



Sage Summit

2011

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

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

Part 2 of 2 – Course Number P-ERP23 and P-ERP23B

CPE Credit

- In order to receive CPE credit for this session, you must be present for the entire session.
 - Session Code: **P-ERP23 (10:00AM) OR P-ERP23B (11:30)**
 - Recommended CPE Credit = **1.5**
 - 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:
 - Describe all of the scripting events
 - Handle Batches in Data Entry Objects, Forms Objects and Update Objects
 - Edit and Delete lines of a Line Entry Object
 - Explore the nuances of Cash Receipts Batch and Deposits
 - Selection criteria with Report Objects
 - Dealing with Lot/Serial Tiers
 - Look at how to deal with things when they go wrong
 - Search Examples for How to do things

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

Changing the Behavior – Business Objects

- If you want to “**PRE**”vent something from happening:
 - Setting a value, do a `oScript.SetError(<reason> As String)` in the `PreValidateColumn()` event
 - Saving a record based on a condition, do a `oScript.SetError(<reason> As String)` in the `PreWrite()` event
 - Deleting a record based on a condition, do a `oScript.SetError(<reason> As String)` in the `PreWrite()` event
 - Exception is the `PreTotals()` cannot prevent it from happening, just a place to perform some action – such as adding a new line prior to calculating totals
- **TIP** – If you want to prevent setting a column that is validated against a Service Object, you must first obtain the object handle and invoke the `ReadAdditional(<dataSourceName> As String)` before doing a `GetValue` to test the value of a column

Changing the Behavior – Business Objects

- Post Events are used to do additional processing based on the successful:
 - `PostValidateColumn()` - Setting a value of a column using `SetValue()`
 - `PostWrite()` - Writing a row
 - `PostDelete()` - Deleting a row
 - `PostRead()` - Reading a row, via `SetKey()` or `MoveXxxx()`
- `ScriptInitialization()` – runs only once when script is first run. Can be used to initialize or store values that won't change during the life of the object (e.g. is user a member of a role)
- `SetDefaultValues()` – if factory defaults or Advanced Field Settings don't do the trick

Let's Take a Look – AR_Customer_bus

- [Cust_Pre_Val_Credit_Limit.txt](#) – Prevent a user from changing the Credit Limit field unless the user is a member of the “salesmgr” security role
- [Cust_Prevalidate_Termscode.txt](#) - Don't allow selection of a terms code that has a discount percentage.
- [Cust_Prewrite.txt](#) – Don't allow a customer to be saved if terms code is 00 and default payment type is not CASH

Advanced Tip

- Since scripts can only be run within a business object in most cases you cannot attach script that runs during an update process.
- Some updates actually use the business objects to delete the rows from the table and any related tables
- Therefore you can attach script to the PostDelete() ONLY (no PreDelete()) of an object and condition the logic based on oSession.Updating which will let you know the delete is happening during the Posting/Update process and not data entry
- To see if this is available for your case, place oScript.DebugPrint() in the PostDelete and check the debug trace window during the posting process.

Service Object Advanced

- [So_Dtl_Item_Code_Post.vbs](#) – GetResultSets() example from the 4.40 UDSDemo.M4X

Let's Take a Look – Edit and Delete Line (SO_SalesOrder_bus)

- GetEditKey(<lineKey> As String) As String – returns string of edit key to be used in EditLine() and DeleteLine() methods
- [EditLine_SO_SalesOrder_bus.txt](#)
- [DeleteLine_SO_SalesOrder_bus.txt](#)

Working with Batches – Business Objects

- Properties
 - BatchEnabled
 - e.g. oBusObj.BatchEnabled
 - 1=yes, 0=no
- Methods
 - SelectBatch(batch as string)
 - selects an existing batch
 - SelectNewBatch(batch as string, private as string, comment as string)
 - creates a new batch, pass in batch number to create, Y or N for private, and batch comment

Working with Batches – Forms Printing Objects

- Properties
 - BatchEnabled
 - e.g. oBusObj.BatchEnabled
 - 1=yes, 0=no
- Methods
 - SelectBatch(batch as string)
 - selects an existing batch
 - can be called multiple times to print documents across multiple batches
 - Returns 1 for success or 0 if failure. Failure is likely the batch is in use in data entry or is being updated.

Working with Batches – Update Objects

- Properties
 - BatchEnabled
 - e.g. oBusObj.BatchEnabled
 - 1=yes, 0=no
- Methods
 - SelectBatch(batch as string)
 - selects an existing batch
 - can be called multiple times to print register and update multiple batches
 - Returns 1 for success or 0 if failure. Failure is likely the batch is in use in data entry or is being used in from printing.

Let's Take a Look – AR Invoice – Entry, Form, Update

- [Batch_AR_Invoice_bus.txt](#)
- [Batch_AR_InvoicePrinting_rpt.txt](#)
- [Batch_AR_InvoicePrinting_upd.txt](#)
 - GetAdditionalReports(<returnString> As String) – provides a list of additional audit trails available (Gross Profit, etc.) Also includes ALL to print them all
 - Use one of the return values in the ProcessAdditionalReports(<reportToPrint> As String)

AR_CashReceipts_bus

- Unique in that it has a DepositObject property which is an object handle for the deposit that the cash receipts record belongs to.
- DepositObject has the GetNextDepositNo(<depositNo> As String). This returns the next deposit number into the string argument
- Use this in the oARCashReceipt.SetValue("DepositNo\$", depNo) method to assign the cash receipt deposit number
- Can also support Batches just like previous example

Let's Take a Look – AR_CashReceipt_bus

- [Batch_AR_CashReceipts_bus.TXT](#)

Advanced Reports – AR_AgedInvoiceReport_rpt

- There is a business object associated with each report object for the SY_ReportSetting table.
- Use SelectReportSetting(<reportSetting> As String) to establish the report setting to use
- SetKeyValue() calls can be made for ModuleCode\$, ReportID\$, ReportSetting\$ and RowKey\$ followed by a SetKey(). If used on STANDARD report setting this will create a new row.
- SetValue() calls can then be made for SelectField\$, SelectFieldValue\$, Tag\$, Operand\$, Value1\$ and Value2\$
- See example code

Let's Take a Look – Advanced AR_AgedInvoiceReport_rpt

- [ReportAdvanced_AR_AgedInvoice_rpt.txt](#)
- Example shows how to set selection criteria at run-time using the BOI.

Working with Lot/Serial Tiers

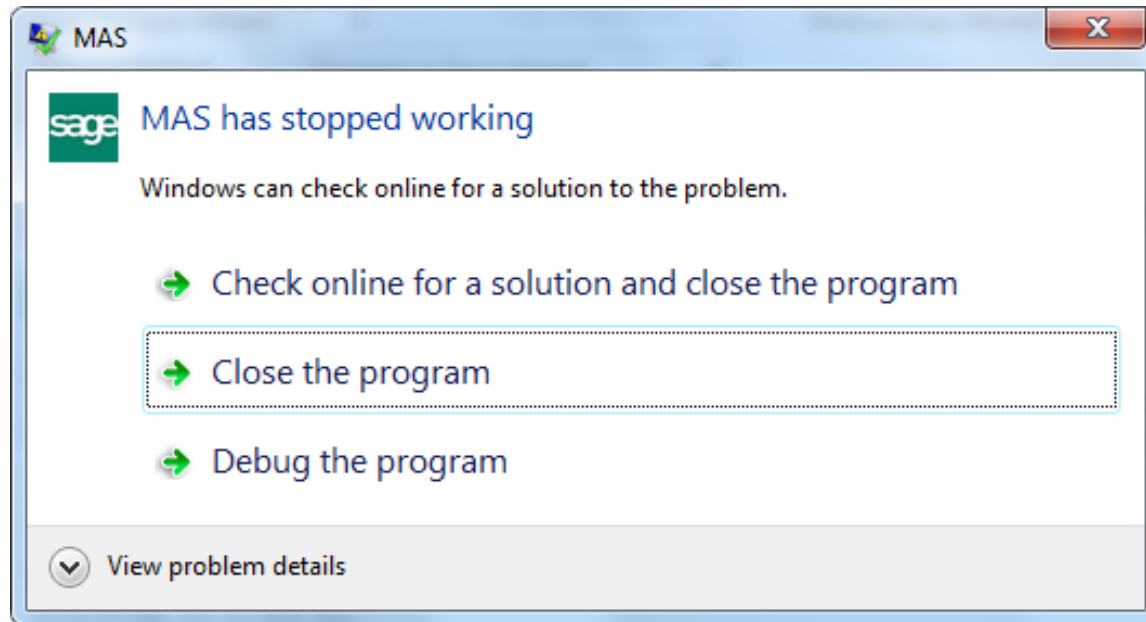
- Regardless of object (PO_Receipt_bus, SO_Invoice_bus, IM_Transaction_bus, etc.), there is a Distribution property which is an object handle to the Tier Distribution object for the detail line. Will only have a value if the detail line is a lot or serialized item code.
- oDist.AddDistributionLine(<lotOrSn> As String) will establish a new line for the provided lot or serial number passed in as an argument
- oDist.GetSerialRecords(<startingSN> As String) will auto increment based on the starting serial number passed in as the argument for the entire remaining quantity.

Let's Take a Look – Advanced IM_Transaction_bus

- [LotSerialAdv_IM_Transaction_bus.txt](#)
- Example shows how to create an Inventory Receipt for Lot and Serial Numbers. Including multiple lots for a detail line, and auto increment on a serialized item
- Example also prints the register and updates the receipt.

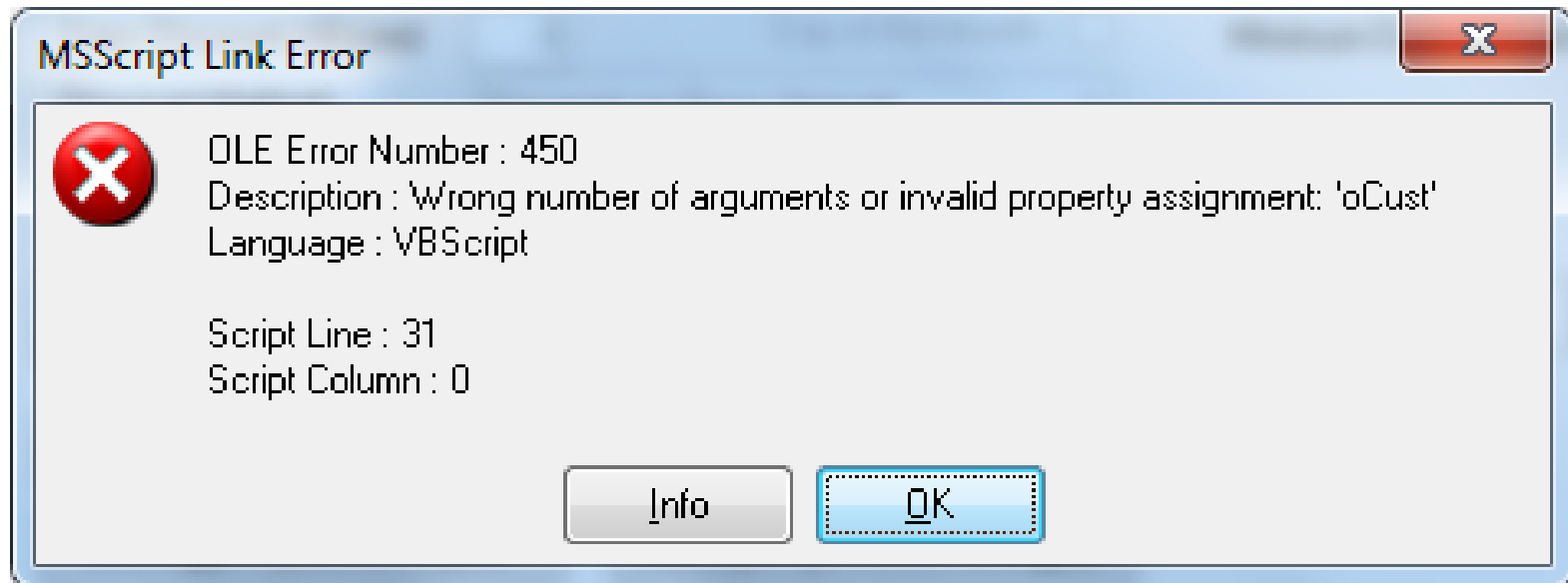
What Can Go Wrong?

- 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



What Can Go Wrong

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



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)



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)
- ***Your mission*** is to get the Murphy's Law Quick Print button to work – There are several “planted” errors in the script

How to Find Examples

- Put all of the provided sample code in a single directory and use windows search for the keyword you're interested in
 - Want to know how to add lines to an order? Search the directory for files that contain “LinesAdded” or “AddLine()”
 - Want to know how or when to de-activate a procedure? Search for “oScript.Deactivate”
 - Want to know how to do a loop? Search for “While”. Or “Do Until”
 - Want to know how to deal with Lot/Serial? Search for “.Distribution”

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?

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:
 - 10:00 (P-ERP23)
 - 11:30 (P-ERP23B)



Contact Us

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