Extending the Data Warehouse: An Introduction to OLE DB

S. David Riba, JADE Tech, Inc., Clearwater, FL

ABSTRACT

Now that you have your data warehoused, how do you share that information with others who need access to it? SAS® software provides a variety of tools to access and retrieve data from multidimensional databases (MDDBs). However, MDDBs are not accessible to non-SAS users in your organization. Until now. Starting with Version 7 of the SAS System, OLE DB can be used to access the information stored in SAS MDDBs and CFO Vision FDBs.

OLE DB can be used to access the information stored in ANY multidimensional database that is OLE DB compliant. Starting with the release of Windows 2000 Professional, OLE DB will be installed as part of the operating system. In Microsoft Office 2000, all member applications will be OLE DB consumers. The same is true for Lotus SmartSuite member applications. Thus, any OLE DB compliant consumer (such as Microsoft Excel, Lotus 1-2-3, Knosys ProClarity, and many others) will be able to directly query a SAS MDDB file using OLE DB on all major platforms, including Windows NT, UNIX, and mainframes.

This paper will provide an introduction to OLE DB. It will describe OLE DB and how it works. The SAS implementation of OLE DB will be demonstrated, along with other OLE DB consumers such as Microsoft Excel and Lotus 1-2-3. The paper is intended as a beginning paper for the advanced user.

INTRODUCTION

In 1995, I first presented *ODBC: Windows to the Outside World*, an introduction to the topic of Open DataBase Connectivity and how to use it effectively. Since then, ODBC has become a useful tool for many SAS sites for cross application access to data.

However, ODBC has its limitations. To address these limitations, and to provide a more flexible means of access to many different types of data, Microsoft proposed a new standard for data access technology, OLE DB.

At SUGI 24, Thomas Cox presented *What's Up with OLE DB?*, which provides an excellent overview of OLE DB architecture and concepts. This paper will attempt to provide some additional background on OLE DB, and present some techniques for using OLE DB to access both SAS and non-SAS data.

WHAT IS WRONG WITH ODBC?

ODBC was originally proposed by Microsoft in the early 1990's to solve the problem of accessing relational and tabular data on the Microsoft Windows® platform. It subsequently was ported to other platforms and is now a universal method to share data between non-compatible applications on a wide variety of hardware platforms.

The ODBC architecture consisted of four tiers:

- ODBC Driver Manager
- ODBC Drivers
- Applications
- Data Sources defined to ODBC

The ODBC Driver Manager is typically supplied by the operating system vendor (e.g. Microsoft).

ODBC Drivers are typically supplied by the vendor of a specific type of data to be accessed with ODBC (e.g. Oracle, SAS, Lotus, etc.).

Data Sources are defined by the end user to identify a specific type of data to be read or written.

However, there are several limitations that affect the use of ODBC. These limitations include:

- ODBC uses SQL as the common language
- ODBC Drivers are written to different Levels
- Deployment issues

Since ODBC is based on SQL as the common language for communicating between different types of data, the structure of SQL limits the types of data that can be accessed. SQL is best at accessing relational and tabular data. SQL can not be used effectively to access hierarchical, multi-dimensional (MDDB), and free-form data. In addition, there are many different interpretations of the SQL language, with differing levels of conformance to the SQL specifications. This can cause problems when trying to construct a SQL query for ODBC.

ODBC Drivers are individual DLLs written by different software vendors to support their specific data structures. However, the ODBC standard specifies several different levels of ODBC compliance, Core, Level 1, and Level 2. Each ODBC driver is written to one of these compliance levels based on the features that they support.

Deployment of ODBC has been a problem at many sites. In order for ODBC to function properly, the server, operating system, network transport, client software, and drivers must all be compatible. If this does not happen, troubleshooting ODBC problems can be extremely difficult.

WHAT IS OLE DB?

In the Summer of 1996, Microsoft published the first specifications for OLE DB (**O**bject **L**inking and **E**mbedding for **D**ata**B**ases). OLE DB was intended to overcome the limitations that had become apparent in ODBC.

According to Microsoft:

> OLE DB is an open specification designed to build on the success of ODBC by providing an open standard for accessing all kinds of data throughout the enterprise. OLE DB is a core technology supporting universal data access. Whereas ODBC was created to access relational databases, OLE DB is designed for the relational and nonrelational information sources, such as mail stores, text and graphical data for the Web, directory services, and IMS and VSAM data stored in the mainframe. OLE DB components consist of data providers, which expose data; data consumers, which use data; and service components, which process and transport data (for example, query processors and cursor engines). These components are designed to integrate smoothly to help OLE DB component vendors quickly bring high-quality OLE DB components to market. OLE DB includes a bridge to ODBC to enable continued support for the broad range of ODBC relational database drivers available today.

OLE DB is intended to be an open standard capable of providing access to any type of data. As such, OLE DB will be useful for:

- Relational Data
- MultiDimensional Data (MDDB)
- Non-Relational Data
  - E-Mail
  - Web Data
  - Directory Services
  - IMS and VSAM data

Since OLE DB is an object oriented interface to data, it is intended for use in thin client (web based) environments. With OLE DB many different types of data can be accessed in a common manner using web browsers, database software, stand alone applications, etc.

IS ODBC OBSOLETE?

In many respects, OLE DB can be considered a superset to ODBC. OLE DB includes all of the functionality of ODBC, adds additional functionality, and is easier to use. In fact, OLE DB ships with an ODBC Provider (more on this later). Thus, OLE DB can perform all of the functionality previously performed by ODBC, with the additional capabilities built into OLE DB.

However, OLE DB is not intended to replace ODBC. The ODBC technology has matured and is widely used. ODBC is still the best choice when accessing standard relational databases from a non-OLE environment. OLE DB is still an evolving standard. It is important to realize that Microsoft is fully integrating OLE DB into its product line (operating systems and Office suite) as their solution for universal data access. In addition, other vendors such as SAS Institute are adopting OLE DB as a standard for external data access.

In deciding whether to use ODBC or OLE DB, here are some recommendations:

- Use ODBC to access standard relational databases from a non-OLE environment
- Use OLE DB for non-SQL data
- Use OLE DB in an OLE environment
- OLE DB is the only option for building interoperable database components (COM)

UNDERSTANDING OLE DB

Technologically, there is a fundamental difference in the architectures of ODBC and OLE DB.

- ODBC is a procedural based specification
- OLE DB is a component based specification
- ODBC is designed to provide access primarily to relational data using SQL in a multiplatform environment.
- OLE DB is designed to provide access to all types of data in an OLE **C**omponent **O**bject **M**odel (COM) environment.

OLE DB includes the Structured Query Language (SQL) functionality defined in ODBC, but it also defines interfaces suitable for gaining access to data other than SQL data. *OLE DB is intended to be an open standard capable of providing access to any type of data*. In addition, OLE DB for OLAP (**O**n**L**ine **A**nalytical **P**rocessing) is intended to extend the features of OLE DB to OLAP data.

Microsoft intended OLE DB to be the foundation of their Universal Data Access strategy. To accomplish this, OLE DB is tightly coupled with ADO (**A**ctiveX **D**ata **O**bjects). ADO is the application level interface (API) that provides the Component Object Model (COM) objects. Specifically, OLE DB is intended to be an object oriented solution for data access. COM (**C**omponent **O**bject **M**odels) is Microsoft's solution for object oriented programming. Virtually all of the Windows oriented environments (Visual Basic, Visual C++, Delphi, PowerBuilder, etc.) all support COM components. While Visual Basic originally created VBX files, COM files have the extension of OCX (OLE Custom Control). OLE DB uses ADO (ActiveX Data Objects) which are COM compliant objects that are designed for data access.

Thus, OLE DB is the interface to various types of data, which uses ADO object oriented components to access the data. This is a very superficial view, and the reader is encouraged to research ADO and COM in more detail for a better understanding of how they interrelate.

OLE DB for DATA MINING

In February, 2000, Microsoft updated a draft of a specification for OLE DB for Data Mining. The specification is currently at Version 0.9. In cooperation with over 40 vendors in the business intelligence field, the OLE DB for DM specification defines a common interface for data mining applications. The goal of this specification is to provide an industry standard for data mining so that different data mining algorithms from various data mining vendors can be easily plugged into user applications.

OLE DB for DM specifies the API between data mining consumers and data mining providers. The specification does not add any new OLE DB interfaces. Instead, the specification defines a simple query language similar to SQL syntax and specialized schema rowsets so consumer applications can communicate with data mining providers.

Like OLE DB, this area is still evolving and maturing. The OLE DB for DM specification is currently in public beta. However, the potential benefits from OLE DB for Data Mining are significant.

PROVIDERS AND CONSUMERS

OLE DB introduces two new terms to the computer lexicon -- "provider" and "consumer". These terms refer to the application software and how they interact with OLE DB.

- Providers are applications that implement OLE DB interfaces
- Consumers are applications that request an OLE DB interface

The OLE DB specifications define four categories:

- Data Providers
- Data Consumers
- Data Service Providers
- Business Component Developers

An OLE DB Provider implements OLE DB interfaces. Microsoft Windows 2000, Windows NT, and the SAS Local Data Provider are all OLE DB providers. Other OLE DB provider applications include Microsoft SQL Server, DB2, AS400, Oracle, Red Brick, Sybase, Lotus Notes, and Microsoft Exchange. OLE DB providers allow consumers to access data in a uniform way through a known set of documented interfaces. In a sense, an OLE DB provider is similar to an ODBC driver that provides a uniform mechanism for accessing relational data. OLE DB providers not only provide a mechanism for relational data but also for nonrelational types of data.

OLE DB Consumers use, or consume, OLE DB interfaces. Any application that connects to a database using OLE DB is a consumer application. Many applications are already OLE DB consumers. Microsoft Office 2000, Lotus Notes, and SAS/Access to OLE DB are all examples of OLE DB consumers.

The other two categories, Data Service Providers and Business Component Developers, are concerned with building and deploying OLE DB components and will not be discussed in this paper.

THE ODBC PROVIDER FOR OLE DB

OLE DB ships with an ODBC Provider. The ODBC Provider maps OLE DB interfaces to ODBC APIs. With the ODBC Provider, OLE DB consumers can connect to a database server through the existing ODBC drivers as follows:

- Consumers call an OLE DB interface on the ODBC Provider
- The ODBC Provider invokes corresponding ODBC APIs
- The ODBC Provider sends the requests to an ODBC driver
- Responses are returned to the OLE DB consumer

Because the ODBC Provider allows OLE DB consumers to use existing ODBC drivers, there may be some performance concerns about the additional layer of the ODBC Provider on top of the existing ODBC driver manager. The design goal of the ODBC Provider is to implement all the functionality of the ODBC driver manager. Thus, the ODBC Driver Manger is not needed. However, the ODBC Provider still requires the ODBC Driver Manager to support connection pooling with ODBC applications.

CONSUMER INTERFACES TO OLE DB

Since OLE DB is intended as an object oriented interface to many different types of data, consumer applications must communicate with the OLE DB objects. This is done by means of program code written in one of the object oriented programming languages. Visual Basic, Visual C++, VB Script, and Java are among the languages that support OLE DB. The syntax of each of these languages needed to communicate with the OLE DB objects are beyond the scope of this paper. Any application that supports OLE DB as a consumer has a scripting language that can be used to communicate with the OLE DB objects.

Appendix A to this paper is a sample of HTML which uses VB Script to read a SAS dataset from within a web page. The VB Script invokes the SAS Local Data Provider to open and read a SAS dataset without the need to open SAS software in the background. This VB Script could be generalized to use any OLE DB provider as a variable, instead of hard coding the value in the script. The key here is that an HTML page, irrespective of operating system or location, can now access and retrieve data across a thin client.

SAS INTERFACES TO OLE DB

As part of the Nashville Project, SAS Institute is in the process of implementing support for OLE DB. The current initiatives are:

- SAS/Access to OLE DB
- OLE DB for OLAP
- SAS Local Data Provider
- SAS/Share Data Provider
- SAS IOM Data Provider
- SAS SPD Server Data Provider

The status of each of these initiatives is listed as of April, 2000.

SAS/Access to OLE DB is production effective with Version 8.0. OLE DB for OLAP is a pass-through component of SAS/Access to OLE DB. It is also production in Version 8.0. SAS/MDDB server and CFO Vision Release 2.2 both have OLE DB for OLAP compliant interfaces.

The SAS Local Data Provider was formerly named the SAS OLE DB BaseProvider. It is experimental in Version 8.0. The role of the SAS Local Data

Provider is to provide access to local SAS datasets from non-SAS applications on the same machine.

The SAS/Share Data Provider was formerly named the SAS/Share OLE DB Provider. It is experimental in Version 8.0. It is intended to provide access to non-local SAS datasets managed by a SAS/Share server. It can surface any data that SAS can process.

The SAS IOM Server OLE DB Provider has been renamed to the SAS IOM Data Provider. It is currently considered Developers Release. It will be production in the Version 8.0 Maintenance Release. SAS Integration Technologies use the IOM Server to provide access to OLE DB data managed by the Integration Technologies object server.

The SAS SPD Server has an ODBC interface and would be accessible using the ODBC Provider for OLE DB . It is due out later this year.

SAS/ACCESS TO OLE DB

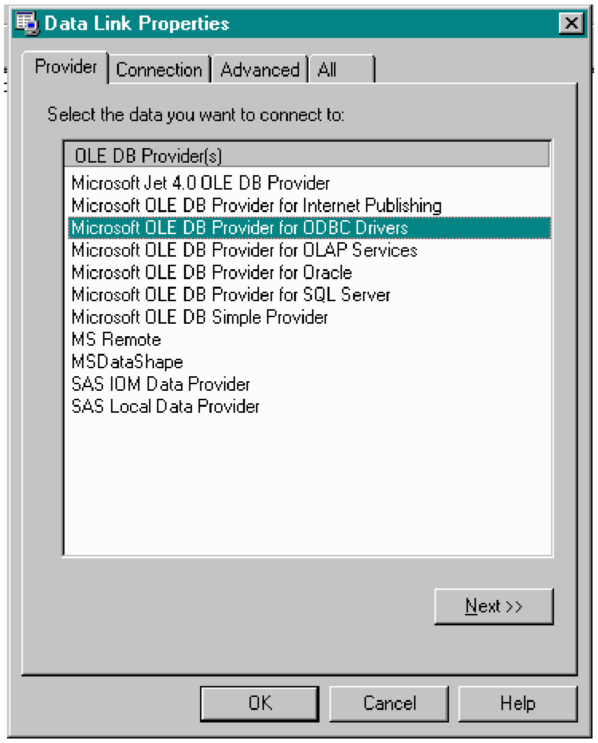
SAS/Access to OLE DB is a separately licensed product from SAS Institute. Like the other SAS/Access products, SAS/Access to OLE DB provides an interface between the SAS System and OLE DB compliant data. The programming syntax is similar to that already used by the other SAS/Access products.

With SAS/Access to OLE DB, SAS has introduced a library engine, the OLEDB engine. Libraries using the OLEDB engine can be defined either with SAS program statements or with the new Library Wizard.

Defining a library containing OLE DB compliant data, can be as simple as:

LIBNAME *libref* OLEDB ;

SAS will then prompt the user to define the rest of the information necessary. A popup window is presented to define the properties of the data link. The popup window contains three tabs where the Properties, Connection, and Advanced features can be set. The fourth tab, All, displays the initialization properties that have been defined for the data.



Select the data you want to connect to:

OLE DB Provider(s)
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MS Remote
MSDataShape
SAS IOM Data Provider
SAS Local Data Provider

Alternatively, the user can complete all of the required data, such as:

LIBNAME *libref* OLEDB
PROVIDER = *'oledb_provider'*
PROPERTIES =
("userid" = *userid*
password = *xxxxxx*
"data source" = *dsname*) ;

The details of configuring an OLE DB connection are beyond the scope of this paper. However, the SAS Version 8 OnLine Docs contained a very complete description of the specific process required to define OLE DB data to SAS.

Since SAS/Access to OLE DB functions the same as all of the other SAS/Access products, the same SQL syntax is valid for OLE DB data. The SQL Procedure Pass Through Facility is common across all of the SAS/Access product line.

For example, a Microsoft Access table can be opened in SAS using the Microsoft Jet OLE DB Engine as follows:

```
   PROC SQL feedback ;
      CONNECT to OLEDB (
        PROVIDER = “Microsoft.Jet.OLEDB.4.0”
        PROPERTIES =
           (“data source” = “ ds_name ” )) ;
      CREATE TABLE sugi as
       SELECT * from CONNECTION TO oledb
          (select * from table_name ) ;
  QUIT ;
```

Each OLE DB Provider must be defined to the system. The Data Link Properties popup will identify each of the Providers available. However, to use the Provider with the SQL Pass Through Facility, the proper Provider ID must be used.

CONCLUSION

OLE DB is an emerging standard which will provide a common means of accessing many different types of information. Through the use of object oriented programming languages such as Visual Basic, Visual C++, Java, or SAS software, OLE DB data objects can be accessed from almost any application. OLE DB is particularly useful with thin client applications, such as web browsers, and with complex data such as multi-dimensional databases. As database vendors implement OLE DB support in their products, the inherent capabilities of OLE DB will become apparent to anyone who needs the ability to access and exchange information between many different types of applications.

TRADEMARK INFORMATION

ABOUT JADE Tech, Inc.

For more information about JADE Tech, Inc. visit our website at.

HTTP://WWW.JADETEK.COM

sas® SAS Alliance Quality Partner

AUTHOR

The author may be contacted at:

S. David Riba

JADE Tech, Inc.

P O Box 4517
Clearwater, FL 33758
(727) 726-6099
INTERNET: DAVE@JADETEK.COM

REFERENCES

Cox, Thomas W. (1999). What’s Up with OLE DB? *Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference*. SAS Institute, Inc., Cary, NC. Pp. 808-817

Maata, B., Alam, M., Maartens, G., McRoberts, R., Pelie, C., and Sharma, P. (1998). *Fast Path to AS/400 Client/Server Using AS/400 OleDB Support*. IBM Corp.

Microsoft Corp, Microsoft OLE DB online library

Microsoft Corp, (2000) *OLE DB for Data Mining DRAFT Specification, Version 0.9*.

SAS Institute, Inc. *SAS OnLine Docs*

AUTHOR BIO

S. David Riba is CEO of JADE Tech, Inc., a SAS Institute Quality Partner who specializes entirely in applications development, consulting and training in the SAS System. Dave has presented papers and assisted in various capacities at SUGI, SESUG, NESUG, MWSUG, SCSUG, and PharmaSUG.

His major areas of interest are efficient programming techniques and applications development using the SAS System. His SAS software product specialties are SAS/AF and FRAME technology, SAS/EIS, SAS/IntrNet, AppDev Studio, CFO/Vision, and the SAS Collaborative Server. Dave is a SAS Certified Professional.

Dave is currently the Co-Chair of SSU 2001, the combined SouthEast and SouthCentral SAS Users Group conference to be held in New Orleans in August, 2001.

APPENDIX A

Sample HTML to access SAS data using OLE DB

```
<html>
<head>
<title>OLE DB Using ADO Example</title>

<table bgcolor="#333333" cellpadding=1 border=0 width="100%">
<tr><td><table bgcolor="#9999CC" cellpadding=1 border=0 width="100%">
<tr><td align="center"><h1><font color="white">View SAS Data</font></h1>
</td></tr></table>
</td></tr></table><br>

<script language="vbscript">

'ADO Constants
Const adCmdTableDirect = 512
Const adOpenStatic = 3
Const adLockOptimistic = 3

' Error handling routine
Sub GetErrors( conn )

If conn.Errors.Count = 0 Then Exit Sub
i = 1
msg = ""
For Each adoErr in conn.Errors
    msg = msg & "ADO Error " & i & ":" & vbNewLine
    msg = msg & "Source: " & adoErr.Source & vbNewLine
    msg = msg & "Number: " & adoErr.Number & " (0x" & Hex(adoErr.Number) & ")" & vbNewLine
    msg = msg & "NativeError: " & adoErr.NativeError & vbNewLine
    msg = msg & "Description: " & adoErr.Description & vbNewLine
    i = i + 1
Next
MsgBox msg, vbExclamation
conn.Errors.Clear

End Sub

' Format a recordset as an HTML table
Sub DisplayRecordset( rs )

document.write "<table border=1 cellpadding=3 align=cenTer><tr>"
for each column in rs.Fields
    document.write "<th>" & column.Name & "</th>"
next
document.writeln "</tr>"
do until rs.Eof
    for each column in rs.Fields
        document.write "<td"
        val = column.Value
        if IsNull( val ) then
            val = "-null-"
        end if
        if Trim(  val ) = "" then
            val = "&nbsp;"
        end if
        if IsNumeric( val ) then
            document.write " align=center"
        end if
        document.write ">" & val & "</td>"
    next
    document.writeln "</tr>"
    rs.MoveNext
loop
document.write "</table>"

End Sub

' Main routine
On Error Resume Next

Sub OpenTable_OnClick

On Error Resume Next
strDirectory = document.form1.sasDirectory.value
strDataset = document.form1.sasDataset.value
strConnect = "Provider=sas.LocalProvider.1;Data Source=" & strDirectory

Set rs = CreateObject( "ADODB.Recordset" )
rs.Open strDataset, strConnect, adOpenStatic, adLockOptimistic, adCmdTableDirect
Call GetErrors( conn )
Call DisplayRecordset( rs )
rs.Close
rs = Nothing

End Sub
</script>
</head>
<body>
<form name="form1">
    <table border=1 cellpadding=3>
        <tr>
            <td>SAS Directory:</td>
            <td><input type="text" name="sasDirectory" size=50></td>
        </tr>
        <tr>
            <td>SAS Dataset Name:</td>
            <td><input type="text" name="sasDataset" size=50></td>
        </tr>
    </table>
    <table border=0 cellpadding=3>
        <tr>
            <td><input type="button" name="OpenTable" value="Open Table"></td>
            <td><input type="reset" value="Reset Form"></td>
        </tr>
    </table>
</form>
</body>
</html>
```