

Interfacing Microsoft Access® and SAS® Using OLE and ODBC

Kayhan Najmabadi, STATPROBE, Inc., Ann Arbor, Michigan

Carl R. Haske, STATPROBE, Inc., Ann Arbor, Michigan

Raymond Xu, STATPROBE, Inc., Ann Arbor, Michigan

ABSTRACT

As an OLE Automation controller, Microsoft Access can manipulate any object in another application's type library. SAS 6.12 can function as an OLE Automation server, and we have built an application that uses SAS in this way. This third and final paper of the present group brings out this brand-new possibility offered by SAS 6.12 and shows what advantage can be taken of new horizons in the enhanced SAS System.

VISUAL BASIC DISCUSSION

As was mentioned in the abstract, the key coding part in our application is the use of OLE Automation. An example of the Visual Basic code segment that is responsible for OLE Automation and uses SAS as an OLE Automation server is shown below:

```
Dim OleSAS As Object
Set OleSAS = createObject("SAS.Application")
OleSAS.submit("%include _
'h:\users\knajmaba\sasdev\decod.sas';run;")
```

To create an instance of the SAS System (that is, invoke a SAS session), we create an OLE object using the SAS System program identifier as it is listed in the Windows registry. The SAS program identifier is SAS.Application. The identifier OleSAS is set to the new SAS session.

In the second line of code we are using the submit method of the Object "OleSAS" to run a SAS program that will decode the chosen data set.

EXAMPLE

We have opted for MS Access as our main platform for launching and controlling the data transferring operation. For this purpose, we created a user interface. This user interface, based on Access Forms, is organized like a wizard tool and guides the user throughout the process. To realize our goal, we have applied OLE automation and ODBC together.

In the first screen, the user is invited to choose the folder that contains SAS data sets.

1. Use the browse button to select the folder which contains all data sets you want to export to Microsoft Access, and choose one of the data sets. Browse

2. System will read all the data sets names in the following folder

3. System names the decoded data set, imported Access table, verification data set as following.

DecodedDataSetName	ImportedTableName	ExportedDataSetName
INJECT	INJECT	INJECTV
AE	AE	AEV
BRONCH	BRONCH	BRONCHV
COMPTON	COMPTON	COMPTONV

Out Next

Figure 1. Read Data Sets form

The Visual Basic code that is attached to the On Click Event of the command button "Read SAS Datasets' Name" is as follows:

```
Private Sub cmdReadDatasetName_Click()
On Error GoTo Err_cmdReadData_setName_Click

    If IsNull(Me![txtReadPath]) Then
        MsgBox "Locate the data sets folder first", 16
        Exit Sub
    End If

    Dim rstDSNames As Recordset
    Dim strDatasetName As String

    ' Use OLE dialogue Control to select one
    ' SAS dataset name at Ordup folder
    ' Global string 'strSASOriDupPath' returns
    ' the folder with '\*.sd2'

    ' Clean table tblDatasetNames records
    CurrentDb.Execute "Delete * From _
tblDatasetNamesDM"

    Set rstDSNames = _
CurrentDb.OpenRecordset("tblDatasetNamesDM",
dbOpenTable)

    strDatasetName = Dir(strSASOriDupPath)
    ' Read the first dataset path
    rstDSNames.AddNew
    ' Update tblDatasetNames table 1st record
    rstDSNames.Fields(1).Value = _
Left$(strDatasetName, _
(InStr(strDatasetName, ".") - 1))
    rstDSNames.Fields(0).Value = _
Left$(strDatasetName, _
(InStr(strDatasetName, ".") - 1))
    rstDSNames.Fields(2).Value = _
ReturnShortStr(strDatasetName) & "v"
    rstDSNames.UPDATE

    Do
        strDatasetName = Dir
        ' Read the following dataset path
        If strDatasetName = "" Then
            ' No dataset at all, Null is returned
            Me![SubfrmDatasetNames].Requery
            ' Display tblDatasetNames table
            Me![SubfrmDatasetNames].Visible = True
            GoTo Exit_cmdReadDatasetName_Click
        End If
        rstDSNames.AddNew
        ' Update following records
        rstDSNames.Fields(1).Value = _
Left$(strDatasetName, _
(InStr(strDatasetName, ".") - 1))
```

```

        rstDSNames.Fields(0).Value = _
        Left$(strDatasetName, _
            (InStr(strDatasetName, ".") - 1))
        rstDSNames.Fields(2).Value = _
        ReturnShortStr(strDatasetName) & "V"
        rstDSNames.UPDATE
    Loop

Exit_cmdReadDatasetName_Click:
    rstDSNames.Close
    Exit Sub

Err_cmdReadDatasetName_Click:
    Select Case Err.Number
        Case 3021
            ' No record at tblDatasetNames table, or
            MoveFirst, MoveNext error
            Resume Next ' Pass the command
        Case Else
            Error.Description
            MsgBox Err.Description
            Resume
    End Select
Exit_cmdReadDatasetName_Click:
    End Select
End Sub

```

Listing 1. Selecting the data set

The second screen lets the user select the data sets needed for the transfer.

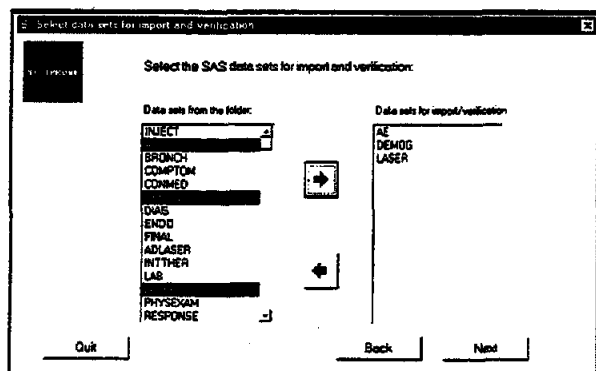


Figure 2. Select Data Sets form

A custom-made Visual Basic code is acting behind the mover command:

```

Private Sub cmdSelect_Click()
On Error GoTo Err_cmdSelect_Click
    Dim rstSelectedName As Recordset
    Dim varDatasetName As Variant

    CurrentDb.Execute "Delete * From _
tblDatasetNames"
    Set rstSelectedName =
CurrentDb.OpenRecordset("tblDatasetNames" _
, dbOpenDynaset)

    For Each varDatasetName In _
Me![lstOriDataset].ItemsSelected()
        rstSelectedName.AddNew
    Next varDatasetName

```

```

        rstSelectedName.Fields(0).Value = _
Me![lstOriDataset].ItemData(varDatasetName)
        rstSelectedName.Fields(1).Value = _
Me![lstOriDataset].Column(1, varDatasetName)
        rstSelectedName.Fields(2).Value = _
Me![lstOriDataset].Column(2, varDatasetName)
        rstSelectedName.UPDATE
    Next varDatasetName
    rstSelectedName.Close
    Me![lstSelectedDataset].Requery

Exit_cmdSelect_Click:
    Exit Sub

```

Listing 2. Mover functionality

The next screen deals with ODBC issues. SAS ODBC Driver is called, and the user will enter/check the ODBC specification. This portion of the process is not covered in this paper; for a comprehensive discussion of this topic, see Riba and Riba (1996).

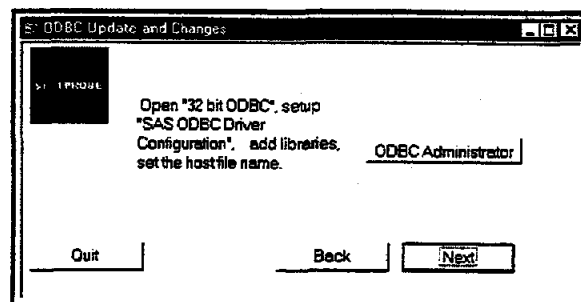


Figure 3. ODBC Setup form

In the fourth screen, the user will launch the decoding process in a SAS environment. The OLE technique is used here.

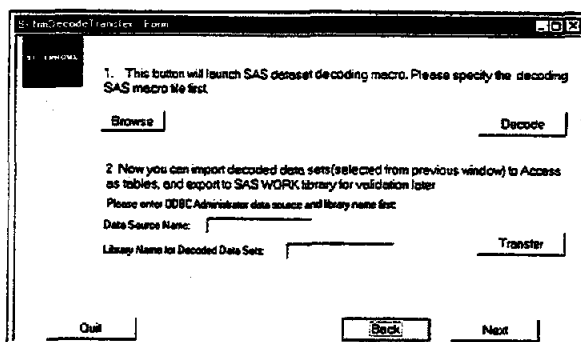


Figure 4. Decode and Transfer form

With the next step, the Visual Basic code invokes ODBC to import the selected data sets to MS Access and immediately afterward exports a copy of the same data set to the SAS system.

The code that is attached to the Decode command button constitutes the heart of our discussion. With a few lines of Visual Basic code, we establish the connection to the SAS application, and we control it from our Access platform.

```

Private Sub cmdDecode_Click()
    Dim OleSAS As Object
    Set OleSAS = CREATE("SAS.Application")

    DoCmd.Echo False, _
        "Decoding the data sets"

    OleSAS.Submit _
        ("%include'h:\data\sas\access\" & _
        & Me![txtSASDecodeFile] & ";run;")

    DoCmd.Echo True

End Sub

```

Listing 3. OLE Automation

Note that the program above runs a SAS macro file, which is defined by the user, to decode the original data set at h:\data\access\q1\photofrip***\sas\oridup.

A partial view of the Visual Basic coding responsible for the transfer is given below.

```

Private Sub cmdTransfer_Click()
    On Error GoTo cmdTransfer_Click_Err

    If IsNull(Me![txtDataSourceName]) Or _
        IsNull(Me![txtLibrary]) Then
        MsgBox "Please fill in data source and _
        library name"
        Exit Sub
    End If

    Dim rstTableName As Recordset
    Dim intTableNum As Integer, _
        intTableLoop As Integer
    Dim strActiveDecoded As String, _
        strActiveImported As String, _
        strActiveExported As String

    '--Open the "tblDataSetName" table at
    Access
    '--system, set up recordset

    Set rstTableName = _
        CurrentDb().OpenRecordset _
        ("tblDataSetNames", dbOpenTable)
    intTableNum = rstTableName.RecordCount

    '-- Loop through the recordset to
    execute import and export

    rstTableName.MoveFirst

    For intTableLoop = 0 To intTableNum - 1
        strActiveDecoded = _
            rstTableName.Fields(0).Value
        strActiveImported = _
            rstTableName.Fields(1).Value
        strActiveExported = _
            rstTableName.Fields(2).Value

        '--import from SAS dataset according to
        ODBC data source and library
        DoCmd.Echo False, "Importingtable" & _
            strActiveImported
    
```

```

DoCmd.TransferDatabase acImport, _
    "ODBC Database", ("ODBC;DSN=" & _
    Me![txtDataSourceName] & ";TABLE=" & _
    Me![txtLibrary]), acTable, _
    strActiveDecoded, strActiveImported, _
    False

DoCmd.Echo True

    '--export to SAS WORK library

DoCmd.Echo False, "Exporting table " & _
    strActiveImported
DoCmd.TransferDatabase acExport, _
    "ODBC Database", ("ODBC;DSN=" & _
    Me![txtDataSourceName] & ";TABLE=" & _
    Me![txtLibrary]), acTable, _
    strActiveImported, strActiveExported, False

DoCmd.Echo True

    rstTableName.MoveNext
    strActiveDecoded = ""
    strActiveImported = ""
    strActiveExported = ""

Next intTableLoop

rstTableName.Close

cmdTransfer_Click_Exit:
    Exit Sub

cmdTransfer_Click_Err:
    Select Case Err.Number
        Case 1648 ' ODBC failed to open
            Resume
        Case 1157
            Resume
        Case Else
            Resume
        ' MsgBox Err.Description
        ' Resume cmdTransfer_Click_Exit
    End Select

End Sub

```

Listing 4. OLE Automation and ODBC calls

Since this operation stores the data in the WORK directory, the user must continue the validation process by stopping the server and calling the validation macro.

In the appendix to this paper, we reproduce the SAS report that results after the validation program has been run.

A sample of the SAS data set transferred to the Access table is shown on the next page.

P	Y	PT	PTMTCOD	VISIT	DOB	EDOS	SEX	HT	W	GEN	CH
P006	BCCA	14	SC	1	BASLINE	18APR22	MALE	157	44.5		
P006	BCCA	15	SH	1	BASLINE	12APR92	MALE	170	85.4		
P006	BCCA	16	FE	1	BASLINE	27DEC92	MALE	175	83.2		
P006	BCCA	17	LK	1	BASLINE	16NOV92	MALE		89		
P006	BCCA	18	JH	1	BASLINE	30JUL28	MALE				
P006	BCCA	19	KT	1	BASLINE	02AUG21	MALE	165	89		
P006	BCCA	20	GS	1	BASLINE	10DEC36	MALE		80		
P006	BCCA	21	NE	1	BASLINE	14JAN03	MALE		82		
P006	BCCA	22	LH	1	BASLINE	12FEB28	MALE				
P006	BCCA	23	JT	1	BASLINE	27AUG90	MALE				
P006	BCCA	24	RD	1	BASLINE	21APR91	MALE				
P006	BCCA	25	GM	1	BASLINE	30AUG21	MALE		92		
P006	BCCA	26	EW	1	BASLINE	05OCT18	FEMALE				
P006	BCCA	27	HH	1	BASLINE	12JUL31	MALE				
P006	BCCA	28	CK	1	BASLINE	18MAR28	FEMALE				
P006	BCCA	29	ML	1	BASLINE	17JUL13	MALE	160	81		

Figure 4. Partial view of the Access table

CONCLUSION

This application has used new techniques included in SAS 6.12. It makes extensive use of OLE and ODBC. Although SAS 6.12 comes with an elaborate Import/Export utility, the system setup by our team establishes a direct connection between SAS and MS Access and therefore is easier to use. By connecting SAS data with MS Access, one can take advantage of the flexibility and simplicity of MS Access in creating new queries, forms, and reports. SAS 6.12 allows us to put SAS in the server side of client/server applications. We can develop extremely user-friendly user interfaces that use the sheer power of SAS for summarizing, manipulating, and analysis.

REFERENCES

- Clegg, Jennifer, and Rigsbee, Carol (1997). "OLE and the SAS System for Windows Release 6.12," *Proceedings of the Twenty-Second Annual SAS® Users Group International Conference*, 257, Cary, NC: SAS Institute Inc.
- Riba, S. David, and Riba, Elizabeth A. (1996). "ODBC: Windows to the Outside World," *Proceedings of the Twenty-First Annual SAS® Users Group International Conference*, 548-557, Cary, NC: SAS Institute Inc.
- Mehler, Gary (1997). "Using the SAS System Release 6.12 on Windows," *Proceedings of the Twenty-Second Annual SAS® Users Group International Conference*, 294, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The authors extend their thanks to Alexandra Mancini of QLT PhotoTherapeutics Inc. for her generous authorization to reproduce a portion of data processed by STATPROBE, Inc. The authors also thank Paul Schwankl for his assistance in the preparation of this paper.

SAS and SAS/AF are registered trademarks of SAS Institute Inc. MS Access and Windows 95 are trademarks of Microsoft in the USA and other countries. * indicates USA registration.

AUTHORS' ADDRESSES

Kayhan Najmabadi, Ph.D.
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000 x161
E-mail: knajmabadi@statprobe.com

Carl R. Haske, Ph.D.
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000 x115
E-mail: chaske@statprobe.com

Raymond Xu, M.S.
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000 x641
E-mail: rxu@statprobe.com

APPENDIX

Part of SAS Report Generated by the Validation Macro

The SAS System

15:36 Friday, January 31, 1997 1

COMPARE Procedure

Comparison of DECODED.ADLASER with TOBECOMP.ADLASER

(Method=EXACT)

Data Set Summary

Dataset	Created	Modified	NVar	NObs	Label
DECODED.ADLASER	31JAN97:15:07:12	31JAN97:15:07:12	24	29	*
TOBECOMP.ADLASER	31JAN97:15:24:27	31JAN97:15:24:29	24	29	

Variables Summary

Number of Variables in Common: 24.

Observation Summary

Observation	Base	Compare
First Obs	1	1
Last Obs	29	29

Number of Observations in Common: 29.

Total Number of Observations Read from DECODED.ADLASER: 29.

Total Number of Observations Read from TOBECOMP.ADLASER: 29.

Number of Observations with Some Compared Variables Unequal: 0.

Number of Observations with All Compared Variables Equal: 29.

NOTE: No unequal values were found. All values compared are exactly equal.