XML for Analysis Specification

Version 1.1

Microsoft Corporation
Hyperion Solutions Corporation

Updated: 11/20/2002

Notice and Disclaimer

Permission to copy and display the XML for Analysis Specification, in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the XML for Analysis Specification, or portions thereof, that you make:

1. A link or URL to the XML for Analysis Specification at this location: http://xmla.org.

2. The copyright notice as follows:

© 2002 Microsoft Corporation. Portions © 2002 Hyperion Solutions Corporation. All rights reserved.

Microsoft Corporation and Hyperion Solutions Corporation (the "Authors") may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in this Notice and Disclaimer or a written license agreement from the Authors, the furnishing of this document does not give you any license to any patents, trademarks, copyrights, or other intellectual property. Implementation of this Specification may require patent licenses from one or more of the Authors. Anyone desiring to implement this Specification should inquire whether or not there is a need for a license from each of the Authors and whether such license is available before implementing this Specification.

THE XML FOR ANALYSIS SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE XML FOR ANALYSIS SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE XML FOR ANALYSIS SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the XML for Analysis Specification or its contents

Table of Contents

Deleted: 32
Deleted: 32
Deleted: 39
Deleted: 39
Deleted: 39
Deleted: 41
Deleted: 56
Deleted: 60
Deleted: 63

Deleted: 64
Deleted: 64
Deleted: 64
Deleted: 74
Deleted: 74
Deleted: 76
Deleted: 76
Deleted: 78
Deleted: 78
Deleted: 79
Deleted: 80
Deleted: 80
Deleted: 81
Deleted: 105

Executive Summary

XML for Analysis is a Simple Object Access Protocol (SOAP)-based XML API, designed specifically for standardizing the data access interaction between a client application and a data provider working over the Web.

Under traditional data access techniques, such as OLE DB and ODBC, a client component that is tightly coupled to the data provider server must be installed on the client machine in order for an application to be able to access data from a data provider. Tightly coupled client components can create dependencies on a specific hardware platform, a specific operating system, a specific interface model, a specific programming language, and a specific match between versions of client and server components.

The requirement to install client components and the dependencies associated with tightly coupled architectures are unsuitable for the loosely coupled, stateless, cross-platform, and language independent environment of the Internet. To provide reliable data access to Web applications the Internet, mobile devices, and cross-platform desktops need a standard methodology that does not require component downloads to the client.

Extensible Markup Language (XML) is generic and can be universally accessed. What if, instead of invoking the proprietary interface of a client component, you could call methods and transfer data through XML HTTP messages without any client component? What if the application developer could build client components without concern for tight coupling to a server component or application? What if an application, developed with any programming language and running on any platform, could access data from any place on the Web without having to plan for specific platform support or even a specific provider version? This specification answers these questions with XML for Analysis.

XML for Analysis advances the concepts of OLE DB by providing standardized universal data access to any standard data source residing over the Web without the need to deploy a client component that exposes COM interfaces. XML for Analysis is optimized for the Web by minimizing roundtrips to the server and targeting stateless client requests to maximize the scalability and robustness of a data source.

This specification defines two methods, **Discover** and **Execute**, which consume and send XML for stateless data discovery and manipulation.

The specification is built upon the open Internet standards of HTTP, XML, and SOAP, and is not bound to any specific language or technology. The specification references OLE DB so that application developers already familiar with OLE DB can see how XML for Analysis can be mapped and implemented. These references also provide background information on the OLE DB definitions that the specification extends.

Audience

This specification targets application developers and assumes the following:

- Knowledge of XML
- Knowledge of SOAP
- Understanding of online analytical processing (OLAP) and data mining
- Working knowledge of OLE DB and OLE DB for OLAP

For more information about these areas, see Appendix E.

Design Goals

The primary goals of this specification include the following:

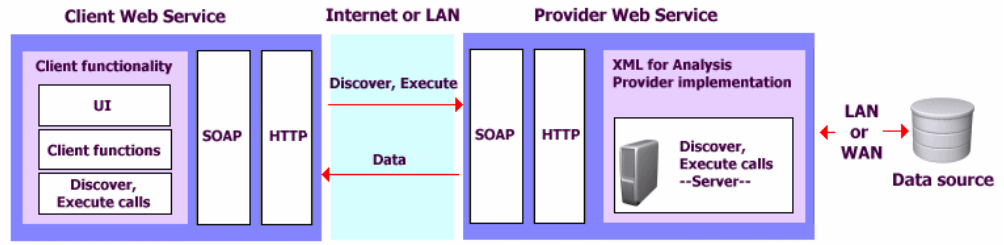
- Provide a standard data access API to remote data access providers that can be used universally on the Internet or intranet for multidimensional data
- Optimize a stateless architecture, requiring no client components for the Web, with minimal roundtrips
- Support technologically independent implementations using any tool, programming language, technology, hardware platform, or device
- Build on open Internet standards, such as SOAP, XML, and HTTP
- Leverage and reuse successful OLE DB design concepts, so that OLE DB for OLAP applications and OLE DB providers can be easily enabled for XML for Analysis
- Work efficiently with standard data sources, such as relational OLAP and data mining

Design Summary

The design centers around an XML-based communication API, called XML for Analysis, which defines two generally accessible methods: **Discover** and **Execute**. Because XML allows for a loosely coupled client and server architecture, both methods handle incoming and outgoing information in XML format. This API is optimized for the Internet, where roundtrips to the server are expensive in terms of time and resources, and where stateful connections to the data limit user connections on the server.

Discover is used to obtain information and meta data from a Web Service. This information can include a list available data sources and data about the provider for a particular data source. Properties are used to define and shape what data is obtained. The client application may need many types of information; **Discover** allows you to specify this in a common way. This generic interface and use of properties allows extensibility without rewriting existing functions.

Execute is used to execute Multidimensional Expressions (MDX) or other provider-specific commands against a particular XML for Analysis data source. The following diagram illustrates one possible implementation of an *n*-tiered application.



Provided with the URL for a server hosting a Web service, the client sends **Discover** and **Execute** calls using the SOAP and HTTP protocols to the server. The server instantiates the XML for Analysis provider, which handles the **Discover** and **Execute** calls. The XML for Analysis provider fetches the data, packages it into XML, and then sends the requested data as XML to the client.

The **Discover** and **Execute** methods enable users to determine what can be queried on a particular server and, based on this, submit commands to be executed. The following scenario illustrates how an Internet application or a Web Service could use these methods.

Part I – XML for Analysis

Introduction to XML for Analysis

XML for Analysis specifies a SOAP-based XML communication API that supports the exchange of analytical data between clients and servers on any platform and with any language.

Methods

The following methods provide a standard way for XML applications to access basic information from the server. Because these methods are invoked using the SOAP protocol, they accept input and deliver output in XML. By default, these methods are stateless, so the server context ends at the completion of any command. For information about how to make stateful calls, see "Support for Statefulness in XML for Analysis."

The simplified interface model has two methods. The **Discover** method obtains information, and the **Execute** method sends action requests to a server. The XML namespace for these methods is `"urn:schemas-microsoft-com:xml-analysis"`.

Connection information is supplied in each method call with the connection properties.

Discover

The **Discover** method can be used to retrieve information, such as the list of available data sources on a server or details about a specific data source. The data retrieved with the **Discover** method depends on the values of the parameters passed to it.

Namespace

```
urn:schemas-microsoft-com:xml-analysis
```

SOAP Action

```
"urn:schemas-microsoft-com:xml-analysis:Discover"
```

Syntax

```
Discover (
   [in] RequestType As EnumString,
   [in] Restrictions As Restrictions,
   [in] Properties As Properties,
   [out] Result As Rowset)
```

Parameters

RequestType [in]

This required parameter consists of a RequestTypes enumeration value, which determines the type of information to be returned. The RequestTypes enumeration is used by the **Discover** method to determine the structure and content of the rowset returned in the *Result* parameter. The format of the *Restrictions* parameter and the resulting XML result set is also dependent on the value specified in this parameter. This enumeration can be extended to support provider-specific enumeration strings.

Each RequestTypes enumeration value corresponds to a return rowset. For rowset definitions, see "XML for Analysis Rowsets." Support is required for the following explicitly named RequestTypes enumeration values.

Enumeration value	Description
DISCOVER_DATASOURCES	Returns a list of XML for Analysis data sources available on the server or Web Service. (For an example of how these may be published, see "XML for Analysis Implementation Walkthrough.")
DISCOVER_PROPERTIES	Returns a list of information and values about the requested properties that are supported by the specified data source (provider).
DISCOVER_SCHEMA_ROWSETS	Returns the names, values, and other information of all supported RequestTypes enumeration values (including those listed here), and any additional provider-specific enumeration values.
DISCOVER_ENUMERATORS	Returns a list of names, data types, and enumeration values of enumerators supported by a specific data source's provider.
DISCOVER_KEYWORDS	Returns a rowset containing a list of keywords reserved by the provider.
DISCOVER_LITERALS	Returns information about literals supported by the data source provider.
Schema Rowset Constant	Given a constant that corresponds to one of the schema rowset names defined by OLE DB, such as MDSCHEMA_CUBES, returns the OLE DB schema rowset in XML format. Note that providers may also extend OLEDB by providing additional provider-specific schema rowsets. The schema rowsets that tabular data providers (TDP) and multidimensional data providers (MDP) are required to support are listed in the section "DISCOVER_SCHEMA_ROWSETS Rowset."

Restrictions [in]

This parameter, of the Restrictions data type, enables the user to restrict the data returned in *Result*. The *Result* columns are defined by the rowset specified in the *RequestType* parameter. Some columns of *Result* can be used to filter the rows returned. For these columns and those that can be restricted, see the rowset tables in "XML for Analysis Rowsets." To obtain the restriction information for provider-specific schema rowsets, use the DISCOVER_SCHEMA_ROWSETS request type.

This parameter must be included, but it can be empty.

Properties [in]

This parameter, of the Properties data type, consists of a collection of XML for Analysis properties. Each property enables the user to control some aspect of the **Discover** method, such as specifying the return format of the result set, the timeout, and specifying the locale in which the data should be formatted.

The available properties and their values can be obtained by using the DISCOVER_PROPERTIES request type with the **Discover** method. Standard XML for Analysis properties are detailed in "XML for Analysis Properties."

There is no required order for the properties listed in the *Properties* parameter. This parameter must be included, but it can be empty.

Result [out]

This required parameter contains the result set returned by the provider as a **Rowset** object.

The columns and content of the result set are specified by the values specified in the *RequestType* and *Restrictions* parameters. The column layout of the returned result set is also determined by the value specified in *RequestType*. For more information about the rowset layouts that correspond to for each *RequestType* value, see "XML for Analysis Rowsets."

For more information about the Rowset data type, see "Data Types Used in XML for Analysis."

Example

In the following sample, the client sends the XML **Discover** call to request a list of cubes from the **FoodMart 2000** catalog:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_CUBES</RequestType>
  <Restrictions>
   <RestrictionList>
    <CATALOG_NAME>
     FoodMart 2000
```

```
  </CATALOG_NAME>
 </RestrictionList>
</Restrictions>
```

```
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=MSOLAP;Data Source=local;
     </DataSourceInfo>
     <Catalog>
      Foodmart 2000
     </Catalog>
     <Format>
      Tabular
     </Format>
    </PropertyList>
   </Properties>
  </Discover>
```

The provider returns the following result to the client:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <DiscoverResponse xmlns="urn:schemas-microsoft-com:xml-analysis">
   <return>
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset">
     <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <!-- The XML schema definition of the result comes here -->
      ...
     </xsd:schema>
     <row>
      <CATALOG_NAME>FoodMart 2000</CATALOG_NAME>
      <CUBE_NAME>Sales</CUBE_NAME>
      ...
     </row>
```

```
     <row>
      <CATALOG_NAME>FoodMart 2000</CATALOG_NAME>
      <CUBE_NAME>Warehouse</CUBE_NAME>
      ...
     </row>
     ...
    </root>
   </return>
  </DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Execute

The **Execute** method is used for sending action requests to the server. This includes requests involving data transfer, such as retrieving or updating data on the server.

Namespace

```
urn:schemas-microsoft-com:xml-analysis
```

SOAP Action

```
"urn:schemas-microsoft-com:xml-analysis:Execute"
```

Syntax

```
Execute (
   [in] Command As Command,
   [in] Properties As Properties,
   [out] Result As Resultset)
```

Parameters

Command [in]

This required parameter is of Command data type and consists of a provider-specific statement to be executed. XML for Analysis multidimensional providers must support the mdXML language, but they can also support other commands as needed.

Properties [in]

This parameter is of the Properties data type and consists of a collection of XML for Analysis properties. Each property allows the user to control some aspect of the **Execute** method, such as defining the information required for the connection, specifying the return format of the result set, or specifying the locale in which the data should be formatted.

The available properties and their values can be obtained by using the DISCOVER_PROPERTIES request type with the **Discover** method. Standard XML for Analysis properties are detailed in "XML for Analysis Properties."

There is no required order for the properties listed in the *Properties* parameter. This parameter must be included, but it can be empty.

Result [out]

This parameter contains the **Resultset** result returned by the provider. The *Command* parameter and values in the *Properties* parameter define the shape of the result set. If no shape-defining properties are passed, the XML for Analysis provider may use a default shape.

The two result set formats defined by this specification are Tabular and Multidimensional, as specified by the client through the Format property. OLAP data lends itself to the Multidimensional format (although the Tabular format can also be used). A provider may support additional rowset types, and clients aware of the specialized types can request them.

Example

The following is an example of an **Execute** method call with <Statement> set to an OLAP MDX SELECT statement:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <Command>
   <Statement>
    select [Measures].members on Columns from Sales
   </Statement>
  <Command>
  <Properties>
   <PropertyList>
```

<DataSourceInfo>

```
Provider=Essbase;Data Source=local;
     </DataSourceInfo>
     <Catalog>Foodmart 2000</Catalog>
     <Format>Multidimensional</Format>
     <AxisFormat>ClusterFormat</AxisFormat>
    </PropertyList>
   </Properties>
  </Execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This is the abbreviated response for the preceding method call:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:ExecuteResponse
  xmlns:m="urn:schemas-microsoft-com:xml-analysis">
  <m:return
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xars="urn:schemas-microsoft-com:xars">
     ...<!—The schema for the data goes here. -- >
    </xsd:schema>
    ... <!—The data in MDDataSet format goes here. -- >
   </root>
  </m:return>
  </m:ExecuteResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Data Types Used in XML for Analysis

The following alphabetical list describes XML for Analysis data types and notes those data types that use standard XML data types. For more information about the XML Schema types, see http://www.w3.org/TR/xmlschema-2/. To view the schema structure, see http://www.w3.org/2001/XMLSchema-datatypes.xsd.

Boolean

The Boolean type uses the standard XML boolean data type.

Decimal

The Decimal type noted uses the standard XML decimal data type.

Integer

The Integer type noted in this document refers to the standard XML int data type.

EnumString

The EnumString data type defines a set of named constants for a given enumerator (enum). EnumString uses the standard XML string data type. The specific values for each of the named constants are specified with the enumerator definition.

MDDataSet

The MDDataSet format is one of the formats that can be returned in the *Result* parameter of the **Execute** method. This one is used for multidimensional data. Representing OLAP data in XML requires an OLAP-oriented rowset (or dataset), which is noted here. The XML namespace for the MDDataSet data type is "`urn:schemas-microsoft-com:xml-analysis:mddataset`".

For basic information about the OLE DB for OLAP dataset structures, see "MDDataset Data Type Mapping to OLE DB." For a full XML Schema Definition (XSD) sample of the MDDataSet, see Appendix D.

This specification defines the following XML structure for OLAP results. An MDDataSet consists of these main sections:

- OLAPInfo: Defines the structure of the result, listing and describing the cube, axes, and cells that will followCubeInfo: Contains the collection of cube elements
- Axes: Contains the data for the axes as defined in the OLAPInfo structure
- CellData: Contains the data for the cells as defined in the OLAPInfo structure

Comment: Any reason for "Cube" rather than cube? Grammatically, should be "cube".

Providers can add additional annotations to the structure as long as they do not change the behavior and meaning of the schema defined here. This open content schema model allows new elements and attributes to be added from other namespaces but does not allow the semantics of defined elements and attributes to be changed.

OLAPInfo

The OLAPInfo section contains three elements, CubeInfo, AxisInfo, and CellInfo. The CubeInfo section contains a collection of cube elements. To define the structure, `<OlapInfo>` defines axes using the `<AxesInfo>` element (note the plural, Axes). Axes consists of a set of `<AxisInfo>` elements (note the singular, Axis) that alias to an ordinal, such as `name="Axis0"`. The dimension hierarchies are then listed with their property definitions. In the example that follows, the standard member properties are represented in `<HierarchyInfo>` element by UName, Caption, LName, and LNum, as well as the nonstandard DisplayInfo element. For the Store hierarchy, the additional (nonstandard) member property, with the space character, [Store].[Store SQFT] is illustrated.

Comment: Small note. Do we really need the plural/singular notes? Seems to me that plural collections and singular items are common in XML and object orientated programming.

```
<OlapInfo>
   <CubeInfo>
      <Cube>
            <CubeName>cubename</CubeName>
      </Cube>
    </CubeInfo>
   <AxesInfo>
      <AxisInfo name="Axis0">
         <HierarchyInfo name="Measures">
            <UName name="[Measures].[MEMBER_UNIQUE_NAME]"></UName>
               <Caption name="[Measures].[MEMBER_CAPTION]"></Caption>
                  <LName name="[Measures].[LEVEL_UNIQUE_NAME]"></LName>
                     <LNum name="[Measures].[LEVEL_NUMBER]"></LNum>
                        <DisplayInfo
name="[Measures].[DISPLAY_INFO]"></DisplayInfo>
                  </HierarchyInfo>
               </AxisInfo>
                  <AxisInfo name="Axis1">
                     <HierarchyInfo name="Store">
                           <UName
name="[Store].[MEMBER_UNIQUE_NAME]"></UName>
                           <Caption
name="[Store].[MEMBER_CAPTION]"></Caption>
                           <LName
name="[Store].[LEVEL_UNIQUE_NAME]"></LName>
                           <LNum name="[Store].[LEVEL_NUMBER]"></LNum>
                           <DisplayInfo
name="[Store].[DISPLAY_INFO]"></DisplayInfo>
```

```
                        <Store_x0020_SQFT name="[Store].[Store
Name].[Store SQFT]"></Store_x0020_SQFT>
                    </HierarchyInfo>
                    <HierarchyInfo name="Time">
                        <UName
name="[Time].[MEMBER_UNIQUE_NAME]"></UName>
                        <Caption
name="[Time].[MEMBER_CAPTION]"></Caption>
                        <LName
name="[Time].[LEVEL_UNIQUE_NAME]"></LName>
                        <LNum name="[Time].[LEVEL_NUMBER]"></LNum>
                        <DisplayInfo
name="[Time].[DISPLAY_INFO]"></DisplayInfo>
                    </HierarchyInfo>
                </AxisInfo>
```

The last thing defined by the OLAPInfo structure is the properties (column definitions) for the cells. This allows cells to contain additional properties. The properties in this example are Value, FmtValue, and the custom property FormatString.

```
<OlapInfo>
 ...
            <CellInfo>
                <Value name="VALUE"></Value>
                <FmtValue name="FORMATTED_VALUE"></FmtValue>
                <FormatString name="FORMAT_STRING"></FormatString>
            </CellInfo>
```

HierarchyInfo Standard Elements

The following standard elements are required for the `<HierarchyInfo>` element. MDSCHEMA references refer to the OLE DB for OLAP schema definition.

The name attribute of the HierarchyInfo element should contain the HIERARCHY_UNIQUE_NAME, as defined in OLEDB.

Element	Description
UName	MEMBER_UNIQUE_NAME property from OLE DB axis rowset
Caption	MEMBER_CAPTION property from OLE DB axis rowset
LName	LEVEL_UNIQUE_NAME property from OLE DB axis rowset
LNum	LEVEL_NUMBER property from OLE DB axis rowset

CellInfo Standard Elements

The following are the standard elements for the `<CellInfo>` element. Whether or not they are returned for any particular query depends on the query itself.

Element	Description
Value	VALUE property from OLE DB cell properties
FmtValue	FORMATTED_VALUE property from OLE DB cell properties
ForeColor	FORE_COLOR property from OLE DB cell properties
BackColor	BACK_COLOR property from OLE DB cell properties

Using Defaults in CellInfo and AxisInfo

A provider can optionally specify default values for individual member or cell properties in the AxisInfo or CellInfo section. This can provide a smaller result if the property always or almost always has the same value.

To indicate a default value for a property, the `<Default>` element can optionally be specified as a subelement of one of the member or cell property elements. For instance, to specify a default value for Store SQFT, the provider would be specified it as follows:

```
<Store_x0020_SQFT name="Store SQFT">
 <Default>5000</Default>
</Store_x0020_SQFT>
```

Therefore, the absence of a member or cell property in the result indicates that the stated default is the value for the member property or the cell property. In the following result, in which the output for the <Store_x0020_SQFT> element is absent, the value for <Store_x0020_SQFT> is 5000 (the default value that was defined earlier):

```
<Member Hierarchy="Store">
      <UName>[Store].[CA]</UName>
      <Caption>CA</Caption>
      <LName>[Store].[State]</LName>
      <LNum>2</LNum>
</Member>
```

If the element is present but without a value, this implies an empty string result (""), as shown in the following example:

```
<Store_x0020_SQFT />
```

Typically, if a property is null, it is simply omitted. However, if a default value has been defined for a property, then to indicate a null value for a property, use the nil attribute from the XML Schema specification, as follows:

```
<Store_x0020_SQFT xsi:nil='true' />
```

Axes

Under Axes, the Axis items are listed in the order that they occur in the dataset, starting at zero. The **AxisFormat** property setting determines how Axis elements are formatted. All XML for Analysis providers must support the following values for the property AxisFormat:

- ClusterFormat
- TupleFormat
- CustomFormat

Support of the CustomFormat value as a distinct format is optional for a provider. If a client requests CustomFormat, the provider may choose, at its discretion, to return one of the TupleFormat and ClusterFormat formats. While providers must support all three of the above values, clients can request the format they want; therefore clients may choose not to make use of all three available formats.

Why Different Formats?

The TupleFormat and ClusterFormat settings for the AxisFormat property provide two different ways of representing tuples. The MDDataset definition gives the provider two ways to specify tuples as multidimensional tuples or as a Cartesian product. This provides a client application a choice between simplicity and minimizing space requirements.

An axis represents a set of tuples, where all tuples in the set have the same dimensionality. A set can be represented in different ways with different advantages. For example, the following set of four tuples can be represented as a collection of two-dimensional tuples or a Cartesian product of two one-dimensional sets.

1999	1999	2000	2000
Actual	Budget	Actual	Budget

The following line represents the set of four tuples as collection of two-dimensional tuples:

```
{ ( 1999, Actual ), ( 1999, Budget ), ( 2000, Actual ), ( 2000, Budget ) }
```

The following line represents the set of four tuples as a Cartesian product of two one-dimensional sets:

```
{ 1999, 2000 } x { Actual, Budget }
```

Both representations have advantages and disadvantages. Two-dimensional tuples are simpler for client tools to use. A Cartesian product of one-dimensional sets uses less space and preserves the multidimensional nature of the set.

The following table lists operations that can be used to define and characterize the structure and members of an axis.

Operation	**Description**
Member	The smallest unit of an axis representing the member of a dimension hierarchy
Tuple	A vector of members from different dimension hierarchies
Members	A set of Member objects from the same dimension hierarchy
Tuples	A collection of Tuple objects with the same dimensionality
Union	A union of sets
CrossProduct	A Cartesian product of sets

Based on the previous example, these operations translate the two-dimensional tuples and Cartesian product of one-dimensional sets as follows.

Two-dimensional tuples

```
Tuples (
Tuple( Member(1999), Member(Actual) ),
Tuple( Member(1999), Member(Budget) ),
Tuple( Member(2000), Member(Actual) ),
Tuple( Member(2000), Member(Budget) )
)
```

Cartesian product of one-dimensional sets

```
CrossProduct (
Members (Member(1999), Member(2000) ),
Members (Member(Actual), Member(Budget) )
)
```

The XML representation of these operations follows these rules (where *member_properties* value refers to the list of member properties defined in the corresponding AxisInfo section):

member : <Member> *member_properties* </Member>

tuple : <Tuple> *member_list* </Tuple>

set : <Members> *member_list* </Members>

set : <Tuples> *tuple_list* </Tuples>

set : <CrossProduct> *set_list* </CrossProduct>

set : <Union> *set_list* </Union>

member_list : *member* [*member* ...]

tuple_list : *tuple* [*tuple* ...]

set_list : *set* [*set* ...]

As shown above, the same set can have different representations using different operations. The client can request a specific representation using the AxisFormat property.

TupleFormat

In TupleFormat, an axis is represented as a set of tuples. The following operations must be used in the specified order:

```
<Axis>
      <Tuples>
            <Tuple>
                  <Member Hierarchy="name">
```

In addition, `<Member>` elements must have the **Hierarchy** attribute that specifies the hierarchy name of the member.

The following example illustrates the TupleFormat.

1999	**1999**	**2000**
Actual	Budget	Budget

```
<Axes>
 <Axis name="Axis0">
  <Tuples>
   <Tuple>
    <Member Hierarchy="Time">
     <UName>[Time].[1999]</UName>
     …
    </Member>
    <Member Hierarchy="Category">
     <UName>[Scenario].[Actual]</UName>
     …
    </Member>
   </Tuple>
   <Tuple>
    <Member Hierarchy="Time">
     <UName>[Time].[1999]</UName>
     …
    </Member>
    <Member Hierarchy="Category">
     <UName>[Scenario].[Budget]</UName>
     …
    </Member>
   </Tuple>
   <Tuple>
    <Member Hierarchy="Time">
     <UName>[Time].[2000]</UName>
     …
    </Member>
```

```
    <Member Hierarchy="Category">
     <UName>[Scenario].[Budget]</UName>
     …
    </Member>
   </Tuple>
  </Tuples>
 </Axis>
 ...
</Axes>
```

ClusterFormat

In ClusterFormat, an axis is represented as a set of clusters. Each cluster represents a crossproduct of members from different dimension hierarchies. Providers will define their own provider-specific clustering algorithms. The following operations must be used in the specified order:

```
<Axis>
 <CrossProduct Size="size">
  <Members Hierarchy="name">
   <Member>
```

For representing objects as clusters, the `<CrossProduct>` element must have a **Size** attribute indicating the number of tuples that results from the product of individual Member sets within the CrossProduct. The `<Members>` element must have a **Hierarchy** attribute that specifies the dimension hierarchy name of all members in the set.

A crossproduct may contain members from a single dimension hierarchy.

The following example illustrates two clusters:

1999	**1999**	**2000**	**2000**	**2001**
Actual	Budget	Actual	Budget	Budget
		cluster 1		*cluster 2*

```
<Axes>
 <Axis name="Axis0">
  <CrossProduct Size = "4">
   <Members Hierarchy="Time">
    <Member>
     <UName>[Time].[1999]</UName>
     …
    </Member>
    <Member>
     <UName>[Time].[2000]</UName>
     …
    </Member>
   </Members>
   <Members Hierarchy="Category">
    <Member>
     <UName>[Scenario].[Actual]</UName>
     …
    </Member>
    <Member>
     <UName>[Scenario].[Budget]</UName>
     …
    </Member>
   </Members>
  </CrossProduct>
  <CrossProduct Size = "1">
   <Members Hierarchy="Time">
```

```
    <Member>
     <UName>[Time].[2001]</UName>
     …
    </Member>
   </Members>
   <Members Hierarchy="Category">
    <Member>
     <UName>[Scenario].[Budget]</UName>
     …
    </Member>
   </Members>
  </CrossProduct>
 </Axis>
 …
</Axes>
```

CustomFormat

CustomFormat allows the provider to generate the axes in any valid combination of the operations defined in the sections above, with following restrictions:

- Only `<Union>`, `<CrossProduct>`, `<Members>`, and `<Tuples>` elements can occur as the first child of an axis.
- A `<Member>` element under the `<Tuple>` element must contain a **Hierarchy** attribute indicating the name of the hierarchy the member belongs to.
- A `<Members>`element must contain a **Hierarchy** attribute indicating the hierarchy name of all members in the set.

The CustomFormat gives the most flexibility and power to a provider to optimize the axis representation.

This section is an example of what a provider may choose to return for CustomFormat.

WA	WA	CA	CA
Umbrella	Umbrella	Sunglasses	Sunglasses
Actual	Budget	Actual	Budget

A provider can choose to generate the representation below for the CustomFormat result. In this example, a set of tuples is crossjoined with a set of members.

```
CrossProduct(
Tuples (
Tuple ( Member(WA), Member(Umbrella) ),
Tuple ( Member(CA), Member(Sunglasses) ) ),
      Members ( Member(Actual), Member(Budget) )
)
```

The above theoretical formulation can be represented in XML as follows:

```
<Axis name="Axis0">
 <CrossProduct>
  <Tuples>
   <Tuple>
    <Member Hierarchy="Store">
<UName>[Store].[WA]</UName>
...
</Member>
    <Member Hierarchy="Product">
<UName>[Product].[Umbrella]</UName>
...
</Member>
   </Tuple>
   <Tuple>
    <Member Hierarchy="Store">
<UName>[Store].[CA]</UName>
...
</Member>
```

```
    <Member Hierarchy="Product">
<UName>[Product].[Sunglasses]</UName>
...
</Member>
   </Tuple>
  </Tuples>
  <Members Hierarchy="Category">
  <Member>
  <UName>[Category].[Actual]</UName>
  ...
  </Member>
  <Member>
  <UName>[Category].[Budget]</UName>
  ...
  </Member>
  </Members>
 </CrossProduct>
</Axis>
```

CellData

The Axes section is followed by the CellData section, which contains the property values for each cell. A mandatory **CellOrdinal** attribute indicates the ordinal of the cell. **CellOrdinal** is numbered 0 to *n-1*, for *n* cells. Cell elements can be missing if all cell properties are the default (NULL is the default if no default has been specified). Note that the type of the `<Value>` element must be specified in the CellData section, while other standard properties, whose type is defined in the schema need not have a type specified.

```
<CellData>
   <Cell CellOrdinal="0">
      <Value xsi:type="xsd:double">16890</Value>
      <FmtValue>16,890.00</FmtValue>
      <FormatString>Standard</FormatString>
   </Cell>
   <Cell CellOrdinal="1">
      <Value xsi:type="xsd:int">50</Value>
      <FmtValue>50</FmtValue>
      <FormatString>Standard</FormatString>
   </Cell>
   <Cell CellOrdinal="2">
      <Value xsi:type="xsd:double">36175.2</Value>
      <FmtValue>$36,175.20</FmtValue>
      <FormatString>Currency</FormatString>
   </Cell>
</CellData>
```

No property is returned more than once in the CellData section, even if an MDX command has multiple references to a cell property. For example, provided the following MDX query

```
Select from sales cell properties Value, FormattedValue, Value
```

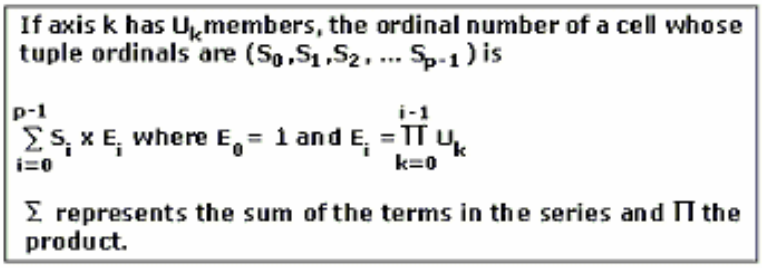
The CellInfo section of OlapInfo contains the same sequence of cell properties:

```
<CellInfo>
   <CellOrdinal/>
   <Value/>
   <FmtValue/>
   <Value/>
</CellInfo>
```

However, the CellData section eliminates the duplication,returning the data only once:

```
<Cell CellOrdinal="0">
   <Value xsi:type="xsd:int">10</Value>
   <FmtValue>$10.00</FmtValue>
</Cell>
```

The axis reference for a cell can be calculated based on CellOrdinal. Conceptually, cells are numbered in a dataset as if the dataset were a *p*-dimensional array, where *p* is the number of axes. Cells are addressed in row-major order. The following illustration shows the formula for calculating the ordinal number of a cell.



We will apply the above formula to the result set shown in the following table. The query asked for four measures on columns and a crossjoin of two states with four quarters on rows. In following the dataset result, the **CellOrdinal** property for the part of the dataset result shown in the box is the set {9, 10, 11, 13, 14, 15, 17, 18, 19}. This is because the cells are numbered in row major order, starting with a **CellOrdinal** of zero for the upper left cell.

Next, we apply the above formula to the cell that is {CA, Q3, Store Cost}. Axis k=0 has U_k=4 members and axis k=1 has U_k=8 tuples. P is the total number of axes in the query, here equal to 2. S_0, the initial summation is i=0 to 1. For i=0, the tuple ordinal on axis 0 of {Store Cost} is 1. For i = 1, the tuple ordinal of {CA, Q3} is 2.

For i=0, E_i = 1, so for i = 0 the sum is 1 * 1 = 1 and for i=1, the sum is 2 (tuple ordinal) * 4 (the value of E_i, computed as 1 * 4), or 8, and so the sum is equal to 1 + 8 = 9, the cell ordinal for that cell.

		Unit Sales	Store Cost	Store Sales	Sales Count
CA	Q1	16,890.00	14,431.09	$36,175.20	5498
	Q2	18,052.00	15,332.02	$38,396.75	5915
	Q3	18,370.00	15,672.83	$39,394.05	6014
	Q4	21,436.00	18,094.50	$45,201.84	7015
OR	Q1	19,287.00	16,081.07	$40,170.29	6184
	Q2	15,079.00	12,678.96	$31,772.88	4799
	Q3	16,940.00	14,273.78	$35,880.46	5432
	Q4	16,353.00	13,738.68	$34,453.44	5196

The complete XML output for the above dataset is shown in Appendix D.

Command

The Command data type is an XML document type. In this version of the XML for Analysis specification the Command data type consists solely of the <Statement> tag, of type string, which contains the text for the command statement. For example, the <Statement> element with an MDX statement may look like this:

```
<Statement>
  SELECT Measures.MEMBERS on columns from Sales
</Statement>
```

In a future version of this specification, the XML document for the Command data type will be expanded beyond the single <Statement> element defined in this specification.

The XML for Analysis specification requires that multidimensional providers support the mdXML language. The mdXML language will be based on MDX; currently mdXML consists solely of the <Statement> element. For multidimensional providers, the <Statement> element must contain an MDX language statement. Future enhancements to mdXML will make additional elements beyond the <Statement> element available. The <Statement> element will continue to support a complete MDX statement as type string, even if it is expanded to also allow for other XML elements.

The <Statement> element may be empty, as in <Statement/>. This can be used for the Command in which BeginSession or EndSession is sent in the SOAP header.

In addition to future enhancements of mdXML, the MDX language itself is extensible.Providers can add extensions to the language to support additional features that are not provided in the base language set. For more information about mdXML, please see section "Relationship between MDX and mdXML.

Properties

The Properties data type represents a collection of XML for Analysis properties. Each property is defined by an XML element, and the value of the property is the data contained by the XML element. The name of the XML element corresponds to the name of the property.

Each provider can extend the set of properties, but provider-specific property names must be well-formed XML tags.

An example follows:

```
<PropertyList>
 <DataSourceInfo>
  Provider=MSOLAP;Data Source=local;
 </DataSourceInfo>
 <Catalog>
  Foodmart 2000
 </Catalog>
 <Format>
  Multidimensional
 </Format>
</PropertyList>
```

Restrictions

The Restrictions data type represents a collection of restrictions to be applied during the execution of a **Discover** method. The Restriction name specifies the rowset column name that is being restricted. The Restriction value defines the data for the column.

Each provider can add new schema rowsets, but columns that can be restricted should have names that meet the well-formedness constraints of XML.

The following example sends a restriction for a column name in the MDSCHEMA_CUBES schema rowset:

```
<RestrictionList>
 <CATALOG_NAME>
  FoodMart 2000
 </CATALOG_NAME>
 ...
</RestrictionList>
```

When needed, a column can be restricted with multiple values. Each value is represented in a `<Value>` element. An example follows:

```
<RestrictionList>
 <LiteralName>
  <Value>DBLITERAL_QUOTE_PREFIX</Value>
  <Value>DBLITERAL_QUOTE_SUFFIX</Value>
  <Value>DBLITERAL_ESCAPE_UNDERSCORE_PREFIX</Value>
  <Value>DBLITERAL_ESCAPE_UNDERSCORE_SUFFIX</Value>
 </LiteralName>
 ...
</RestrictionList>
```

The Members schema rowset has a special restriction that does not correspond to a column in the rowset result. This special restriction operator is called the TREE_OP restriction. The type of the TREE_OP restriction is integer. The TREE_OP restriction is a bitmask, so bitwise combinations of its values are valid. The following table contains the possible values for the TREE_OP restriction.

Symbolic constant (OLEDB)	Integer value	Description
MDTREEOP_CHILDREN	1	Returns only the immediate children.
MDTREEOP_SIBLINGS	2	Returns members on the same level.
MDTREEOP_PARENT	4	Returns only the immediate parent.
MDTREEOP_SELF	8	Returns the immediate member in the list of returned rows.
MDTREEOP_DESCENDANTS	16	Returns all descendants.
MDTREEOP_ANCESTORS	32	Returns all ancestors.

Resultset

The Resultset data type is a self-describing XML result set. The type of data it will contain is indicated by the XML for Analysis Format property.

By default, the XML schema is returned with the result set. This can be changed using the Content property, described in "XML for Analysis Properties."

Rowset

The XML schema embedded within the rowset defines the specific structure of the **Rowset** return data type. The general structure of the XML for Analysis rowset is similar to the Microsoft® SQL Server™ 2000 rowset format obtained with the FOR XML RAW

clause, but it is element-centric rather than attribute-centric, and it allows hierarchical data.

XML does not allow certain characters as element and attribute names. XML for Analysis supports encoding as defined by SQL Server 2000 to address this XML constraint. For column names that contain invalid XML name characters (according to the XML 1.0 specification), the nonvalid Unicode characters are encoded using the corresponding hexadecimal values. These are escaped as _x*HHHH*_ where *HHHH* stands for the four-digit hexadecimal UCS-2 code for the character in most-significant bit first order. For example, the name "Order Details" is encoded as Order_*x0020*_Details, where the space character is replaced by the corresponding hexadecimal code.

Encoding can make Extensible Style Language (XSL) transformations difficult. To support a quick lookup of the actual unencoded column names, add the **sql:field** attribute into the XML rowset schema with each column. This attribute resides in the "`urn:schemas-microsoft-com:xml-sql`" namespace.

An example follows:

```
<xsd:element name="Order_x0020_Details" type="string" sql:field="Order
Details" />
```

A Null value for a column within a row can be expressed in the following ways:

- A missing column element implies that the column is null.
- You can use xsi:nil='true' attribute to indicate that the column element has a null value.

For example, if a row has a single column called Store_Name and its value is null, it can be represented as either:

```
      <row>
      </row>
-Or-
      <row>
         <Store_name xsi:nil='true'/>
      </row>
```

For flat data, the XML for Analysis rowset format appears as in the following example. The column names, which are specific to the query, are defined in the schema as the element names. A pair of <row> tags encapsulates each row:

```
<root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sql="urn:schemas-microsoft-com:xml-sql">
   </xsd:element>
      <xsd:complexType name="row">
```

```
         <xsd:choice maxOccurs="unbounded" minOccurs="0">
            <xsd:element name="CATALOG_NAME" type="xsd:string"
sql:field="CATALOG_NAME"></xsd:element>
            <xsd:element name="DESCRIPTION" type="xsd:string"
sql:field="DESCRIPTION"></xsd:element>
            <xsd:element name="ROLES" type="xsd:string"
sql:field="ROLES"></xsd:element>
            <xsd:element name="DATE_MODIFIED" type="xsd:time"
sql:field="DATE_MODIFIED"></xsd:element>
         </xsd:choice>
      </xsd:complexType>
   </xsd:schema>
   <row>
      <CATALOG_NAME>FoodMart 2000</CATALOG_NAME>
      <DESCRIPTION></DESCRIPTION>
      <ROLES>All Users</ROLES>
      <DATE_MODIFIED>3/11/2001 6:49:36 PM</DATE_MODIFIED>
   </row>
   …
</Root>
```

For hierarchical data (or nested rowsets), such as that returned by OLE DB for data mining queries, the XML for Analysis rowset format appears as in the following example. The structure of the rows is not changed, but the data-specific schema defines an element subtype that contains the nested data. In this case, the nested element is <NODE_DISTRIBUTION>.

```
<root xmlns=”urn:schemas-microsoft-com:xml-analysis:rowset">
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:xml-sql">
      <xsd:complexType name="row">
         <xsd:choice maxOccurs="unbounded" minOccurs="0">
            <xsd:sequence maxOccurs="unbounded" minOccurs="0">
               <xsd:element name="NODE_DISTRIBUTION"
sql:field="NODE_DISTRIBUTION">
                  <xsd:complexType>
                     <xsd:choice maxOccurs="unbounded" minOccurs="0">
                        <xsd:element name="ATTRIBUTE_NAME"
type="xsd:string" sql:field="ATTRIBUTE_NAME"></xsd:element>
                           <xsd:element name="ATTRIBUTE_VALUE"
type="xsd:string" sql:field="ATTRIBUTE_VALUE"></xsd:element>
                           <xsd:element name="SUPPORT" type="xsd:double"
sql:field="SUPPORT"></xsd:element>
                        <xsd:element name="PROBABILITY" type="xsd:double"
sql:field="PROBABILITY"></xsd:element>
                        <xsd:element name="VARIANCE" type="xsd:double"
sql:field="VARIANCE"></xsd:element>
                        <xsd:element name="VALUETYPE" type="xsd:int"
sql:field="VALUETYPE"></xsd:element>
                     </xsd:choice>
                  </xsd:complexType>
               </xsd:element>
            </xsd:sequence>
            <xsd:element name="MODEL_CATALOG" type="xsd:string"
sql:field="MODEL_CATALOG"></xsd:element>
            <xsd:element name="MODEL_SCHEMA" type="xsd:string"
sql:field="MODEL_SCHEMA"></xsd:element>
            <xsd:element name="MODEL_NAME" type="xsd:string"
sql:field="MODEL_NAME"></xsd:element>
            <xsd:element name="ATTRIBUTE_NAME" type="xsd:string"
sql:field="ATTRIBUTE_NAME"></xsd:element>
```

```
            <xsd:element name="NODE_NAME" type="xsd:string"
sql:field="NODE_NAME"></xsd:element>
            <xsd:element name="NODE_UNIQUE_NAME" type="xsd:string"
sql:field="NODE_UNIQUE_NAME"></xsd:element>
            <xsd:element name="NODE_TYPE" type="xsd:unsignedInt"
sql:field="NODE_TYPE"></xsd:element>
            <xsd:element name="NODE_GUID" type="xsd:string"
sql:field="NODE_GUID"></xsd:element>
            <xsd:element name="NODE_CAPTION" type="xsd:string"
sql:field="NODE_CAPTION"></xsd:element>
            <xsd:element name="CHILDREN_CARDINALITY"
type="xsd:unsignedInt" sql:field="CHILDREN_CARDINALITY"></xsd:element>
            <xsd:element name="PARENT_UNIQUE_NAME" type="xsd:string"
sql:field="PARENT_UNIQUE_NAME"></xsd:element>
            <xsd:element name="NODE_DESCRIPTION" type="xsd:string"
sql:field="NODE_DESCRIPTION"></xsd:element>
            <xsd:element name="NODE_RULE" type="xsd:string"
sql:field="NODE_RULE"></xsd:element>
            <xsd:element name="MARGINAL_RULE" type="xsd:string"
sql:field="MARGINAL_RULE"></xsd:element>
            <xsd:element name="NODE_PROBABILITY" type="xsd:double"
sql:field="NODE_PROBABILITY"></xsd:element>
            <xsd:element name="MARGINAL_PROBABILITY" type="xsd:double"
sql:field="MARGINAL_PROBABILITY"></xsd:element>
            <xsd:element name="NODE_SUPPORT" sql:type="xsd:double"
sql:field="NODE_SUPPORT"></xsd:element>
            <xsd:element name="MSOLAP_MODEL_COLUMN" sql:type="xsd:string"
sql:field="MSOLAP_MODEL_COLUMN"></xsd:element>
            <xsd:element name="MSOLAP_NODE_SCORE" sql:type="xsd:double"
sql:field="MSOLAP_NODE_SCORE"></xsd:element>
            <xsd:element name="MSOLAP_NODE_SHORT_CAPTION"
sql:type="xsd:string" sql:field="MSOLAP_NODE_SHORT_CAPTION"></xsd:element>
         </xsd:choice>
      </xsd:complexType>
   </xsd:schema>
   <row>
      <MODEL_CATALOG>FoodMart 2000</MODEL_CATALOG>
      <MODEL_NAME>customer pattern discovery</MODEL_NAME>
```

<ATTRIBUTE_NAME>Customers.Name.Member Card</ATTRIBUTE_NAME>
<NODE_NAME>2147483652</NODE_NAME>
<NODE_UNIQUE_NAME>2147483652</NODE_UNIQUE_NAME>
<NODE_TYPE>2</NODE_TYPE>
<NODE_CAPTION>All</NODE_CAPTION>
<CHILDREN_CARDINALITY>8</CHILDREN_CARDINALITY>
<PARENT_UNIQUE_NAME>0</PARENT_UNIQUE_NAME>
<NODE_DESCRIPTION>All</NODE_DESCRIPTION>
<NODE_RULE></NODE_RULE>
<MARGINAL_RULE></MARGINAL_RULE>
<NODE_PROBABILITY>1</NODE_PROBABILITY>
<MARGINAL_PROBABILITY>1</MARGINAL_PROBABILITY>
<NODE_DISTRIBUTION>
<ATTRIBUTE_NAME>Customers.Name.Member Card</ATTRIBUTE_NAME>
<ATTRIBUTE_VALUE>missing</ATTRIBUTE_VALUE>
<SUPPORT>0</SUPPORT>
<PROBABILITY>0</PROBABILITY>
<VARIANCE>0</VARIANCE>
<VALUETYPE>1</VALUETYPE></NODE_DISTRIBUTION>
<NODE_DISTRIBUTION>
<ATTRIBUTE_NAME>Customers.Name.Member

```
Card</ATTRIBUTE_NAME>
         <ATTRIBUTE_VALUE>Bronze</ATTRIBUTE_VALUE>
         <SUPPORT>3077</SUPPORT>
         <PROBABILITY>0.551334886221107</PROBABILITY>
         <VARIANCE>0</VARIANCE>
         <VALUETYPE>4</VALUETYPE></NODE_DISTRIBUTION>
      <NODE_DISTRIBUTION>
         <ATTRIBUTE_NAME>Customers.Name.Member Card</ATTRIBUTE_NAME>
         <ATTRIBUTE_VALUE>Golden</ATTRIBUTE_VALUE>
         <SUPPORT>659</SUPPORT>
         <PROBABILITY>0.118079197276474</PROBABILITY>
         <VARIANCE>0</VARIANCE>
         <VALUETYPE>4</VALUETYPE></NODE_DISTRIBUTION>
      <NODE_DISTRIBUTION>
         <ATTRIBUTE_NAME>Customers.Name.Member Card</ATTRIBUTE_NAME>
         <ATTRIBUTE_VALUE>Normal</ATTRIBUTE_VALUE>
         <SUPPORT>1332</SUPPORT>
         <PROBABILITY>0.238666905572478</PROBABILITY>
         <VARIANCE>0</VARIANCE>
         <VALUETYPE>4</VALUETYPE></NODE_DISTRIBUTION>
      <NODE_DISTRIBUTION>
         <ATTRIBUTE_NAME>Customers.Name.Member Card</ATTRIBUTE_NAME>
         <ATTRIBUTE_VALUE>Silver</ATTRIBUTE_VALUE>
         <SUPPORT>513</SUPPORT>
         <PROBABILITY>9.19190109299409E-02</PROBABILITY>
         <VARIANCE>0</VARIANCE>
         <VALUETYPE>4</VALUETYPE></NODE_DISTRIBUTION>
      <NODE_SUPPORT>5581</NODE_SUPPORT>
      <MSOLAP_MODEL_COLUMN>Customers.Name.Member
```

```
Card</MSOLAP_MODEL_COLUMN>
      <MSOLAP_NODE_SCORE>1948.401692055</MSOLAP_NODE_SCORE>
      <MSOLAP_NODE_SHORT_CAPTION>All</MSOLAP_NODE_SHORT_CAPTION>
</row>
</root>
```

String

The String type corresponds to the standard XML string data type.

UnsignedInt

The UnsignedInt data type corresponds to the XML unsignedInt schema type.

EmptyResult

Some XML for Analysis commands are not expected to return a result. For commands that do not return a result, the following namespace on the <root> return element is used:is:

```
urn:schemas-microsoft-com:xml-analysis:empty
```

The root element of an empty result looks like the following:

```
<root xmlns="urn:scshemas-microsoft-com:xml-analysis:empty"/>
```

XML for Analysis Rowsets

Information returned in the *Result* parameter of the **Discover** method is structured according to the rowset column layouts detailed in this section.

All columns noted in the following rowsets are required, and they must be returned in the order shown. However, additional columns (which should be ignored by clients not expecting them) can be added at the end, and some columns can contain null data for information that does not apply.

The following sections describe the columns in each rowset. Each section includes a table that provides the following information for each column.

Column heading	Contents
Column name	The name of the column in the output rowset.
Type	A description of the data type for the column. For more information on data types supported by XML for Analysis, see "Data Types Used in XML for Analysis."
Description	A brief description of the purpose of the column.

Restriction	Indicates whether the column can be used to restrict the returned rowset by inclusion in the *Restrictions* parameter of the **Discover** method. *Yes* means that the column is available to use as a *Restrictions* item to filter results by this field.
Nullable	Indicates whether the data must be returned or if a null string is allowed if the column does not apply. *Yes* means nulls are allowed, and the data is optional. *No* means that the data is required.

DISCOVER_DATASOURCES Rowset

When the **Discover** method is called with the DISCOVER_DATASOURCES enumeration value in the *RequestType* parameter, it returns the DISCOVER_DATASOURCES rowset in the *Result* parameter. This request type returns a list of published data sources (in an implementation specific way) from a URL of the application Web server, so the client can select one with which to connect.

The client selects a data source by setting the DataSourceInfo property in the *Properties* parameter which is sent with the *Command* parameter *by* the **Execute** method. A client should not construct the contents of the DataSourceInfo property to send to the server. Instead the client should use the **Discover** method to find the data sources that the provider supports. The client then sends back the value for the DataSourceInfo property that it gets from the DISCOVER_DATASOURCES rowset.

Column name	Type	Description	Restriction	Nullable
DataSourceName	string	The name of the data source, such as **FoodMart 2000**.	Yes	No
DataSourceDescription	string	A description of the data source, as entered by the publisher.	No	Yes
URL	string	The unique path that shows where to invoke the XML for Analysis methods for that data source.	Yes	Yes
DataSourceInfo	string	A string containing any additional information required to connect to the data source. This can include the Initial Catalog property or other information for the provider. Example: `"Provider=MSOLAP; Data Source=Local;"`	No	Yes
ProviderName	string	The name of the provider behind the data source. Example:	Yes	Yes

"MSDASQL"

Column name (*continued*)	Type (*continued*)	Description (*continued*)	Restriction (*continued*)	Nullable (*continued*)
ProviderType	array	The types of data supported by the provider. May include one or more of the following types. Example follows this table. **TDP**: tabular data provider. **MDP**: multidimensional data provider. **DMP**: data mining provider. A DMP provider implements the OLE DB for Data Mining specification.	Yes	No

Column name (*continued*)	Type (*continued*)	Description (*continued*)	Restriction (*continued*)	Nullable (*continued*)
AuthenticationMode	EnumString	Specification of what type of security mode the data source uses. Values can be one of the following: **Unauthenticated**: no user ID or password needs to be sent. **Authenticated**: User ID and Password must be included in the information required for the connection. **Integrated**: the data source uses the underlying security to determine authorization, such as Integrated Security provided by Microsoft Internet Information Services (IIS).	Yes	No

The *ProviderType* array has an element for each type that the provider supports. For instance, a provider that supported TDP, MDP, and DMP produces the following array:

```
<ProviderType><MDP/><TDP/><DMP/></ProviderType>
```

DISCOVER_PROPERTIES Rowset

When the **Discover** method is called with the DISCOVER_PROPERTIES enumeration value in the *RequestType* parameter, it returns the DISCOVER_PROPERTIES rowset in the *Result* parameter. This request type returns information about the standard and provider-specific properties supported by an XML for Analysis Provider. Properties that are not supported by a provider are not listed in the return result set.

Column name	Type	Description	Restriction	Nullable
PropertyName	string	The name of the property.	Yes, as an array	No
PropertyDescription	string	A localizable text description of the property.	No	Yes
PropertyType	string	The XML data type of the property.	No	Yes
PropertyAccessType	EnumString	Access for the property. The value can be Read, Write, or ReadWrite.	No	No
IsRequired	boolean	True if a property is required, false if it is not required.	No	Yes
Value	string	The current value of the property.	No	Yes

DISCOVER_SCHEMA_ROWSETS Rowset

When the **Discover** method is called with the DISCOVER_SCHEMA_ROWSETS enumeration value in the *RequestType* parameter, it returns the DISCOVER_SCHEMA_ROWSETS rowset in the *Result* parameter. This request type retrieves a list of all **RequestTypes** enumeration values supported by the provider.

Column name	Type	Description	Restriction	Nullable
SchemaName	String array	The name of the schema/request. This returns the values in the **RequestTypes** enumeration, plus any additional types supported by the provider. The provider defines rowset structures for the additional types.	Yes	No
Restrictions	array	An array of the restrictions supported by provider. An example follows this table.	No	Yes
Description	string	A localizable description of the schema.	No	Yes

The result returned in the restrictions array might look like the following, for a provider that supported three restrictions for the DBSCHEMA_MEMBERS schema rowset. The elements refer to column names in the schema.

```
<Restrictions>
    <CATALOG_NAME type="string" />
    <SCHEMA_NAME type="string" />
    <CUBE_NAME type="string" />
</Restrictions>
```

The following table indicates which OLE DB schema rowsets are required for XML for Analysis tabular data providers and multidimensional data providers. In some cases, some columns within the schema rowsets, which are required for OLE DB for OLAP providers, are optional for XML for Analysis providers. These schema rowsets are indicated with an asterisk (*) in the following table; the details of the optional columns are listed following this table.

OLE DB schema rowset	Required for	Description
DBSCHEMA_CATALOGS	TDP, MDP, DMP	Catalogs available for a server instance of the provider
DBSCHEMA_COLUMNS	TDP, DMP	Enumeration of the columns of tables
DBSCHEMA_PROVIDER_TYPES	TDP, DMP	Enumeration of the base data types supported by the provider
DBSCHEMA_TABLES	TDP, DMP	Enumeration of tables in the catalog
DBSCHEMA_TABLES_INFO	TDP, DMP	Enumeration of tables in the catalog
MDSCHEMA_ACTIONS	MDP	Enumeration of available actions
MDSCHEMA_CUBES	MDP	Enumeration of cubes in catalog
MDSCHEMA_DIMENSIONS	MDP	Enumeration of dimensions for all cubes
MDSCHEMA_FUNCTIONS*	MDP	Enumeration of MDX functions supported by provider
MDSCHEMA_HIERARCHIES*	MDP	Enumeration of hierarchies in all dimensions
MDSCHEMA_MEASURES	MDP	Enumeration of measures in all cubes
MDSCHEMA_MEMBERS*	MDP	Enumeration of all members in all dimensions of all cubes
MDSCHEMA_PROPERTIES*	MDP	Enumeration of user-defined properties available for cells and members
MDSCHEMA_SETS	MDP	Enumeration of available sets in a catalog.

The schema rowsets marked with an asterisk (*) in the above table have columns that, although required for OLE DB for OLAP providers, are considered optional for XML for Analysis providers. These optional columns are listed in the following table.

OLE DB schema rowset	OLE DB required columns that are optional for XML for Analysis providers
MDSCHEMA_FUNCTIONS	ORIGIN, INTERFACE_NAME
MDSCHEMA_HIERARCHIES	STRUCTURE
MDSCHEMA_MEMBERS	LEVEL_UNIQUE_NAME, LEVEL_NUMBER, PARENT_LEVEL
MDSCHEMA_PROPERTIES	LEVEL_UNIQUE_NAME

The OLE DB for OLAP MDSCHEMA_LEVELS schema rowset is not required of XML for Analysis MDP providers, although providers may optionally choose to support it. Therefore, columns that reference levels in other schema rowsets have also become optional, as stated above. This is because different multidimensional providers use the term *level* with different meanings (some providers number them from the top down and others from the bottom up). It is envisioned that additional schema rowsets for levels will be added in a future version of this specification.

DISCOVER_ENUMERATORS Rowset

When the **Discover** method is called with the DISCOVER_ENUMERATORS enumeration value in the *RequestType* parameter, it returns the DISCOVER_ENUMERATORS rowset in the *Result* parameter. This request type queries a provider for supported enumerators, including data types and values. By supporting this request, a provider publishes all enumeration constants that it recognizes.

For each enumerator, there are multiple elements, one for each value in the enumeration. The rowset that represents this is flat, and the name of the enumerator may be repeated for elements belonging to the same enumeration.

Column name	Type	Description	Restriction	Nullable
EnumName	string	The name of the enumerator that contains a set of values.	Yes, as an array	No
EnumDescription	string	A localizable description of the enumerator.	No	Yes
EnumType	string	The data type of the Enum values.	No	No
ElementName	string	The name of one of the value elements in the enumerator set. Example: TDP	No	No
ElementDescription	string	A localizable description of the element (optional).	No	Yes
ElementValue	string	The value of the element. Example: 01	No	Yes

DISCOVER_KEYWORDS Rowset

When the **Discover** method is called with the DISCOVER_KEYWORDS enumeration value in the *RequestType* parameter, it returns the DISCOVER_KEYWORDS rowset in the *Result* parameter. This request type lists keywords reserved by a provider.

Each keyword returned is a row in the DISCOVER_KEYWORDS rowset.

Column name	Type	Description	Restriction	Nullable
Keyword	string	A list of all the keywords reserved by a provider. Example: AND	Yes, as an array	No

DISCOVER_LITERALS Rowset

When the **Discover** method is called with the DISCOVER_LITERALS enumeration value in the *RequestType* parameter, the DISCOVER_LITERALS rowset is returned in the *Result* parameter. This request type queries a provider for information on supported literals, including data types and values.

Each literal returned is a row in the DISCOVER_LITERALS rowset.

Column name	Type	Description	Restriction	Nullable
LiteralName	string	The name of the literal described in the row. Example: DBLITERAL_LIKE_PERCENT	Yes, as an array	No
LiteralValue	string	Contains the actual literal value. Example, if **LiteralName** is DBLITERAL_LIKE_PERCENT and the percent character (%) is used to match zero or more characters in a LIKE clause, this column's value would be "%".	No	Yes
LiteralInvalidChars	string	The characters, in the literal, that are not valid. For example, if table names can contain anything other than a numeric character, this string would be "0123456789".	No	Yes
LiteralInvalidStartingChars	string	The characters that are not valid as the first character of the literal. If the literal can start with any valid character, this is null.	No	Yes
LiteralMaxLength	integer	The maximum number of characters in the literal. If there is no maximum or the maximum is unknown, the value is –1.	No	Yes

XML for Analysis Properties

This section describes the properties required for XML for Analysis.

Column	Contents
Name	The name of the property.
Type	The data type of the property. For more information about data types used in this specification, see "Data Types Used in XML for Analysis."
R/W	The read/write behavior of the property.
Default value	The default value of the property.
Usage	The method (and *RequestType*, if appropriate) or methods in which the property can be used.
Description	A basic description of the behavior of the property.

The following table shows specific information for each property.

Name	Type	R/W	Default value	Usage	Description
AxisFormat	Enumeration	W		**Execute method**	Client asks for the MDDataSet axis to be formatted in one of these ways: TupleFormat, ClusterFormat, CustomFormat.
BeginRange*	int	W	-1	**Execute** method	An integer value corresponding to a CellOrdinal used to restrict an MDDataSet returned by a command to a specific range of cells. Used in conjunction with the **EndRange** property. If unspecified, all cells are returned in the rowset. The value -1 means unspecified.

Name (*continued*)	Type (*continued*)	R/W	Default value (*continued*)	Usage (*continued*)	Description (*continued*)
Catalog	string	R/W	Empty string	**Discover** method **Execute** method	Specifies the initial catalog or database on which to connect.
Content	EnumString	W	Schema Data	**Discover** method **Execute** method	An enumerator that specifies what type of data is returned in the result set. None: Allows the structure of the command to be verified, but not executed. Analogous to using Prepare to check syntax, and so on. Schema: Contains the XML schema (which indicates column information, and so on) that relates to the requested query. Data: Contains only the data that was requested. SchemaData: Returns both the schema information as well as the data.
Cube	string	R/W	Empty string	**Execute**	The cube context for the *Command* parameter. If the command contains a cube name (such as an MDX FROM clause) the setting of this property is ignored.
DataSourceInfo	string	R/W	Empty string	**Discover** method **Execute** method	A string containing provider specific information, required to access the data source.

Name (*continued*)	**Type (*continued*)**	**R/W**	**Default value (*continued*)**	**Usage (*continued*)**	**Description (*continued*)**
EndRange *	int	W	-1	**Execute** method	An integer value corresponding to a CellOrdinal used to restrict an MDDataSet returned by a command to a specific range of cells. Used in conjunction with the BeginRange property. If unspecified, all cells are returned in the rowset. The value -1 means unspecified.
Format	EnumString	W	Native	**Discover** method **Execute** method	Enumerator that determines the format of the returned result set. Values include: Tabular: a flat or hierarchical rowset. Similar to the XML RAW format in SQL. The Format property should be set to Tabular for OLE DB for Data Mining commands. Multidimensional: Indicates that the result set will use the MDDataSet format (**Execute** method only). Native: The client does not request a specific format, so the provider may return the format appropriate to the query. (The actual result type is identified by namespace of the result.)

Name (*continued*)	Type (*continued*)	R/W	Default value (*continued*)	Usage (*continued*)	Description (*continued*)
LocaleIdentifier	unsignedInt	R/W	None	**Discover** method **Execute** method	Use this to read or set the numeric locale identifier for this request. The default is provider-specific. For the complete hexadecimal list of language identifiers, search on "Language Identifiers" in the MSDN® Library at http://www.msdn.microsoft.com.
MDXSupport	EnumString	R	Core	**Discover** method	Enumeration that describes the degree of MDX support. At initial release Core is the only value in the enumeration. In future releases, other values will be defined for this enumeration.
Password	string		Empty string	(Deprecated)	This property is deprecated in XMLA 1.1. To support legacy applications, the provider accepts but ignores the Password property setting when it is used with the **Discover** and **Execute** method
ProviderName	string	R	Empty string	**Discover** method	The XML for Analysis Provider name.
ProviderVersion	string	R	Empty string	**Discover** method	The version number of the XML for Analysis Provider (implementation). The version value should be a four-part format, with each part separated by

Comment: Not sure what "no effect by the provider means." No effect on the provider?" "Does not affect the provider?"

a decimal.

Name (*continued*)	Type (*continued*)	R/W	Default value (*continued*)	Usage (*continued*)	Description (*continued*)
StateSupport	EnumString	R	None	**Discover** method	Property that specifies the degree of support in the provider for state. For information about state in XML for Analysis, see "Support for Statefulness in XML for Analysis." Minimum enumeration values are as follows: None – No support for sessions or stateful operations. Sessions – Provider supports sessions.
Timeout	UnsignedInt	R/W	Undefined	**Discover** method **Execute** method	A numeric time-out specifying in seconds the amount of time to wait for a request to be successful.
UserName	string	R	Empty string	**Discover** method **Execute** method (Deprecated)	Returns the UserName the server associates with the command. This property is deprecated as writeable in XMLA 1.1. To support legacy applications, servers accept but ignore the password setting when it is used with the Execute method.

* The range values for cell coordinates begin at 0 (zero). –1 means undefined, or all in a range.

The following table contains examples of range value pairs and their behavior. In general, the following must be true to return a result set BeginRange <= EndRange. If BeginRange > EndRange, the range is not valid and no results are returned.

BeginRange	EndRange	Behavior
-1	-1	All cells or rows. This is the default behavior.
0	-1	All cells or rows ranging from the first, or 0-th, to undefined, or all at the ending side.
15	-1	Returns Cell 15 through to the end of the dataset.
-1	0	First cell as the first item in the range is undefined (or all), and the ending item is the 0-th element.
15	50	Returns the range of OLAP cells 15 to 50 inclusively. BeginRange <= EndRange.
2	1	No cells will be returned, because the range is not valid (the begin value is greater than the end value). BeginRange > EndRange.

Error Handling in XML for Analysis

Errors are handled differently, depending on their type. The following types of errors can occur:

- Failure to execute a method call
- Success in executing a method call, but with errors or warnings
- Success in executing a method call, but errors within the result set

Failure to execute a method call is reported though a SOAP Fault message. When this occurs, *Result* is not returned. When the method completes with errors or warnings, these are returned to the client with *Result*.

Immediately preceeding the closing `</root>` element, an optional `<Messages>` section may follow all other sections of the result in either of the following two cases:

- The method call itself did not fail, but a server failure occurred after the method callsucceeded.
- The server uses it to return a warning when the command is successful.

The `<Messages>` section contains one or more of the optional elements, `<Error>` or `<Warning>`. The structure of these elements is identical to the elements of the SOAP fault, except that an additional `<WarningCode>` element is available. Either one can contain the attributes ErrorCode (or WarningCode for `<Warning>`), Description, Source, and HelpFile. (For details about these XML attributes, see the table under "SOAP Fault Error Example" later in this document.

A sample Messages section with two warnings follows.

```
<root>
```

```
…
<Messages>
     <Warning
          WarningCode=”1234”
          Description=”description”
          Source=”XML for Analysis Provider”
          HelpFile = “”
     />
     <Warning
          WarningCode=”5678”
          Description=”description”
          Source=”XML for Analysis Provider”
          HelpFile = “”
     />
</Messages>
</root>
```

If the server encounters an error after it begins serializing the output, it should output an `<Exception>` element in a different namespace at the point of the error, as follows:

```
<Exception xmlns=urn:schemas-microsoft-com:xml-analysis:exception>
```

The server should then close all open elements so that the XML document received by the client is valid.. Finally, the server should output the Messages section containing the description of the error followed by the closing </root> element.

MDDataSet Error Example

Errors specific to cells or data in *Result* are embedded within the result set in the appropriate location. The MDDataSet data type is covered in "MDDataSet" under "Data Types Used in XML for Analysis." The following is an example of the result if there is an error in an MDDataSet:

```
<CellData>
 ...
 <Cell CellOrdinal=”10”>
  <Value>
   <Error>
    <ErrorCode>2148497527</ErrorCode>
    <Description>Security Error.</Description>
   </Error>
  </Value>
  </Cell>
</CellData>
```

SOAP Fault Error Example

The SOAP fault codes relating to this specification begin with "XMLForAnalysis" followed by a period and the hexadecimal HR result code. For example, an error code of "`0x80000005`" would be formatted as "`XMLForAnalysis.0x80000005`".

For more information about the SOAP fault format, see http://www.w3.org/TR/SOAP/#_Ref477795996.

The following table shows the XML for Analysis fault code information that is contained in the **detail** section of the SOAP response. The columns are the attributes of an error in the detail section of a SOAP fault.

Column name	Type	Description	Nullable
ErrorCode	UnsignedInt	Return code that indicates the success or failure of the method. Note: The hexadecimal value must be converted to an **Unsignedint** value.	No
Description	String	Error text and description returned by the component that generated the error.	Yes
Source	String	Name of the component that generated the error.	Yes
HelpFile	String	Path or URL to the Help file or topic that describes the error.	Yes

The following is an example of a SOAP fault for a failed method call:

```
<?xml version="1.0"?>
 <SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Fault>
  <faultcode>XMLAnalysisError.0x80000005</faultcode>
  <faultstring>The XML for Analysis provider encountered an
error</faultstring>
  <faultactor>XML for Analysis Provider</faultactor>
  <detail>
```

```
    <Error
    ErrorCode="2147483653"
    Description="An unexpected error has occurred"
    Source="XML for Analysis Provider"
    HelpFile=""
    />
  </detail>
 </SOAP-ENV:Fault>
</SOAP-ENV:Envelope>
```

Support for Statefulness in XML for Analysis

XML for Analysis is stateless by default. (Statelessness is a condition during which the server does not remember the identity or context of a client after a method call is completed). In order to support statefulness (a condition during which the server preserves the identity and context of a client from method invocation to method invocation), sessions are supported for the provider. Sessions are useful for a series of statements that should be performed together. An example of this is the creation of a calculated member that is used in subsequent queries.

Support for sessions on the provider is optional. The client can test for support by inspecting the value of the XML for Analysis property, StateSupport, through the **Discover** method. The minimum value for state to occur is *Sessions*. For more information about the StateSupport property, see "XML for Analysis Properties."

In general, sessions follow the behavior outlined by the OLE DB specification as follows:

- Sessions define transaction and command context scope.
- Multiple commands can be executed in the context of a single session.
- Support for transactions in the XML for Analysis context are handled with provider-specific commands sent with the **Execute** method.

XML for Analysis defines a way to support state (sessions) in a Web environment in a mode similar to the approach used by the Distributed Authoring and Versioning (DAV) protocol to implement locking in a loosely coupled environment. This specification parallels DAV in that the provider is allowed to expire sessions due to various reasons (for example, timeout or connection error). This also means that the Web Services must be aware and ready to handle sets of commands that were interrupted and must be restarted.

The SOAP specification recommends using SOAP headers for building up new protocols on top of SOAP messages. The following table lists the SOAP header elements and attributes that XML for Analysis defines for initiating, maintain, and closing a session.

SOAP header	Description
BeginSession	Requests the provider to create a new session. The provider should respond by constructing a new session and returning the session ID as part of the Session header in the SOAP response.
SessionId	The value area contains the session ID that must be used in each method call for the rest of the session. The provider in the SOAP response sends this tag and the client must also send this attribute with each Session header element.
Session	For every method call that is to occur in the session, this header must be used, and the session ID must be included in the value area of the header.
EndSession	To terminate the session, use this header. The session ID must be included with the value area.

The following example shows how sessions are supported:

1. To begin the session, add a BeginSession header in SOAP to the outbound XML for Analysis method call from the client. The value area is initially blank because the session ID is not yet known.

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Header>
  <XA:BeginSession
   xmlns:XA="urn:schemas-microsoft-com:xml-analysis"
   xsi:type="xsd:int"
   mustUnderstand="1"/>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  ...<!-- Discover or Execute call goes here.-->
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2. The SOAP response message from the provider includes the session ID in the return header area, using the XML for Analysis header tag <SessionId>.

```
<SOAP-ENV:Header>
 <XA:Session
  xmlns:XA="urn:schemas-microsoft-com:xml-analysis"
  SessionId="581"/>
</SOAP-ENV:Header>
```

3. For each method call in the session, the Session header must be added, containing the session ID returned from the provider.

```
<SOAP-ENV:Header>
 <XA:Session
  xmlns:XA="urn:schemas-microsoft-com:xml-analysis"
  mustUnderstand="1"
  SessionId="581"/>
</SOAP-ENV:Header>
```

4. When the session is complete, the <EndSession> tag is used, containing the related session ID value.

```
<SOAP-ENV:Header>
 <XA:EndSession
  xmlns:XA="urn:schemas-microsoft-com:xml-analysis"
  xsi:type="xsd:int"
  mustUnderstand="1"
  SessionId="581"/>
</SOAP-ENV:Header>
```

A session ID does not guarantee that a session remains valid. If the session expires (for example, if it times out or the connection is lost), the provider can choose to end and roll back that session's actions. As a result, all subsequent method calls from the client on a session ID fail with an error signaling a nonvalid session. A client handles this condition and must be prepared to resend the session method calls from the beginning.

XML for Analysis with Data Mining

The XML for Analysis **Execute** method supports the execution of data mining commands. Data mining command support is provider specific. Data mining commands are also supported through the **Execute** interface, if the results are returned either as an **MDDataSet** or in Tabular form. Data mining providers are required to support Tabular results at a minimum.

As an example, accessing OLE DB for Data Mining information through the XML for Analysis API is not significantly different from obtaining data from a relational data source. The result of OLE DB for Data Mining commands is a flat rowset or a flat rowset in a hierarchical arrangement. The Format property should be set to **Tabular** for OLE DB for Data Mining commands.

XML for Analysis providers are not required to support OLE DB for Data Mining-specific commands. Providers that do support OLE DB for data mining will expose themselves by setting one of the array elements in ProviderType to DMP.

Part II – Appendices

This section includes additional topics and related information to help you understand and implement this specification.

Appendix A: Implementation Notes

This appendix includes discussions on implementation considerations.

XML for Analysis Implementation Walkthrough

To better illustrate how the XML for Analysis API is used, this section provides an example walkthrough of a simple client/server interaction. This includes how an implementation can get the list of data sources.

The steps in the walkthrough detail a client communicating with a server to get a list of data sources and then using one of the data sources to run an OLAP query.

The following notation shows which steps represent the specification and which steps are examples of an implementation:

- (I) The step is an example of an implementation.
- (S) The step shown is an item implemented as outlined in this specification.

Finding an XML for Analysis Service

1. (I) The client application is aware of a server URL that supports Web Services. This could be found through browsing a Universal Description, Discovery, and Integration (UDDI) business registry.
2. (I) The client sends a request to the URL to find out whether it supports the XML for Analysis Web Service (for example, using Discovery Protocol (DISCO)).
3. (I) DISCO returns the URL of the WSDL file for the XML for Analysis Web Service (for example, XMLAnalysis.wsdl).
4. (I) The client sends a request for the WSDL file.
5. (I) The Web server sends back XMLAnalysis.wsdl, which defines the supported methods: **Discover** and **Execute**.
6. (I) The client confirms from the WSDL that both client and server use the same methods.
7. (I) The WSDL file contains the URL that should be used for the XML for Analysis Web Service (for example, XMLAnalysis.asp).

The following is an example of the WSDL that a service might return for XML for Analysis:

```
<?xml version='1.0' encoding='UTF-8' ?>
 <!-- Generated 02/08/01 by Microsoft SOAP Toolkit WSDL File Generator,
Version 1.1 -->
<definitions  name ='MSXmlAnalysis'   targetNamespace =
```

```
'http://schemas.microsoft.com/xmla/MSXmlAnalysis/wsdl'
    xmlns:wsdlns='http://schemas.microsoft.com/xmla/MSXmlAnalysis/wsdl'
    xmlns:typens='http://schemas.microsoft.com/xmla/MSXmlAnalysis/type'
    xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
    xmlns:xsd='http://www.w3.org/2000/10/XMLSchema'
    xmlns:stk='http://schemas.microsoft.com/soap-toolkit/wsdl-
extension'
    xmlns='http://schemas.xmlsoap.org/wsdl/'>
  <types>
    <schema targetNamespace='urn:schemas-microsoft-com:xml-analysis'
      xmlns='http://www.w3.org/2000/10/XMLSchema'>
    </schema>
  </types>
  <message name='XmlAnalysis.Execute'>
    <part name='Command' type='xsd:xmldom'/>
    <part name='Properties' type='xsd:xmldom'/>
  </message>
  <message name='XmlAnalysis.ExecuteResponse'>
    <part name='return' type='xsd:xmldom'/>
  </message>
  <message name='XmlAnalysis.Discover'>
    <part name='RequestType' type='xsd:string'/>
    <part name='Restrictions' type='xsd:xmldom'/>
    <part name='Properties' type='xsd:xmldom'/>
  </message>
  <message name='XmlAnalysis.DiscoverResponse'>
    <part name='return' type='xsd:xmldom'/>
  </message>
  <portType name='XmlAnalysisSoapPort'>
    <operation name='Execute' parameterOrder='Command Properties'>
      <input message='wsdlns:XmlAnalysis.Execute' />
      <output message='wsdlns:XmlAnalysis.ExecuteResponse' />
    </operation>
    <operation name='Discover' parameterOrder='RequestType Restrictions
```

```
Properties'>
      <input message='wsdlns:XmlAnalysis.Discover' />
      <output message='wsdlns:XmlAnalysis.DiscoverResponse' />
    </operation>
  </portType>
  <binding name='XmlAnalysisSoapBinding'
type='wsdlns:XmlAnalysisSoapPort' >
    <stk:binding preferredEncoding='UTF-8'/>
    <soap:binding style='rpc'
transport='http://schemas.xmlsoap.org/soap/http' />
    <operation name='Execute' >
      <soap:operation soapAction='urn:schemas-microsoft-com:xml-
analysis:Execute' />
      <input>
        <soap:body use='encoded' namespace='urn:schemas-microsoft-
com:xml-analysis'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:body use='encoded' namespace='urn:schemas-microsoft-
com:xml-analysis'
            encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
/>
      </output>
    </operation>
    <operation name='Discover' >
      <soap:operation soapAction='urn:schemas-microsoft-com:xml-
analysis:Discover' />
      <input>
        <soap:body use='encoded' namespace='urn:schemas-microsoft-
com:xml-analysis'
           encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:body use='encoded' namespace='urn:schemas-microsoft-
com:xml-analysis'
```

```
            encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
/>
      </output>
    </operation>
  </binding>
  <service name='MSXmlAnalysis' >
    <port name='XmlAnalysisSoapPort'
binding='wsdlns:XmlAnalysisSoapBinding' >
      <soap:address location='http://localhost/xmla/msxisapi.dll' />
    </port>
  </service>
</definitions>
```

Obtaining a Data Source

1. (S) The client looks for OLAP data sources by using the URL and method information obtained above to send a **Discover** call of *RequestType* DISCOVER_DATASOURCES. In the *Restrictions* parameter, it specifies MDP (multidimensional data) for **ProviderType**.
2. (I) The list of published data sources is an XML file on the server, maintained by the application administrator. XMLAnalysis.asp retrieves the XML document, and sends it to the client application.
3. (I) The client parses the rowset, and selects the data source to use. If the URL listed for the data source is different from the URL first used for the original **Discover** method, then the second, data-source-specific, URL must be used for all further **Discover** and **Execute** calls to work with that data on that source.

Using the Data Source

1. (S) The client sends a **Discover** command to the chosen OLAP data source. To obtain a list of all cubes that are available, use the *RequestType* MDSCHEMA_CUBES. *Restrictions* contains the name of a database to search for **FoodMart 2000**. The information needed to connect to the provider is included in the *Properties* parameter.

4. The following is a sample of the XML sent from the client for this call:

```
SOAPAction: "urn:schemas-microsoft-com:xml-analysis:Discover"
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
  <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <RequestType>MDSCHEMA_CUBES</RequestType>
    <Restrictions>
      <RestrictionList>
      <CATALOG_NAME>
       FoodMart 2000
      </CATALOG_NAME>
     </RestrictionList>
    </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=MSOLAP;Data Source=local;
     </DataSourceInfo>
     <Catalog>
      Foodmart 2000
     </Catalog>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2. (I) The XML for Analysis provider processes the request and sends it to the OLAP data source. After the available cube data is received, the provider packages it as XML and sends it back to the requesting client application.
5. The following is a sample of the XML sent from the server with the data:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <DiscoverResponse xmlns="urn:schemas-microsoft-com:xml-analysis">
   <return>
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset">
     <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <!-- The XML schema definition of the result comes here -->
      ...
     </xsd:schema>
     <row>
      <CATALOG_NAME>FoodMart 2000</CATALOG_NAME>
      <CUBE_NAME>Sales</CUBE_NAME>
      ...
     </row>
     <row>
      <CATALOG_NAME>FoodMart 2000</CATALOG_NAME>
      <CUBE_NAME>Warehouse</CUBE_NAME>
      ...
     </row>
     ...
    </root>
   </return>
  </DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3. (S) The client chooses a cube, and then it sends an **Execute** containing a `<Statement>` element that contains an MDX SELECT statement: `"Select Measures.Members on Columns from Sales"`. Connection information is again provided in the *Properties* parameter.

```
SOAPAction: "urn:schemas-microsoft-com:xml-analysis:Execute"
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
  <Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <Command>
<Statement>
select [Measures].members on Columns from Sales
</Statement>
</Command>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=MSOLAP;Data Source=local;
     </DataSourceInfo>
     <Catalog>
      Foodmart 2000
     </Catalog>
     <Format>
      Multidimensional
     </Format>
       <AxisFormat>
         TupleFormat
      </AxisFormat>
    </PropertyList>
   </Properties>
  </Execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4. (I) The XML for Analysis provider parses the request and sends it to the data source to be filled.
6. (I) When the dataset is returned, the provider packages it to XML, and sends it back to the requesting client application as illustrated in the following example:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <ExecuteResponse xmlns="urn:schemas-microsoft-com:xml-analysis">
   <return>
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset">
     <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <!-- The XML schema definition of the result comes here -->
      ...
     </xsd:schema>
     <OlapInfo>
      <!—  Dimension information comes here  -->
     </OlapInfo>
     <Axes>
      <!—  Axis information for the MDDataSet result axes comes here
-->
     </Axes>
     <CellData>
      <!—  Cell data values and properties come here  -->
     </CellData>
    </root>
   </return>
  </ExecuteResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5. (I) The client receives the result set, and performs any additional tasks required by the user. For example, a client application could format the result set with XSL, and present the data as a table that the user can browse on a Web page. A client application can also cache the result set locally to minimize roundtrips when the user refreshes displayed data.

XML for Analysis and Non-Web Applications

The XML for Analysis API is optimized for Web applications. However, this does not prevent this methodology from being leveraged for LAN-oriented applications. The following applications can benefit from this XML-based API:

- Client/server applications that require technology flexibility between clients and the server
- Client/server applications that target multiple operating systems
- Clients that do not require significant state in order to increase server capacity

Appendix B: Quick SOAP Glossary

Simple Object Access Protocol (SOAP) is an industry standard for using XML to represent data and commands in an extensible way. Because SOAP is an integral part of this specification, this section provides background information and defines SOAP terminology.

Web Service

Broadly speaking, a Web Service is an application delivered as a service that can be integrated with other Web Services using Internet standards. It is a URL-addressable resource that programmatically returns information to clients who want to use it.

Web Services represent black-box functionality that can be reused without having to deal specifically with how a service is implemented. Web Services provide well-defined interfaces, called contracts, which describe the provided services. A Web Service can use SOAP to specify its message formats.

SOAP

SOAP is a lightweight protocol for exchanging information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it
- A set of encoding rules for expressing instances of application-defined data types
- A convention for representing remote procedure calls and responses

Here is a sample SOAP request:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:GetLastTradePrice xmlns:m="Some-URI">
   <symbol>DIS</symbol>
  </m:GetLastTradePrice>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The corresponding data response might look like this:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse xmlns:m="Some-URI">
   <Price>34.5</Price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web Services Description Language (WSDL)

As communications protocols and message formats are standardized in the Web community, it becomes increasingly possible and important to be able to describe the communications in a structured way. Web Services Description Language (WSDL) addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions:

- Messages, which are abstract descriptions of the data being exchanged
- Port types, which are abstract collections of operations

The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding; a collection of ports defines a service.

For a link to more complete information about WSDL, see Appendix E.

Appendix C: XML for Analysis to OLE DB Mapping

As XML for Analysis builds on definitions outlined in OLE DB, further information can be gained by referring to the OLE DB specification for the areas mentioned.

Function Mapping

The following table maps OLE DB and OLE DB for OLAP to equivalent XML for Analysis actions. Not all OLE DB functions have an XML for Analysis mapping, because not all actions apply. For example, methods that target navigational methods, which would be used to manipulate a rowset after it is received, do not apply, because the client handles that.

OLE DB interface and command	XML implementation
IColumnsInfo::GetColumnInfo	**Execute** method *Properties*: Content = Schema
ICommandPrepare::Prepare	**Execute** method *Properties*: Content = None (In order to validate a command only)
ICommandProperties::GetProperties	**Discover** method *RequestType*: DISCOVER_PROPERTIES
ICommandProperties::SetProperties	**Execute** method *Properties*: <Specify the property to be updated> Note: The property must have read/write permissions.
ICommandText::SetCommandText	**Execute** method *Command* parameter.
IDBInfo::GetKeywords	**Discover** method *RequestType*: DISCOVER_KEYWORDS
IDBInfo::GetLiteralInfo	**Discover** method *RequestType*: DISCOVER_LITERALS

OLE DB interface and command (*continued*)	XML implementation (*continued*)
IDBProperties::GetProperties	**Discover** method *RequestType*: DISCOVER_PROPERTIES
IDBProperties::GetPropertyInfo	**Discover** method *RequestType*: DISCOVER_PROPERTIES
IDBProperties::SetProperties	**Execute** method *Properties*: <Specify the property to be updated> Note: The property must have read/write permissions.
IGetDataSource::GetDataSource	**Discover** method *RequestType* is DISCOVER_DATASOURCES
IMDDataset::FreeAxisInfo	(Not a direct mapping)
IMDDataset::GetAxisRowset **IMDDataset::GetCellData** **IMDDataset::GetSpecification**	To obtain the meta data structure (rows and columns, and so on, information) of the rowset, use the **Execute** method: *Properties:* Content = Schema
IMDRangeRowset::GetRangeRowset	(Not a direct mapping.) **Execute** method *Commands*: <Provider-specific query statement> *Properties*: BeginRange = *integer value* EndRange = *integer value*
ISessionProperties::GetProperties	**Discover** method *RequestType*: DISCOVER_PROPERTIES (Session is implied)
ISessionProperties::SetProperties	**Execute** method *Properties*: <Specify the property to be updated> Note: The property must have read/write permissions.

Properties Mapping

The following table lists some common OLE DB properties to equivalent XML for Analysis properties.

OLE DB property	XML implementation
DBPROP_INIT_PROVIDERSTRING (**DBPROPSET_DBINIT** Property Set)	DataSourceInfo property
DBPROP_COMMANDTIMEOUT **DBPROP_INIT_TIMEOUT** **DBPROP_INIT_GENERALTIMEOUT** (**DBPROPSET_DBINIT** Property Set)	Timeout property

RequestTypes Mapping

This table shows the mapping to OLE DB of the **RequestTypes** enumeration values.

OLE DB mapping	XML for Analysis request type
Not applicable	DISCOVER_DATASOURCES
IDBProperties::GetPropertyInfo and **IDBProperties::GetProperties** functions	DISCOVER_PROPERTIES
IDBSchemaRowset::GetSchemas	DISCOVER_SCHEMA_ROWSETS
Not applicable	DISCOVER_ENUMERATORS
IDBInfo::GetKeywords	DISCOVER_KEYWORDS
IDBInfo::GetLiteralInfo	DISCOVER_LITERALS
The OLE DB schema rowset names and definitions are listed in "Appendix B: Schema Rowsets" in the OLE DB specification.	<Schema Rowset Constant>

OLE DB to XML Data Type Mapping

For reference, the following table maps OLE DB data types to published XML schema types.

More information and definitions of the XML schema types are available on http://www.w3.org/TR/xmlschema-2/. To view the XML schema structure, see the W3C Web site.

OLE DB type	XML schema type
DBTYPE_I1	byte
DBTYPE_I2	short
DBTYPE_I4	int
DBTYPE_I8	long
DBTYPE_UI1	unsignedByte
DBTYPE_UI2	unsignedShort
DBTYPE_UI4	unsignedInt
DBTYPE_UI8	unsignedLong
DBTYPE_R4	float
DBTYPE_R8	double
DBTYPE_BOOL	boolean
DBTYPE_CY	decimal
DBTYPE_ERROR	string
DBTYPE_DECIMAL	decimal
DBTYPE_NUMERIC	decimal
DBTYPE_DATE	date
DBTYPE_DBTIMESTAMP	time
DBTYPE_GUID	string
DBTYPE_BYTES	binary
DBTYPE_STR	string
DBTYPE_WSTR	string
DBTYPE_BSTR	string
DBTYPE_VARIANT	string

MDDataSet Data Type Mapping to OLE DB

This is the OLE DB mapping for the XML for Analysis data type of **MDDataset** and further reference information.

OLE DB implementation	XML for Analysis implementation
OLE DB for OLAP dataset type Accessed through the **IMDDataset** interface	MDDataset data type

Relationship between MDX and mdXML

Multidimensional Expressions (MDX) is the multidimensional expression language defined in the OLE DB for OLAP specification. The mdXML language is an XML-encapsulated version of the MDX language. As of the initial release of this specification, the only XML element for mdXML is the `<Statement>` element, which for multidimensional providers consists of an MDX language statement, described previously in this specification. In the future mdXML will be extended to have additional elements and features. The extensions to mdXML will be based on the MDX language, which will continue to remain available in XML for Analysis, via the `<Statement>` element.

The MDX language itself is extensible so that providers can add extensions to the language to support additional features not provided in the base language set. A future version of this specification will define an mdXML language based upon MDX.

The MDX language specification is part of the OLE DB for OLAP specification and can be found at the location referenced for OLE DB for OLAP information in Appendix E.

Appendix D: MDDataSet Example

The following is a complete example of an MDDataSet reply result set, with AxisFormat=TupleFormat. The XSD for the MDDataSet is available as a separate file. See http://xmla.org.

```
<?xml version="1.0" encoding="UTF-16"?>
<root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

The **MDDataSet** query result is shown below.

```
    <OlapInfo>
        <CubeInfo>
            <Cube>
                  <CubeName>Sales</CubeName>
            </Cube>
         </CubeInfo>
        <AxesInfo>
            <AxisInfo name="Axis0">
                <HierarchyInfo name="Measures">
                    <UName name="[Measures].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Measures].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Measures].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Measures].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Measures].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
            </AxisInfo>
            <AxisInfo name="Axis1">
                <HierarchyInfo name="Store">
                    <UName name="[Store].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Store].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Store].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Store].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Store].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Time">
                    <UName name="[Time].[MEMBER_UNIQUE_NAME]"></UName>
```

<Caption

```
name="[Time].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Time].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Time].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Time].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
            </AxisInfo>
            <AxisInfo name="SlicerAxis">
                <HierarchyInfo name="Product">
                    <UName name="[Product].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Product].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Product].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Product].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Product].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Promotion Media">
                    <UName name="[Promotion
Media].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Promotion
Media].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Promotion
Media].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Promotion Media].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo name="[Promotion
Media].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Promotions">
                    <UName
```

```
name="[Promotions].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption
name="[Promotions].[MEMBER_CAPTION]"></Caption>
                    <LName
name="[Promotions].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Promotions].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Promotions].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Customers">
                    <UName
name="[Customers].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption
name="[Customers].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Customers].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Customers].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Customers].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Education Level">
                    <UName name="[Education
Level].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Education
Level].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Education
Level].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Education Level].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo name="[Education
Level].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Gender">
                    <UName
```

```
name="[Gender].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Gender].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Gender].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Gender].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo
name="[Gender].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Marital Status">
                    <UName name="[Marital
Status].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Marital
Status].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Marital
Status].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Marital Status].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo name="[Marital
Status].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Store Size in SQFT">
                    <UName name="[Store Size in
SQFT].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Store Size in
SQFT].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Store Size in
SQFT].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Store Size in
SQFT].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo name="[Store Size in
SQFT].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Store Type">
                    <UName name="[Store
```

```
Type].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Store
Type].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Store
Type].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Store Type].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo name="[Store
Type].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
                <HierarchyInfo name="Yearly Income">
                    <UName name="[Yearly
Income].[MEMBER_UNIQUE_NAME]"></UName>
                    <Caption name="[Yearly
Income].[MEMBER_CAPTION]"></Caption>
                    <LName name="[Yearly
Income].[LEVEL_UNIQUE_NAME]"></LName>
                    <LNum name="[Yearly Income].[LEVEL_NUMBER]"></LNum>
                    <DisplayInfo name="[Yearly
Income].[DISPLAY_INFO]"></DisplayInfo>
                </HierarchyInfo>
            </AxisInfo>
        </AxesInfo>
        <CellInfo>
            <Value name="VALUE"></Value>
            <FmtValue name="FORMATTED_VALUE"></FmtValue>
        </CellInfo>
    </OlapInfo>
```

```
<Axes>
    <Axis name="Axis0">
        <Tuples>
            <Tuple>
                <Member Hierarchy="Measures">
                    <UName>[Measures].[Unit Sales]</UName>
                    <Caption>Unit Sales</Caption>
                    <LName>[Measures].[MeasuresLevel]</LName>
                    <LNum>0</LNum>
                    <DisplayInfo>131072</DisplayInfo>
                </Member>
            </Tuple>
            <Tuple>
                <Member Hierarchy="Measures">
                    <UName>[Measures].[Store Cost]</UName>
                    <Caption>Store Cost</Caption>
                    <LName>[Measures].[MeasuresLevel]</LName>
                    <LNum>0</LNum>
                    <DisplayInfo>131072</DisplayInfo>
                </Member>
            </Tuple>
            <Tuple>
                <Member Hierarchy="Measures">
                    <UName>[Measures].[Store Sales]</UName>
                    <Caption>Store Sales</Caption>
                    <LName>[Measures].[MeasuresLevel]</LName>
                    <LNum>0</LNum>
                    <DisplayInfo>131072</DisplayInfo>
                </Member>
            </Tuple>
```

```
        <Tuple>
            <Member Hierarchy="Measures">
                <UName>[Measures].[Sales Count]</UName>
                <Caption>Sales Count</Caption>
                <LName>[Measures].[MeasuresLevel]</LName>
                <LNum>0</LNum>
                <DisplayInfo>131072</DisplayInfo>
            </Member>
        </Tuple>
    </Tuples>
</Axis>
<Axis name="Axis1">
    <Tuples>
        <Tuple>
            <Member Hierarchy="Store">
                <UName>[Store].[All Stores].[USA].[CA]</UName>
                <Caption>CA</Caption>
                <LName>[Store].[Store State]</LName>
                <LNum>2</LNum>
                <DisplayInfo>131077</DisplayInfo>
            </Member>
            <Member Hierarchy="Time">
                <UName>[Time].[1997].[Q1]</UName>
                <Caption>Q1</Caption>
                <LName>[Time].[Quarter]</LName>
                <LNum>1</LNum>
                <DisplayInfo>131075</DisplayInfo>
            </Member>
        </Tuple>
        <Tuple>
            <Member Hierarchy="Store">
                <UName>[Store].[All Stores].[USA].[CA]</UName>
                <Caption>CA</Caption>
                <LName>[Store].[Store State]</LName>
                <LNum>2</LNum>
                <DisplayInfo>131077</DisplayInfo>
```

```
        </Member>
        <Member Hierarchy="Time">
            <UName>[Time].[1997].[Q2]</UName>
            <Caption>Q2</Caption>
            <LName>[Time].[Quarter]</LName>
            <LNum>1</LNum>
            <DisplayInfo>131075</DisplayInfo>
        </Member>
    </Tuple>
    <Tuple>
        <Member Hierarchy="Store">
            <UName>[Store].[All Stores].[USA].[CA]</UName>
            <Caption>CA</Caption>
            <LName>[Store].[Store State]</LName>
            <LNum>2</LNum>
            <DisplayInfo>131077</DisplayInfo>
        </Member>
        <Member Hierarchy="Time">
            <UName>[Time].[1997].[Q3]</UName>
            <Caption>Q3</Caption>
            <LName>[Time].[Quarter]</LName>
            <LNum>1</LNum>
            <DisplayInfo>131075</DisplayInfo>
        </Member>
    </Tuple>
    <Tuple>
        <Member Hierarchy="Store">
            <UName>[Store].[All Stores].[USA].[CA]</UName>
            <Caption>CA</Caption>
            <LName>[Store].[Store State]</LName>
            <LNum>2</LNum>
            <DisplayInfo>131077</DisplayInfo>
        </Member>
        <Member Hierarchy="Time">
            <UName>[Time].[1997].[Q4]</UName>
            <Caption>Q4</Caption>
```

```
            <LName>[Time].[Quarter]</LName>
            <LNum>1</LNum>
            <DisplayInfo>131075</DisplayInfo>
        </Member>
    </Tuple>
    <Tuple>
        <Member Hierarchy="Store">
            <UName>[Store].[All Stores].[USA].[OR]</UName>
            <Caption>OR</Caption>
            <LName>[Store].[Store State]</LName>
            <LNum>2</LNum>
            <DisplayInfo>131074</DisplayInfo>
        </Member>
        <Member Hierarchy="Time">
            <UName>[Time].[1997].[Q1]</UName>
            <Caption>Q1</Caption>
            <LName>[Time].[Quarter]</LName>
            <LNum>1</LNum>
            <DisplayInfo>131075</DisplayInfo>
        </Member>
    </Tuple>
    <Tuple>
        <Member Hierarchy="Store">
            <UName>[Store].[All Stores].[USA].[OR]</UName>
            <Caption>OR</Caption>
            <LName>[Store].[Store State]</LName>
            <LNum>2</LNum>
            <DisplayInfo>131074</DisplayInfo>
        </Member>
        <Member Hierarchy="Time">
            <UName>[Time].[1997].[Q2]</UName>
            <Caption>Q2</Caption>
            <LName>[Time].[Quarter]</LName>
            <LNum>1</LNum>
            <DisplayInfo>131075</DisplayInfo>
        </Member>
```

```
        </Tuple>
        <Tuple>
            <Member Hierarchy="Store">
                <UName>[Store].[All Stores].[USA].[OR]</UName>
                <Caption>OR</Caption>
                <LName>[Store].[Store State]</LName>
                <LNum>2</LNum>
                <DisplayInfo>131074</DisplayInfo>
            </Member>
            <Member Hierarchy="Time">
                <UName>[Time].[1997].[Q3]</UName>
                <Caption>Q3</Caption>
                <LName>[Time].[Quarter]</LName>
                <LNum>1</LNum>
                <DisplayInfo>131075</DisplayInfo>
            </Member>
        </Tuple>
        <Tuple>
            <Member Hierarchy="Store">
                <UName>[Store].[All Stores].[USA].[OR]</UName>
                <Caption>OR</Caption>
                <LName>[Store].[Store State]</LName>
                <LNum>2</LNum>
                <DisplayInfo>131074</DisplayInfo>
            </Member>
            <Member Hierarchy="Time">
                <UName>[Time].[1997].[Q4]</UName>
                <Caption>Q4</Caption>
                <LName>[Time].[Quarter]</LName>
                <LNum>1</LNum>
                <DisplayInfo>131075</DisplayInfo>
            </Member>
        </Tuple>
    </Tuples>
</Axis>
<Axis name="SlicerAxis">
```

```
<Tuples>
    <Tuple>
        <Member Hierarchy="Product">
            <UName>[Product].[All Products]</UName>
            <Caption>All Products</Caption>
            <LName>[Product].[(All)]</LName>
            <LNum>0</LNum>
            <DisplayInfo>3</DisplayInfo>
        </Member>
        <Member Hierarchy="Promotion Media">
            <UName>[Promotion Media].[All Media]</UName>
            <Caption>All Media</Caption>
            <LName>[Promotion
```

```
Media].[(All)]</LName>
                        <LNum>0</LNum>

                        <DisplayInfo>14</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Promotions">
                        <UName>[Promotions].[All Promotions]</UName>
                        <Caption>All Promotions</Caption>
                        <LName>[Promotions].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>51</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Customers">
                        <UName>[Customers].[All Customers]</UName>
                        <Caption>All Customers</Caption>
                        <LName>[Customers].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>3</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Education Level">
                        <UName>[Education Level].[All Education
Level]</UName>
                        <Caption>All Education Level</Caption>
                        <LName>[Education Level].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>5</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Gender">
                        <UName>[Gender].[All Gender]</UName>
                        <Caption>All Gender</Caption>
                        <LName>[Gender].[(All)]</LName>
```

```
                        <LNum>0</LNum>
                        <DisplayInfo>2</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Marital Status">
                        <UName>[Marital Status].[All Marital
Status]</UName>
                        <Caption>All Marital Status</Caption>
                        <LName>[Marital Status].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>2</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Store Size in SQFT">
                        <UName>[Store Size in SQFT].[All Store Size in
SQFT]</UName>
                        <Caption>All Store Size in SQFT</Caption>
                        <LName>[Store Size in SQFT].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>21</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Store Type">
                        <UName>[Store Type].[All Store Type]</UName>
                        <Caption>All Store Type</Caption>
                        <LName>[Store Type].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>6</DisplayInfo>
                    </Member>
                    <Member Hierarchy="Yearly Income">
                        <UName>[Yearly Income].[All Yearly Income]</UName>
                        <Caption>All Yearly Income</Caption>
                        <LName>[Yearly Income].[(All)]</LName>
                        <LNum>0</LNum>
                        <DisplayInfo>8</DisplayInfo>
                    </Member>
                </Tuple>
            </Tuples>
        </Axis>
```

```
</Axes>
<CellData>
    <Cell CellOrdinal="0">
        <Value xsi:type="xsd:double">16890</Value>
        <FmtValue>16,890.00</FmtValue>
    </Cell>
    <Cell CellOrdinal="1">
        <Value xsi:type="xsd:double">14431.0851</Value>
        <FmtValue>14,431.09</FmtValue>
    </Cell>
    <Cell CellOrdinal="2">
        <Value xsi:type="xsd:double">36175.2</Value>
        <FmtValue>$36,175.20</FmtValue>
    </Cell>
    <Cell CellOrdinal="3">
        <Value xsi:type="xsd:int">5498</Value>
        <FmtValue>5498</FmtValue>
    </Cell>
    <Cell CellOrdinal="4">
        <Value xsi:type="xsd:double">18052</Value>
        <FmtValue>18,052.00</FmtValue>
    </Cell>
    <Cell CellOrdinal="5">
        <Value xsi:type="xsd:double">15332.0164</Value>
        <FmtValue>15,332.02</FmtValue>
    </Cell>
```

```
<Cell CellOrdinal="6">
    <Value xsi:type="xsd:double">38396.75</Value>
    <FmtValue>$38,396.75</FmtValue>
</Cell>
<Cell CellOrdinal="7">
    <Value xsi:type="xsd:int">5915</Value>
    <FmtValue>5915</FmtValue>
</Cell>
<Cell CellOrdinal="8">
    <Value xsi:type="xsd:double">18370</Value>
    <FmtValue>18,370.00</FmtValue>
</Cell>
<Cell CellOrdinal="9">
    <Value xsi:type="xsd:double">15672.8256</Value>
    <FmtValue>15,672.83</FmtValue>
</Cell>
<Cell CellOrdinal="10">
    <Value xsi:type="xsd:double">39394.05</Value>
    <FmtValue>$39,394.05</FmtValue>
</Cell>
<Cell CellOrdinal="11">
    <Value xsi:type="xsd:int">6014</Value>
    <FmtValue>6014</FmtValue>
</Cell>
<Cell CellOrdinal="12">
    <Value xsi:type="xsd:double">21436</Value>
    <FmtValue>21,436.00</FmtValue>
</Cell>
<Cell CellOrdinal="13">
    <Value xsi:type="xsd:double">18094.498</Value>
    <FmtValue>18,094.50</FmtValue>
</Cell>
```

```
<Cell CellOrdinal="14">
    <Value xsi:type="xsd:double">45201.84</Value>
    <FmtValue>$45,201.84</FmtValue>
</Cell>
<Cell CellOrdinal="15">
    <Value xsi:type="xsd:int">7015</Value>
    <FmtValue>7015</FmtValue>
</Cell>
<Cell CellOrdinal="16">
    <Value xsi:type="xsd:double">19287</Value>
    <FmtValue>19,287.00</FmtValue>
</Cell>
<Cell CellOrdinal="17">
    <Value xsi:type="xsd:double">16081.0735</Value>
    <FmtValue>16,081.07</FmtValue>
</Cell>
<Cell CellOrdinal="18">
    <Value xsi:type="xsd:double">40170.29</Value>
    <FmtValue>$40,170.29</FmtValue>
</Cell>
<Cell CellOrdinal="19">
    <Value xsi:type="xsd:int">6184</Value>
    <FmtValue>6184</FmtValue>
</Cell>
<Cell CellOrdinal="20">
    <Value xsi:type="xsd:double">15079</Value>
    <FmtValue>15,079.00</FmtValue>
</Cell>
<Cell CellOrdinal="21">
    <Value xsi:type="xsd:double">12678.9611</Value>
    <FmtValue>12,678.96</FmtValue>
</Cell>
```

```
<Cell CellOrdinal="22">
    <Value xsi:type="xsd:double">31772.88</Value>
    <FmtValue>$31,772.88</FmtValue>
</Cell>
<Cell CellOrdinal="23">
    <Value xsi:type="xsd:int">4799</Value>
    <FmtValue>4799</FmtValue>
</Cell>
<Cell CellOrdinal="24">
    <Value xsi:type="xsd:double">16940</Value>
    <FmtValue>16,940.00</FmtValue>
</Cell>
<Cell CellOrdinal="25">
    <Value xsi:type="xsd:double">14273.7838</Value>
    <FmtValue>14,273.78</FmtValue>
</Cell>
<Cell CellOrdinal="26">
    <Value xsi:type="xsd:double">35880.46</Value>
    <FmtValue>$35,880.46</FmtValue>
</Cell>
<Cell CellOrdinal="27">
    <Value xsi:type="xsd:int">5432</Value>
    <FmtValue>5432</FmtValue>
</Cell>
<Cell CellOrdinal="28">
    <Value xsi:type="xsd:double">16353</Value>
    <FmtValue>16,353.00</FmtValue>
</Cell>
<Cell CellOrdinal="29">
    <Value xsi:type="xsd:double">13738.6822</Value>
    <FmtValue>13,738.68</FmtValue>
</Cell>
```

```
        <Cell CellOrdinal="30">
            <Value xsi:type="xsd:double">34453.44</Value>
            <FmtValue>$34,453.44</FmtValue>
        </Cell>
        <Cell CellOrdinal="31">
            <Value xsi:type="xsd:int">5196</Value>
            <FmtValue>5196</FmtValue>
        </Cell>
    </CellData>
```

Appendix E: Links to Referenced Technologies and Standards

This section provides links to further information about technologies referred to by this specification. The technologies are listed alphabetically.

DAV

For information about Distributed Authoring and Versioning (DAV):

http://msdn.microsoft.com/library/periodic/period99/DAV.HTM

For the specification, see the IETF WEBDAV Working Group Web site:

http://www.ics.uci.edu/pub/ietf/webdav/

.NET

For information about the .NET framework:

http://msdn.microsoft.com/net/

OLE DB specification

For information about OLE DB and Microsoft Data Access Components (MDAC):

http://www.microsoft.com/Data/oledb/default.htm

For the specification and Data Access Software Development Kit (SDK) download:

http://msdn.microsoft.com/library/psdk/dasdk/mdac3sc7.htm

OLE DB for Data Mining

http://www.microsoft.com/data/oledb/dm.htm

OLE DB for OLAP

For information about OLE DB for OLAP, and to download the OLE DB for OLAP specification:

http://www.microsoft.com/data/oledb/olap/default.htm

WSDL

For information about Web Services Description Language (WSDL), the replacement for SDL:

http://msdn.microsoft.com/xml/general/wsdl.asp

SOAP

For Microsoft information about the SOAP protocol:

http://msdn.microsoft.com/xml/general/soaptemplate.asp

For the World Wide Web Consortium (W3C) specification for SOAP:

http://www.w3.org/TR/SOAP/

Microsoft SQL Server

For more information about the SQL Server 2000 XML RAW rowset format, search for "XML, RAW" in SQL Server Books Online.

For more information about XML Encoding, search for "XML Encoding" in SQL Server Books Online.

UDDI

For the Universal Description, Discovery, and Integration (UDDI) Web site, where white papers and technical specifications can be found:

http://www.uddi.com

XML

For the standard for Extensible Markup Language (XML):

http://www.w3.org/XML/

For information about XML on the Microsoft Web site, navigate to:

http://msdn.microsoft.com/xml/default.asp