and Registers, as well as related recaps and error logs are printed from within the Update (_UPD) objects

Report Objects - Key Methods() and Properties

- SelectReportSetting(<savedReportSetting As String)
  - If no saved settings available can ALWAYS use “STANDARD” which is factory defaults
  - Use SY_ReportSetting_svc to determine any saved settings
- ProcessReport(<destination As String)
  - “PREVIEW”
  - “DEFERRED”
  - “PRINT” – based on default printer for report setting
  - “EXPORT”, “EXPORTDATA” (warning prompts user for type and location)

FORM Objects - Key Methods() and Properties

- QuickPrint As String
  - Not all forms support this – Check Object Reference (e.g. AR_Statement_rpt does not, SO_SalesOrderPrinting_rpt does)
- ***TIP***: Must select a template (e.g. marbled, plain paper, etc.) first via the UI or script will not work

Let’s Take a Look – Report and Form Objects

- ReportSimple_AR_AgedInvoice_rpt.txt – Report Object Script
- QuickPrint_SO_SalesOrderPrinting_rpt.txt – Forms Object

Summary – Report and Form Objects

- What we've done
  - Used SelectReportSetting() for both STANDARD and saved setting (even had to set one up since it failed the first time)
  - Used ProcessReport() for all output types
  - Used oScript.GetStorageVar() to retrieve the sales order number created and saved off in another script, to use to set the QuickPrint property of a FORM object

Update Objects (_UPD) - Summary

- Update Objects are used to post data entry transaction through to the history tables and to post GL transaction associated with those entries.
- Encompasses both the Register (audit trail report) and the update posting logic.
- The Register must be “Printed” either to hard copy, paperless office or deferred print in order for the transactions to be updated. (Previewed registers can only be updated with appropriate security event rights)
- If errors are detected during the register printing, the update cannot continue and an error log can be printed

Update Objects - Key Methods() and Properties

- ProcessReport(<destination As String)
  - “PREVIEW” – cannot update unless Security Event rights
  - “DEFERRED”
  - “PRINT” – based on default printer for report setting
  - “EXPORT”, “EXPORTDATA” (warning prompts user for type and location)
- Update() – Executes the update process

Let’s Take a Look – Update Objects

- InvoiceOrder_SO_Invoice_bus.txt – Invoice Order Script (Bonus Coverage!!)
- Update_SO_SalesJournal_upd.txt – Update Object

Summary – Update Objects

- What we've done
  - Covered how to invoice a sales order and copy detail lines from the sales order to the invoice (Bonus coverage)
  - Used the ProcessReport() to print the main Audit Trail Register. Same method as in the reports, so all output types are available
    - NOTE: PREVIEW will not allow update unless have proper security event rights
    - ProcessReport() can fail if there is an error condition.
  - Used the Update() method to post the Invoices through to GL, Open Invoice, History, etc.

User Interface Objects (_UI) - Summary

- User Interface objects are used to display the Sage ERP MAS 90 and MAS 200 data entry screens
- The object class names are used to obtain security rights at a task level.
- As an alternative, can be launched in a separate process using the oSession.InvokeTask() method

UI Objects - Key Methods() and Properties

- Process() – displays UI
- Process(<recordToDisplay As String)
  - Will bring the record (e.g. oARCust.Process(“01ABF”) – will display customer 01-ABF

Let’s Take a Look – UI Objects

- UI_AR_Customer_ui.txt – AR_Customer_UI

Questions?

Additional Learning Opportunities

- For information about additional learning opportunities visit www.sageu.com (Sage University).
- Training options include:
  - Anytime Learning—Recorded online training sessions.
  - Realtime Learning—Live, online learning.
  - Replay Learning—Recordings of live classes.

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:
  - Monday (P-ERP22B)
  - Tuesday (P-ERP22)

Contact Us

- Presenter Contact Information:
  - Steve Malmgren, Elliott Pritchard
  - Twitter @swmalm
- Follow us on Twitter: @Sage_Summit,
  - Use the official Summit hashtag: #SageSummit
- **Thank you for your participation.**

Summary

- <It’s a good idea to summarize your key points at the end of your session and focus on how to apply what they have learned. Try to relate your summary points to your learning objectives to reinforce these objectives for session participants.>

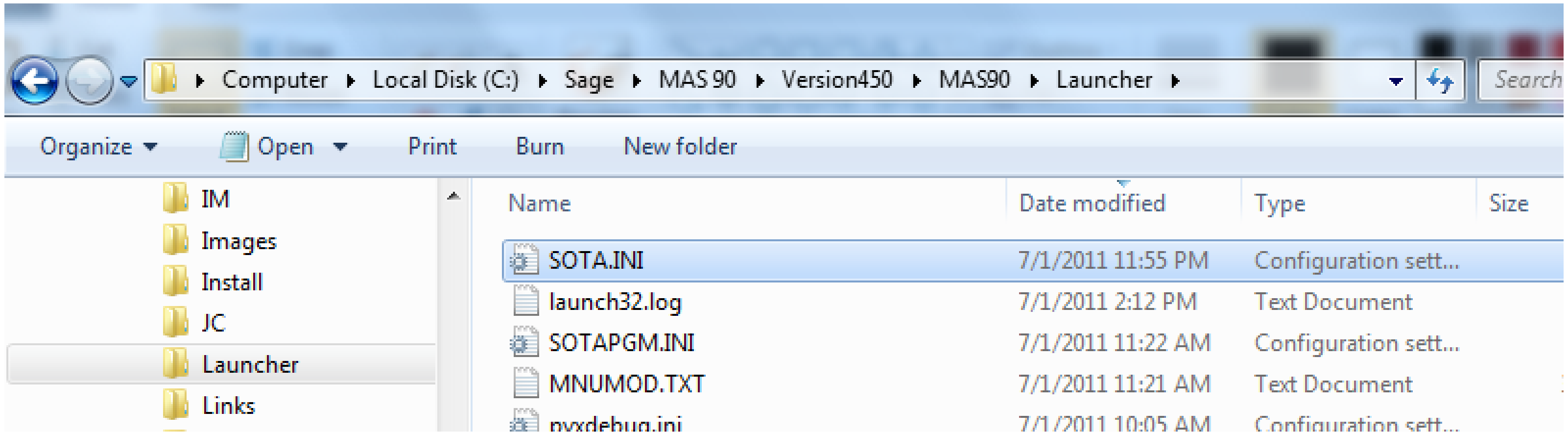
Appendix

- 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

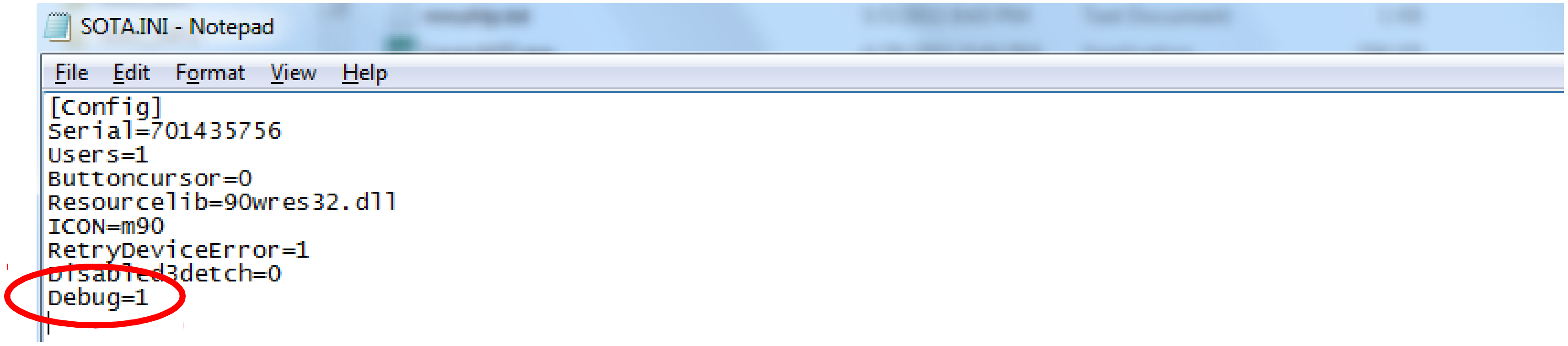
Back

Sage 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...	

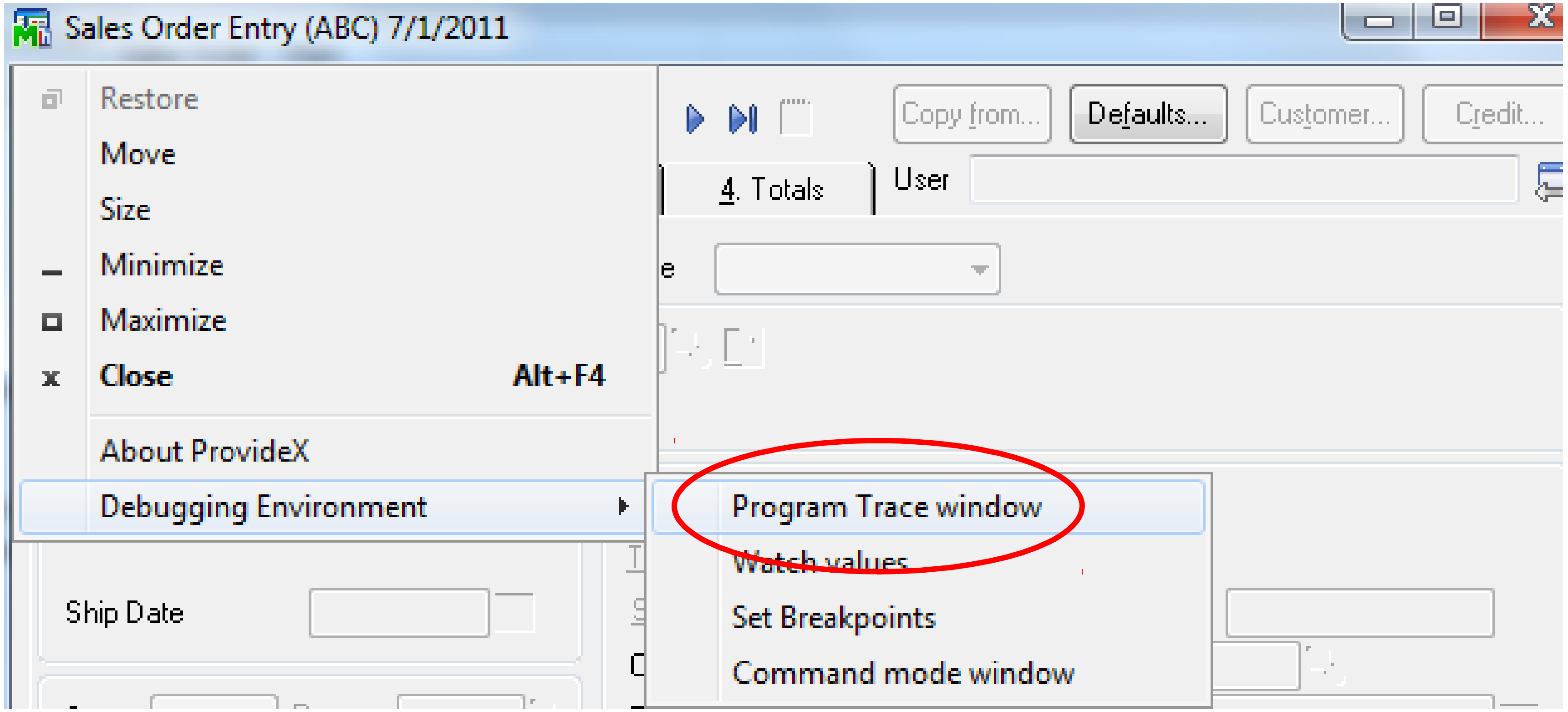


```
[Config]
Serial=701435756
Users=1
Buttoncursor=0
Resourcelib=90wres32.dll
ICON=m90
RetryDeviceError=1
Disabled3detch=0
Debug=1
```

sage

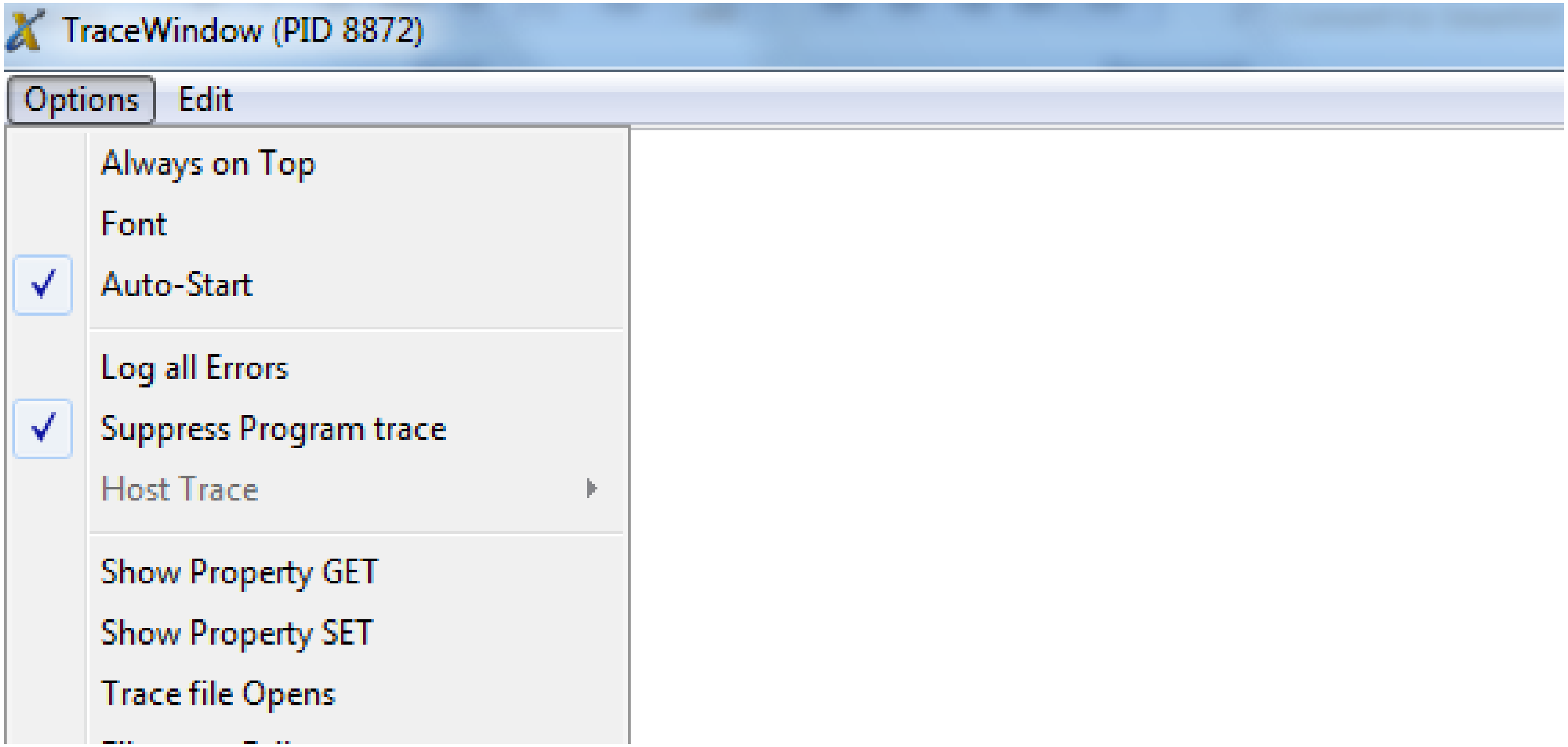
Sage Summit 2011

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

sage

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`

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

sage