Enterprise Integration Technologies

What is it and what can it do for me?

Scott Vodicka, SAS, Cary, NC

ABSTRACT

Today, businesses are faced with many challenges such as increased competition, acquisitions and mergers, new worldwide markets, and changing hardware and software environments. To survive under these conditions companies must integrate their applications throughout the enterprise. Enterprise Integration Technologies provides the ability to integrate your SAS ® applications in the diverse information systems arena.

Enterprise Integration Technologies (EIT) is a new bundle of products from SAS that include SAS/CONNECT® software, SAS/IntrNet® software, and SAS Integration Technologies. SAS Integration Technologies is a new product with Version 8 of the SAS System. The products in this bundle support application integration and access to a SAS server for SAS clients, Web clients, and other thin clients. These products may be licensed individually. To fully enable your SAS server you may license all three products as the Enterprise Integration Technologies bundle.

Our business intelligence, data warehousing, data mining, and decision support applications are enabled through the technologies in EIT. In the past, these applications have relied on SAS/CONNECT software to provide client/server capabilities between a SAS client and server. SAS/IntrNet software opened them up to the world of Web based clients. Now, with the introduction of SAS Integration Technologies a new generation of SAS applications can be delivered, achieving the business requirements for supporting more data, more users, and more questions with fewer resources. The Enterprise Integration Technologies bundle allows you to fully enable your server platform to integrate with other applications, opens The SAS System on your server to all clients, and lets you proactively deliver information.

INTRODUCTION

SAS is known for its award winning solutions; at the core of these solutions are the technologies from SAS that can now be licensed as a product bundle in Enterprise Integration Technologies. SAS/CONNECT and SAS/IntrNet software provided a foundation on which successful SAS solutions are built. SAS Integration Technologies adds to the foundation by providing additional opportunities to create open client access to SAS software.

Each of these products can be licensed independently but if you need to fully enable your SAS server then they may also be licensed as part of the Enterprise Integration Technologies bundle.

This paper will explain the capabilities of SAS/CONNECT, SAS/IntrNet, and SAS Integration Technologies. The sections on SAS/CONNECT and SAS/IntrNet will be overviews of these technologies, and the sections on Integration Technologies will explain what Integration Technologies is and what these technologies can do for you.

SAS/CONNECT

SAS/CONNECT software provides the essential tools for sharing data and applications across multiple computing environments. It extends the power of the SAS System's MultiVendor Architecture by giving you control over where and how to execute each part of a SAS application.

SAS/CONNECT software provides middleware connectivity for transparent access to enterprise-wide data in virtually any format or location. Local SAS clients can connect to multiple data sources on different platforms and can even combine diverse data types. With SAS/CONNECT software, data can be extracted for local processing or accessed interactively on a remote platform.

SAS/CONNECT software's unique Remote Computing Services provide dynamic relocation of applications logic. Applications developed on one system can be easily moved or even divided for execution on multiple diverse systems. Because the SAS System's unique MultiVendor Architecture ensures the same robust functionality across all computing platforms, the same capabilities available on a client system are also available on any server

When do you use SAS/CONNECT software? When you have SAS applications that need to access data on other systems, that want to partition the logic in the application so that sections of the program execute on another system, or that need to transfer data from the remote system to the local system or vice versa. SAS/CONNECT can provide all of these capabilities for your SAS applications.

SAS/CONNECT software provides the core connectivity between a SAS client and a SAS server.

SAS/IntrNet

SAS/IntrNet software extends SAS software's powerful data retrieval and analysis functionality to the Web. SAS customers now have more flexibility to deploy SAS applications throughout their organization. These applications, powered by SAS/IntrNet software, will enable customers to deliver core SAS functionality to desktops worldwide. Whether you're building internet, intranet or extranet applications, SAS/IntrNet software can assist you in delivering business critical information to users all over the world.

SAS/IntrNet software helps organizations with:

Report Distribution - Share reports generated by SAS applications to anyone within and outside the organization who has a Web browser. It also allows for the provision of ad-hoc reporting applications to those audiences. Organizations can therefore extend their applications' user base quickly and at minimal cost since all the user needs is a Web browser.

Application Distribution - Build and distribute applications via the Internet. This offers reduction in applications maintenance costs since the application needs to be updated only once on the server as opposed to on multiple user machines running multiple operating systems. Also, users can Web-enable existing SAS applications without needing to know CGI programming (commonly used for building dynamic Web applications).

Thin Clients - Deploy sophisticated applications across the Web, reducing the need for additional desktop disk space. SAS/IntrNet software eliminates the necessity of physically deploying an application in order to make it available on each desktop.
One of the distinct advantages of SAS/IntrNet software is the ability to quickly build and deploy powerful decision support applications without having to do any CGI programming. In addition, SAS/IntrNet software provides robust Java tools for

building thin client interfaces to SAS. Since SAS applications are used for such a wide range of functions, SAS/IntrNet software reduces the development time and costs that are normally associated with Web enablement

When do you use SAS/IntrNet software? When you need Web based thin-client interfaces for the computing and data services provided by the SAS System.

SAS Integration Technologies

SAS Integration Technologies is a great name for the new product released with SAS Version 8. The name says it all. SAS Integration Technologies gives you new ways to integrate your open client applications with the SAS server, and new ways to integrate your SAS applications and solutions with other applications and systems. SAS Integration Technologies does this by supporting industry standard technologies and with a new framework for proactively delivering information.

The industry standard technologies are:

- Standard component models: COM/DCOM, CORBA
- Third party messaging software. IBM's MQSeries and Microsoft's MSMQ
- Open Network Directory standard. Light Weight Directory Access Protocol (LDAP)

The new framework with Integration Technologies is the Publishing Framework.

Standard Component Models

Seeing the need to support more and different types of clients the SAS System has evolved from supporting SAS based clients, to Web based clients, and now with the introduction of SAS Integration Technologies the support for "Open Clients" – client applications that support the distributed object model standards of DCOM or CORBA – is available. The SAS System's rich set of features and its ability to access a wide variety of data can now be leveraged by applications developed in languages such as C++, Java, Microsoft's Visual Basic, and other languages that support distributed object programming.

The problem can be defined as follows. You have a large knowledge base in your employees that develop applications in component-based languages. You have the SAS System on a server platform near your data. You know that the SAS System gives you the capability to access this data seamlessly and gives you a rich set of data analysis capabilities. But, you do not know how to let your application developers communicate with the SAS System, how to use it to provide your client application with access to the wide range of data sources, and how your client application can take advantage of SAS processing on the server near the data and displaying the results in your client application.

The answer to this problem is SAS Integration Technologies and the Integrated Object Model, referred to as IOM. IOM is a distributed object model that provides application developers a set of objects they can script in their client application and have the distributed object provide the requested services on the SAS Server. IOM provides your client application developers open access to the wide range of features that are a part of the SAS system.

IOM consists of a hierarchy of objects depicted in Figure 1. The object at the top of the hierarchy is the SAS Workspace. From the SAS Workspace object the application can create the following objects:

- *Data Service object* - to access SAS Library information and access data via standard API's such as ADO, OLE DB, and JDBC.
- *File Service object* - provides access to the files and filerefs defined to the SAS System
- *Language Service object* - provides the interface for executing SAS Systems 4GL language programs on the SAS server. The SAS programs may either reside in the client side source code or as a stored SAS program on the server that is callable by name.
- *Utility Service object* - provides a number of services that control various utility features with the SAS Workspace. These features include result packages, formats, informats, and options.

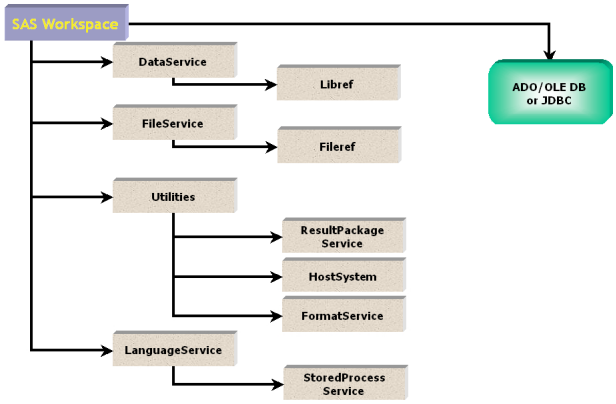

Figure 1 - IOM Object Hierarchy

Lets take a look at a sample VB program that connects to the SAS Server, runs a sample SAS program, and displays the results.

To start the SAS session you create the SAS Workspace object. Since the server name has not been specified a local SAS session will be started.

```
Set sWorkspace = New SAS.Workspace
```

Now lets create the language service object that will be used to submit our SAS program.

```
Set sLangServ = sWorkspace.LanguageService
```

Using the language service object submit the SAS program.

```
sLangServ.Submit ("proc univariate
data=sashelp.retail; run;")
```

Using the language service object loop through and write each line of the output to the textbox.

```
tbOutput.text = tbOutput.text + vbCrLf + "***
Printing LIST for UNIVARIATE ***" + vbCrLf
more = True
While (more)
    sLangServ.FlushListLines 18, cc, lt,
lines
    n = UBound(lines)
    If n < 0 Then
        more = False
    Else
        For i = 0 To UBound(lines)
           tbOutput.text = tbOutput.text +
lines(i) + vbCrLf
        Next i
    End If
Wend

tbOutput.text = tbOutput.text + "*** End of
LIST for UNIVARIATE ***" + vbCrLf
```

Display the closing message, end the SAS session, and delete the Workspace object.

```
tbOutput.text = tbOutput.text + "*** Closing
SAS Session ***" + vbCrLf
```

```
sWorkspace.Close
sWorkspace = Nothing

End Sub
```

This is a simple example that only utilizes a couple of the objects in IOM. In a production application you would take advantage of many of the features presented in the other object's properties and methods. Most importantly would be to take advantage of the strong output formatting capabilities of the Output Delivery System in your SAS programs on the server and using the Result Package features of the Utility Service object to retrieve these results for display on the client.

This example utilizes a local version of the SAS System but it could very easily be changed to use SAS Software on a remote Windows NT server, a UNIX server, or a OS\390 session on the mainframe. To change this program from using the local SAS System to using the SAS System on a server only the one line where the Workspace object is created would have to be changed. This is powerful. A client application can be written that is driving the SAS system on a remote server, even the mainframe! Let your client application do what it does best: manage the user interaction, and let the powerful servers and the SAS System do their job of analyzing the often-large sets of data.

Third Party Messaging software

Messaging software is another form of middleware technology used to integrate disparate applications throughout your enterprise. It is often referred to as MQM – Message Queuing Middleware, or MOM – Message Oriented Middleware.

Messaging software provides the infrastructure to support sending messages between applications spread across your organization. It provides the ability to set up queues for an application to receive messages from any other application that has the capability of interfacing with the MQM software. Part of the infrastructure of messaging software is the messaging server software, the messaging client software, the message queues, and the management of delivering the message from the client application to the defined queue. Messaging software guarantees that once it is given a message to deliver to a queue, it will get there. The message delivery mechanism is often referred to as *store and forward* message delivery where Intermediate queues are used to guarantee delivery of the message to the destination queue.

An important property of messaging software is the ability for an application to send a message to another application without having to create a direct or synchronous connection between each other. Many times these direct, or point-to-point connections are impossible due to the distributed nature of the applications.

Messaging software relies on asynchronous communications between the communicating applications. The application receiving the messages does not have to be available in order for messages to be sent to it. When it comes online all of the messages will be waiting in the message queue for it to process. This is also important for the application sending the message. It can continue with other processing since it does not have to wait on the response for each request. The application can post requests, continue on with other processing, and respond to the replies later on when they are received.

A driving force for the popularity of messaging software is the ability for applications on diverse platforms to communicate with each other. Many applications that require services from other applications in your enterprise were written at different times, using different languages, and running on different platforms. All three of these can serve as barriers to integrating your applications that messaging software can solve. The older legacy applications that were written years ago, yet still have a significant value in your organization may be written in languages that do not support distributed object technology, such DCOM and CORBA, but can take advantage of messaging software to request services from other, perhaps newer, applications.

An example is illustrated in Figure 2 below. Your organization may have applications running in different areas of your organizations. These applications could even be spread across subsidiaries yet require a common set of services.

Figure 2 below is a diagram of applications spread throughout your network and organizations. A message queuing system and message queues have been set up to allow these applications to communicate. The Supply Chain, Transportation, Finance, and Marketing applications all run in different departments or subsidiaries and on a variety of platforms. The Transportation application written in COBOL runs in the shipping subsidiary on a UNIX platform in Denver, CO. The brand new Supply Chain application written in C++ runs at the corporate headquarters in Raleigh, NC on a Windows NT server. Both of these applications need to know the availability of products from the mainframe Availability application written in COBOL that is in Raleigh. Immediately you can see the issues you are faced with: different platforms, distant locations, different applications. Messaging software provides the way to integrate applications like these. The Transportation application posts a message to the Availability applications message queue requesting the number of available products for the current order being filled and continue on with other processing while the request is being processed. It is not held up waiting on the one requests. At the same time the Supply Chain application can post a request to the Availability application. The Availability application can provide services to many applications in this manner. The power of messaging software is in its abilities to allow applications like these to overcome the barriers to integration.

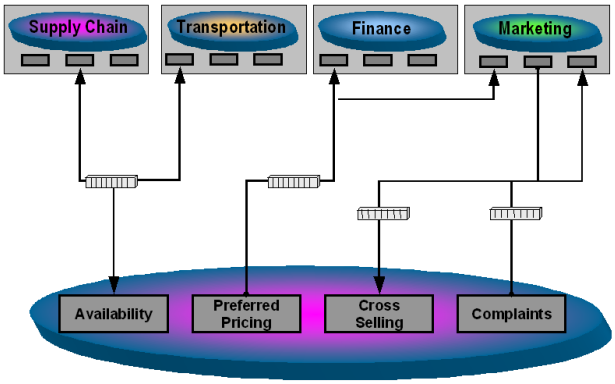

Figure 2 - Integrating Applications with Message Queues

Your SAS applications can now integrate with your enterprise applications. SAS Integration Technologies provides support for two commercial messaging systems: IBM's MQSeries and Microsoft's MSMQ. The support for MQSeries and MSMQ are via three API's. One API that mimics the MQI API for MQSeries, one API that mimics Microsoft's Message Queuing Services API, and the final API that is generic to be used for either system. Using these API's your SAS applications can both send and receive requests via message queues giving them the capability to integrate with and provide services to your enterprise applications.

Open Network Directory support

In today's complicated networking environment, even relatively

small intranets can quickly become cluttered, making it difficult to find useful or necessary resources. Administration also becomes problematic as users access those resources from multiple platforms. Security must be administered in multiple places in disparate ways. Directory services provide a repository for user data, resource data, and security data that can be administered in one place using one interface. LDAP (Lightweight Directory Access Protocol) is the protocol used to communicate with Enterprise Directory Servers.

The Enterprise Directory servers provide a common location accessible to all applications throughout the distributed enterprise. Since many languages have interfaces for accessing Directory Servers using the LDAP protocol, the information in the server is accessible to applications written in a variety of languages running on various platforms.

Information is stored in an Enterprise Directory Server in a hierarchical tree format. For example: The entries for several of the departments in SAS are depicted in Figure 3. At the root of the tree is an entry for the country. The country entry is specified as `c=US`. The company entry is specified `as o=SAS, c=US`. As you can see as you proceed down the levels of the tree you in order to specify an entry name you append the levels from above to the name. A name that specifies an entry is called a Distinguished Name. So the Distinguished Name for the Marketing Department (MKT) is: `ou=MKT, o=SAS, c=US`. The abbreviations on the left side of the equal signs are defined as:

```
c=   Country
O=   Organization
Ou=  Organizational Unit
```

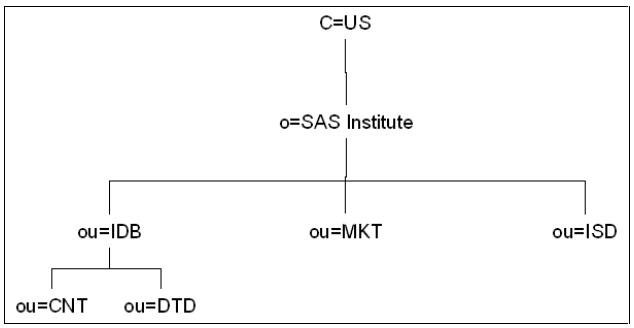

Figure 3 - Sample Directory Structure

Entries in an Enterprise Directory server have a defined set of required information. Each entry type is an object that is defined by the Directory Schema. An important feature of the entries is that they may have other information associated them by name-value pairs. This gives applications the flexibility to define the information it needs associated with each entry.

For a user entry you would normally keep track of typical data such as name, office number, phone numbers, email address, etc. along with any other information that applications may need for each user. A sample entry for a person would be:

```
cn=John Doe,o=SAS,c=US
sn=Doe
objectclass=top
objectclass=person
objectclass=sasPerson
l=Cary
title=SR SYST DEV
ou=CNT
ou=IDB
mail=John.Doe@sas.com
telephonenumber=5555
roomnumber=4411 - 123
uid=SASABC
```

Enterprise Directory Servers are gaining in popularity so applications that truly integrate into the enterprise architecture must have the ability to search, read, and update information in these servers. To that end SAS Integration Technologies provides a set of interfaces for the SAS 4GL and SCL languages. Additionally the SAS Publishing Framework relies on a LDAP server to store information such as users, channels, and data source information.

Using the LDAP interfaces your SAS applications may search, read, and update information in the Enterprise Directory server. This opens even more integration options for your SAS programs to allow them to access the enterprise information available in the Enterprise Directory server.

Publishing Framework

In the business world you have people and processes that generate information called *information producers*, and you have people and processes that consume information called *information consumers.* How do the information producers deliver information to the information consumers making sure that the consumers who need the information are actually the ones who receive it? On the other hand how do the information consumers make sure they are using the correct information? Is the report they are looking at the most recent? Is the set of data they are using for their analysis the correct version? These questions are what the Publishing Framework answers.

The *Publishing Framework* provided by SAS Integration Technologies lets information producers proactively deliver information to users in the format that they need it when they need it. We all publish information just about everyday using email. When we publish information we have to know exactly who is interested in the information we are publishing. Yes, we have personal distribution lists, but as users interests change we have to add or remove users from the distribution list ourselves. The Publishing Framework lets you leave these manual requirements behind by letting your SAS applications dynamically generate the publishing content and letting the end users subscribe to a channel where the information is published. When a user is no longer interested in the information delivered via the channel they unsubscribe from the channel without having to involve the owner of the channel.

What can the Publishing Framework publish to?

- To email addresses
- To message queues
- To a channel
- To an archive

How is the publish content generated?

- SAS programs that generate reports and data
- Desktop applications such as spreadsheets, word processors, etc.
- Warehouse processes
- SAS Publish Window

What can be published?

- Text Notification messages – event, data update messages
- URL's – web page address along with descriptive text
- HTML & HTML Framesets
- SAS Data – relational tables, structured data stores
- MIME-associated data stores – PDF, DOC, XLS, PPT, BMP, GIF,…

The text notification messages are email messages that you

generate via the SAS Publish Window, or by your SAS applications. These email messages may be either strictly a text based format or formatted HTML text. You chose the format of the message that is supported by your email viewers. Figure 4 contains two HTML formatted emails that were published by a SAS program to the Business Reports channel. All subscribers to this channel received these reports as soon as they were published enabling them to make informed timely decisions.

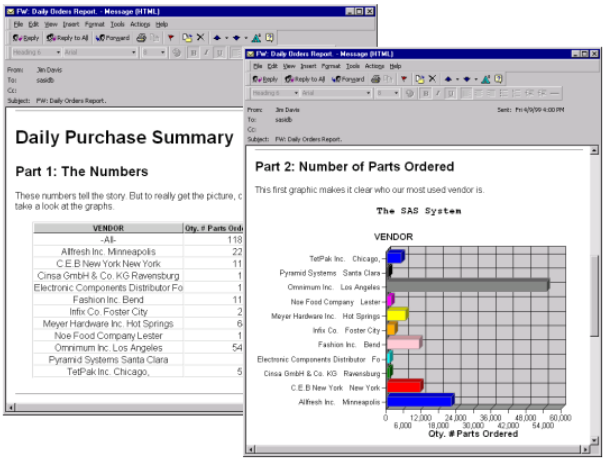

Figure 4 - Published HTML

To send the other types of file based content (data, MIME documents, etc.) with a published email you create a *SAS package file* and insert all the items to be delivered into your package. The package is delivered as an attachment to your email. You can think of the package file as a similar file construct to a zip file. The package file is simply a container for the items to be delivered to the end users along with descriptive information.

The text and HTML email are viewable as delivered, but how do I view the attached package files? The SAS Package Reader application, downloadable from SAS's web site, lets users view SAS Package Files (referred to as SPK files). Once installed on a users computer the user simply double clicks the attached SPK file and the SPK Reader application opens displaying the contents of the package. The user may then browse the items in the package. Figure 5 is a sample displaying the SPK Reader application.

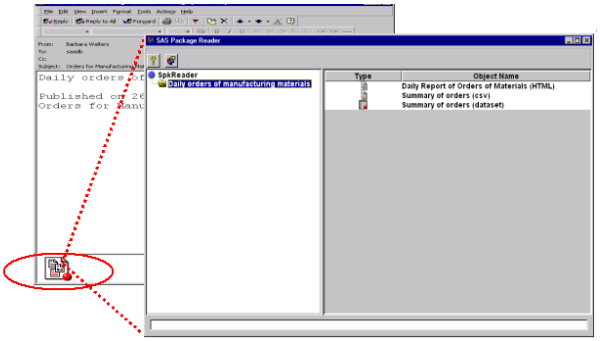

Figure 5 - SAS Package Reader Application

What is a channel? In the context of SAS Integration Technologies Publishing Framework it is a construct used to track the users, queues, or archives for published content to be sent. In Figure 6, a report is published to the Executive Information channel and in this example it only has one subscriber. The other channel publishes both data and a report to the Business Reports channel. Since two people are subscribed to the channel they both receive the data and the report. This channel construct frees you from having to keep the list of information consumers in your application. Instead you just publish to one location, the Business Channel in this example, and the publish framework takes care of publishing to the list of subscribers. Ten more subscribers could be added tomorrow and you would not have to change a thing!

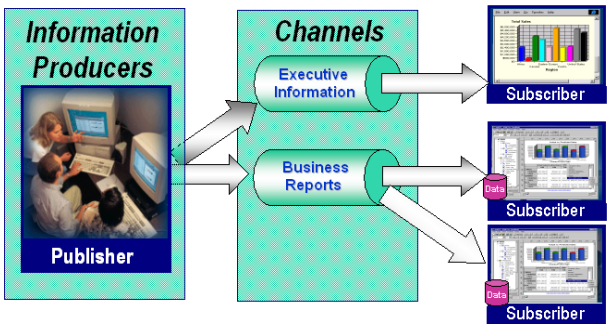

Figure 6 - Publishing to Channels

Why would I want to publish to a message queue? Publishing to a message queue gives you the capability of publishing to an application or process that is monitoring the queue, whether it is a SAS application or other applications using asynchronous communications. This allows you to automate processes or applications in response to messages published from other applications or users.

For example, you could have a data warehouse with distributed datamarts in a wide area network. When the main warehouse update is completed you want the distributed datamarts to automatically update. One way to implement this is to update your warehouse processes that refresh the warehouse by adding a section that upon successful completion publishes a message to a notification channel. The message queues that are subscribed to the channel subsequently receive the message on their queue and the corresponding jobs start the datamart update processes on each server. This combination of the Publishing Framework and the support for Messaging Software is very powerful. In this scenario in order to add another distributed datamart to this configuration that automatically gets updated. All I have to do is replicate the environments, setup a queue for the datamart server, subscribe it to the notification channel, and it will start receiving the update messages so it will know when to refresh its datamart.

SAS Publish Window

One way to publish information that has been discussed thus far is by writing SAS programs to generate the content or packages to publish. Another way to publish information is using the new Publish command available with the desktop interface for SAS. By typing "`publish`" on the SAS command line, in a Display Manager Session, the SAS Publish Window interface will start. Using this interface you can easily point and click to select the items to publish, where to publish the items, and how you would like to publish them. When you have completed making your selections pressing the OK button will run the job to immediately publish your selections.

Publishing from your Warehouse

Seeing the wide range of opportunities that the Publishing Framework can provide to the SAS Data Warehousing solution SAS has enhanced the SAS Warehouse Administrator application with support for the Publish Framework. Four new *addins* have been added to support the publish framework.

- *Publisher: Create Channel* – creates a channel associated with the current item

- *Publisher: Define Package to Publish* – uses the SAS Publish Window with the current item as the default item to publish to define a package to publish
- *Publisher: Show Subscribers* – queries the channel associated with the current item to display the list of subscribers.
- *IT Administrator:* Link to launch the Integration Technologies Administrator application directly.

These addins simplify the process for creating the channel where the item(s) from your warehouse jobs will be published. They help you define the items that will be published, save this information to the Warehouse Metadata, and provide the logic to generate the SAS program to publish the information.
The Show Subscribers addin is a very useful tool when maintaining your warehouse. When you are at the point of maintaining your warehouse, somehow you have to decide which job to fix first. This is where the Show Subscribers addin can help. Use it to query the channel information and using the list of subscribers for each item you can make a more informed decision on which item to fix. If you see the CEO is subscribed to one of the reports being generated by the warehouse processes, then that is the one to consider fixing first.

Integration Technologies Administrator
The Publishing Framework maintains the information on Channels, Subscribers, SAS User ID's, and Archive definitions in a LDAP server. The Integration Technologies Administrator, a.k.a. IT Admin, application is provided for you to maintain all of this information being stored in the LDAP server. This application lets you create SAS Users, create SAS Channels, associate Archive paths with Channels, and add users to your channels.

A common way to operate would be for standard channels to be created and the default set of users added to the channel using IT Admin. As users decide to add or drop from the channel then they can use the Subscription Manager application to do this on their own.

Subscription Manager
The SAS Subscription Manager is a Java applet that runs in a Web browser enabling you to subscribe to and unsubscribe from channels and to specify how information is delivered to you. Managing subscription services is like managing other resources that you may already be familiar with, such as e-mail alias lists, or Internet listservs.

During installation the Subscription Manager is configured to point to the LDAP server containing the channel and user information. The Subscription Manager prompts you for your user ID and password and based on these the initial window will be your subscribed channels. To subscribe yourself to another channel you press the subscribe button to display the list of available channels, pick the channel you would like to subscribe to, press OK and you will be subscribed to the channel.

The Subscription Manager also lets you maintain your personal information kept in the LDAP server on your email format and other information. Additionally it will let you search all channels available to you for key information to help you find the appropriate channel.

CONCLUSION

The Enterprise Integration Technologies (EIT) product bundle fully enables the SAS System to integrate with the variety of clients and systems in your enterprise. The SAS System now supports non-SAS clients and applications through both Distributed Object Technology support and through Messaging Software support. IT can take advantage of enterprise information stored in Directory Systems based on LDAP.

Utilizing these technologies the power, scalability, and rich feature set of the SAS System can be leveraged throughout your organization.

REFERENCES

SAS Integration Technologies Library
http://www.sas.com/rnd/itech/library/index.html

SAS Integration Technologies Overview White paper
Steve Jenisch
Manager, Distributed Technologies Development
SAS Inc.
http://www.sas.com/rnd/itech/papers/oviewSUGI24.html

SAS/IntrNet Product Web pages
http://www.sas.com/web

Microsoft MSMQ Web Site
http://www.microsoft.com/ntserver/appservice/default.asp

DCOM: A Business Overview
Microsoft Web Site Library
http://msdn.microsoft.com/library

Microsoft Component Services
Microsoft Web Site Library
http://msdn.microsoft.com/library

MQSeries: Message Oriented Middleware
IBM MQSeries Web Site
http://www.software.ibm.com/ts/mqseries/index.html

Directory Services Overview
Gary Williams
SAS Inc

Scherberger, Karen, Gartner Group, *Getting the message across*, IBM MQSeries Web Site: Library & Articles

ACKNOWLEDGMENTS

SAS Staff who contributed to the completion of this paper:

Terri Angeli
Steve Jenisch
Dan Tamburro
Don Hatcher
Ellen Kunkel
Aaron Hill
Renee Harper
Distributed Technologies Department – SAS Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Scott Vodicka
SAS Inc.
SAS Campus Dr.
Cary, NC 27513
Work Phone: 919-677-8000
Fax: 919-677-4444
Email: Scott.Vodicka@sas.com
Web: http://www.sas.com

Trademarks