

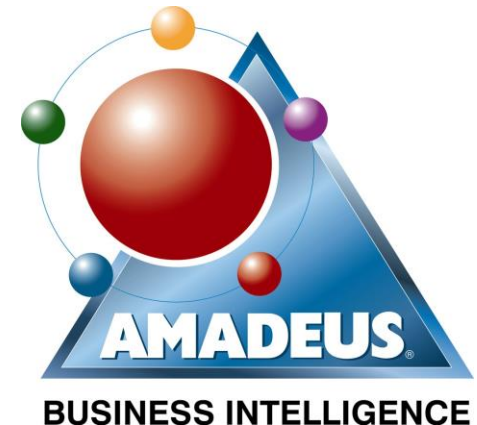


Building Windows Front Ends to SAS Software

Katie Essam

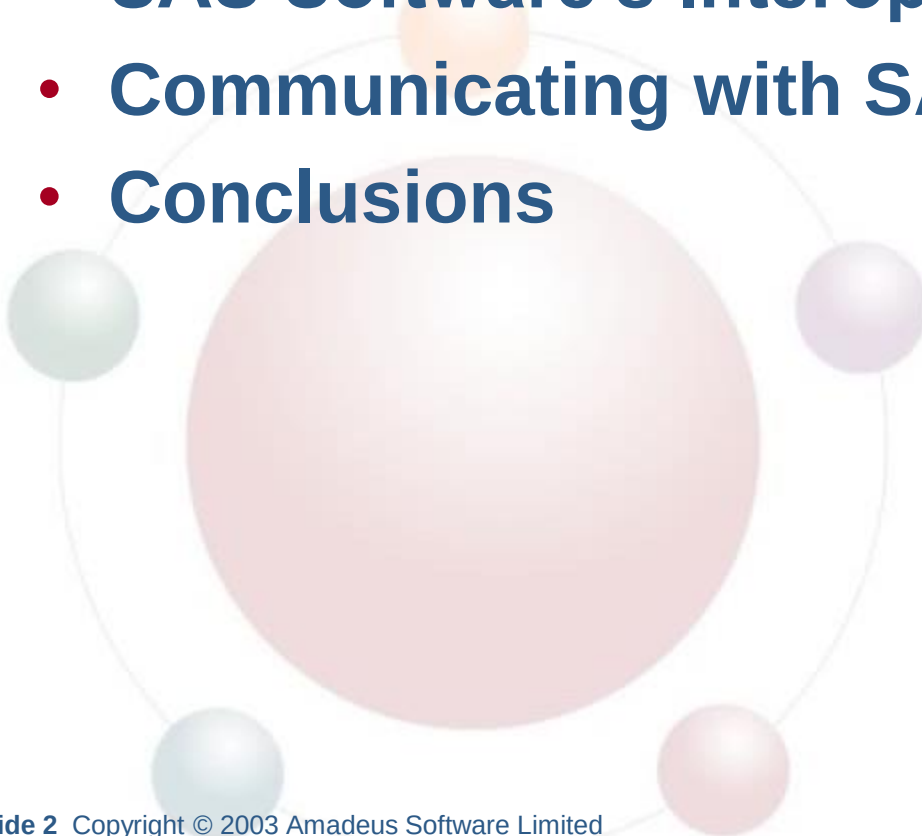
Amadeus Software Limited

20th May 2003



Topics

- Introduction
- What is .NET?
- SAS Software's Interoperability
- Communicating with SAS from VB.NET
- Conclusions



Introduction

- **Prior to Version 8 of SAS**
 - **SAS/AF® Software**
 - Rich Client Interfaces
 - **SAS/IntrNet® Software**
 - Browser based Interfaces
- **Since Version 8**
 - **SAS® Integration Technologies**
 - Enables communication with many Industry Standard languages including Java, C#, C++ and Visual Basic
- **This paper aims to introduce some of the techniques that can be used to access SAS Software from Visual Basic.NET**



What is Microsoft's .NET Framework?

- **New view of the Windows operating system**
- **Hundreds of classes that provide the functionality of Windows**
- **Provides single approach to building**
 - **Rich client applications**
 - **Web client applications**
- **Incorporates CLR (Common Language Runtime) so that all .NET programming languages are equal**

What is Visual Basic.NET?

- **VB.NET is the latest version of the Visual Basic Language**
 - Fully Object Orientated Programming
 - Directly comparable to other Visual Studio Languages
- **From the developer's point of view**
 - Provides Rapid Application Development(RAD) tool
 - Highly reusable code
 - Large pool of resources
- **From the user's point of view**
 - Familiar look and feel
 - Windows applications with rich functionality

What is Microsoft's .NET Framework?

Visual Basic

C#

C++

JScript

J#

ASP.NET

Windows Forms

Common Language Specification (CLS)

Data & XML

Base Class Library

.NET Common Language Runtime (CLR)

Windows API

COM+ Services



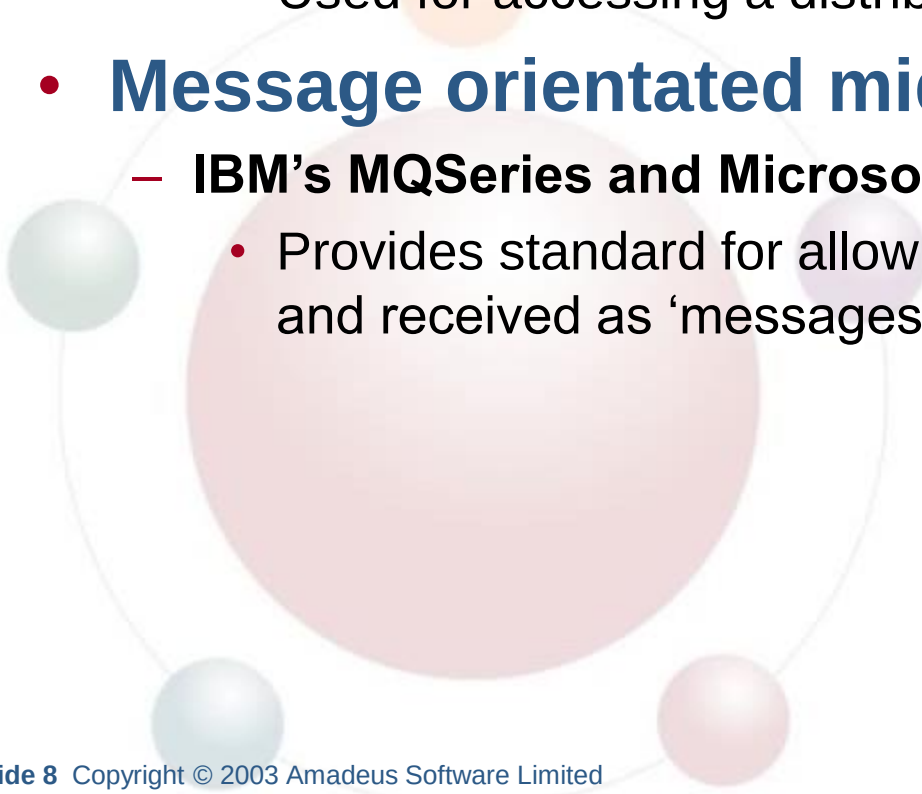
SAS Software's Interoperability

- **Prior to Version 8**
 - **SAS/AF Software**
 - Proprietary software to build Rich Client Interfaces using SCL
 - Version 8 introduced dot notation
- **SAS Integration Technologies**
 - **Foundation to enable seamless integration between SAS and industry standard tools**
 - **Provides middleware to allow programmers working in a number of different languages to quickly create applications that harness the functionality of SAS**



SAS Software's Interoperability

- **Industry Standards supported**
- **Directory Technology**
 - **LDAP (Lightweight Directory Access Protocol)**
 - Used for accessing a distributed store of information
- **Message orientated middleware**
 - **IBM's MQSeries and Microsoft's MSMQ**
 - Provides standard for allowing requests and response to be sent and received as 'messages'



SAS Software's Interoperability

- **Distributed object models**

- **Microsoft's COM/DCOM**

- The Component Object Model (COM) allows objects in different object spaces to 'talk' to each other by calling each other's methods. DCOM (Distributed COM) allows components to communicate with each other over networks

- **CORBA (Common Object Request Broker Architecture)**

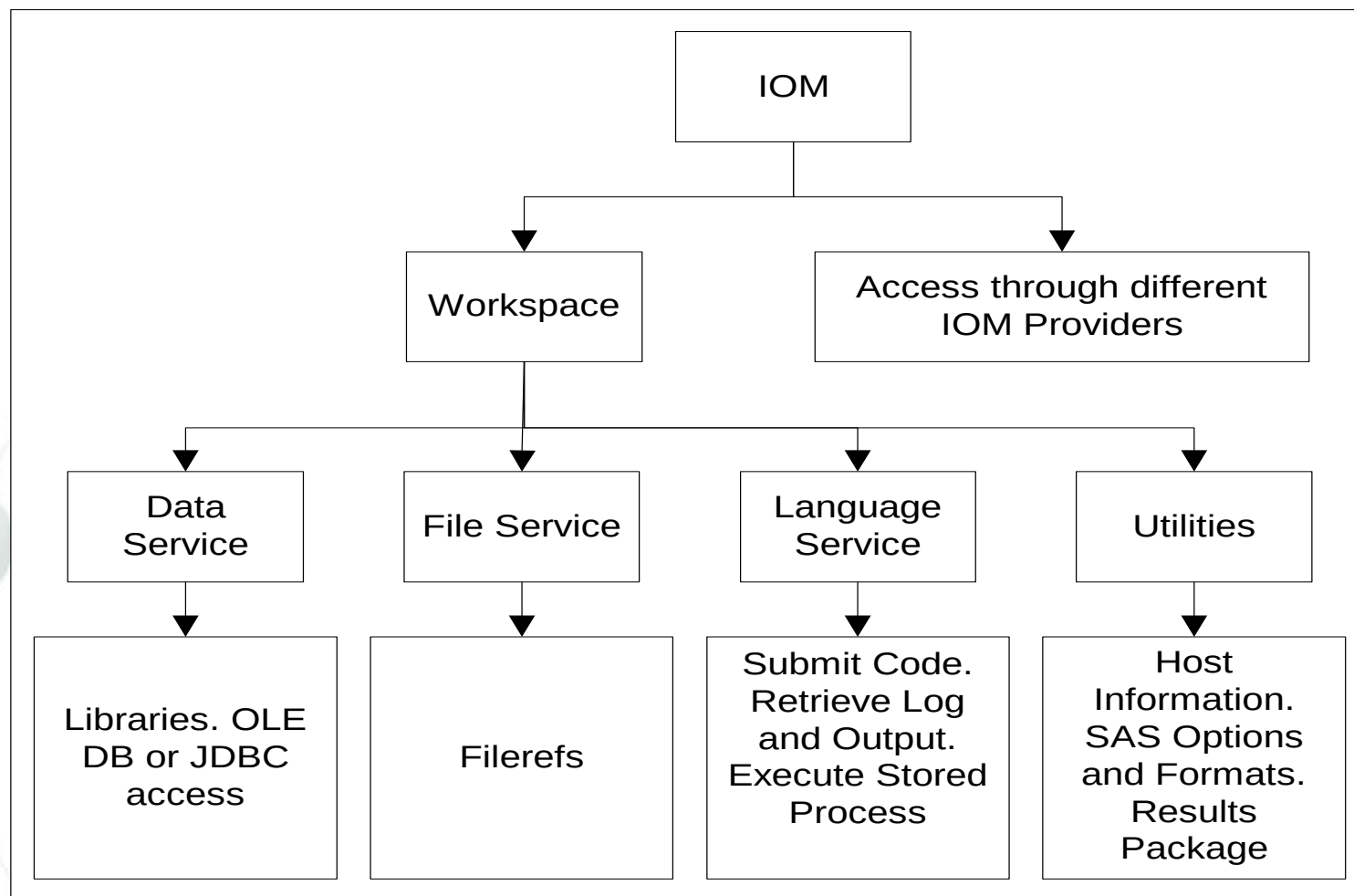
- CORBA is a vendor independent infrastructure to provide communication between applications over networks



Integrated Object Model

- **IOM is an object model in SAS Integration Technologies**
 - Provides a set of objects that interface to Base SAS Software
 - Client applications can connect to an IOM server through a variety of industry standard communication protocols
- **IOM objects mainly handle two tasks**
 - Submit SAS Language programs to SAS
 - Retrieve information and data from SAS

Integrated Object Model



The full object hierarchy can be found at <http://support.sas.com/rnd/itech/doc/dist-obj/iom.html>

Integrated Object Model - The Workspace

- **The root object of the IOM Hierarchy is the Workspace**
 - Can be created via the Workspace Manager
 - Can be thought of as a SAS Session
- **The Workspace Manager is not necessarily required but makes scaling the application to use multiple Workspaces easy**

IOM Data Providers

- **Three SAS Data Providers that allow direct connections to SAS Data Sources**
- **SAS Local Data Provider**
 - Allows manipulation of data sets in your windows environment
 - Does not support SQL command processing
 - Single User update
- **SAS/SHARE Data Provider**
 - Allows manipulation of data sets and views in a SAS/SHARE server.
 - Multi User update
 - Supports SQL processing

IOM Data Providers

- **SAS IOM Data Provider**
 - Allows manipulation of data sets and views in an Integration Technologies object server
 - Supports SQL processing
 - Multi user support
 - Can be used on any platform that the IOM supports
- **All of the providers:**
 - Have basic OLE DB schema rowsets, which enable consumers to obtain metadata about the data source they are using.
 - Have integrated SAS Formatting services (NOTE the Local provider cannot use user defined formats)
- **The examples in this paper use the Local Provider**



Getting SAS to communicate with VB.NET

- .NET can communicate with SAS by making use of any of COM (or DCOM), OLE/DB, ADO.NET or ODBC
- COM/DCOM are used indirectly when using the SAS IOM
- The other Industry Standard can be used to access SAS data directly

Running SAS Programs

- **There are a number of ways of running SAS programs via the IOM**
 - Stored Processes
 - Simple file includes
 - Submitting code created within an application
- **First, need to setup Visual Basic.NET**
 - Install the SAS Integration Technologies Client Side components, executable downloadable from SAS Institute's Web Site
 - Add references to the components to the VB.NET project
 - SAS Integrated Object Model (IOM) (SAS System Version 8.2) Type Library – Type Lib Version 1.1
 - SASWorkspaceManager Type Library 1.1

NOTE: This paper assumes Version 8.2 is being used



Running SAS Programs – Starting SAS

- **Creating a SAS Session**

- Optionally, use the Workspace Manager to manage SAS Workspaces
- Create an instance of the SAS Workspace

```
Public obWsMgr As New SASWorkspaceManager.WorkspaceManager()  
Public obWS As SAS.Workspace  
Public Function StartSAS()  
    Dim errString As String  
    obws = obWsMgr.Workspaces.CreateWorkspaceByServer( _  
        "My workspace", _  
        SASWorkspaceManager.Visibility.VisibilityNone, _  
        Nothing, "", "", errString)  
End Function
```

Running SAS Programs – Stopping SAS

- **To stop a SAS Workspace**
 - Close the Workspace which ends the session
 - Release the object for garbage collection in .NET

```
Public Function EndSAS()  
    obws.Close()  
    obws = Nothing  
End Function
```

In Visual Basic objects persist, or stay in existence, until they are removed. By setting the Workspace to Nothing we remove it from use

Running SAS Programs – %Include

- **To run a SAS Program**
 - Start SAS Session (Workspace)
 - Create SAS code and Submit it
 - End SAS Session (Workspace)
- **Using %Include is a simple way to run a SAS Program**
 - Assumes have a button called Submit and a text box called FileName
 - A String is created which contains the %include statement and the SAS Program name

```
Private Sub Submit_Click(ByVal sender As System.Object, _  
                        ByVal e As System.EventArgs) Handles _  
                        Submit.Click
```

```
    Dim SASCode As String
```

```
    SASCode = "%include '" & fileName.Text & "';"
```

```
    StartSAS()
```

```
    SubmitCode(SASCode)
```

```
    EndSAS()
```

```
End Sub
```

Running SAS Programs – %Include

- **To submit the code**
 - an instance of the SAS Language Service is created.
 - the SAS code String is passed to the SAS Workspace for processing, using the Submit method.
 - Once the obLS object is no longer required, release it by setting it to Nothing

```
Private Function SubmitCode(ByVal SASCode As String) As _  
                                Boolean  
    Dim obLS As SAS.LanguageService = obws.LanguageService  
    obLS.Submit(SASCode)  
    obLS = Nothing  
End Function
```

- **SASCode is a String so we could extend this to provide the User's with an application containing an Editor, from which we could capture the text, for them to write and submit their code**



Running SAS Programs – Stored Process

- A Stored Process in SAS terms is a .sas file with a comment in the header of
***ProcessBody;**
- Business Logic is separate from Application Logic
- Can be thought of as SAS Macro definitions
- Macro variables should be defined as **Variable=Value**

Running SAS Programs – Stored Process

- **To use Stored Process need to**
 - specify the directory (or Repository), program name and any macro variables (not included below)
 - Create the LanguageService and the StoredProcessService
 - Use the StoredProcessService to execute the program
 - Release the LanguageService and StoredProcessService objects for garbage collection

```
Private Function SubmitCode(ByVal dir As String, _  
                            ByVal program As String) As Boolean  
    Dim obLS As SAS.LanguageService = obws.LanguageService  
    Dim obSP As SAS.StoredProcessService =  
        obLS.StoredProcessService  
    obSP.Repository = "file:" & dir  
    obSP.Execute(program, "")  
    obLS = Nothing  
    obSP = Nothing  
End Function
```

Running SAS Programs – Stored Process

- If macro variables are required then they should be passed to the stored process using

```
cmd = "title='This is the title'"  
obSP.Execute(program, cmd)
```

- Any name/value pairs passed in this way replace the *ProcessBody; comment and are treated by the SAS Code as normal macro variables

Running SAS Programs – Events

- The LanguageService also provides functionality to dynamically capture events as they happen in SAS Code that is executing
- Need to declare LanguageService as

```
Dim WithEvents obLS As SAS.LanguageService
```

- Can then access events, for example, can put messages on a StatusBar

```
Private Sub obLS_DatastepComplete() Handles _  
                                     obLS.DatastepComplete  
    StatusBar.Panels(0).Text = "DATA Step has finished"  
End Sub
```

Accessing Data

- **VB.NET uses ADO.NET as it's primary data access and manipulation tool**
- **ADO.NET makes use of OleDb Interfaces to provide standard access methods to a variety of sources, including**
 - **Microsoft Access**
 - **Microsoft SQL Server**
 - **SAS**
- **Connection method differs slightly for SAS but provides the ability to retrieve information from a SAS Data Set, including column names and descriptions, as well as the SAS Data Set itself**

Accessing Data

- **Two objects can be used to access SAS Data**
 - **DataAdapter**
 - Read-write, multi-directional
 - Can perform SQL queries (this is dependent on the IOM Provider being used)
 - **DataReader**
 - Sequential, read-only access
- **For the following examples an OleDbConnection must be created**

```
Public cn As New OleDb.OleDbConnection()
```

The OleDbConnection provides the interface that allows us to establish a connection via an IOM provider to a SAS Data Source.



Accessing Data – VB.NET DataTable

- The DataTable is a key component of ADO.NET
- Represents one table of relational in-memory data
- To create a DataTable

`Public schema As DataTable`

- Enables “disconnected” access
 - Don’t have continuous access to the data source
 - Access can be very fast as working on local copy

Accessing Data – OleDB Schema

- To access the header portion of a SAS Data Set we can use the **GetOleDbSchemaTable** method
 - Specify which Data Provider to connect to SAS with and the directory containing the Data Set we want to access
 - Open the connection
 - Use the connection to get the schema table for a specific data set

```
Public Function Make_Connection(ByVal SASLib As String, _  
                                ByVal SASDataSet As String) As Boolean  
    cn.ConnectionString = _  
        "Provider=sas.LocalProvider;Data Source=" & SASLibrary  
    cn.Open()  
    schema = cn.GetOleDbSchemaTable(OleDbSchemaGuid.Columns, _  
                                     New String() {"", "", SASDataSet})  
End Function
```

Accessing Data – OleDb Schema

- Once the VB.NET DataTable contains the header information from a SAS Data Set we loop round the DataTable retrieving the information of interest
 - In this example, we make a connection to a specific data set and then add the column names to a list box

```
Make_Connection(SASLib,SASDataSet)
Dim i As Integer
For i = 0 To schema.Rows.Count - 1
    ListBox.Items.Add(schema.Rows(i).Item("Column_Name"))
Next
```

Accessing Data – Using a DataAdapter

- We may want to display the whole data set
- Can use either **DataReader** or **DataAdapter** depending on access required
- A DataAdapter can be thought of as a bridge between an ADO.NET DataSet and the external Data Source
- An ADO.NET DataSet is a collection of ADO.NET DataTables

	age	gender	height	weight	status
▶	22	M	64	60	S
	26	F	87	74	M
	29	M	55	62	D
	24	M	84	74	M
	23	F	84	88	P
	30	M	87	76	S
	34	F	78	72	M
	38	F	69	56	M
	56	F	52	60	M
	52	F	66	69	M
	60	F	88.5	71	M
	56	F	61	61	M
	46	F	63	76	W
	65	F	87	81	M
	23	F	78	80	S
	28	M	67	82	

Accessing Data – Using a DataAdapter

- **To populate a DataGrid with a SAS Data Set**
 - Use connection to specify location of SAS Data Set
 - Use an OleDbCommand on the open connection to specify the Data Set name and that we want to access a table (CommandType of TableDirect)
 - Create a DataAdapter using the command specified
 - Use the DataAdapter to Fill a VB.NET DataSet with the contents of a SAS Data Set
 - Set the properties of the DataGrid so it's data source is the VB.NET DataSet and the table to display has the same name as the SAS Data Set



Accessing Data – Using a DataAdapter

```
Private Sub GetData()  
    cn.ConnectionString = _  
        "Provider=sas.LocalProvider;Data Source" & SASLib  
    cn.Open()  
  
    Dim cmd As OleDb.OleDbCommand = cn.CreateCommand()  
    cmd.CommandType = CommandType.TableDirect  
    cmd.CommandText = "demog"  
    Dim oleDA As OleDbDataAdapter = New OleDbDataAdapter(cmd)  
    Dim ds As DataSet = New DataSet()  
    oleDA.Fill(ds, "demog")  
    DataGrid.DataSource = ds  
    DataGrid.DataMember = "demog"  
  
End Sub
```

Conclusions

- **With Integration Technologies SAS have provided us with the mechanism to create User Interfaces either on the Web or Desktop in a variety of languages which can utilise the functionality of SAS as either middle tier or as a back end**
- **This paper has introduced some of the key functions that a developer may wish to incorporate into their applications**



Questions on Building Windows Front Ends to SAS Software?

Katie Essam

Senior Consultant

Amadeus Software Limited

Email: katie.essam@amadeus.co.uk

Web: www.amadeus.co.uk

