

Access to SAS[®] Data Using the Integrated Object Model (IOM) in Version 9.1

Frederick Pratter, Eastern Oregon University, La Grande OR

ABSTRACT

SAS[®] 9.1 Integration Technologies provides a complex set of APIs (Application Programming Interfaces) for access to SAS data from various Microsoft environments (COM, DCOM and .NET) as well as J2SE and J2EE for Java developers. This paper presents an overview of the Integrated Object Model available in SAS Integrated Technologies and suggests further resources for programmers who wish to integrate SAS with Windows clients using Microsoft Office and ASP, Visual Basic or C++ with the .NET framework.

OVERVIEW: SAS INTEGRATION TECHNOLOGIES

SAS[®] 9.1 Integration Technologies, initially released in Version 8, provides access to SAS data and procedures from a variety of clients, including Java, Microsoft Office, Visual Studio, ASP and the Web. Prior to Version 8, it was possible to access SAS from other applications in a limited manner with DDE or OLE automation. SAS Integration Technologies adds a set of powerful and complex tools allowing developers to interact with SAS using the *Integrated Object Model (IOM)* interface. IOM can be used to connect to SAS from “Open Clients”, that is non-proprietary solutions using standard programming languages such as Java, Visual Basic or C++. Of course IOM also supports access to SAS data library and remote computing services from local SAS sessions.

SAS defines the Integrated Object Model as a set of “distributed object interfaces to SAS... IOM enables you to use industry-standard languages, programming tools, and communication protocols to develop client programs that access these services on IOM servers.” (*SAS 9.1.2 Integration Technologies: Technical Overview*, 1). This paper is intended to introduce SAS IT (and the IOM in particular) to that segment of the community of SAS users for whom the very concept of a “distributed object interface” is unfamiliar and potentially scary.

The success of SAS over the last 30 years in no small part has been because very powerful results can be obtained with a relatively small investment in programming effort. IOM comes at the problem from the other side—if you already know how to program in, say, Visual C++, the IOM can allow you to access SAS data and run SAS procedures from within a C++ program. This is an extremely important addition to the overall functionality of SAS, but the range of choices available and the complexity is such that many users may be dissuaded from attempting to enter into this new realm. SAS has provided a number of “roadmaps” to IT (see the references at the end of this paper), but a roadmap is of use only if you know where you want to go. This paper, then, attempts to begin a little further back, by suggesting a number of interesting and rewarding destinations.

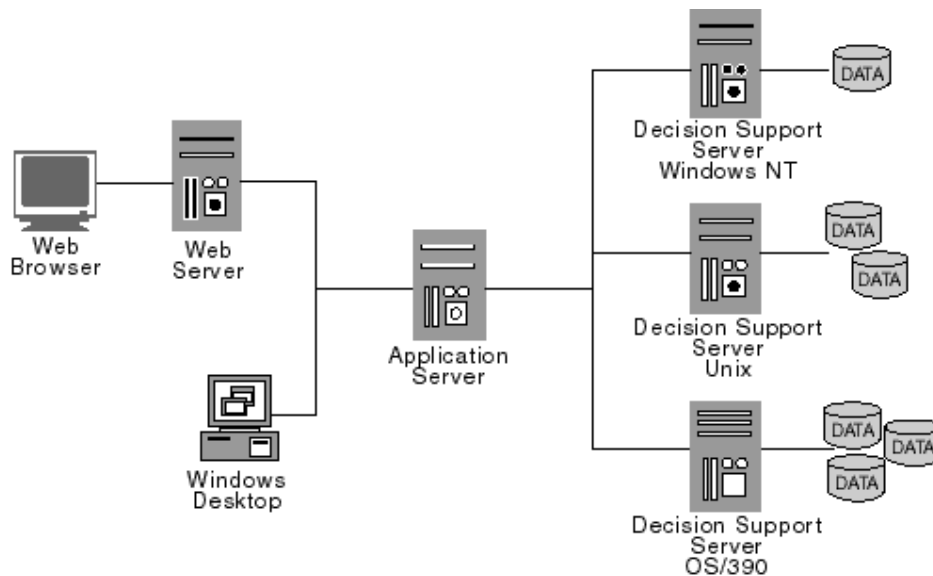
SAS IT includes a number of integration and development tools in addition to the IOM. Version 8 provided several kinds of *IOM servers* (see below), the *Publishing Framework* (PUSH capability), the *Application Messaging* interface, and the *Directory Service Interface* for access to LDAP services. Version 9.1 added a new *Management Console* along with production versions of the *SAS Foundation Services* (for Java programming), the *Stored Process* application interface, the *Web Infrastructure Kit* for building portals, and *BI Web Services*, which allow client applications to run SAS Stored Processes using the *XML for Analysis Version 1.1 Interface*. All of these are designed to make use of the new *SAS Open Metadata Interface*. (See “SAS Integration Technologies: What’s New?” <http://support.sas.com/rnd/itech/updates/new.html>.)

Obviously, in one short presentation it would be quite impossible even to begin to explain all these features. Consequently, this paper will focus as much as possible on what the IOM can do for application developers.

The Integrated Object Model

The first thing necessary to understand about the Integrated Object Model is that it is designed to work in a client-server environment, that is, one in which each computer on the network performs some specific function, such as managing data or Web page requests. In this model, the user interface, functional process logic and data access are each handled by separate modules, most often on separate computers. This separation between the presentation layer, business rules and implementation is central to modern reusable software design.

The diagram below, from the Version 8 SAS IOM documentation, illustrates one such scenario (it gets reproduced a lot):



"Integrated Object Model" *SAS Integration Technologies Release 8.2 Library*.
<<http://support.sas.com/rnd/tech/doc/dist-obj/iom.html>>

In the traditional model of SAS computing, programs to access and manipulate data were all run on the same computer as the data storage, whether on a PC, UNIX or a remote mainframe over a terminal connection. SAS/SHARE™ and to some extent SAS/CONNECT™ made it possible to connect clients to SAS servers over a network, but as the diagram above suggests, SAS Integration Technologies can be used to automate and distribute SAS data processing tasks over a wide variety of platforms.

One important point, which the SAS documentation assumes but does not ever say explicitly, is that IOM is primarily designed for "thin client" applications. If all your users have SAS on their local desktops, you do not need IOM. The exception to this is SAS *Enterprise Guide*, which communicates with SAS using IOM when SAS and EG run on the same machine, but in a "thin client" world, users who need access to SAS data and/or procedures can obtain it via the Web or through a user interface constructed with any of the common development tools such as Visual Basic, C++ and even Microsoft Office.

A second point that should be emphasized is that IOM provides remote users with access not only to SAS data but also to the full capabilities of the SAS procedural language. Most relational database management systems, such as Oracle or SQL Server, have a scripting language for data manipulation in addition to retrieval, but none of these has the power of SAS to carry out complex analyses. The Oracle PL/SQL language, for example, although quite powerful in its own right is hardly capable of running PROC GLM or NLIN!

One finally important point is that IT is heavily Windows-centric. SAS has quite reasonably responded to the almost total domination of Microsoft in the desktop market by aiming many of the features of IOM toward Windows users. The SAS Management Console, for example, which centralizes and simplifies server management, is only available for Windows. The Workspace server can be running on any one of a number of different operating systems, whether

Windows, UNIX, Linux or z/OS, but the client is expected to be using some recent flavor of Windows, either NT, 2000 or XP.

So in general, when do you need SAS IT? If you are distributing thin client applications that require complex statistical and graphical capabilities then the SAS Integrated Object Model provides an elegant solution. If you just want to be able to copy the data from SAS tables into Excel, you probably do not need to use IOM to do it (although you can; see Eberhardt, 2002).

IOM Servers

Another important difference between the client-server world and the traditional SAS model is that SAS IT assumes that you have two kinds of people on your SAS programming staff: developers and system administrators. This is reflected in the materials provided by SAS-- the Version 9.1.2 IT documentation offers the following sets of manuals:

- Technical Overview
- Server Administrator's Guide
- Administrator's Guide
- Administrator's Guide (LDAP Version)
- Developer's Guide
- SAS Web Infrastructure Kit 1.0: Overview
- SAS Web Infrastructure Kit 1.0: Administrator's Guide
- SAS Web Infrastructure Kit 1.0: Developer's Guide

Finding a specific solution sometimes requires a lot of patience as you navigate a web of interrelated topics.

Alas, as is all too frequently the case with system documentation (and SAS is by no means the worst offender) the documentation assumes that you already understand what all this stuff does. Setting up the necessary SAS servers is not a task for the faint of heart. It requires a substantial amount of experience both with SAS and with the specific platform employed, whether Windows, UNIX or mainframe. The documentation is written for systems programmers, not for users, and is necessarily complex given the range of possible options available and the lengthy list of supported features.

Consequently, the initial installation for SAS Release 9.1 Integration Technologies should be done by your SAS System Administrator (if you are lucky enough to have one). Do not attempt to do this process on your desktop system-- designate one or more separate SAS servers for your network, and set aside a block of time for completing all the configuration tasks. It's not hard, it just takes time. Fortunately, you only have to do it once. Incidentally, this is where all those extra SAS user IDs (*sasadm*, *sassrv*, *sasguest*, *sastrust*, *saswdadm* and *sasdemo*) suggested in the installation documentation actually become necessary.

Once the SAS server-side components have been installed, the system administrator will need to run the *Configuration and Management Wizard*, included on a separate disk in the installation materials, to define all of the different kinds of IT servers. (You can configure a system without the wizard, but it is a lot easier if you use it; see "Getting Started Without the SAS Configuration Wizard" in the *SAS 9.1 Integration Technologies Server Administrator's Guide* http://support.sas.com/rnd/itech/doc9/admin_oma/getstart/gswconwiz.html.)

The configuration wizard generates a lengthy HTML document called "*My Configuration Steps.htm*." This document details the numerous specific steps required to complete the deployment. The first step is to start the Metadata server. On Windows, the SAS Metadata Server should be configured to run as a service that is started automatically at boot time. On UNIX or mainframe systems, the system administrator can set up the server to start automatically using the appropriate procedures for the operating platform. Consequently, you probably will not need to manually start or stop the Metadata server, or even worry very much about it, after this first time.

The subsequent steps all use the new *SAS Management Console* (available on the client-side installation disks) to set up metadata repositories, users, authorizations, libraries and servers. (For more information, see "Introduction to SAS Management Console" *SAS OnlineDoc*[®] 9.1 <<http://support.sas.com/91doc/>>.) This Windows-only application can be run on the server (if it is running a Windows OS) or on any Windows client where it has been installed. You can also use the Windows-only *SAS Enterprise Guide Administrator* to manage server connections. Note that you do

not need to have SAS installed on the client system to use these application managers—the Management Console and the Enterprise Guide work perfectly well connecting to SAS on a remote host.

When you start the Management Console, you will need to specify a *metadata profile* that includes the server to which the definitions will be written, the active *metadata repository* and any required connection information such as user IDs and passwords. At the conclusion of the approximately 28-step process detailed in the configuration file, the management console should look a lot like **Figure 1** below (depending on which products you have licensed and installed):

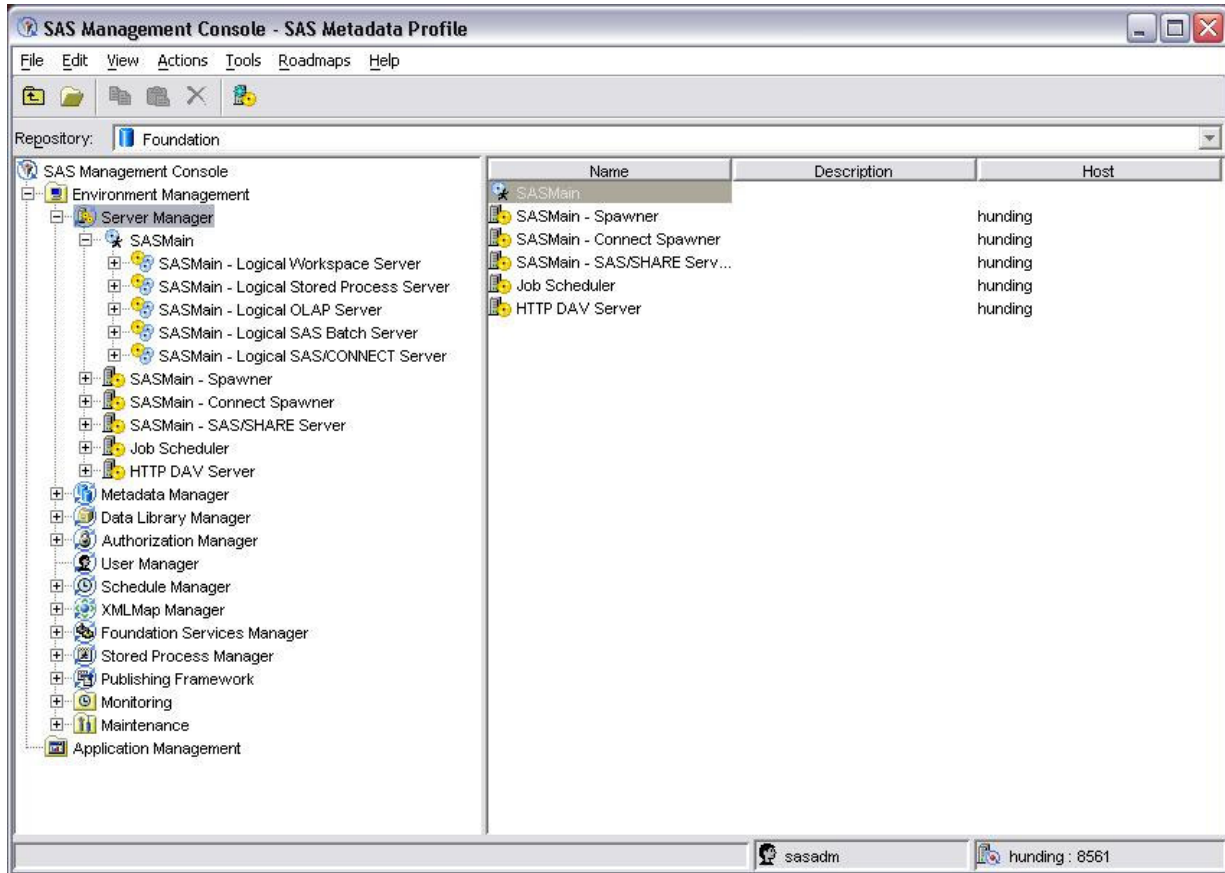


Figure 1. SAS Management Console

In order to make sense of all of the options provided by the Management Console, it is important to understand that the Integrated Object Model is implemented on the client. In other words, as a developer you need to write a program (in SAS, Java, C++ or some other general purpose programming language) that instantiates this interface and that runs on an end users' PC. However to make use of the IOM, it is necessary to have one or more "object servers" running on the network. Four kinds of IOM servers are provided, that can be managed with the console application illustrated above:

- 1) **Metadata server** – store and manage "metadata repositories", that is data about the SAS servers, libraries and stored procedures available
- 2) **Workspace server** – supports the SAS programming environment; functionally equivalent to a Display Manager session
- 3) **Stored process server** – run "canned" SAS programs
- 4) **OLAP server** – delivers data cubes (pre-summarized tabulations) to SAS Enterprise Guide or other OLAP clients.

The OLAP server is a specialized platform for data access provided with SAS *Intelligent Storage* products; space limitation preclude describing this service in detail. The other three kinds of object servers are more general in scope, and it is useful for the developer to have a clear idea of the functions and requirement for each.

The SAS Metadata Server

As noted above, in order for a client program to take advantage of distributed objects, it is necessary to have an instance of SAS running on the host. A SAS object server is an enhanced database engine, essentially SAS/SHARE on steroids. The difference is that while SAS/SHARE only supports remote library services, an IOM server provides the full functionality of a display manager session, by exposing SAS objects that provide data and methods to the client.

Version 9.1 of SAS is based on an entirely new model for storing and managing information about available data sources, business rules and security authorizations, called the SAS *Open Metadata Architecture (OMA)*. Within this architecture, the SAS *Metadata Server* is a shared software application that provides access to metadata, that is, information about data, stored in one or more *Metadata Repositories* on the server. The *Open Metadata Interface* is the API for accessing the metadata server from a variety of environments including Java, Windows applications and of course from SAS programs.

The OMA is intended to provide a single, central point of access for all of the information required by an organization, whether a business, a university or a research group. The advantages claimed for this approach are that it simplifies system support and documentation and helps to ensure data integrity. The disadvantage is that it makes everything more complicated for the SAS system administrator (and to some extent, for the end user). However, once the Metadata server is set up and configured, and procedures for accessing it are documented and disseminated, the result should be greater efficiency and lower maintenance costs.

The SAS Workspace Server

As **Figure 1** illustrates, the Metadata server is used to manage connections to one or more *Logical Workspace Servers* along with OLAP servers, Stored Process servers, and SAS/CONNECT and SAS/SHARE if they are licensed. The Workspace server is the actual connection to a SAS session. While it is technically possible to start a Workspace server by itself (this was the usual way in Release 8), generally you will require an initial call to a Metadata server first. This is because in order to use a specified SAS workspace sever, the developer needs to know how to connect to the host and what information is available there. In the new SAS Open Metadata Architecture, this information is stored in one or more "metadata repositories" and managed by the SAS Management Console, as noted above.

There are three kinds of Workspace servers, depending on the client application and the server operating system:

- 1) Java clients connect to IOM servers using the *IOM Bridge for Java* provided by SAS.
- 2) Windows client applications can connect to IOM servers running in a Windows operating environment using the Microsoft *Component Object Model (COM)* as the server protocol. If the object server and the client are on the same system, the connection uses COM; if the server is on a remote host the connection is made using DCOM, the Microsoft *Distributed Component Object Model*. (See <http://www.microsoft.com/com/> for more information about the range of Microsoft component software models including COM, COM+, DCOM and ActiveX controls.)
- 3) There is also an *IOM Bridge for COM* that allows Windows clients to access servers running on UNIX or a mainframe system such as z/OS. It can also be used to make distributed Windows-Windows connections, and SAS recommends this approach over DCOM since it is somewhat easier to administer.

For Windows applications, IOM supports the OLE DB access protocols that are used by Active Data Objects (ADO). In Java environments, the JDBC 2.0 access protocol is supported.

Whatever method is used the principal interfaces of the SAS Workspace server include:

- *Workspace* – the SAS session
- *Language Service* – submit SAS DATA and PROC steps, retrieve LOG and LIST output, run stored processes

- *Data Service* – manage SAS LIBREFS
- *File Service* – manage FILEREFS
- *Utilities* – formats, options, result packages, host system information

The section following on client implementations illustrates the use of the first two of these: the Workspace and the Language Service. The interested reader is referred to the online *Developer's Guide* for more information and examples about all of these.

The SAS Stored Process Server

The third kind of IOM server available for applications development is a *Stored Process Server*. This is a lot like the Workspace server, with one important difference.

SAS Stored Processes are slightly different than traditional SAS programs in that they must be executed on a SAS Integration Technologies server. They cannot be executed in a normal batch or interactive SAS session. (*SAS Stored Processes – A Roadmap*, 2)

The easiest way to think of stored processes is that they are just “canned” SAS programs that can be run using input parameters supplied at run time in the form of macro variables. The main difference between running a SAS job in a Workspace and running it as a Stored Process is that with the former approach you can generate and submit code from the client *or* run code already existing on the server, while the Stored Process Server requires code to be accessible from the server.

Stored process results can be made available in a variety of ways: via the Web (using an *Information Delivery Portal* or *BI Web Services*), in Microsoft Office (with the *Add-In for Office*), or with the new *Enterprise Guide 3.0*. It is also possible to access stored processes programmatically, as the example in the following section shows.

IOM Client Applications

Windows Client Interface – VB Example

The easiest way to get started programming with the Integrated Object Model is probably in Visual Basic. The following example illustrates how to set run a SAS program to display the records from a dataset in a VB *Message Box*. Obviously, it is possible to update as well as retrieve information from the server and, with ODS, to return SAS output in a variety of formats; this example demonstrates the basic principles for connecting to a remote host and displaying information.

In order to use IOM from a VB project, it is necessary to add in references to the SAS link libraries. To add these in, go to *Project>References* on the main VB menu. Scroll down the list and click to add the following references:

- SAS: Integrated Object Model (SAS System 9.1)
- SASWorkspaceManager 1.1 Type Library

This will allow VB to resolve references to the SAS objects used to connect to the server.

A trivial VB form was constructed (using Visual Studio 6 SP5); see **Figure 2** below. The code behind this form is shown as **Example 1**, attached at the end of this paper. (This program borrows heavily from the first example in Jahn, 2004.)

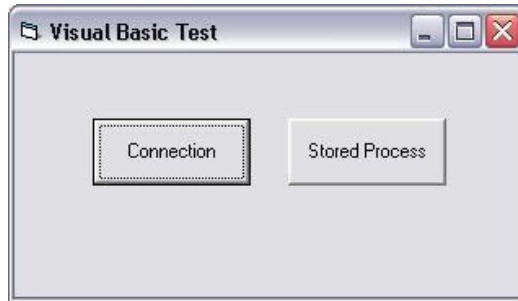


Figure 2. Simple VB Form

When the form is loaded, the program creates a new SAS Workspace on the server using the VB code below. Since the connection protocol type is not specified, the default is to create a COM connection (or in this case DCOM since the client is not running on the same host as the server):

```
' create Workspace server
Dim obServer As New SASWorkspaceManager.ServerDef
obServer.MachineDNSName = "hunding"

Set obSAS = obWSMgr.Workspaces.CreateWorkspaceByServer _
    ("", VisibilityProcess, obServer, "", "", xmlInfo)
```

Changing this form to use a Workspace server on a Linux host, using the *IOM Bridge for COM* is as simple as modifying two of the properties of the server definition object; also, the IOM Bridge requires a valid user name and password, while COM does not.

```
' create Workspace server using IOM Bridge for COM
Dim obServer As New SASWorkspaceManager.ServerDef
obServer.MachineDNSName = "hygelac"
obServer.Protocol = ProtocolBridge
obServer.Port = 8591      ' workspace server

' user name is required
Set obSAS = obWSMgr.Workspaces.CreateWorkspaceByServer _
    ("", VisibilityProcess, obServer, "sassrv", "sasuser", xmlInfo)
```

It is that easy to move your code from a Windows host to a UNIX one. There is one important difference, however. A Windows Metadata server will automatically provide logical workspace server connections as needed, but on a Linux host the IOM server has to be started manually. The following program will start a server process that will respond to one (and only one) connection request:

```
SASROOT=/usr/local/SAS/SAS_9.1
$SASROOT/sas -nonews -noautoexec -memsize max -linesize max -pagesize max \
-log MetadataServer_%Y.%m.%d.log \
-logparm "rollover=auto open=replaceold write=immediate" \
-noterminal -objectserver -objectserverparms "protocol=bridge port=8591"
```

As the log from this shell file illustrates, the Workspace server is automatically closed when the client program completes:

```
NOTE: Copyright (c) 2002-2003 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) 9.1 (TS1M0)
      Licensed to FREDERICK PRATTER, Site 0041553004.
NOTE: This session is executing on the Linux 2.4.20-31.9 platform.

NOTE: SAS initialization used:
      real time          1.77 seconds
```

```

      cpu time          0.23 seconds

20040808:19.29.11.23: 00000007:
      Defined server listen connection (1) on port 8591.
20040808:19.29.18.18: 00000010:
      New client connection (2) accepted from server port 8591 for user sassrv.
      Peer IP address and port are 192.168.0.2:1058.
20040808:19.29.22.41: 00000006:          Server listen connection (1) closed.
20040808:19.29.22.41: 00000006:          Client connection (2) closed.

NOTE: The SAS System used:
      real time          13.29 seconds
      cpu time           0.36 seconds

```

In order to avoid this behavior, it is necessary to create a pooled connection server on the remote host; see “Using Connection Pooling” in the *SAS 9.1.2 Integration Technologies Developer's Guide* <http://support.sas.com/rnd/itech/doc9/dev_guide/dist-obj/javaInt/javaprogram/connpool.html>.

Note that SAS is not installed on the client system—just Visual Studio. The SAS program below is stored on and run by the server:

```

C:\temp>type IOMTest.sas
%let cond=;
*ProcessBody;
proc print data=sashelp.class;
      title "Test Stored Process";
      where &cond;
run;

```

The comment line `*ProcessBody;` is required for the `StoredProcessService` facility (see below) but is otherwise ignored. The macro variable `cond` is initialized to null; the value of this variable can be supplied as a parameter if the program is run as a stored process.

The first button (`cmdTest1_Click—Connection`) uses the `SAS LanguageService` interface to submit a simple SAS program. If the first subprogram is run (by clicking on the `Connection` button), the `where` statement is read just as `where ;` which SAS ignores, so all 19 observations in the source data set are printed.

A screen print of the first message box is shown in **Figure 3**, showing the SAS log generated by the server process.

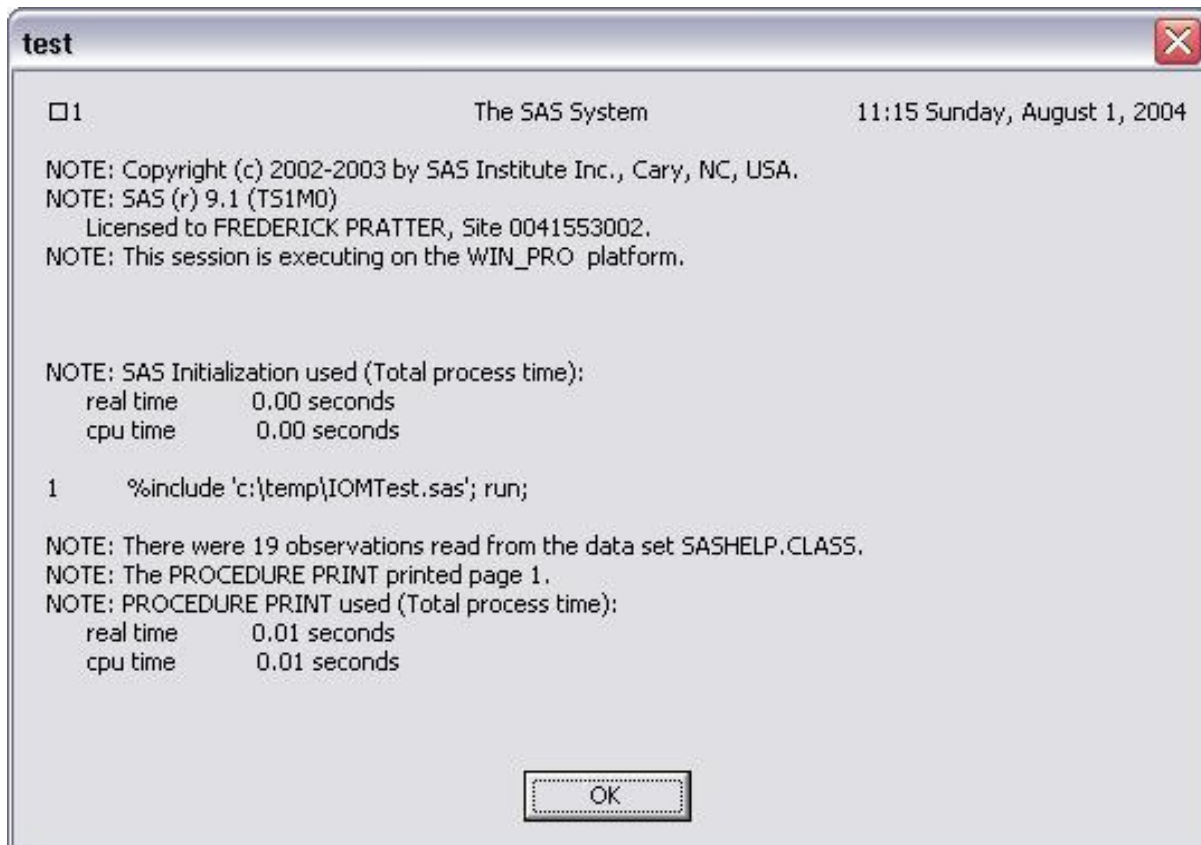


Figure 3. Submit SAS Program

The second subprogram (*cmdTest2_click*-- the **Stored Process** button event handler) uses the `execute` method of the `StoredProcessService` object, instead of the `submit` method of `LanguageService`:

```
'run the stored SAS program
Dim obStoredProcessService As SAS.StoredProcessService
Set obStoredProcessService =
    obSAS.LanguageService.StoredProcessService

obStoredProcessService.Repository = "file:c:\temp"
obStoredProcessService.Execute "IOMtest", "cond='sex eq "M"'"
```

This code passes a name/value pair to the stored procedure. Here the name of the parameter is `cond` and the value `sex eq "M"`; thus only the male students are listed (see **Figure 4** for the SAS log). Note that the line numbers continue from the first log, since SAS considers this a single workspace session. The form `unload` subprogram then closes the workspace.

Note the parallels to the VB code. If you have been paying attention, it is obvious that this program uses the SAS IOM Bridge for COM to connect to a Linux host. The one absolutely critical line in this program is the initialization of the COM library by the statement:

```
HRESULT hr = CoInitialize(NULL);
```

Without this line your application program will abort; see the Microsoft Knowledge Base Article 169496 "INFO: Using ActiveX Data Objects (ADO) via #import in VC++" for the details of this vitally important initial step.

Since the output of this program is identical to **Figure 3** above, it is not shown here.

Java Client Interface

Alan Cooper has written that programmers by nature are inherently willing to trade simplicity for control: "The price of control is always more effort and increased complexity" (Cooper, 1999. p.96). The SAS implementation of the *IOM Bridge for Java* is a classic example of this approach. You can do anything with it, if you are willing to make the investment in learning the API. The following Java program (included as **Example 3** attached) is about as simple as it is possible to make it. It opens a connection to a remote server and runs the same SAS program as in the previous example. It will run on any client that has the SAS Java Foundation classes installed; this particular example was compiled on a Linux workstation. (This code is taken from the example "Connecting With Directly Supplied Server Attributes" *SAS 9.1.2 Integration Technologies Developer's Guide* at http://support.sas.com/rnd/itech/doc9/dev_guide/dist-obj/javaclnt/javaprogram/connfact_direct.html.)

In order to run the code, it is necessary to add the required SAS IOM objects to the Java classpath. The SAS Foundation Services archive (jar) files are installed by default in the directory `SASFoundationServices\1.1\jars` under the SAS root directory. Since SAS is not installed on the Linux client (it is not currently available for this particular flavor of Linux), the single required jar file was simply copied to the run time directory on the client. The code to compile the sample program is:

```
javac -classpath ../sas.svc.connection.jar IOMTest.java
```

The same classpath was used to run the example. The connection log shown below is sent to the standard error output (`stderr`), presumably in order to avoid interactions with the list output. It illustrates the messages and responses sent back and forth over the TCP/IP connection to the IOM Bridge:

```
Aug 1, 2004 12:43:52 PM com.sas.services.connection.AggregationKernel
getConnection
INFO: connection request received

Aug 1, 2004 12:43:53 PM com.sas.services.connection.BridgeServer connect
INFO: properties for new connection: {encryptionContent=all, port=8591,
clsid=440196d4-90f0-11d0-9f41-00a024bb830c,
logfile=java.util.logging.Logger@1125127, encryptionPolicy=none,
userName=sassrv, password=xxxxxxx, protocol=bridge, host=hunding}

Aug 1, 2004 12:43:55 PM com.sas.services.connection.AggregationKernel
getConnection
INFO: request served by unshared connection #0

Aug 1, 2004 12:44:04 PM com.sas.services.connection.AggregationKernel
reactivateConnection
INFO: connection #0 returned to factory by user

Aug 1, 2004 12:44:04 PM com.sas.services.connection.AggregationKernel
destroyConnection
INFO: connection #0 destroyed
```

The log window displayed by Java is shown in **Figure 5**:

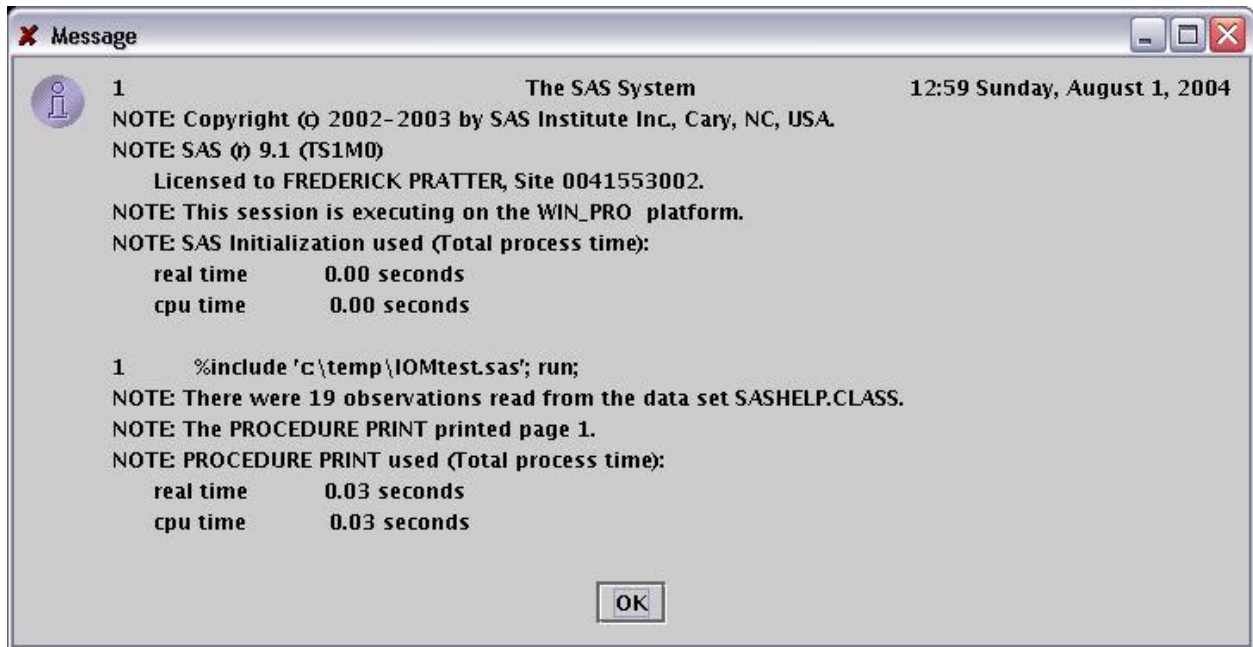


Figure 5. Simple Java Example: Log Output

This is precisely the equivalent of the VB output shown in **Figure 2**.

Conclusion

SAS Integration Technologies offers a way to connect to the flexibility and power of SAS from thin-client workstations. As Peter Eberhardt points out: "Integration technologies moved SAS from a closed, proprietary system out into the open systems area. (Eberhardt, 2003)" The revolutionary implications of this have not yet spread widely to the SAS user community, but an entirely new paradigm for SAS development has now become possible. At the same time, starting with out IT and the Integrated Object Model is very much like those old text-based adventure games that begin: "You are in a cave. There is a key on the floor in front of you. The cave has passages to the E, W, and S. What do you want to do?"

The correct answer is of course "Pick up key."

Code Samples

Example 1 – Connecting with DCOM in Visual Basic

```
Option Explicit

' define a global workspace
Dim obsAS As SAS.Workspace
Dim obWSMgr As New SASWorkspaceManager.WorkspaceManager

Private Sub Form_Load()

    Dim xmlInfo As String

    ' create Workspace server
    Dim obServer As New SASWorkspaceManager.ServerDef
    obServer.MachineDNSName = "hunding"

    Set obsAS = obWSMgr.Workspaces.CreateWorkspaceByServer _
        ("", VisibilityProcess, obServer, "", "", xmlInfo)

End Sub

Private Sub cmdTest1_Click()

    ' use LanguageService to submit code
    obsAS.LanguageService.Submit _
        "%include 'c:\temp\IOMTest.sas'; run;"

    MsgBox obsAS.LanguageService.FlushLog(100000)
    MsgBox obsAS.LanguageService.FlushList(100000)

End Sub

Private Sub cmdTest2_Click()

    'run the stored SAS program
    Dim obStoredProcessService As SAS.StoredProcessService
    Set obStoredProcessService = _
        obsAS.LanguageService.StoredProcessService

    obStoredProcessService.Repository = "file:c:\temp"
    obStoredProcessService.Execute "IOMtest", "cond='sex eq ""M""'"

    MsgBox obsAS.LanguageService.FlushLog(100000)
    MsgBox obsAS.LanguageService.FlushList(1000000)

End Sub

Private Sub Form_Unload(Cancel As Integer)

    obWSMgr.Workspaces.RemoveWorkspaceByUUID _
        obsAS.UniqueIdentifier
    obsAS.Close

End Sub
```

Example 2 – Connecting with the IOM bridge for COM in Visual C++

```
#include <iostream>
#include <stdexcept>
#include <windows.h>

using namespace std;

#import "C:\Program Files\SAS Institute\Shared Files\Integration
Technologies\sas.tlb"
#import "C:\Program Files\SAS Institute\Shared Files\Integration
Technologies\SASWMan.dll"

int main()
{
    SASWorkspaceManager::IWorkspaceManager2Ptr piWorkspaceManager;
    SASWorkspaceManager::IServerDef2Ptr piServerDef = NULL;
    SAS::IWorkspacePtr piWorkspace;

    BSTR xmlInfo;
    HRESULT hr = CoInitialize(NULL);

    hr = piWorkspaceManager.CreateInstance(
        "SASWorkspaceManager.WorkspaceManager.1");

    piServerDef.CreateInstance("SASWorkspaceManager.ServerDef");
    piServerDef->PutMachineDNSName("hygelac");
    piServerDef->Protocol = SASWorkspaceManager::ProtocolBridge;
    piServerDef->put_Port(8591);

    piWorkspace = piWorkspaceManager->Workspaces->CreateWorkspaceByServer(
        _bstr_t(""), //workspace name
        SASWorkspaceManager::VisibilityProcess,
        piServerDef, // server
        _bstr_t("sassrv"), // login
        _bstr_t("sasuser"), // password
        &xmlInfo // connection log
    );

    piWorkspace->LanguageService->Submit(
        "%include '/home/sasadm/IOMTest.sas'; run;");

    MessageBox(NULL,
        piWorkspace->LanguageService->FlushLog(10000),
        "SAS Log",
        MB_OK
    );

    MessageBox(NULL,
        piWorkspace->LanguageService->FlushList(10000),
        "List Output",
        MB_OK
    );

    piWorkspace->Close();

    return(0);
}
```

Example 3 – Using the Java Connection Factory

```
import com.sas.services.connection.Server;
import com.sas.services.connection.BridgeServer;
import com.sas.services.connection.ConnectionFactoryConfiguration;
import com.sas.services.connection.ConnectionFactoryManager;
import com.sas.services.connection.ConnectionFactoryInterface;
import com.sas.services.connection.ConnectionFactoryException;
import com.sas.services.connection.ConnectionInterface;
import com.sas.services.connection.ManualConnectionFactoryConfiguration;

import com.sas.iom.SAS.IWorkspace;
import com.sas.iom.SAS.IWorkspaceHelper;
import com.sas.iom.SAS.ILanguageService;
import com.sas.iom.SAS.ILanguageServicePackage.CarriageControlSeqHolder;
import com.sas.iom.SAS.ILanguageServicePackage.LineTypeSeqHolder;

import com.sas.iom.SAS.IOMDefs.GenericError;
import com.sas.iom.SAS.IOMDefs.StringSeqHolder;

import javax.swing.JOptionPane;

public class IOMTest{

    public IOMTest() throws ConnectionFactoryException, GenericError
    {
        // connection parameters
        String classID = Server.CLSID_SAS;
        String host = "hunding";
        int port = 8591;
        String userName = "sassrv";
        String password = "sasuser";

        // identify the IOM Bridge server (the Workspace server)
        Server server = new BridgeServer(classID,host,port);

        // make a manual connection factory configuration to the server
        ConnectionFactoryConfiguration cxfConfig =
            new ManualConnectionFactoryConfiguration(server);

        // get a connection factory manager
        ConnectionFactoryManager cxfManager =
            new ConnectionFactoryManager();

        // get a connection factory interface from the manager
        ConnectionFactoryInterface cxf = cxfManager.getFactory(cxfConfig);

        // get a connection from the interface
        ConnectionInterface cx = cxf.getConnection(userName,password);

        // create a workspace by "narrowing" the connection to the ORB
        IWorkspace iWorkspace = IWorkspaceHelper.narrow( cx.getObject() );

        // Submit batch SAS code
        ILanguageService sasLanguage = iWorkspace.LanguageService();
        sasLanguage.Submit("%include 'c:\\temp\\IOMtest.sas'; run;");

        // flush log file to string array
        StringSeqHolder logHldr = new StringSeqHolder();
        sasLanguage.FlushLogLines(
            Integer.MAX_VALUE,
            new CarriageControlSeqHolder(),
```

```

        new LineTypeSeqHolder(),
        logHldr);

// display log file
String[] logLines = logHldr.value;
JOptionPane.showMessageDialog(null, logLines);

// flush list file to string array
StringSeqHolder listHldr = new StringSeqHolder();
sasLanguage.FlushListLines(
    Integer.MAX_VALUE,
    new CarriageControlSeqHolder(),
    new LineTypeSeqHolder(),
    listHldr);

// display list file
String[] listLines = listHldr.value;
JOptionPane.showMessageDialog(null, listLines);

iWorkspace.Close();
cx.close();
}

public static void main(String args[]) {
    try {
        new IOMTest();
        System.exit(0);
    }
    catch(Exception ex) {
        ex.printStackTrace();
        System.exit(1);
    }
}
}

```

References

SAS Technical Documentation

URL references are current as of the date of August 2004 but please check the Enterprise Integration Community pages at <http://support.sas.com> for the most up-to-date information.

- SAS OnlineDoc (includes links to all the separate product documentation). <<http://support.sas.com/91doc/docMainpage.jsp>>
- SAS Integration Technologies: Version 9 Documentation. <<http://support.sas.com/rnd/itech/library/library9.html>>
- SAS Integration Technologies: Release 8.2 Documentation. <<http://support.sas.com/rnd/itech/library/library82.html>>
- SAS 9.1 Open Metadata Architecture: Best Practices Guide. <<http://support.sas.com/rnd/eai/openmeta/v9/bestpractices/>>
- SAS 9.1.2 Metadata Server: Setup Guide. <<http://support.sas.com/rnd/eai/openmeta/v9/setup/>>

SAS White Papers

Available from <<http://support.sas.com/rnd/itech/papers/>>

- SAS Stored Processes — A Roadmap <[101519_0504.pdf](#)>
- SAS Integration Technologies: A Roadmap <[48030_0202.pdf](#)>
- SAS Integration Technologies Overview <[oviewSUGI24.html](#)>
- Jahn, Daniel. (2004). "Developing an Open Client in Visual Basic". <[VBClient.pdf](#)>
- Vodicka, Scott. (2000). "Enterprise Integration Technologies: What is it and what can it do for me?" *Proceedings of the 25th Annual SAS Users Group International Conference*, Paper 141 <[eitover.pdf](#)>

User Publications

- Cohen, Barry R. "Using AppDev Studio and Integration Technologies for an Easy and Seamless Interface between Java and Server-Side SAS" *Proceedings of the 29th Annual SAS Users Group International Conference*, Paper 232. March 2000. <<http://www2.sas.com/proceedings/sugi29/232-29.pdf>> (August 1, 2004).
- Cooper, Alan. (1999). *The Inmates Are Running the Asylum*. SAMS, Indianapolis.
- Eberhardt, Peter. "Bring the Data Warehouse to the Office with SAS[®] Integration Technologies" *Proceedings of the 29th Annual SAS Users Group International Conference*, 222. March 2000. <<http://www2.sas.com/proceedings/sugi29/222-29.pdf>>. (August 1, 2004).
- Eberhardt, Peter. "SAS[®] in the Office. IT Works" *Proceedings of the 28th Annual SAS Users Group International Conference*, Paper157. March 2003. <<http://www2.sas.com/proceedings/sugi28/157-28.pdf>> (August 1, 2004).
- Eberhardt, Peter. "Rev up Your Spreadsheets With Some V8 Power" *Proceedings of the 27th Annual SAS Users Group International Conference*, Paper 26. March 2002. <<http://www2.sas.com/proceedings/sugi27/p026-27.pdf>> (August 1, 2004).
- Essam, Katie. "Building Windows Front ends to SAS[®] Software". Amadeus Software Limited. <http://www.amadeus.co.uk/events_resources/conferences/Views 2003/Katie Essam Building Windows Front Ends to SAS Software.doc> (August 1, 2004).
- Nicklin, Clare A. "The Use of Java with the SAS[®] System". Amadeus Software Limited. <http://www.amadeus.co.uk/events_resources/conferences/Views 2003/Clare Nicklin The Use of Java with SAS.doc> (August 1, 2004).
- Nicklin, Clare A. and Daniel Morris. "A Successful Implementation of a Complicated Web-based Application Through webAF[™] and SAS[®] Integration Technologies" *Proceedings of the 28th Annual SAS Users Group International Conference*, Paper 178. March 2003. <<http://www2.sas.com/proceedings/sugi28/178-28.pdf>> (August 1, 2004).
- Nipko, Joseph. "Using SAS Integration Technologies To Interface With Enterprise Applications Written In Visual C++" <<http://www.wuss.org/Conference/papers/CC09.pdf>> (August 1, 2004).

- Pope, Lynn. "Communicating with SAS Using SAS Integration Technologies". Amadeus Software Limited. <[http://www.amadeus.co.uk/about/VIEWS_2004/Papers/Communicating with SAS using SAS Integration Technologies.pdf](http://www.amadeus.co.uk/about/VIEWS_2004/Papers/Communicating_with_SAS_using_SAS_Integration_Technologies.pdf)> (August 1, 2004).
- Silva, Greg. "Using IOM and Visual Basic in SAS Program Development" *Proceedings of the 28th Annual SAS Users Group International Conference*, Paper 32. March 2003. <<http://www2.sas.com/proceedings/sugi28/032-28.pdf>> (August 1, 2004).

Acknowledgments

I would like to particularly thank David Barron at SAS for his unfailing good humor despite my many calls to the Tech Support line. Daniel Jahn was kind enough to read a draft of this paper and make a number of valuable suggestions for improvement. As usual, I take credit for the remaining errors and infelicities. John West, my ever-helpful editor at SAS BBU, provided contacts and support at SAS without which this work would be considerably less than it is. I'll get to thank him again when the 2nd edition of the book comes out.

Trademark Citation

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries: ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contact

Frederick Pratter, Ph.D.
Division of Science, Math and Technology
Eastern Oregon University
La Grande OR 97850
fpratter@eou.edu
<http://www.eou.edu/~fpratter>
541-962-3065