# ODBO, BAPI and XMLA – It's All MDX to Me



## Applies To:

ODBO, OLAP BAPI and XMLA interfaces for connecting to SAP BW.

## Summary

Over the years, there has been a lot of discussion and confusion about what interface to use when connecting to SAP BW. In the days of BW 1.x, things were easy because OLE DB for OLAP (ODBO) was the primary connectivity interface. When BW 2.x came out, SAP also added the OLAP BAPI interface, which some thought was better. Then in BW 3.x, SAP added the XML for Analysis (XMLA) interface and people became confused. In this paper, we will review the three (3) interfaces available for third-party tools to access SAP BW data. We will also review some commercially available applications, how they access BW data, and their performance on some simple queries. Finally, we will provide some recommendations and guidance on when it is appropriate to use OLE DB for OLAP, OLAP BAPI, or XML for Analysis within your own programs.

By: Amyn Rajan, George Chow and Darryl Eckstein

Company: Simba Technologies Inc.

Date: 30 March 2006

# ODBO, BAPI and XMLA – It's All MDX to Me

## Open Analysis Interfaces for BW

There are three different interfaces that third-party client tools can use to query data within SAP BW. The interfaces are:

- OLE DB for OLAP (ODBO)

- OLAP BAPI (Business Application Programming Interface)

- XML for Analysis (XMLA)

Although the interfaces are different, the underlying command language used to retrieve data, MDX, is the same between all three interfaces. Similarly, the ODBO metadata schema format is also used by all three interfaces. The differences between the interfaces are entirely based on the programming model used. The following table summarizes the differences.

| Interface | Query Language | Call Interface | Platform |
|---|---|---|---|
| ODBO | MDX | COM (Component Object Model) protocol | Microsoft Windows platforms |
| OLAP BAPI | MDX | RFC (Remote Function Call) protocol | SAP support platforms[1] |
| XMLA | MDX | XML over HTTP/SOAP | All platforms |

The following diagram better illustrates how these interfaces fit within the BW server. Note that this diagram comes directly from SAP and can be found at http://help.sap.com/saphelp_nw04/helpdata/en/d9/ed8c3c59021315e10000000a114084/content.htm.



---
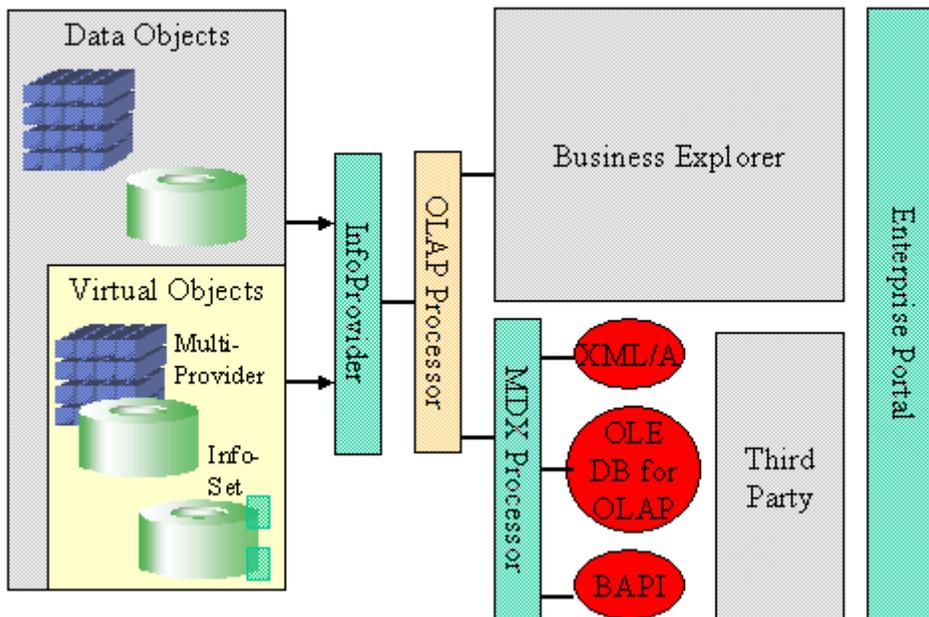
[1] See http://service.sap.com/platforms or http://service.sap.com/pam for a list of supported platforms.

It is important to review the above diagram in some detail. Note that ODBO, XMLA and BAPI go to the same MDX Processor. Therefore, if you factor out any differences in the three interfaces, everything boils down to the MDX call, and all three interfaces access the same MDX processor, so at that point there should be no difference.

Another important point to notice from the above diagram is that Business Explorer (BEx) does not go through any of the standard interfaces and also does not go through the MDX Processor. This is important because people often compare other products against BEx, which often results in an apples to oranges comparison.

## What is OLE DB for OLAP (ODBO)?

ODBO is an extension of Microsoft's OLE DB standard that is designed for connecting to multi-dimensional data sources. It defines an API and a model for interaction between a "consumer" (application) and a "provider" (driver).

More information on ODBO can be found here: http://www.xmlforanalysis.com/odbo.htm or on the Microsoft Developer Network here: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/oledbpart3_ole_db_for_olap.asp.

## What is OLAP BAPI?

OLAP BAPIs (Business Application Programming Interfaces) are SAP defined objects that are implemented as RFC-enabled function modules. The methods defined are similar to ODBO and XMLA, as the metadata concepts and MDX command language are shared by all three interfaces. More information on OLAP BAPI is available within the SAP help portal here: http://help.sap.com/saphelp_nw04s/helpdata/en/e3/e60138fede083de10000009b38f8cf/frameset.htm.

## What is XML for Analysis (XMLA)?

XMLA is a web service standard designed for connecting to multi-dimensional data sources. Similar to ODBO, XMLA defines an XML-based API and interaction model between a "consumer" (application) and a "provider" (web service). Most of the metadata objects and schema rowsets defined within ODBO are also defined within XMLA. XMLA can be thought of as the web service evolution of ODBO.

More information on XML for Analysis can be found here: http://www.xmlforanalysis.com or on the Microsoft Developer Network here: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxmlspec/html/xmlanalysis.asp.

## What is Multi-dimensional eXpressions (MDX)?

MDX is a query language for multi-dimensional data sources. It is defined as part of the ODBO specification, and it has also been adopted as the OLAP query language for XMLA providers. MDX is to OLAP as SQL is to relational databases.

More information on MDX can be found here: http://www.xmlforanalysis.com/mdx.htm or on the Microsoft Developer Network here: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/olapchapter_4_multidimensional_expressions.asp.

## Application Overview

We will examine a number of third-party applications with different interfaces and how they interact with SAP BW. The following table lists the applications and how they interface with SAP BW. Note that BEx Analyzer is included for comparison purposes. It uses an internally defined API for communicating with SAP BW.

| Application | Interface |
|---|---|
| Microsoft Excel 2000 | ODBO |
| Microsoft Excel 2003 | ODBO |
| MicroStrategy 8 | OLAP BAPI |
| Microsoft Reporting Services 8 | XMLA |
| Hyperion Intelligence 8 | ODBO |
| SAP BEx Analyzer (SAP GUI 6.40) | Proprietary |

The performance of applications that use one of these interfaces depends on how efficient the generated MDX is and how the application processes the results of the MDX. If an application generates inefficient MDX, then changing to another interface will not improve the performance. Similarly, if the processing that the application performs after receiving the results is inefficient from the interface, then changing interfaces cannot improve performance.

## Application Tests

To test the performance of the different APIs we'll perform two simple tests using some of the demo business content. The `0D_DX_C01` cube contains demo sales data from over one million different sales transactions. For the first test, we'll retrieve all of the customers and key figures. For the second test, we'll retrieve all of the

products bought by all customers through the Internet distribution channel in 2001. The test details are shown below. For each test, the measures were placed on the column axis. For the last two tests, `0CALQUARTER` was placed on the column axis, depending on the application. Some applications, such as Reporting Services, do not support the placement of non-measure dimensions on the column axis. Regardless, the total number of cells retrieved is the same.

| Test | Data Requested | Filters | # of Rows | # of Columns | # of Cells |
|------|----------------|---------|-----------|--------------|------------|
| Test 1 | `0D_CUSTOMER, Measures` | `<none>` | 5089 | 2 | 10178 |
| Test 2 | `0D_CUSTOMER, 0D_PH2, Measures` | `0D_CHANNEL = Internet and 0CALYEAR=2001` | 11138 | 2 | 22276 |
| Test 3 | `0D_CUSTOMER, 0CALQUARTER, Measures` | `<none>` | 5089 | 16 | 81424 |
| Test 4 | `0D_CUSTOMER, 0DPH2, 0CALQUARTER, Measures` | `0D_CHANNEL = Internet and 0CALYEAR=2001` | 11138 | 8 | 89104 |

Each test was run against the following application at least three times with the fastest time reported below. The time in seconds for all tests is shown below.

| Application | Test 1 Query Time | Test 2 Query Time | Test 3 Query Time | Test 4 Query Time |
|-------------|-------------------|-------------------|-------------------|-------------------|
| Excel 2000 (ODBO) | 5 | 38 | 40 | <failure> |
| Excel 2003 (ODBO) | 4 | 11 | 21 | 50 |
| MicroStrategy Desktop (OLAP BAPI) | 10 | 21 | 65 | 75 |

| | | | | |
|---|---|---|---|---|
| Reporting Services 2005 (Designer) (XMLA) | 12 | 38 | 107 | 140 |
| Reporting Services 2005 (Web Server) (XMLA) | 16 | 37 | 110 | 135 |
| Hyperion Intelligence (ODBO) | 5 | 9 | 20 | 28 |
| BEx Analyzer | 6 | 16 | 25 | 54 |

From the results, you can see that the times for Test 1 are relatively close except for MicroStrategy and Reporting Services. In contrast, the times for the other tests are more interesting. Similarly, when looking at Test 3 and 4, the results for applications using ODBO demonstrates our point very clearly. Even though Hyperion Intelligence turned in the fastest time using ODBO, other applications that also used ODBO trailed it quite consistently.

How can the query time for all of these applications vary for queries that display similar results? There are two reasons for this difference. The first is the amount of processing the application performs after it has retrieved the data from the BW interface. The second is the MDX the application generates to retrieve data. Although each application displays similar results, the MDX generated is different in each case. How that MDX performs is covered in the next section.

## Interface Neutral Tests

Some interface neutral method is required to test the performance of the MDX that each application generates for each test. One candidate is the MDXTEST transaction available within BW. The transaction runs within SAPGUI and interfaces directly with the MDX processor, so none of the data access interface is accessed. The transaction still has some overhead for processing and displaying the results in a grid within SAPGUI; but every MDX query will have this overhead, so it should be relatively the same for each query. Finally, BEx Analyzer does not use MDX, so times cannot be listed here.

| Application | Test 1 | Test 2 | Test 3 | Test 4 |
|---|---|---|---|---|
| Excel 2000 (ODBO) | 2 | 11 | 13 | <failure> |
| Excel 2003 (ODBO) | 2 | 6 | 10 | 14 |
| MicroStrategy Desktop (OLAP | 2 | 4 | 11 | 13 |

| BAPI) | | | | |
|---|---|---|---|---|
| Reporting Services 2003 (XMLA) | 2 | 4 | 13 | 14 |
| Hyperion Intelligence (ODBO) | 2 | 4 | 9 | 13 |

From the timings, a few interesting results are immediately noticeable. Firstly, the MDX that Excel 2000 generates shows how inefficient MDX can impact application performance. Note the MDX from Excel 2000 and Excel 2003 is noticeably slower for Test 2 than the MDX from the other applications. One reason is that both MDX statements use the `DrillDownLevel` MDX statement to retrieve the members from the `0D_CUSTOMER` and `0D_PH2` dimensions instead of simply using `<DIMENSION>.members` or `<LEVEL>.members`. Excel 2000's MDX is even slower because it does not contain the `NON EMPTY` clause. Excel 2000 post filters the empty items from the results retrieved instead of letting SAP BW perform the filtering. The lack of `NON EMPTY` results in a noticeable performance impact. For example, Test 2 returns over 71,000 rows instead of the approximate 10,000 non empty rows. Furthermore, Test 4 fails for Excel 2000 because the result of the MDX statement is too large for the BW server to calculate without running out of memory.

We now need to look at the results more critically. With the exception of Excel 2000, the above interface neutral test times are roughly equivalent for the MDX statements generated by every application. Minor differences between the MDX are magnified with the larger data volumes in Tests 3 and 4. However, the application test times from the previous section vary much more widely. How do we reconcile these two differing trends?

It is understandable why XMLA consistently trails either OLAP BAPI or ODBO. However, there are applications that use ODBO that are both faster and slower than applications that use OLAP BAPI. Also, the two applications that use ODBO (Hyperion Intelligence, and Excel 2003) were all noticeably faster than MicroStrategy (which uses OLAP BAPI). Furthermore, this result is with Excel 2003 using a less efficient MDX query.

We are forced to conclude that the data access interface is not the deciding factor of the application test time. While the MDX generated is important, what and how the application handles the data retrieved is far more important in determining the overall query performance.

## Which Interface?

The previous section shows that when you are developing an application to access data from BW, every interface (ODBO, XMLA or OLAP BAPI) is equal in terms of query performance because all three interfaces access the same MDX Processor. Other considerations are more important, such as deployment scenarios, platform targets and programming language restrictions. For maintenance reasons, you should not try to shoehorn an interface into an application where it does not make technical sense.

## Use ODBO if…

ODBO should be used if you're developing a COM-based app in C++, or using ADOMD from Visual Basic, ASP or C++ on the Windows platform. If you already have an existing ADOMD or COM application and you want to access BW data, then using the ODBO provider is the simplest option available. Existing white papers explain an ODBO-based implementation in detail and provide simple examples showing you how to do this. They are available at the following locations: http://www.simba.com/Knowledge_Center/docs/Connecting_to_SAP_BW_from_ASP_Using_ADO_MD.pdf, and http://www.simba.com/Knowledge_Center/docs/Connecting_to_SAP_BW_Using_Visual_Basic_and_ADO_MD.pdf.

## Use OLAP BAPI if…

You require a cross-platform application that will only be used for SAP supported platforms. OLAP BAPI can be used in C/C++ applications with the RFC library, Java applications using the SAP Java Connector, and .NET applications using the SAP connector for Microsoft .NET. The SAP Java Connector and the SAP connector for Microsoft .NET both use the RFC library internally, providing another layer of indirection. As a result, for the fastest performance possible, the RFC library should be used directly from C or C++. Similarly, if your application is written in C/C++ and cannot use COM, then use the RFC library.

## Use XMLA if…

Your application is web service enabled, or you require a cross-platform application beyond any SAP supported platforms, such as mobile devices or legacy systems. The XMLA standard can be used to connect web service applications with SAP BW. Since XMLA is typically used over HTTP or HTTPS, applications can connect to disparate data sources over the Internet.

Additionally, if you are developing within a .NET application, you can use ADOMD.NET to connect to SAP BW via XMLA. ADOMD.NET is the .NET evolution of ADOMD. Since ADOMD.NET was developed specifically for XMLA and Analysis Services, some features (for example, data mining, KPIs, and parameters) will not work with SAP BW. However, retrieving data via the ADOMD.NET library is simpler than using the SAP connector for Microsoft .NET because ADOMD.NET provides an easier to use class structure and deployment option.

Note that from a performance standpoint, XMLA is slower than ODBO or OLAP BAPI because BW must convert the result of the MDX query into XML. With ODBO and OLAP BAPI, it can return the results directly without the conversion of data into strings.

Another advantage that XMLA has over ODBO and OLAP BAPI is easier deployment. XMLA applications do not require any SAP components for deployment. In contrast, both ODBO and OLAP BAPI require that SAPGUI be installed on each client machine. Furthermore, the license terms do not allow redistributing the SAP Java Connector or the SAP connector for Microsoft .NET to external customers.

## Conclusion

In this article, we investigated the differences between three different APIs to connect to SAP BW – OLE DB for OLAP (ODBO), OLAP BAPI, XML for Analysis (XMLA). We have found that all three APIs access the same MDX Processor in SAP BW, and as such, the differences between the APIs reside largely in their mechanism for access to SAP BW. We also reviewed the performance of third-party applications that access SAP BW cube data. From the test results, the data access interface that each application uses does not affect application performance. Ultimately, the strength or weakness of an application will be determined by two

factors: the efficiency of the MDX it generates when accessing SAP BW and how it processes the data that it receives from SAP BW. Finally, we reviewed some considerations when determining what interface to use in development.

## Author Bio

### About Simba Technologies

Simba Technologies Incorporated (http://www.simba.com) builds development tools that make it easier to connect disparate data analysis products to each other via standards such as, ODBC, JDBC, OLE DB, OLE DB for OLAP (ODBO), and XML for Analysis (XMLA). Independent software vendors that want to extend their proprietary architectures to include advanced analysis capabilities look to Simba for strategic data connectivity solutions. Customers use Simba to leverage and extend their proprietary data through high performance, robust, fully customized, standards-based data access solutions that bring out the strengths of their optimized data stores. Through standards-based tools, Simba solves complex connectivity challenges, enabling customers to focus on their core businesses.

### About the Authors

Amyn Rajan is President and CEO of Simba Technologies. Amyn has over 14 years experience in custom software development, and is responsible for driving Simba's success as a leader in data connectivity solutions.

George Chow is a Development Manager with Simba Technologies. George has over 14 years experience in software and technology, and is a specialist in SAP BW, database access and data connectivity solutions.

Darryl Eckstein is a Senior Computer Scientist with Simba Technologies. Darryl has over seven years experience in software development, and is a specialist in SAP BW and OLAP data access.

Prepared for SAP Developer Network
©2006 Simba Technologies Inc.

Simba Technologies Incorporated
1090 Homer Street, Suite 200
Vancouver, BC  Canada
V6B 2W9

Tel.    +1.604.633.0008

Fax.    +1.604.633.0004

Email.   solutions at simba.com

www.simba.com

**SAP** SAP DEVELOPER NETWORK

## Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.