# Oracle® Essbase Provider Services

## Administration Guide

Release 11.1.2.2.000

ORACLE

ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM

Provider Services Administration Guide, 11.1.2.2.000



Authors: EPM Information Development Team

# Contents

# Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

# 1 Provider Services Overview

**In This Chapter**

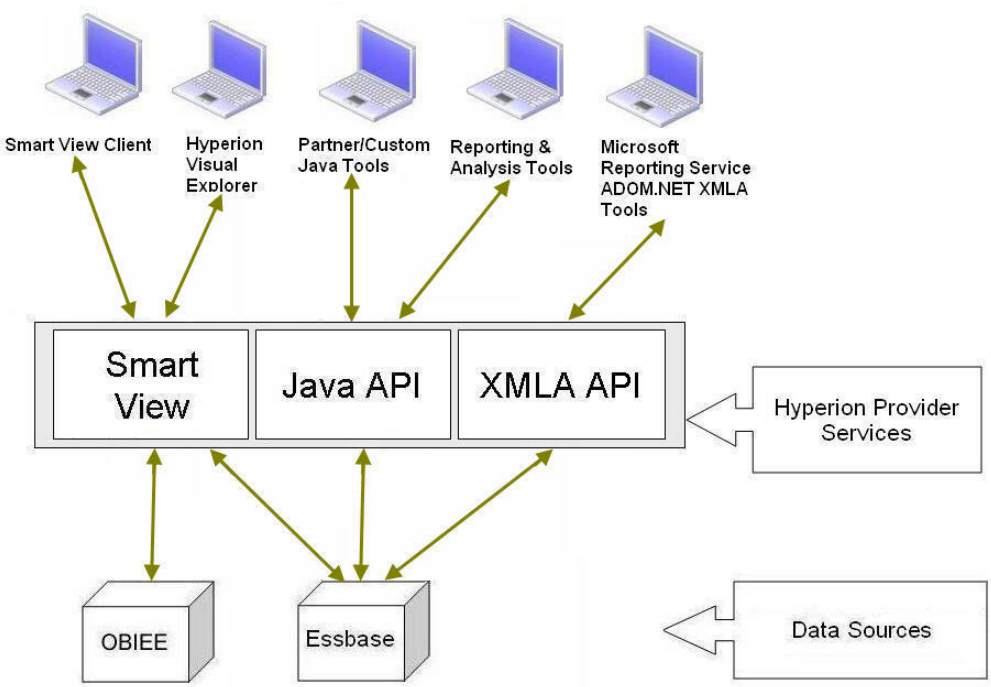


## Provider Services Introduction

Oracle Essbase Provider Services is a middle-tier data-source provider to Oracle Essbase for Java API, Oracle Hyperion Smart View for Office, and XMLA clients and to Oracle Business Intelligence Enterprise Edition for Smart View. Provider Services supports highly concurrent analytical scenarios and provides scalability and reliability in a distributed Web-enabled enterprise environment.

Figure 1 illustrates the relationship of Provider Services to Essbase, and to its Java API, Smart View, and XMLA clients.

Figure 1 Provider Services Architecture



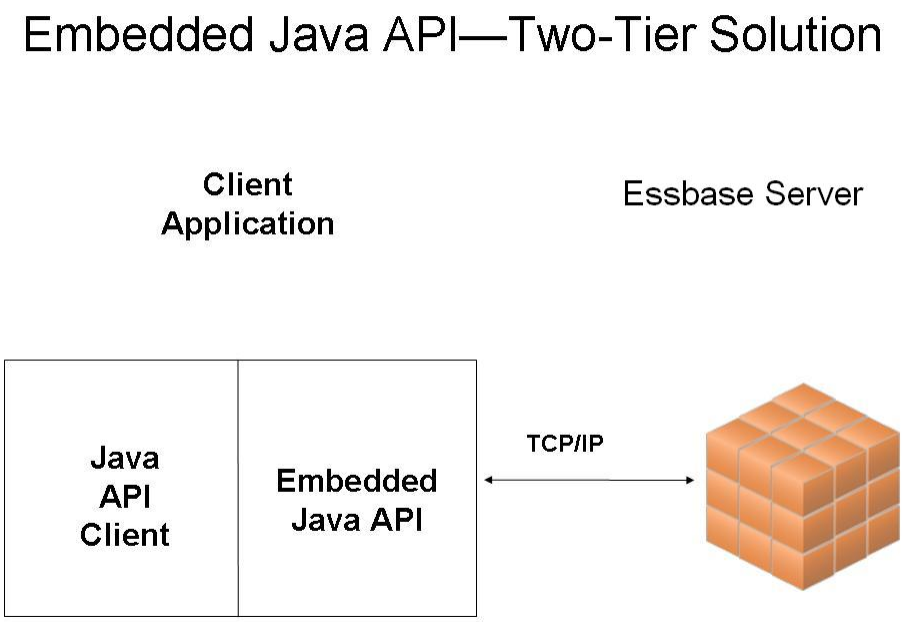

# Java API

Java API is available in embedded and three-tier deployments. Both provide a 100% Java implementation. With a complete Java solution, platform independence is achieved.

Embedded Java API is provided through .jar files and related property files that a Java API client can embed within their application. Java API clients communicate to Essbase through Java API. No installer is required, and no middle-tier server, such as Provider Services, is required to service Java API client requests. However, Java API can be embedded in a Java client application in a two-tier solution or in Hyperion products for the middle-tier application of a three-tier solution.High availability and clustering is not available with embedded Java APIs. You must use Java API with Provider Services to enable high availability and clustering.

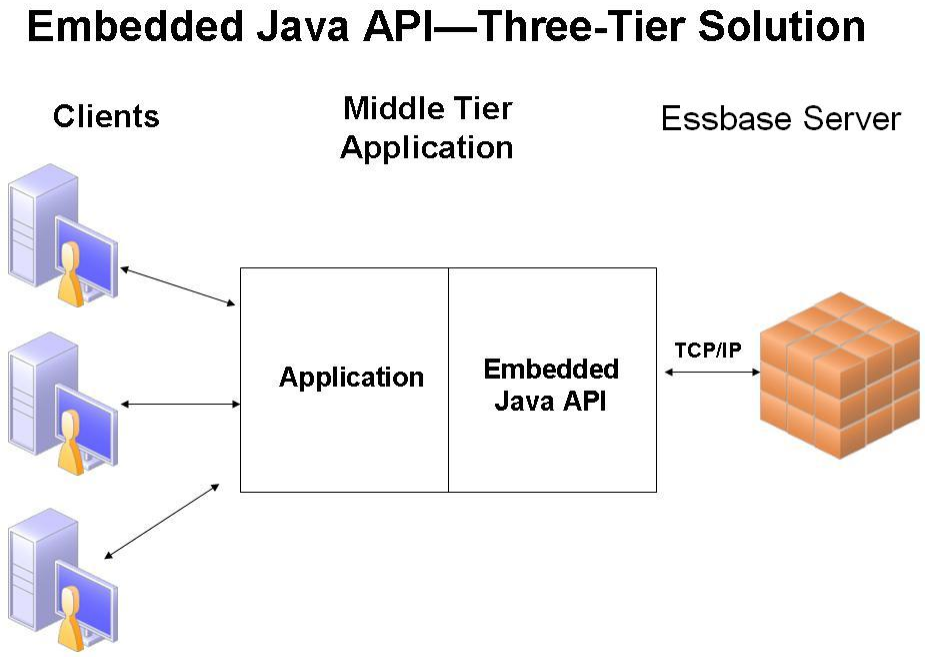
You can switch from embedded Java API to three-tier mode. Through Java API, products such as Web Analysis and Production Reporting can use the high-availability features of Provider Services. The URL for connecting Provider Services to Java API clients: `http://server_name:port/aps/JAPI`.

Figure 2 Embedded Java API - Two-Tier Solution




You can embed Java API in the middle tier of an application as shown in Figure 3:

Figure 3 Embedded Java API - Three-Tier Solution



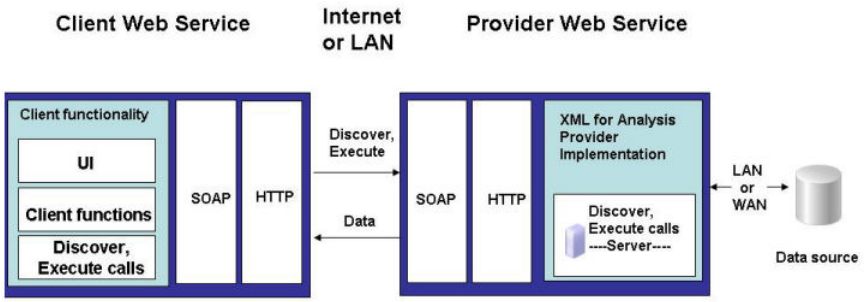
# Smart View

Smart View provides a common Microsoft Office interface for Essbase, Oracle BI EE, Oracle Hyperion Reporting and Analysis, and Oracle Hyperion Financial Management. To use Smart View with Essbase and Oracle BI EE, you need Provider Services as a middle-tier server. The URL for connecting Provider Services to Smart View clients is: http://server_name:port/aps/SmartView.

# XML for Analysis

XML for Analysis (XMLA) is an open, industry-standard Web service interface for online analytical processing. The open architecture of XMLA enables development on any language, platform, or operating system. Provider Services provides high availability for XMLA for Essbase. Using Provider Services and XMLA, Microsoft Reporting Services generates and publishes reports for Essbase. The URL for connecting Provider Services to XMLA clients: http://server_name:port/aps/XMLA.

Figure 4 XMLA System Architecture




# Essbase Web Services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. Web services use XML to code and decode data, and SOAP (Simple Object Access Protocol) to transport it. Web services are defined using WSDL (Web Service Description Language).

Essbase Web Services support access to and administration of Essbase applications and cubes. Essbase Web Services include the following modules:

- Datasource
- Administration
- Data and Metadata Query

# EPM System

Provider Services is part of Oracle Enterprise Performance Management System, a comprehensive business performance management system that integrates modular suites of financial management applications with the most comprehensive business intelligence capabilities for reporting and analysis.

# Provider Services Documentation

Installation, configuration, deployment, and other related information for Provider Services may be found in the Hyperion EPM System documentation set, which comprises the following guides:

- *Oracle Hyperion Enterprise Performance Management System Installation Start Here*
- *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide*
- *Oracle Hyperion Enterprise Performance Management System Security Administration Guide*
- *Oracle Hyperion Enterprise Performance User and Role Security Guide*
- *Oracle Hyperion Enterprise Performance Management System High Availability and Disaster Recovery Guide*
- *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Troubleshooting Guide*
- *Oracle Hyperion Enterprise Performance Management System Backup and Recovery Guide*
- *Oracle Hyperion Enterprise Performance Management System Lifecycle Management Guide*
- *Oracle Hyperion Enterprise Performance Management System Certification Matrix*

# 2 Administering Provider Services

**In This Chapter**



## Administering Provider Services

To perform administrative functions in Provider Services in Essbase, you must have an administrator role for the designated Essbase Server in Oracle Hyperion Shared Services. If you have an admin role, you are automatically given permission to add or administer Provider Services for that Essbase after logging into Oracle Essbase Administration Services. Provider Services communicates with the designated Essbase server and grants or denies administrator permissions based on the your role in that Essbase instance.

Use Administration Services Console to administer Provider Services:

- "Adding Provider Services" on page 15
- "Removing Provider Services" on page 16
- "Connecting to a Stand-alone Essbase Server" on page 17
- "Editing the Authenticating Essbase Server" on page 16
- "Connecting to Provider Services" on page 17
- "Monitoring Sessions" on page 18
- "Specifying Session Timeout" on page 18
- "Specifying Maximum Rows and Columns" on page 19

## Adding Provider Services

You can manage Oracle BI EE and Essbase connections through the Smart View Panel in Smart View. For Essbase only, to add Provider Services through Administration Services, use the following procedure.

➤ To add Provider Services:

1. From Enterprise View or a custom view, select the **Hyperion Provider Services** node.
2. Right-click and select **Add Hyperion Provider Services**.
3. In **Add Hyperion Provider Services**, in **Provider Name**, enter the Provider Services server name, for example, `localhost`.
4. Click the **URL** text box. This copies the URL of the Provider Services server you entered in the previous step. For example, `http://localhost:13080/aps/APS`.
5. In **Authenticating Essbase Server**, select the name of the Essbase server from the dropdown list. You must have an administrator role in this Essbase server to perform administrative actions.
6. Click **OK**.

   The provider name is displayed under the Provider Services node.

## Editing the Authenticating Essbase Server

You can manage Oracle BI EE and Essbase connections through Smart View. For Essbase only, to edit Provider Services through Administration Services, use the following procedure.

You can edit the **Authenticating Essbase Server** that you specified in "Adding Provider Services" on page 15 while adding a Provider Services server in Administration Services.

➤ To edit the authenticating Essbase Server:

1. From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select a provider.
2. Right-click and select **Edit Authenticating Essbase Server**.

   A dialog box where you can edit and specify another authenticating Essbase server is displayed.
3. Click **OK**.

## Removing Provider Services

You can manage Oracle BI EE and Essbase connections through Smart View. For Essbase only, to remove Provider Services through Administration Services, use the following procedure.

➤ To remove Provider Services:

1. From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select a provider.
2. Right-click and select **Remove**.
3. In **Remove Hyperion Provider Services**, click **Yes**.

## Connecting to a Stand-alone Essbase Server

You can manage Oracle BI EE and Essbase connections through Smart View. For Essbase only, to connect Provider Services through Administration Services, use the following procedure.

Through Administration Services Console, Provider Services can connect to stand-alone Essbase Servers or Essbase Server clusters. Smart View, Java API, and XMLA users connect to Essbase Servers through Provider Services. To users, the accessed database is transparent. From their perspective, they connect to, and retrieve data from, one data source.

**Note:** To enable users to select any stand-alone Essbase Server, add the stand-alone server to Provider Services through Administration Services Console. Add Essbase Server to the User Properties window in Administration Services Console before adding the stand-alone server to Provider Services.

➤ To connect to a stand-alone Essbase Server:

1. From Enterprise View or a custom view, select the **Essbase Servers** node to add Essbase Servers to administer.
2. Right-click and select **Add Essbase Server**.
3. In **Add Analytic Server**, enter the Essbase Server name, user name, and password, confirm the password, and click **OK**.
4. Repeat step 2 to add additional Essbase Servers.
5. From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select a provider.
6. Right-click and select **Create**, then **Create Stand-alone Server.**
7. In **Add Stand-alone Server**, from the list of servers added in step 3, select a server.
8. Click **OK**.

   The name of the stand-alone Essbase Server is displayed under the Stand-alone Server node.

   **Note:** Alternatively, if you have existing stand-alone servers, you can select the Stand-alone Server node under a provider's name, right-click, and select Create Stand-alone Server.

## Connecting to Provider Services

You can manage Oracle BI EE and Essbase connections through Smart View. For Essbase only, to connect Provider Services through Administration Services, use the following procedure.

Start all Essbase Servers associated with Provider Services, as stand-alone servers or in a cluster. Ensure that Provider Services is connected so that clients can connect to it.

➤ To connect to Provider Services:

1. From Enterprise View or a custom view, select the server node under the **Hyperion Provider Services** node.
2. Right-click and select **Connect**.

   Provider Services is now online.

## Monitoring Sessions

Use the sessions window to monitor sessions of users connected to Provider Services. You can view sessions of all users or specific users and which session types are running—Smart View, Java API, or XMLA.

➤ To monitor Provider Services sessions:

1. From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select a provider.
2. Right-click and select **Sessions**.

   The Provider Services Sessions window is displayed:

   - **Session**—Active session ID
   - **Session Type**—Type of request, from stand-alone server or cluster
   - **Mode**—Stand-alone server mode (server) or Analytic Cluster mode (cluster)
   - **User**—The use who generated the request
   - **Analytic Server**— Essbase Server to which the request was made
   - **Application**—Application name
   - **Database**—Database name
   - **Request Time**—Time of request
   - **Request**—Name of current running request, if any
3. To see one user's sessions, select **Show sessions for user** and select from user lists.
4. To see a session, select **Show sessions for type** and select **JAVA**, **XMLA**, or **SMARTVIEW**
5. Click **Refresh** to update the view.

## Specifying Session Timeout

You can specify how many minutes the session can be inactive before timing out.

➤ To specify the session timeout limit:

1. From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select the **Provider** node.
2. Right-click and select **Edit,** then **Properties**.
3. In **Hyperion Provider Services Properties**, select **Settings**.

4 In **Idle Session timeout in minutes**, specify how long the session can be inactive before timing out (default is 60). If the session times out, Smart View users must reconnect to Provider Services.

5 Click **Apply**.

6 Click **Close**.

## Specifying Maximum Rows and Columns

Administrators can specify maximum values for rows and columns to be retrieved in a Smart View grid. By default, Provider Services installations set a maximum of 5000 rows and a maximum of 255 columns. If *all* Smart View users are using Excel 2007 or later, the administrator can increase these maximum values for rows and columns. However, if some or all Smart View users are using Excel 2003, then the default values of 5000 rows and 255 columns (the limits set by Excel 2003) must be used.

Changes to the maximum row and column properties take effect only after the Smart View client connects to a new session of Provider Services.

➤ To specify maximum rows and columns:

1 Open Administration Services.

2 From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select the **Provider** node.

3 Right-click and select **Edit,** then **Properties**.

4 In **Hyperion Provider Services Properties**, select **Settings**.

5 In **Maximum number of rows**, specify the number of rows to retrieve (default is 5,000).

6 In **Maximum number of columns**, specify the number of columns to retrieve (default is 255).

7 Click **Apply**.

8 Click **Close**.

## Automatically Deploying Client Upgrades

You can enable automatic deployment of new Smart View client releases.

➤ To automatically deploy Smart View clients:

1 From Enterprise View or a custom view, under the **Hyperion Provider Services** node, select the **Provider** node.

2 Right-click and select **Edit,** then **Properties**.

3 In **Hyperion Provider Services Properties**, select **Client Deployment**.

4 Select an option:

- **Force Smart View client to upgrade**—Users must upgrade to continue using Smart View.

- **Warn Smart View client to upgrade**—Informs users of available Smart View upgrade. Users can continue using Smart View clients without upgrading.
- **Apply Smart View client to upgrade**—Enables the administrator to apply new versions of Oracle Hyperion Smart View for Office and inform users without requiring Provider Services restart.

5 **Click Apply.**

6 **In** `ORACLE_HOME/common/epmstatic/wspace/SmartView`**, open** `version.xml`.

7 **Add the following Provider Services URL to** `version.xml`:

```
http://<server name>:13080/aps/APS?downloadClient
```

This sample `version.xml` shows where to place the URL:

```
<?xml version="1.0" encoding="utf-8"?>
<CommonAddinVersion>
    <internalVersion>
        <major>
        4
        </major>
        <minor>
        2.1.0.0
        </minor>
    </internalVersion>
    <externalVersion>
        11.1.2.1.00
    </externalVersion>
    <installFile>
        http://<server name>:13080/aps/APS?downloadClient
    </installFile>
</CommonAddinVersion>
```

# Updating References to Rehosted Servers

If you are upgrading to this release by installing EPM System products on a new host machine, you must update Provider Services references to any of the following to reflect the new host name and port number.

- Essbase servers
- Active-active Essbase clusters configured by Provider Services
- Oracle Business Intelligence Enterprise Edition servers

See the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide* "Updating References to a Rehosted Environment" section for general rehosting information and for information about updating Provider Services references to Essbase servers.

➤ To update Provider Services references to active-active Essbase clusters configured by Provider Services or to Oracle BI EE servers:

1 **Navigate to** *EPM_ORACLE_INSTANCE*`/bin/upgrades`.

2 **From a command prompt, run the following script:**

```
ApsUpdateEssbaseServer.bat|sh fromHost toHost
```

where `fromHost` is the host name of the original host, and `toHost` is the name of the new host.

Execute this script once for each reference to be updated.

# Logging

Provider Services uses the Oracle Diagnostic Logging framework (ODL) for logging purposes. See the *Oracle Hyperion Enterprise Performance Management System Troubleshooting Guide*.

# Setting TCP/IP Socket Communication

In the TCP/IP socket communication between Provider Services Java API and Essbase, you do not need to configure the socket timeout. By default, control returns to the client when the socket communication completes, when the server resets the socket state or closes, or when the socket times out because TCP/IP timed out. However, you can specify the network operation timeout within which the control returns to the client.

➤ To specify network operation timeout:

1 **Navigate to** *EPM_ORACLE_INSTANCE*`\bin\essbase.properties`.

2 **Double-click** `essbase.properties` **to open the file.**

3 **Set** `olap.server.netSocketTryInfinite=false`.

4 **Set** `olap.server.netRetryCount= xxx`**, where the total network operation timeout =** `olap.server.netRetryCount` **x** `olap.server.netSocketTimeOut` **in milliseconds.**

# Configuring Options in essbase.properties

The following Provider Services options are configurable only in `essbase.properties`:

**Table 1** essbase.properties Settings

| Setting | Description |
|---|---|
| olap.server.netConnectRetry | The number of attempts a client makes to connect to an Essbase server |
| olap.server.netDelay | The time that the thread waits before attempting another connect against Essbase |
| olap.server.netRetryCount | The number of times an API can retry a unsuccessful network operation |
| olap.server.netLoopIPAddresses | For connections to hosts with both IPv4 and IPv6 network interfaces, enables performance benefit |
| olap.server.netSocketTimeOut | The time that a network operation can be blocked before it times out |

| Setting | Description |
|---|---|
| olap.server.netSocketTryInfinite | Indicates that the client will keep trying infinitely on a network operation |

# Configuring essbase.properties

➤ To edit `essbase.properties`:

1. **Navigate to** *EPM_ORACLE_INSTANCE*`\bin\essbase.properties`.
2. **Double-click** `essbase.properties` **to open the file.**
3. **Enter each setting on a separate line.**

   Semicolon terminators are not required.
4. **Save and close** `essbase.properties`.
5. **Restart the Provider Services server.**

# Configurable Options in essbase.properties

## olap.server.netConnectRetry

### Description

The number of attempts a client makes to connect to an Essbase Server before failing and reporting an error.

Some causes of connection failures: network congestion, server inaccessibility, and network interruption.

### Syntax

olap.server.netConnectRetry=*n*

### Parameters

*n* - An integer value (default is 3).

### Example

```
olap.server.netConnectRetry=20
```

## olap.server.netDelay

### Description

The time in milliseconds that the thread waits before attempting another connect against Essbase.

### Syntax

olap.server.netDelay=*n*

### Parameters

*n* - Integer value of 100 or greater, expressed in milliseconds (default is 200).

### Example

```
olap.server.netDelay=300
```

## olap.server.netRetryCount

### Description

The number of times an API can retry a unsuccessful network operation before failing and reporting an error. If olap.server.netSocketTryInfinite is true, then olap.server.netRetryCount is ineffective.

### Syntax

olap.server.netRetryCount=*n*

### Parameters

*n* - An integer value (default value is 600 retries).

### Example

```
olap.server.netRetryCount=400
```

## olap.server.netLoopIPAddresses

### Description

If Provider Services is needed to connect to hosts that have both IPv4 and IPv6 network interfaces enabled but only one is being used, this property can be set to false to get a performance benefit.

When set to false, Provider Services will not loop through all the interfaces while connecting and instead use only the default one returned by host network environment.

### Syntax

olap.server.netLoopIPAddresses=boolean

### Parameters

True or false - (default is true).

### Sample

```
olap.server.netLoopIPAddresses=true
```

## olap.server.netSocketTimeOut

### Description

The maximum time in milliseconds that a network operation can be blocked before the operation times out. A timeout of zero is interpreted as an infinite timeout.

### Syntax

olap.server.netSocketTimeOut=*n*

### Parameters

*n* - Integer value of 0 or above, expressed in milliseconds (default is 200).

### Example

```
olap.server.netSocketTimeOut=120000
```

## olap.server.netSocketTryInfinite

### Description

Indicates that the client will keep trying infinitely on a network operation. If olap.server.netSocketTryInfinite is true, then olap.server.netRetryCount is ineffective.

### Syntax

olap.server.netSocketTryInfinite=boolean

### Parameters

True or false - (default is true).

### Sample

```
olap.server.netSocketTryInfinite=true
```

# Clustering

For information about using Provider Services to cluster Essbase databases, see *Oracle Hyperion Enterprise Performance Management System High Availability and Disaster Recovery Guide*.

For information about clustering Provider Services, see "Clustering Web Applications" in *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide*.

# 3 Working with XMLA

In This Chapter



## Key Features

XML for Analysis (XMLA) is an open industry-standard Web service interface designed for online analytical processing. XMLA is a set of XML Message Interfaces built on the open standards of HTTP, XML, and Simple Object Access Protocol (SOAP). XMLA, which is not bound to any language, platform, or operating system, provides standardized data access between client applications and any multidimensional data source on the Web.

For more information on XMLA, visit www.xmla.org.

Key XMLA features:

- Support for flattened rowsets
- Support for stateful sessions
- Backward XMLA level representation (level 1 is the top level)
- User authentication through basic HTTP authentication
- XMLA High-Availability functionality through Provider Services
- XMLA administration and monitoring through Administration Services

**Note:** XMLA is available for use with Essbase only.

## Methods

The following methods provide a standard way for XML applications to access basic information from the server. Because these methods are invoked using SOAP, 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.

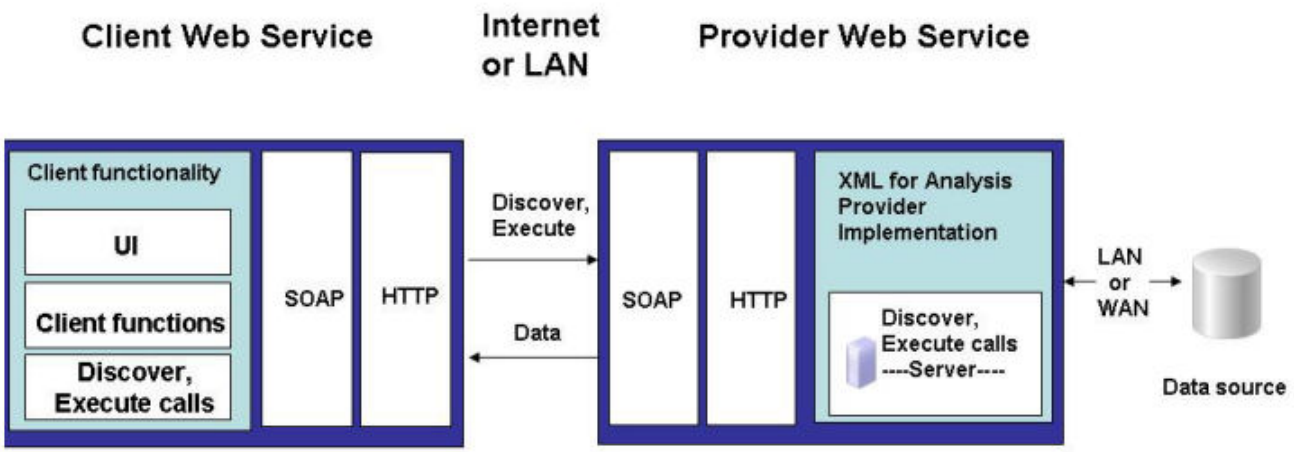
The simplified interface model has two methods.

- Discover
- Execute

**Discover** obtains information and metadata from a Web Service. This information can include a list of available data sources and data about a data source provider. Properties define and shape the data obtained. Discover allows you to specify the types of information that the client application needs. The use of generic interface and properties enables extensibility without necessitating rewriting existing functions.

**Execute** executes Multidimensional Expressions (MDX) or other provider-specific commands against an XMLA data source. The following diagram illustrates a possible implementation of an n-tiered application.

Figure 5 XMLA Architecture




Provided with the URL for a server hosting a Web Service, the client uses SOAP and HTTP protocols to send Discover and Execute calls to the server. The server instantiates the XMLA provider, which handles the calls. The XMLA provider fetches the data, packages it into XML, and sends the data to the client.

The Discover and Execute methods enable users to determine what can be queried on a server and, based on this, submit commands to be executed.

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 retrieves information, such as the list of data sources on a server or details about a 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 comprises a RequestType enumeration value, which determines the type of information to be returned. The RequestType enumeration is used by the Discover method to determine the structure and content of the rowset returned in the Result parameter. The Restrictions parameter format and XML result set are also dependent on the value specified in this parameter. This enumeration can be extended to support provider-specific enumeration strings.

Each RequestType enumeration value corresponds to a return rowset. For rowset definitions, see "XMLA Rowsets" on page 33. Support is required for the following explicitly named RequestType enumeration values.

| Enumeration value | Description |
|---|---|
| DISCOVER_ DATASOURCES | Returns a list of XMLA data sources available on the server or Web Service. (For an example of how these may be published, see "XMLA Implementation Walkthrough" in the *XML for Analysis Specification,* available on the Hyperion Developer Network.) |
| 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 RequestType 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 the provider of a specific data source. |
| 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 also may 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. Result columns are defined by the rowset specified in the RequestType parameter. Some

columns of Result can filter the rows returned. For these columns and those that can be restricted, see the rowset tables in "XMLA Rowsets" on page 33. To obtain the restriction information for provider-specific schema rowsets, use the DISCOVER_SCHEMA_ROWSETS request type. This parameter can be empty, but it must be included.

**Properties [in]**

This parameter, of the Properties data type, comprises a collection of XMLA properties. Each property enables users to control some aspect of the Discover method, such as specifying the return format of the result set, the timeout, or the locale in which the data should be formatted.

You can obtain the available properties by using the DISCOVER_PROPERTIES request type with the Discover method.

The properties in the Properties parameter have no required order. This parameter can be empty, but it must be included.

**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 in the RequestType and Restrictions parameters. The column layout of the returned result set also is determined by the value specified in RequestType. For information about the rowset layouts that correspond to for each RequestType value, see "XMLA Rowsets" on page 33.

**Example**

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

```
<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>Demo</CATALOG_NAME>
     </RestrictionList>
    </Restrictions>
    <Properties>
     <PropertyList>
      <DataSourceInfo>
       Provider=Essbase;Data Source=localhost
      </DataSourceInfo>
      <Format>Tabular</Format>
     </PropertyList>
    </Properties>
   </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CATALOG_NAME" type="xsd:string"
         sql:field="CATALOG_NAME"/>
        <xsd:element name="CUBE_NAME" type="xsd:string"
         sql:field="CUBE_NAME"/>
        <xsd:element name="CUBE_TYPE" type="xsd:string"
         sql:field="CUBE_TYPE"/>
        <xsd:element name="LAST_SCHEMA_UPDATE" type="xsd:dateTime"
         sql:field="LAST_SCHEMA_UPDATE" minOccurs="0"/>
        <xsd:element name="DESCRIPTION" type="xsd:string"
         sql:field="DESCRIPTION" minOccurs="0"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <CATALOG_NAME>Demo</CATALOG_NAME>
      <CUBE_NAME>Demo.Basic</CUBE_NAME>
      <CUBE_TYPE>CUBE</CUBE_TYPE>
     </row>
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Execute

The Execute method sends action requests, including those involving data transfer, such as retrieving or updating data on the server, to 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 an MDX statement to be executed.

**Properties [in]**

This parameter is of the Properties data type and consists of a collection of XMLA 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.

The properties in the Properties parameter have no required order. This parameter can be empty, but it must be included.

**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 XMLA 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 also can 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 MDX SELECT statement:

```
<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  CrossJoin([Measures].CHILDREN , [Market].CHILDREN)
    on columns,  [Product].Members on rows
    from Sample.Basic
   </Statement>
  </Command>
  <Properties>
   <PropertyList>
    <DataSourceInfo>
     Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Catalog>Sample</Catalog>
    <Format>Multidimensional</Format>
    <AxisFormat>TupleFormat</AxisFormat>
    <Content>SchemaData</Content>
   </PropertyList></Properties>
  </Execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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>
```

# XMLA Rowsets

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

## CATALOGS Rowset

The CATALOGS rowset identifies the physical attributes associated with catalogs accessible from Analytic Services.

**GUID:** DBSCHEMA_CATALOGS

the section called "Flattened Rowset Examples" describes the rowset structure.

**Table 2 CATALOGS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| DESCRIPTION | Always null |

### Request Example

```
<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>DBSCHEMA_CATALOGS</RequestType>
   <Restrictions>
    <RestrictionList></RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response Example (truncated)

```
<?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:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
```

```
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CATALOG_NAME" type="xsd:string"
         sql:field="CATALOG_NAME"/>
        <xsd:element name="DESCRIPTION" type="xsd:string"
         sql:field="DESCRIPTION" minOccurs="0"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <CATALOG_NAME>Demo</CATALOG_NAME>
     </row>
     < ..............More Rows..... >
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## MDSCHEMA_CUBES Rowset

The CUBES rowset contains information about the available cubes in a schema (or the catalog, if the provider does not support schemas).

**GUID**: MDSCHEMA_CUBES

Table 3 describes the rowset structure.

**Table 3 MDSCHEMA_CUBES Rowset structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| CUBE_TYPE | “CUBE” |
| LAST_SCHEMA_UPDATE | Time stamp of last outline update |
| DESCRIPTION | Database description |

### Request Example

```
<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>Demo</CATALOG_NAME>
     </RestrictionList>
    </Restrictions>
    <Properties>
     <PropertyList>
      <DataSourceInfo>
       Provider=Essbase;Data Source=localhost
      </DataSourceInfo>
      <Format>Tabular</Format>
     </PropertyList>
    </Properties>
   </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response 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>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CATALOG_NAME" type="xsd:string"
         sql:field="CATALOG_NAME"/>
        <xsd:element name="CUBE_NAME" type="xsd:string"
         sql:field="CUBE_NAME"/>
        <xsd:element name="CUBE_TYPE" type="xsd:string"
         sql:field="CUBE_TYPE"/>
        <xsd:element name="LAST_SCHEMA_UPDATE" type="xsd:dateTime"
         sql:field="LAST_SCHEMA_UPDATE" minOccurs="0"/>
        <xsd:element name="DESCRIPTION" type="xsd:string"
         sql:field="DESCRIPTION" minOccurs="0"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <CATALOG_NAME>Demo</CATALOG_NAME>
```

```
      <CUBE_NAME>Demo.Basic</CUBE_NAME>
      <CUBE_TYPE>CUBE</CUBE_TYPE>
     </row>
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## MDSCHEMA_DIMENSIONS Rowset

The DIMENSIONS rowset contains information about the dimensions in a given cube. Each dimension has one row.

**GUID:** MDSCHEMA_DIMENSIONS

Table 4 describes the rowset structure.

**Table 4 MDSCHEMA_DIMENSIONS Rowset structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| DIMENSION_NAME | Dimension name |
| DIMENSION_UNIQUE_NAME | Dimension name |
| DIMENSION_CAPTION | Dimension name |
| DIMENSION_ORDINAL | Dimension number. First dimension is 1, second is 2, and so on |
| DIMENSION_TYPE | If Essbase dimension type is:<br>● TIME: MD_DIMTYPE_TIME<br>● ACCOUNTS: MD_DIMTYPE_MEASURE<br>● ALL OTHER: MD_DIMTYPE_OTHER |
| DIMENSION_CARDINALITY | Number of members in the dimension |
| DEFAULT_HIERARCHY | Dimension name |
| DESCRIPTION | Comment added for the dimension |
| DIMENSION_UNIQUE_SETTINGS | 2 |
| DIMENSION_IS_VISIBLE | True always |

### Request Example

```
<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_DIMENSIONS</RequestType>
    <Restrictions>
     <RestrictionList>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Basic</CUBE_NAME>
     </RestrictionList>
    </Restrictions>
    <Properties>
     <PropertyList>
      <DataSourceInfo>Provider=Essbase;Data Source=localhost
      </DataSourceInfo>
      <Format>Tabular</Format>
     </PropertyList>
    </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Response Example**(truncated)

```
<?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:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="CATALOG_NAME" type="xsd:string"
        sql:field="CATALOG_NAME"/>
       <xsd:element name="CUBE_NAME" type="xsd:string"
        sql:field="CUBE_NAME"/>
       <xsd:element name="DIMENSION_NAME" type="xsd:string"
        sql:field="DIMENSION_NAME"/>
       <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
        sql:field="DIMENSION_UNIQUE_NAME"/>
       <xsd:element name="DIMENSION_CAPTION" type="xsd:string"
        sql:field="DIMENSION_CAPTION"/>
```

```
        <xsd:element name="DIMENSION_ORDINAL" type="xsd:unsignedInt"
         sql:field="DIMENSION_ORDINAL"/>
        <xsd:element name="DIMENSION_TYPE" type="xsd:short"
         sql:field="DIMENSION_TYPE"/>
        <xsd:element name="DIMENSION_CARDINALITY" type="xsd:unsignedInt"
         sql:field="DIMENSION_CARDINALITY"/>
        <xsd:element name="DEFAULT_HIERARCHY" type="xsd:string"
         sql:field="DEFAULT_HIERARCHY"/>
        <xsd:element name="DESCRIPTION" type="xsd:string"
         sql:field="DESCRIPTION" minOccurs="0"/>
        <xsd:element name="DIMENSION_UNIQUE_SETTINGS" type="xsd:int"
         sql:field="DIMENSION_UNIQUE_SETTINGS"/>
        <xsd:element name="DIMENSION_IS_VISIBLE" type="xsd:boolean"
         sql:field="DIMENSION_IS_VISIBLE"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Sample.Basic</CUBE_NAME>
      <DIMENSION_NAME>Year</DIMENSION_NAME>
      <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
      <DIMENSION_CAPTION>Year</DIMENSION_CAPTION>
      <DIMENSION_ORDINAL>1</DIMENSION_ORDINAL>
      <DIMENSION_TYPE>1</DIMENSION_TYPE>
      <DIMENSION_CARDINALITY>19</DIMENSION_CARDINALITY>
      <DEFAULT_HIERARCHY>[Year]</DEFAULT_HIERARCHY>
      <DIMENSION_UNIQUE_SETTINGS>2</DIMENSION_UNIQUE_SETTINGS>
      <DIMENSION_IS_VISIBLE>true</DIMENSION_IS_VISIBLE>
     </row>
     < ..............More Rows..... >
    </root>
   </m:return>
  </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# MDSCHEMA_FUNCTIONS Rowset

The FUNCTIONS rowset exposes all functions supported by the MDP. Default sort order: ORIGIN, INTERFACE_NAME, and FUNCTION_NAME.

**GUID**: MDSCHEMA_FUNCTIONS

Table 5describes the rowset structure.

**Table 5 MDSCHEMA_FUNCTIONS Rowset structure**

| Column Name | Essbase Mapping |
|---|---|
| FUNCTION_NAME | Name of the function |
| DESCRIPTION | Description of the function |
| PARAM_LIST | A comma delimited list of parameters |

| Column Name | Essbase Mapping |
|---|---|
| RETURN_TYPE | Always 12 |
| ORIGIN | 1 (always:MDX functions) |
| INTERFACE_NAME | One of the following: Member, Set, Tuple, Numeric, Dimension, Level, Boolean |
| OBJECT | One of the following values: Set, Member, Tuple, Level, Hierarchy, Dimension |
| HELP_CONTEXT | Help context ID for the function |
| CAPTION | Display caption of the function |

### Request Example

```
<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_FUNCTIONS</RequestType>
   <Restrictions><RestrictionList></RestrictionList></Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response Example (truncated)

```
<?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:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
```

```
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="FUNCTION_NAME" type="xsd:string"
        sql:field="FUNCTION_NAME"/>
       <xsd:element name="DESCRIPTION" type="xsd:string"
        sql:field="DESCRIPTION"/>
       <xsd:element name="PARAMETER_LIST" type="xsd:string"
        sql:field="PARAMETER_LIST"/>
       <xsd:element name="RETURN_TYPE" type="xsd:int"
        sql:field="RETURN_TYPE"/>
       <xsd:element name="ORIGIN" type="xsd:int"
        sql:field="ORIGIN"/>
       <xsd:element name="INTERFACE_NAME" type="xsd:string"
        sql:field="INTERFACE_NAME"/>
       <xsd:element name="OBJECT" type="xsd:string"
        sql:field="OBJECT" minOccurs="0"/>
       <xsd:element name="HELP_CONTEXT" type="xsd:int"
        sql:field="HELP_CONTEXT" minOccurs="0"/>
       <xsd:element name="CAPTION" type="xsd:string"
        sql:field="CAPTION"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
    <!-- Begin: All MDX functions that return a Member
     (INTERFACE_NAME=Member) -->
    <row>
     <FUNCTION_NAME>Ancestor</FUNCTION_NAME>
     <DESCRIPTION>Given the input member, returns the ancestor
      at the specified level.</DESCRIPTION>
     <PARAMETER_LIST>Member, Level | Numeric Expression</PARAMETER_LIST>
     <RETURN_TYPE>12</RETURN_TYPE>
     <ORIGIN>1</ORIGIN>
     <INTERFACE_NAME>Member</INTERFACE_NAME>
     <HELP_CONTEXT>9142</HELP_CONTEXT>
     <CAPTION>Ancestor</CAPTION>
    </row>
    < ..............More Rows..... >
   </root>
  </m:return>
 </m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## MDSCHEMA_HIERARCHIES Rowset

The HIERARCHIES rowset contains information about the hierarchies available in a dimension.

**GUID**: MDSCHEMA_HIERARCHIES

Table 6describes the rowset structure.

**Table 6** MDSCHEMA_HIERARCHIES Rowset Structure

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| DIMENSION_UNIQUE_NAME | Dimension name |
| HIERARCHY_NAME | Dimension name |
| HIERARCHY_UNIQUE_NAME | Dimension name |
| HIERARCHY_CAPTION | Dimension name |
| DIMENSION_TYPE | If Essbase dimension type is:<br>● TIME: MD_DIMTYPE_TIME<br>● ACCOUNTS: MD_DIMTYPE_MEASSURE<br>● ALL OTHER: MD_DIMTYPE_OTHER |
| HIERARCHY_CARDINALITY | Number of members in the dimension |
| DEFAULT_MEMBER | Dimension name |
| ALL_MEMBER | Dimension name |
| DESCRIPTION | Dimension comment |
| STRUCTURE | MD_STRUCTURE_UNBALANCED(2) |
| HIERARCHY_UNIQUE_SETTINGS | 2 |
| HIERARCHY_IS_VISIBLE | True |

## Request Example

```
<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_HIERARCHIES</RequestType>
    <Restrictions>
     <RestrictionList>
      <CUBE_NAME>Sample.Basic</CUBE_NAME>
      <DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
     </RestrictionList>
    </Restrictions>
    <Properties>
     <PropertyList>
      <DataSourceInfo>Provider=Essbase;Data Source=localhost
      </DataSourceInfo>
      <Format>Tabular</Format>
     </PropertyList>
```

```
    </Properties>
   </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response 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>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CATALOG_NAME" type="xsd:string"
         sql:field="CATALOG_NAME"/>
        <xsd:element name="CUBE_NAME" type="xsd:string"
         sql:field="CUBE_NAME"/>
        <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
         sql:field="DIMENSION_UNIQUE_NAME"/>
        <xsd:element name="HIERARCHY_NAME" type="xsd:string"
         sql:field="HIERARCHY_NAME"/>
        <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
         sql:field="HIERARCHY_UNIQUE_NAME"/>
        <xsd:element name="HIERARCHY_CAPTION" type="xsd:string"
         sql:field="HIERARCHY_CAPTION"/>
        <xsd:element name="DIMENSION_TYPE" type="xsd:short"
         sql:field="DIMENSION_TYPE"/>
        <xsd:element name="HIERARCHY_CARDINALITY" type="xsd:unsignedInt"
         sql:field="HIERARCHY_CARDINALITY"/>
        <xsd:element name="DEFAULT_MEMBER" type="xsd:string"
         sql:field="DEFAULT_MEMBER"/>
        <xsd:element name="ALL_MEMBER" type="xsd:string"
         sql:field="ALL_MEMBER"/>
        <xsd:element name="DESCRIPTION" type="xsd:string"
         sql:field="DESCRIPTION" minOccurs="0"/>
        <xsd:element name="STRUCTURE" type="xsd:int"
         sql:field="STRUCTURE"/>
```

```
        <xsd:element name="HIERARCHY_UNIQUE_SETTINGS" type="xsd:int"
         sql:field="HIERARCHY_UNIQUE_SETTINGS"/>
        <xsd:element name="HIERARCHY_IS_VISIBLE" type="xsd:boolean"
         sql:field="HIERARCHY_IS_VISIBLE"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Sample.Basic</CUBE_NAME>
      <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
      <HIERARCHY_NAME>Year</HIERARCHY_NAME>
      <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
      <HIERARCHY_CAPTION>Year</HIERARCHY_CAPTION>
      <DIMENSION_TYPE>1</DIMENSION_TYPE>
      <HIERARCHY_CARDINALITY>19</HIERARCHY_CARDINALITY>
      <DEFAULT_MEMBER>[Year]</DEFAULT_MEMBER>
      <ALL_MEMBER>[Year]</ALL_MEMBER>
      <STRUCTURE>2</STRUCTURE>
      <HIERARCHY_UNIQUE_SETTINGS>2</HIERARCHY_UNIQUE_SETTINGS>
      <HIERARCHY_IS_VISIBLE>true</HIERARCHY_IS_VISIBLE>
     </row>
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## MDSCHEMA_MEASURES Rowset

The MEASURES rowset contains information about the available measures.

**GUID**: MDSCHEMA_MEASURES

Table 7describes the rowset structure.

**Table 7 MDSCHEMA_MEASURES Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| MEASURE_NAME | Member names in the Accounts dimension |
| MEASURE_UNIQUE_NAME | Above member name |
| MEASURE_CAPTION | Above member name |

| Column Name | Essbase Mapping |
|---|---|
| MEASURE_AGGREGATOR | Essbase ADDITION: 1<br>Essbase SUBSTRACTION: 17<br>Essbase MULTIPLICATION:18<br>Essbase DIVISION:19<br>Essbase PERCENT:20<br>Essbase NOOP: 21 |
| DESCRIPTION | Member comment |
| DATA_TYPE | 5 |
| EXPRESSION | Member formula |
| MEASURE_IS_VISIBLE | True |

### Request Example

```
<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_MEASURES</RequestType>
    <Restrictions>
     <RestrictionList>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Basic</CUBE_NAME>
     </RestrictionList>
    </Restrictions>
    <Properties>
     <PropertyList>
      <DataSourceInfo>Provider=Essbase;Data Source=localhost
      </DataSourceInfo>
      <Format>Tabular</Format>
     </PropertyList>
    </Properties>
   </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response Example(truncated)

```
<?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:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
  targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:xml-sql"
  elementFormDefault="qualified">
  <xsd:element name="root">
   <xsd:complexType>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="row" type="row"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="row">
   <xsd:sequence maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="CATALOG_NAME" type="xsd:string"
     sql:field="CATALOG_NAME"/>
    <xsd:element name="CUBE_NAME" type="xsd:string"
     sql:field="CUBE_NAME"/>
    <xsd:element name="MEASURE_NAME" type="xsd:string"
     sql:field="MEASURE_NAME"/>
    <xsd:element name="MEASURE_UNIQUE_NAME" type="xsd:string"
     sql:field="MEASURE_UNIQUE_NAME"/>
    <xsd:element name="MEASURE_CAPTION" type="xsd:string"
     sql:field="MEASURE_CAPTION"/>
    <xsd:element name="MEASURE_AGGREGATOR" type="xsd:int"
     sql:field="MEASURE_AGGREGATOR"/>
    <xsd:element name="DESCRIPTION" type="xsd:string"
     sql:field="DESCRIPTION" minOccurs="0"/>
    <xsd:element name="DATA_TYPE" type="xsd:unsignedShort"
     sql:field="DATA_TYPE"/>
    <xsd:element name="NUMERIC_PRECISION" type="xsd:unsignedShort"
     sql:field="NUMERIC_PRECISION"/>
    <xsd:element name="NUMERIC_SCALE" type="xsd:short"
     sql:field="NUMERIC_SCALE"/>
    <xsd:element name="EXPRESSION" type="xsd:string"
     sql:field="EXPRESSION" minOccurs="0"/>
    <xsd:element name="MEASURE_IS_VISIBLE" type="xsd:boolean"
     sql:field="MEASURE_IS_VISIBLE"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
 <row>
  <CATALOG_NAME>Sample</CATALOG_NAME>
  <CUBE_NAME>Sample.Basic</CUBE_NAME>
  <MEASURE_NAME>Measures</MEASURE_NAME>
  <MEASURE_UNIQUE_NAME>[Measures]</MEASURE_UNIQUE_NAME>
  <MEASURE_CAPTION>Measures</MEASURE_CAPTION>
  <MEASURE_AGGREGATOR>0</MEASURE_AGGREGATOR>
  <DATA_TYPE>5</DATA_TYPE>
  <NUMERIC_PRECISION>0</NUMERIC_PRECISION>
  <NUMERIC_SCALE>0</NUMERIC_SCALE>
  <MEASURE_IS_VISIBLE>true</MEASURE_IS_VISIBLE>
 </row>
 < ..............More Rows..... >
</root>
```

```
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# MDSCHEMA_MEMBERS Rowset

The MEMBERS rowset contains information about the available members.

**GUID:** MDSCHEMA_MEMBERS

Table 8 describes the rowset structure.

**Table 8 MDSCHEMA_MEMBERS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| DIMENSION_UNIQUE_NAME | Dimension name |
| HIERARCHY_UNIQUE_NAME | Dimension name |
| LEVEL_UNIQUE_NAME | Level name |
| LEVEL_NUMBER | Level number |
| GENERATION_NUMBER | Generation number |
| MEMBER_ORDINAL | Member number |
| MEMBER_NAME | Member name |
| MEMBER_UNIQUE_NAME | Unique member name |
| MEMBER_TYPE | 1 (REGULAR) |
| MEMBER_CAPTION | Member name |
| MEMBER_ALIAS | Default alias |
| CHILDREN_CARDINALITY | Child count |
| PARENT_LEVEL | Level number of the parent. For dimension, same level number as the dimension level number |
| PARENT_UNIQUE_NAME | Name of the parent. For dimension, same name as the dimension name |
| PARENT_COUNT | Always 1 |
| DESCRIPTION | Member comment |

### Request Example

```
<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"
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
 <SOAP-ENV:Header>
   <wsse:Security>
    <wsse:UsernameToken>
      <wsse:Username>system</wsse:Username>
      <wsse:Password>password</wsse:Password>
    </wsse:UsernameToken>
   </wsse:Security>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <RequestType>MDSCHEMA_MEMBERS</RequestType>
   <Restrictions>
    <RestrictionList>
     <CATALOG_NAME>Sample</CATALOG_NAME>
     <CUBE_NAME>Basic</CUBE_NAME>
     <DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
    </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Response Example**(truncated)

```
<?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:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
    <m:return xsi:type="xsd:string"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
        targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:sql="urn:schemas-microsoft-com:xml-sql"
        elementFormDefault="qualified">
        <xsd:element name="root">
          <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
              <xsd:element name="row" type="row"/>
            </xsd:sequence>
```

```
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="row">
   <xsd:sequence maxOccurs="unbounded" minOccurs="0">
     <xsd:element name="CATALOG_NAME" type="xsd:string"
      sql:field="CATALOG_NAME"/>
     <xsd:element name="CUBE_NAME" type="xsd:string"
      sql:field="CUBE_NAME"/>
     <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
      sql:field="DIMENSION_UNIQUE_NAME"/>
     <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
      sql:field="HIERARCHY_UNIQUE_NAME"/>
     <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string"
      sql:field="LEVEL_UNIQUE_NAME"/>
     <xsd:element name="LEVEL_NUMBER" type="xsd:unsignedInt"
      sql:field="LEVEL_NUMBER"/>
     <xsd:element name="GENERATION_NUMBER" type="xsd:unsignedInt"
      sql:field="GENERATION_NUMBER"/>
     <xsd:element name="MEMBER_ORDINAL" type="xsd:unsignedInt"
      sql:field="MEMBER_ORDINAL"/>
     <xsd:element name="MEMBER_NAME" type="xsd:string"
      sql:field="MEMBER_NAME"/>
     <xsd:element name="MEMBER_UNIQUE_NAME" type="xsd:string"
      sql:field="MEMBER_UNIQUE_NAME"/>
     <xsd:element name="MEMBER_TYPE" type="xsd:int"
      sql:field="MEMBER_TYPE"/>
     <xsd:element name="MEMBER_CAPTION" type="xsd:string"
      sql:field="MEMBER_CAPTION"/>
     <xsd:element name="MEMBER_ALIAS" type="xsd:string"
      sql:field="MEMBER_ALIAS" minOccurs="0"/>
     <xsd:element name="CHILDREN_CARDINALITY" type="xsd:unsignedInt"
      sql:field="CHILDREN_CARDINALITY"/>
     <xsd:element name="PARENT_LEVEL" type="xsd:unsignedInt"
      sql:field="PARENT_LEVEL"/>
     <xsd:element name="PARENT_UNIQUE_NAME" type="xsd:string"
      sql:field="PARENT_UNIQUE_NAME"/>
     <xsd:element name="PARENT_COUNT" type="xsd:unsignedInt"
      sql:field="PARENT_COUNT"/>
     <xsd:element name="DESCRIPTION" type="xsd:string"
      sql:field="DESCRIPTION" minOccurs="0" />
    </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
<row>
 <CATALOG_NAME>Sample</CATALOG_NAME>
 <CUBE_NAME>Sample.Basic</CUBE_NAME>
 <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
 <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
 <LEVEL_UNIQUE_NAME>[Year].Levels(2)</LEVEL_UNIQUE_NAME>
 <LEVEL_NUMBER>2</LEVEL_NUMBER>
 <GENERATION_NUMBER>1</GENERATION_NUMBER>
 <MEMBER_ORDINAL>1</MEMBER_ORDINAL>
 <MEMBER_NAME>Jan</MEMBER_NAME>
 <MEMBER_UNIQUE_NAME>[Jan]</MEMBER_UNIQUE_NAME>
 <MEMBER_TYPE>1</MEMBER_TYPE>
 <MEMBER_CAPTION>Jan</MEMBER_CAPTION>
 <CHILDREN_CARDINALITY>0</CHILDREN_CARDINALITY>
```

```
        <PARENT_LEVEL>1</PARENT_LEVEL>
        <PARENT_UNIQUE_NAME>[Qtr1]</PARENT_UNIQUE_NAME>
        <PARENT_COUNT>1</PARENT_COUNT>
       </row>
       < ..............More Rows..... >
      </root>
     </m:return>
    </m:DiscoverResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## MDSCHEMA_PROPERTIES Rowset

The PROPERTIES rowset contains information about the available properties for each level of the dimension, assuming that each level defines a class of members. The properties of all members in this class are the same. For a data store that does not support named levels, a dummy level includes all members in the dimension. The name of this level is the same as the name of the dimension.

The default sort order: PROPERTY_TYPE, CATALOG_NAME, SCHEMA_NAME, CUBE_NAME, DIMENSION_UNIQUE_NAME, HIERARCHY_UNIQUE_NAME, and LEVEL_UNIQUE_NAME.

**GUID:** MDSCHEMA_PROPERTIES

Table 9 describes the rowset structure.

**Table 9 MDSCHEMA_PROPERTIES Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| HIERARCHY_UNIQUE_NAME | Dimension name |
| LEVEL_UNIQUE_NAME | Dimension name |
| PROPERTY_TYPE | 1 (MDPROP_MEMBER) |
| PROPERTY_NAME | One of the following:<br>● For attribute dimension, the name of the dimension is the name of the property<br>● For UDA, the UDA name<br>● For aliases, the alias name |
| PROPERTY_CAPTION | One of the following:<br>● For attribute dimensions, the attribute dimension name<br>● For UDA, the UDA name<br>● For aliases, the alias name |

| Column Name | Essbase Mapping |
|---|---|
| DATA_TYPE | 1 (double) – attribute dimension<br>2 (boolean) – attribute dimension<br>3 (string) – attribute dimension, UDA or alias<br>4 (integer) – attribute dimension |
| CHARACTER_MAXIMUM_LENGTH | 80 (for UDA or an attribute dimension)<br>30 (for alias) |
| CHARACTER_OCTET_LENGTH | 320 (for UDA or an attribute dimension)<br>120 (for alias) |
| PROPERTY_CONTENT_TYPE | 0 (MD_PROPTYPE_REGULAR) |
| SQL_COLUMN_NAME | One of the following:<br>● For attribute dimensions, the attribute dimension name<br>● For UDA, the UDA name<br>● For aliases, the alias name |
| PROPERTY_ORIGIN | 1 (MD_USER_DEFINED) |
| PROPERTY_ATTRIBUTE_HIERARCHY_NAME | For attribute dimensions, the attribute dimension name |
| PROPERTY_CARDINALITY | ONE (for UDA and aliases)<br>MANY (for attribute dimension) |
| PROPERTY_IS_VISIBLE | True |

### Request Example

```
<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_PROPERTIES</RequestType>
    <Restrictions>
     <RestrictionList>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Basic</CUBE_NAME>
      <DIMENSION_UNIQUE_NAME>Product</DIMENSION_UNIQUE_NAME>
      <LEVEL_UNIQUE_NAME>SKU</LEVEL_UNIQUE_NAME>
     </RestrictionList>
    </Restrictions>
    <Properties>
     <PropertyList>
      <DataSourceInfo>
       Provider=Essbase;Data Source=localhost
      </DataSourceInfo>
      <Format>Tabular</Format>
```

```
     </PropertyList>
    </Properties>
   </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Response Example**(truncated)

```
<?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:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
    <m:return xsi:type="xsd:string"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
       targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:sql="urn:schemas-microsoft-com:xml-sql"
       elementFormDefault="qualified">
       <xsd:element name="root">
        <xsd:complexType>
         <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="row" type="row"/>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
       <xsd:complexType name="row">
        <xsd:sequence maxOccurs="unbounded" minOccurs="0">
         <xsd:element name="CATALOG_NAME" type="xsd:string"
          sql:field="CATALOG_NAME"/>
         <xsd:element name="CUBE_NAME" type="xsd:string"
          sql:field="CUBE_NAME"/>
         <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
          sql:field="DIMENSION_UNIQUE_NAME"/>
         <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
          sql:field="HIERARCHY_UNIQUE_NAME"/>
         <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string"
          sql:field="LEVEL_UNIQUE_NAME" minOccurs ="0"/>
         <xsd:element name="MEMBER_UNIQUE_NAME" type="xsd:string"
          sql:field="MEMBER_UNIQUE_NAME" minOccurs ="0"/>
         <xsd:element name="PROPERTY_TYPE" type="xsd:short"
          sql:field="PROPERTY_TYPE" minOccurs ="0"/>
         <xsd:element name="PROPERTY_NAME" type="xsd:string"
          sql:field="PROPERTY_NAME" minOccurs ="0"/>
         <xsd:element name="PROPERTY_CAPTION" type="xsd:string"
          sql:field="PROPERTY_CAPTION" minOccurs ="0"/>
         <xsd:element name="DATA_TYPE" type="xsd:unsignedShort"
          sql:field="DATA_TYPE" minOccurs ="0"/>
         <xsd:element name="CHARACTER_MAXIMUM_LENGTH"
          type="xsd:unsignedInt"
          sql:field="CHARACTER_MAXIMUM_LENGTH" minOccurs ="0"/>
         <xsd:element name="CHARACTER_OCTET_LENGTH" type="xsd:unsignedInt"
```

```
          sql:field="CHARACTER_OCTET_LENGTH" minOccurs ="0"/>
         <xsd:element name="NUMERIC_PRECISION" type="xsd:unsignedShort"
          sql:field="NUMERIC_PRECISION" minOccurs ="0"/>
         <xsd:element name="NUMERIC_SCALE" type="xsd:short"
          sql:field="NUMERIC_SCALE" minOccurs ="0"/>
         <xsd:element name="DESCRIPTION" type="xsd:string"
          sql:field="DESCRIPTION" minOccurs ="0"/>
         <xsd:element name="PROPERTY_CONTENT_TYPE" type="xsd:short"
          sql:field="PROPERTY_CONTENT_TYPE" minOccurs ="0"/>
         <xsd:element name="SQL_COLUMN_NAME" type="xsd:string"
          sql:field="SQL_COLUMN_NAME" minOccurs ="0"/>
         <xsd:element name="LANGUAGE" type="xsd:unsignedShort"
          sql:field="LANGUAGE" minOccurs ="0"/>
         <xsd:element name="PROPERTY_ORIGIN" type="xsd:unsignedShort"
          sql:field="PROPERTY_ORIGIN" minOccurs ="0"/>
         <xsd:element name="PROPERTY_ATTRIBUTE_HIERARCHY_NAME"
          type="xsd:string"
          sql:field="PROPERTY_ATTRIBUTE_HIERARCHY_NAME" minOccurs ="0"/>
         <xsd:element name="PROPERTY_CARDINALITY" type="xsd:string"
          sql:field="PROPERTY_CARDINALITY" minOccurs ="0"/>
         <xsd:element name="MIME_TYPE" type="xsd:string"
          sql:field="MIME_TYPE" minOccurs ="0"/>
         <xsd:element name="PROPERTY_IS_VISIBLE" type="xsd:boolean"
          sql:field="PROPERTY_IS_VISIBLE" minOccurs ="0"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:schema>
      <row>
       <CATALOG_NAME>Sample</CATALOG_NAME>
       <CUBE_NAME>Sample.Basic</CUBE_NAME>
       <DIMENSION_UNIQUE_NAME>[Product]</DIMENSION_UNIQUE_NAME>
       <HIERARCHY_UNIQUE_NAME>[Product]</HIERARCHY_UNIQUE_NAME>
       <LEVEL_UNIQUE_NAME>[Product]</LEVEL_UNIQUE_NAME>
       <PROPERTY_TYPE>1</PROPERTY_TYPE>
       <PROPERTY_NAME>Caffeinated</PROPERTY_NAME>
       <PROPERTY_CAPTION>Caffeinated</PROPERTY_CAPTION>
       <DATA_TYPE>2</DATA_TYPE>
       <PROPERTY_CONTENT_TYPE>0</PROPERTY_CONTENT_TYPE>
       <SQL_COLUMN_NAME>Caffeinated</SQL_COLUMN_NAME>
       <PROPERTY_ORIGIN>1</PROPERTY_ORIGIN>
       <PROPERTY_ATTRIBUTE_HIERARCHY_NAME>Caffeinated
       </PROPERTY_ATTRIBUTE_HIERARCHY_NAME>
       <PROPERTY_CARDINALITY>MANY</PROPERTY_CARDINALITY>
       <PROPERTY_IS_VISIBLE>true</PROPERTY_IS_VISIBLE>
      </row>
      < ..............More Rows..............  >
     </root>
    </m:return>
   </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## MDSCHEMA_SETS Rowset

The SETS rowset contains information about the sets in a schema (or the catalog, if the provider does not support schemas).

**GUID:** MDSCHEMA_SETS

Table 10describes the rowset structure.

**Table 10 MDSCHEMA_SETS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| SET_NAME | Name of the set |
| SCOPE | Session |

## MDSCHEMA_LEVELS Rowset

The LEVELS rowset contains information about the levels available in a dimension.

**GUID:** MDSCHEMA_LEVELS

Table 11 describes the rowset structure.

**Table 11 MDSCHEMA_LEVELS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| CATALOG_NAME | Application name |
| CUBE_NAME | Database name |
| DIMENSION_UNIQUE_NAME | Name of the dimension to which the level belongs |
| HIERARCHY_UNIQUE_NAME | Name of the dimension to which the level belongs |
| LEVEL_NAME | Unique level name |
| LEVEL_UNIQUE_NAME | Unique level name |
| LEVEL_CAPTION | Level name |
| LEVEL_NUMBER | Level number |
| LEVEL_CARDINALITY | Number of members in the level |
| LEVEL_TYPE | MDLEVEL_TYPE_ALL (for dimension level)<br>MDLEVEL_TYPE_TIME (for dimension type TIME)<br>MDLEVEL_TYPE_REGULAR (for all others) |
| LEVEL_UNIQUE_SETTINGS | 2 (MDDIMENSIONS_MEMBER_NAME_UNIQUE) |
| LEVEL_IS_VISIBLE | True |
| ESSBASE_GEN_UNIQUE_NAME | Generation unique name |

| Column Name | Essbase Mapping |
|---|---|
| ESSBASE_GEN_CAPTION | Generation caption |

### Request Example

```
<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_LEVELS</RequestType>
   <Restrictions>
    <RestrictionList>
     <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Basic</CUBE_NAME>
      <DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
     </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response 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>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
```

```
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CATALOG_NAME" type="xsd:string"
         sql:field="CATALOG_NAME"/>
        <xsd:element name="CUBE_NAME" type="xsd:string"
         sql:field="CUBE_NAME"/>
        <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
         sql:field="DIMENSION_UNIQUE_NAME"/>
        <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
         sql:field="HIERARCHY_UNIQUE_NAME"/>
        <xsd:element name="LEVEL_NAME" type="xsd:string"
         sql:field="LEVEL_NAME"/>
        <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string"
         sql:field="LEVEL_UNIQUE_NAME"/>
        <xsd:element name="LEVEL_CAPTION" type="xsd:string"
         sql:field="LEVEL_CAPTION"/>
        <xsd:element name="LEVEL_NUMBER" type="xsd:unsignedInt"
         sql:field="LEVEL_NUMBER"/>
        <xsd:element name="LEVEL_CARDINALITY" type="xsd:unsignedInt"
         sql:field="LEVEL_CARDINALITY"/>
        <xsd:element name="LEVEL_TYPE" type="xsd:int"
         sql:field="LEVEL_TYPE"/>
        <xsd:element name="LEVEL_UNIQUE_SETTINGS" type="xsd:int"
         sql:field="LEVEL_UNIQUE_SETTINGS"/>
        <xsd:element name="LEVEL_IS_VISIBLE" type="xsd:boolean"
         sql:field="LEVEL_IS_VISIBLE"/>
        <xsd:element name="DESCRIPTION" type="xsd:string"
         sql:field="DESCRIPTION" minOccurs="0"/>
             <xsd:element name="ESSBASE_GEN_UNIQUE_NAME" type="xsd:string"
                 sql:field="ESSBASE_GEN_UNIQUE_NAME"/>
             <xsd:element name="ESSBASE_GEN_CAPTION" type="xsd:string"
                 sql:field="ESSBASE_GEN_CAPTION"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Sample.Basic</CUBE_NAME>
      <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
      <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
      <LEVEL_NAME>[Year].Levels(2)</LEVEL_NAME>
      <LEVEL_UNIQUE_NAME>[Year].Levels(2)</LEVEL_UNIQUE_NAME>
      <LEVEL_CAPTION>[Year].Level 2</LEVEL_CAPTION>
      <LEVEL_NUMBER>2</LEVEL_NUMBER>
      <LEVEL_CARDINALITY>12</LEVEL_CARDINALITY>
      <LEVEL_TYPE>4</LEVEL_TYPE>
      <LEVEL_UNIQUE_SETTINGS>2</LEVEL_UNIQUE_SETTINGS>
      <LEVEL_IS_VISIBLE>true</LEVEL_IS_VISIBLE>
            <ESSBASE_GEN_UNIQUE_NAME>[Year].[Months]</ESSBASE_GEN_UNIQUE_NAME>
            <ESSBASE_GEN_CAPTION>[Year].Months</ESSBASE_GEN_CAPTION>
     </row>
     < ..............More Rows..... >
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# DISCOVER_SCHEMA_ROWSETS Rowset

GUID: DISCOVER_SCHEMA_ROWSETS

Table 12describes the rowset structure.

**Table 12 DISCOVER_SCHEMA Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| SchemaName | 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. |
| Restrictions | List of restrictions allowed |
| Description | Description of the schema |

# DISCOVER_DATASOURCES Rowset

GUID: DISCOVER_DATASOURCES

Table 13 describes the rowset structure.

**Table 13 DISCOVER_DATASOURCES Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| DataSourceName | Name of the data source |
| DataSourceDescription | Description of the data source |
| DataSourceInfo | Provider=Essbase Data Source= name of the Analytic Server |
| ProviderName | XMLA for Essbase |
| ProviderType | MDP |
| AuthenticationMode | Authenticated |

# DISCOVER_PROPERTIES Rowset

GUID: DISCOVER_PROPERTIES

Table 14 describes the rowset structure.

**Table 14 DISCOVER_PROPERTIES Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| PropertyName | Name of the property |
| PropertyDescription | Description of the property |
| PropertyType | XML data type of the property. |

| Column Name | Essbase Mapping |
|---|---|
| PropertyAccessType | Access for the property. The value can be Read, Write, or ReadWrite |
| IsRequired | True if a property is required, false if it is not required |
| Value | Current value of the property |

### Request Example

```
<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>DISCOVER_PROPERTIES</RequestType>
    <Restrictions>
     <RestrictionList></RestrictionList>
    </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response 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>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
```

```
      </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="PropertyName" type="xsd:string"
         sql:field="PropertyName"/>
        <xsd:element name="PropertyDescription" type="xsd:string"
         sql:field="PropertyDescription"/>
        <xsd:element name="PropertyType" type="xsd:string"
         sql:field="PropertyType"/>
        <xsd:element name="PropertyAccessType" type="xsd:string"
         sql:field="PropertyAccessType"/>
        <xsd:element name="IsRequired" type="xsd:boolean"
         sql:field="IsRequired"/>
        <xsd:element name="Value" type="xsd:string"
         sql:field="Value"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <PropertyName>ProviderName</PropertyName>
      <PropertyDescription>The name of the Analytic Services Provider
      </PropertyDescription>
      <PropertyType>string</PropertyType>
      <PropertyAccessType>Read</PropertyAccessType>
      <IsRequired>false</IsRequired>
      <Value>Analytic Services XML for Analysis Provider</Value>
     </row>
     < ..............More Rows.............. >
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# DISCOVER_ENUMERATORS Rowset

**GUID:** DISCOVER_ENUMERATORS

Table 15 describes the rowset structure.

**Table 15 DISCOVER_ENUMERATORS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| EnumName | Name of the enumerator that contains a set of values |
| EnumDescription | Description of the enumerator |
| ElementName | Name of one of the value elements in the enumerator set<br>Example: TDP |
| ElementDescription | Description of the element |
| EnumType | Data type of the Enum values |

| Column Name | Essbase Mapping |
|---|---|
| ElementValue | Value of the element<br>Example: 01 |

### Request Example

```
<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>DISCOVER_ENUMERATORS</RequestType>
  <Restrictions>
   <RestrictionList></RestrictionList>
  </Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>
     Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Format>Tabular</Format>
   </PropertyList>
  </Properties>
 </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response 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>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
```

```
      <xsd:complexType  name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="EnumName" type="xsd:string"
         sql:field="EnumName"/>
        <xsd:element name="EnumDescription" type="xsd:string"
         sql:field="EnumDescription" minOccurs="0"/>
        <xsd:element name="ElementName" type="xsd:string"
         sql:field="ElementName"/>
        <xsd:element name="ElementDescription" type="xsd:string"
         sql:field="ElementDescription" minOccurs="0"/>
        <xsd:element name="ElementValue" type="xsd:string"
         sql:field="ElementValue" minOccurs="0"/>
        <xsd:element name="EnumType" type="xsd:string"
         sql:field="EnumType"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row>
      <EnumName>ProviderType</EnumName>
      <ElementName>TDP</ElementName>
      <EnumType>string</EnumType>
     </row>
     < ..............More Rows.............. >
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# DISCOVER_KEYWORDS Rowset

**GUID:** DISCOVER_KEYWORDS

Table 16 describes the rowset structure.

**Table 16 DISCOVER_KEYWORDS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| Keyword | A list of keywords reserved by a provider<br>Example: AND |

### Request Example

```
<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>DISCOVER_KEYWORDS</RequestType>
   <Restrictions>
    <RestrictionList></RestrictionList>
   </Restrictions>
   <Properties>
```

```
    <PropertyList>
     <DataSourceInfo>
      Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response 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>
  <m:DiscoverResponse
   xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
       <xsd:element name="root">
        <xsd:complexType>
         <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="row" type="row"/>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="Keyword" type="xsd:string"
         sql:field="Keyword"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:schema>
     <row><Keyword>aggregate</Keyword></row>
     <row><Keyword>ancestors</Keyword></row>
     < ................More Rows..... >
    </root>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## DISCOVER_LITERALS Rowset

**GUID:** DISCOVER_LITERALS

the section called "Example 1" describes the rowset structure.

**Table 17 DISCOVER_LITERALS Rowset Structure**

| Column Name | Essbase Mapping |
|---|---|
| LiteralName | Name of the literal described in the row<br>Example: DBLITERAL_LIKE_PERCENT |
| LiteralValue | Contains the literal value<br>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 "%." |
| LiteralInvalidChars | Characters, in the literal, that are not valid<br>Example: If table names can contain anything other than a numeric character, this string would be "0123456789" |
| LiteralInvalidStartingChars | Characters that are not valid as the first character of the literal. If the literal can start with any valid character, this is null. |
| LiteralMaxLength | Maximum number of characters in the literal. If there is no maximum or the maximum is unknown, the value is -1. |

# Flattened Rowset Examples

Flattening a rowset is a way to present multidimensional data in a grid. This two-dimensional, tabular presentation of data can facilitate understanding of the output of a multidimensional XMLA request.

## MDX Examples

The following examples illustrate flattened rowsets as MDX queries and results. MDX is used for ease of presentation; however, the example queries are intended to be considered in terms of XMLA SOAP requests. Remember that in XMLA, level 0 represents a dimension, rather than a leaf member, as in MDX. Therefore, although these examples are in MDX, the levels are reversed as if they were in XMLA.

### Example 1

The following query requests all members of level 1.

```
SELECT NON EMPTY {[Profit]} ON COLUMNS,
NON EMPTY [Product].Levels(1).ALLMEMBERS ON ROWS
FROM Sample.Basic
```

This query has the following result:

| [Product].[Family].[MEMBER_CAPTION] | [Profit] |
|---|---|
| **100** | 30468 |
| **200** | 27954 |
| **300** | 25799 |
| **400** | 21301 |
| **Diet** | 28826 |

## Example 2

The following query requests a maximum of two levels. The flattening of rowsets includes level 1 in this request for levels(2). When using flattened rowsets, if you query for level N, levels 1 through N are returned.

```
SELECT NON EMPTY {[Profit] } ON COLUMNS,
NON EMPTY [Product].Levels(2).ALLMEMBERS ON ROWS
FROM Sample.Basic
```

This query has the following result (truncated):

| [Product].[Family].[MEMBER_CAPTION] | [Product].[SKU].[MEMBER_CAPTION] | [Profit] |
|---|---|---|
| **100** | **100-10** | 22777 |
| **100** | **100-20** | 5708 |
| **100** | **100-30** | 1983 |
| **200** | **200-10** | 7201 |
| **200** | **200-20** | 12025 |
| **200** | **200-30** | 4636 |
| **200** | **200-40** | 4092 |
| **...** | **...** | ... |

## Example 3

The following query builds on the previous, and also asks for the result set to include the member unique name and level number properties for the set of levels 1 through N, where N=2. Each member and each property is allotted a row.

```
SELECT NON EMPTY {[Profit]} ON COLUMNS,
NON EMPTY [Product].Levels(2).ALLMEMBERS
 DIMENSION PROPERTIES MEMBER_UNIQUE_NAME, LEVEL_NUMBER
ON ROWS
FROM Sample.Basic
```

This query has the following result (truncated):

| [Product]. [Family]. [MEMBER_ UNIQUE_ NAME] | [Product]. [Family]. LEVEL_ NUMBER | [Product]. [SKU]. [MEMBER_ UNIQUE_ NAME] | [Product]. [SKU]. LEVEL_ NUMBER | [Profit] |
|---|---|---|---|---|
| **[100]** | 1 | [100-10] | 2 | 22777 |
| **[100]** | 1 | [100-20] | 2 | 5708 |
| **[100]** | 1 | [100-30] | 2 | 1983 |
| **[200]** | 1 | [200-10] | 2 | 7201 |
| **[200]** | 1 | [200-20] | 2 | 12025 |
| **[200]** | 1 | [200-30] | 2 | 4636 |
| **[200]** | 1 | [200-40] | 2 | 4092 |
| **[300]** | 1 | [300-10] | 2 | 12195 |
| **[300]** | 1 | [300-20] | 2 | 2511 |
| **[300]** | 1 | [300-30] | 2 | 2511 |
| **...** | ... | ... | ... | ... |

## Example 4

By implementing CrossJoin in a flattened rowsets query, you can use multiple dimensions (at least two). In this example, Market and Product dimensions are requested. For each dimension, the same logic as in previous examples applies: Each dimension, level, and property is allotted one column (in this case, one level and one property are requested).

```
SELECT NON EMPTY {[Profit] } ON COLUMNS,
NON EMPTY Crossjoin ([Market].Levels(1).AllMembers,[Product].Levels(1).ALLMEMBERS)
  DIMENSION PROPERTIES MEMBER_CAPTION
ON ROWS
FROM Sample.Basic
```

This query has the following result (truncated):

| [Market]. Levels(1). [MEMBER_CAPTION] | [Product]. [Family]. [MEMBER_CAPTION] | [Profit] |
|---|---|---|
| **East** | **Colas** | 12656 |
| **East** | **Root Beer** | 2534 |
| **East** | **Cream Soda** | 2627 |
| **East** | **Fruit Soda** | 6344 |
| **East** | **Diet Drinks** | 2408 |
| **West** | **Colas** | 3549 |

| [Market]. Levels(1). [MEMBER_CAPTION] | [Product]. [Family]. [MEMBER_CAPTION] | [Profit] |
|---|---|---|
| **West** | **Root Beer** | 9727 |
| **West** | **Cream Soda** | 10731 |
| **West** | **Fruit Soda** | 5854 |
| **West** | **Diet Drinks** | 8087 |
| **...** | **...** | ... |

## Example 5

In this example, CrossJoin is used to request levels 1–2 for Market and Product.

```
SELECT NON EMPTY { [Profit] } ON COLUMNS,
NON EMPTY Crossjoin ([Market].Levels(2).AllMembers,[Product].Levels(2).ALLMEMBERS)
   DIMENSION PROPERTIES MEMBER_CAPTION
ON ROWS
FROM Sample.Basic
```

This query has the following result (truncated):

| [Market]. Levels(1). [MEMBER_ CAPTION] | [Market]. Levels(2). [MEMBER_ CAPTION] | [Product]. [Family]. [MEMBER_ CAPTION] | [Product]. [SKU]. [MEMBER_ CAPTION] | [Profit] |
|---|---|---|---|---|
| **East** | **New York** | **Colas** | **Cola** | 3498 |
| **East** | **New York** | **Root Beer** | **Old Fashioned** | -2594 |
| **East** | **New York** | **Root Beer** | **Birch Beer** | 3086 |
| **East** | **New York** | **Cream Soda** | **Dark Cream** | 2496 |
| **East** | **New York** | **Cream Drinks** | **Vanilla Cream** | -1952 |
| **East** | **New York** | **Fruit Soda** | **Grape** | 1329 |
| **East** | **New York** | **Fruit Soda** | **Orange** | 1388 |
| **East** | **New York** | **Fruit Soda** | **Strawberry** | 951 |
| **...** | **...** | **...** | **...** | ... |

## Example 6

The following example uses CrossJoin to represent multiple dimensions, requests a different number of levels for each dimension, and requests multiple properties.

```
SELECT NON EMPTY { [Profit] } ON COLUMNS,
NON EMPTY Crossjoin ([Market].Levels(1).AllMembers,[Product].Levels(2).ALLMEMBERS)
   DIMENSION PROPERTIES MEMBER_CAPTION, LEVEL_NUMBER
ON ROWS
FROM Sample.Basic
```

This query has the following result (truncated):

| [Market]. Levels(1). [MEMBER_ CAPTION] | [Market]. Levels(1). [LEVEL_ NUMBER] | [Product]. [Family]. [MEMBER_ CAPTION] | [Market]. Levels(1). [LEVEL_ NUMBER] | [Product]. [SKU]. [MEMBER_ CAPTION] | [Market]. Levels(1). [LEVEL_ NUMBER] | [Profit] |
|---|---|---|---|---|---|---|
| **East** | 1 | **Colas** | 1 | **Cola** | 2 | 11129 |
| **East** | 1 | **Colas** | 1 | **Diet Cola** | 2 | 1114 |
| **East** | 1 | **Colas** | 1 | **Caffeine Free Cola** | 2 | 413 |
| **East** | 1 | **Root Beer** | 1 | **Old Fashioned** | 2 | -2540 |
| **East** | 1 | **Root Beer** | 1 | **Diet Root Beer** | 2 | 982 |
| **East** | 1 | **Root Beer** | 1 | **Birch Beer** | 2 | 4092 |
| **East** | 1 | **Cream Soda** | 1 | **Dark Cream** | 2 | 3233 |
| **East** | 1 | **Cream Soda** | 1 | **Vanilla Cream** | 2 | -918 |
| **...** | ... | **...** | ... | **...** | ... | ... |

## Example 7

The following example uses multiple, nested CrossJoins.

```
SELECT NON EMPTY { [Profit] } ON COLUMNS,
NON EMPTY {CROSSJOIN
                    (
                      CROSSJOIN( [Market].Levels(1).ALLMEMBERS,
                                 [Product].[Family].ALLMEMBERS
                               ),
                      [Year].Levels(1).ALLMEMBERS
                     )
           } DIMENSION PROPERTIES MEMBER_CAPTION
ON ROWS FROM Sample.Basic
```

This query has the following result (truncated):

| [Market]. Levels(1). [MEMBER_ CAPTION] | [Product]. [Family]. [MEMBER_ CAPTION] | [Year]. Levels(1). [MEMBER_ CAPTION] | [Profit] |
|---|---|---|---|
| East | Colas | Qtr1 | 2747 |
| East | Colas | Qtr2 | 3352 |
| East | Colas | Qtr3 | 3740 |
| East | Colas | Qtr4 | 2817 |
| East | Root Beer | Qtr1 | 562 |
| East | Root Beer | Qtr2 | 610 |

| [Market]. Levels(1). [MEMBER_ CAPTION] | [Product]. [Family]. [MEMBER_ CAPTION] | [Year]. Levels(1). [MEMBER_ CAPTION] | [Profit] |
|---|---|---|---|
| East | Root Beer | Qtr3 | 372 |
| East | Root Beer | Qtr4 | 990 |
| ... | ... | ... | ... |

## XMLA Examples

The following examples illustrate an XMLA response and request.

This is an example of a flattened rowset request. To flatten the result, you must use Tabular format in the PropertyList element, as shown in the example.

```
<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>
     WITH MEMBER [Year].[calctest] AS '4'
     SELECT NON EMPTY { [Profit] } ON COLUMNS,
     NON EMPTY {[Year].ALLMEMBERS } ON ROWS
     FROM Sample.Basic
    </Statement>
   </Command>
   <Properties>
    <PropertyList>
     <DataSourceInfo>Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Catalog>Sample</Catalog>
     <Format>Tabular</Format>
     <AxisFormat>TupleFormat</AxisFormat>
    </PropertyList>
   </Properties>
  </Execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

An example of a flattened rowset response:

```
<?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:rowset"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
 xmlns:sql="urn:schemas-microsoft-com:xml-sql"
 elementFormDefault="qualified">
 <xsd:element name="root">
  <xsd:complexType>
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="row" type="row" />
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:complexType name="row">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
   <xsd:element name="column1" type="xsd:string"
    sql:field="[Year].Levels(1).[MEMBER_CAPTION]" minOccurs="0"/>
   <xsd:element name="column2" type="xsd:string"
    sql:field="[Year].Levels(2).[MEMBER_CAPTION]" minOccurs="0"/>
   <xsd:element name= "column3" type="xsd:double"
    sql:field= "[Profit]" minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>

<row>
 <column3>105522.000000</column3>
</row>
<row>
 <column1>Qtr1</column1>
 <column3>24703.000000</column3>
</row>
<row>
 <column1>Qtr1</column1>
 <column2>Jan</column2>
 <column3>8024.000000</column3>
</row>
<row>
 <column1>Qtr1</column1>
 <column2>Feb</column2>
 <column3>8346.000000</column3>
</row>
<row>
 <column1>Qtr1</column1>
 <column2>Mar</column2>
 <column3>8333.000000</column3>
</row>
<row>
 <column1>Qtr2</column1>
 <column3>27107.000000</column3>
</row>
<row>
 <column1>Qtr2</column1>
 <column2>Apr</column2>
 <column3>8644.000000</column3>
</row>
<row>
 <column1>Qtr2</column1>
```

```
        <column2>May</column2>
        <column3>8929.000000</column3>
       </row>
       <row>
        <column1>Qtr2</column1>
        <column2>Jun</column2>
        <column3>9534.000000</column3>
       </row>
       <row>
        <column1>Qtr3</column1>
        <column3>27912.000000</column3>
       </row>
       <row>
        <column1>Qtr3</column1>
        <column2>Jul</column2>
        <column3>9878.000000</column3>
       </row>
       <row>
        <column1>Qtr3</column1>
        <column2>Aug</column2>
        <column3>9545.000000</column3>
       </row>
       <row>
        <column1>Qtr3</column1>
        <column2>Sep</column2>
        <column3>8489.000000</column3>
       </row>
       <row>
        <column1>Qtr4</column1>
        <column3>25800.000000</column3>
       </row>
       <row>
        <column1>Qtr4</column1>
        <column2>Oct</column2>
        <column3>8653.000000</column3>
       </row>
       <row>
        <column1>Qtr4</column1>
        <column2>Nov</column2>
        <column3>8367.000000</column3>
       </row>
       <row>
        <column1>Qtr4</column1>
        <column2>Dec</column2>
        <column3>8780.000000</column3>
       </row>
       <row>
        <column1>calctest</column1>
        <column3>4.000000</column3>
       </row>
      </root>
     </m:return>
    </m:ExecuteResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# 4 Working with Java API

In This Chapter



## Key Features

- "100% Pure Java" solution
- Embedded versus three-tier (APS) Java API deployment
- Pure Java implementation is more efficient than JNI wrapper implementations around CAPI
- Java API can be embedded in the client Java application of a two-tier solution
- Java API can be embedded in the mid-tier server of a three-tier Java application
- Embedded Java API is a set of JAR and property files
- Through Embedded Java API, client applications communicate directly to Analytic Server
- Easy to deploy and use
- Easy to switch between embedded and three-tier modes

## Embedded JAPI

JAPI can be embedded in a front-end Java application of a two-tier architecture or in the middle tier of a multi-tier architecture. JAPI communicates directly and through TCP/IP to Analytic Servers. No mid-tier Provider Services server is necessary.

A sample batch file `runsamplesEmbedded.cmd` in *`EPM_ORACLE_HOME`*`/common/EssbaseJavaAPI/11.1.2.0/samples/japi` illustrates the use of a JAPI sample in Embedded mode.

➤ To set up a client application to use embedded Java API:

1. Set ESS_ES_HOME to the root of Embedded JAPI installation and pass it to the JVM. For example:

   ```
   java -DESS_ES_HOME=<root of Embedded JAPI installation>
   ```

Include the following JAR files, which are necessary for Embedded JAPI to work, in your CLASSPATH:

- `%MIDDLEWARE_HOME%\EPMSystem11R1\common\EssbaseJavaAPI\11.1.2.0\lib\ess_japi.jar`
- `%MIDDLEWARE_HOME%\EPMSystem11R1\common\EssbaseJavaAPI\11.1.2.0\lib\ess_es_server.jar`
- `%MIDDLEWARE_HOME%\EPMSystem11R1\common\essbase-studio-sdk\11.1.2.0\lib\cpld.jar`
- `%MIDDLEWARE_HOME%oracle_common\modules\oracle.odl_11.1.1\ojdl.jar`

To switch between Embedded JAPI and three-tier APS JAPI, the value of the String providerUrl in the signOn API must change. For Embedded JAPI, this is the String `embedded`. For three-tier Provider Services mode, it is the URL to the Provider Services instance.

A sample batch file `runsamplesAPS.cmd` in *`EPM_ORACLE_INSTANCE`*`/common/EssbaseJavaAPI/11.1.2.0/samples/japi` illustrates the use of a JAPI sample in three-tier Provider Services mode.

The API is the same for both Embedded JAPI as well as three-tier JAPI through Provider Services. The difference between the two is the parameter providerUrl.

**signOn APIs in Iessbase interface — Embedded JAPI**

```
public IEssDomain signOn(java.lang.String userName,
     java.lang.String password,
     boolean passwordIsToken,
     java.lang.String userNameAs,
     java.lang.String providerUrl
throws EssException
```

*Parameters:*

userName—The user name. Can be null if password is cssToken and the passwordIsToken flag is true.

password—The user password. Cannot be null. If the passwordIsToken flag is true, this represents the cssToken string.

passwordIsToken—A boolean indicating whether the password is cssToken string.

userNameAs—The user name you want to impersonate. If null, no impersonation occurs.

providerUrl—The URL of the Provider Services servlet (For embedded mode pass "embedded"). In the embedded mode, the JAPI client and provider are in the same process space, and JAPI talks directly to the OLAP server. (No separate provider application needs to be running.)

*Returns*:

The signed on domain.

**signOn APIs in Iessbase interface — Three tiered JAPI**

```
public IEssOlapServer signOn(java.lang.String userName,
     java.lang.String password,
     boolean passwordIsToken,
```

```
     java.lang.String userNameAs,
     java.lang.String providerUrl,
     java.lang.String olapServerName)
throws EssException
```

*Parameters*:

userName—The user name. Can be null if password is cssToken and the passwordIsToken flag is true.

password—The user password. Cannot be null. If the passwordIsToken flag is true, this represents the cssToken string.

passwordIsToken—A boolean indicating whether the password is cssToken string.

userNameAs—The user name you want to impersonate. If null, no impersonation occurs.

providerUrl—The URL of the Provider Services servlet (For embedded mode pass "embedded"). In the embedded mode, the JAPI client and provider are in the same process space, and JAPI talks to Analytic Server directly. (No separate provider application needs to be running.)

olapServerName—The host name where Essbase Server is running.

*Returns*:

The connected Essbase Server instance.

# 5 Working with Essbase Web Services

In This Chapter



## Key Features

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. Web services use XML to code and decode data, and SOAP (Simple Object Access Protocol) to transport it. Web services are defined using WSDL (Web Service Description Language).

Essbase Web Services support access to and administration of Essbase applications and cubes. Essbase Web Services include the following modules:

- Datasource
- Administration
- Data and Metadata Query

## Deploying Web Services

Oracle recommends deploying Web Services in secure mode with Oracle Web Services Manager (OWSM).

**Note:** You must also configure the Provider Services deployment in Oracle WebLogic Server to support OWSM. For information on OWSM, see the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

➤ To configure Web Services to support OWSM:

1 **Stop Provider Services.**

2. **Find and open** `essbase.properties` **in** *`Middleware_HOME`*`\user_projects\domains\epmsystem1\aps\bin`.
3. **Set** `essbase.webservices.disable.owsm` **to** `False`.
4. **Save and close** `essbase.properties`.
5. **Restart Provider Services.**

To use Web Services in non-OWSM mode, set the `essbase.webservices.disable.owsm` property in `essbase.properties` to `True` Client applications should send the non-OWSM or native Essbase user credentials using a SOAP message header in `ClientMessageHandler` similar to the following:

```
SOAPElement el = header.addHeaderElement(envelope.createName("Parameters", "", "http://
context.webservices.epm.oracle/"));
                SOAPElement el0 = el.addChildElement(envelope.createName("UserName", "",
"http://context.webservices.epm.oracle/"));
                 el0.setValue(s_userName);
                SOAPElement el1 = el.addChildElement(envelope.createName("Password", "",
"http://context.webservices.epm.oracle/"));
                 el1.setValue(s_password);
                 SOAPElement el2 = el.addChildElement(envelope.createName("IsToken", "",
"http://context.webservices.epm.oracle/"));
                 el2.setValue(Boolean.toString(b_isPwdToken));
                 SOAPElement el3 = el.addChildElement(envelope.createName("UserNameAs",
"", "http://context.webservices.epm.oracle/"));
                 el3.setValue(userNameAs);
```

You will also need to modify your Oracle WebLogic Server deployment to use Web Services in non-OWSM mode.

# Datasource

You use the Datasource service to identify and connect with Essbase servers, applications, cubes, and dimensions. You can perform the following operations:

- GetTypes()

  ```
  Input: none
  Output: BaseObject[] (array)
  ```

- Get()

  ```
  Input: URI
  Output: BaseObject
  ```

- Create()

  ```
  Input: BaseObject
  Output: BaseObject
  ```

- Rename()

  ```
  Input: URI, String(newName)
  Output: None
  ```

- Update()

```
Input: URI, Base Object (application or cube)
Output: BaseObject
```

- Delete()

```
Input: URI
Output: None
```

- List()

```
Input: UriType, URI
Output: BaseObject[]
```

# Administration

You use Administration Web Services to start, monitor, or stop Essbase servers, applications, and cubes and to perform operations, such as executing MaxL scripts. You can perform the following operations:

- Start()

```
Input: URI (Essbase application or cube)
Output: Void
```

- Stop()

```
Input: URI (Essbase application or cube)
Output: Void
```

- Ping()

```
Input: URI (Essbase application or cube)
Output: Packet round trip time in milliseconds
```

- Perform()

```
Input: MaxL statement
Output: Two-dimensional string array of MaxL result set
```

# Query

You use Query Web Services to perform data and metadata queries, update data, and retrieve metadata from Essbase servers, applications, and cubes. You can perform the following operations:

## Data Queries

- Retrieve()

```
Input:
    URI: (Essbase cube)
    Options: Grid operation parameters
    Grid: Input grid
Output: Output grid
```

- UpdateData()

```
Input:
    Cube: URI (Essbase cube)
    Options: Grid operation parameters
    Grid: Input grid
Output: Output Grid
```

- zoomIn()

```
Input:
    URI: (Essbase cube)
    ZoomOptions: Zoom in operation parameters
    Range: Grid range to zoom in on
    Grid: Input grid
Output: Output grid
```

- zoomOut()

```
Input:
    URI: (Essbase cube)
    ZoomOptions: Zoom out operation parameters
    Range: Grid range to zoom out of
    Grid: Input grid
Output: Output grid
```

- pivot()

```
Input:
    URI: (Essbase cube)
    Options: Grid operations parameters
    FromPivotLocation: Location member is pivoted from
    ToPivotLocation: Location member is pivoted to

Output: Grid
```

  Pivot pivots the grid on a selected member to a new location. The resulting grid is returned.

- keepOnly()

```
Input:
    URI: Essbase cube location
    Options: Grid operation parameters
    Range: Grid range to keep
    Grid: Input grid
Output: Output grid
```

- removeOnly()

```
Input:
    URI: Essbase cube location
    Options: Grid operation parameters
    Range: Grid range to remove
    Grid: Input grid
Output: Output grid
```

- execute()

```
Input:
    Cube: URI (Essbase cube)
    Text: MDX query statement
    Options: MDX query options
Output: Either an MDX MDData result set or a grid based on resultFormat specified as
part of input Options
```

## Metadata Queries

- queryMemberHeader

  ```
  In: URI (Essbase cube), QueryMemberOptions
  Out: Essbase member array
  ```

- queryMember

  ```
  In: URI (Essbase cube), QueryMemberOptions
  Out: Essbase member array
  ```

- queryMemberReport

  ```
  In: URI (Essbase cube), QueryReportOptions
  Out: Essbase member array
  ```

`QueryMemberOptions` is an object containing two other objects, `QueryOption` and `QueryType`:

```
QueryOption fields in QueryMemberOptions
 QUERY_OPTION_MEMBERSONLY
     QUERY_OPTION_ALIASESONLY
     QUERY_OPTION_MEMBERSANDALIASES
     QUERY_OPTION_COUNTONLY
     QUERY_OPTION_NOTOTALCOUNTS
     QUERY_OPTION_INCLUDEHYBRIDANALYSIS
     QUERY_OPTION_EXCLUDEHYBRIDANALYSIS
     QUERY_OPTION_FORCECASESENSITIVE
     QUERY_OPTION_FORCEIGNORECASE
     QUERY_OPTION_UNIQUENAME
     QUERY_OPTION_USESUBSTITUTIONVAR

QueryType  fields in QueryMemberOptions
     QUERY_TYPE_CHILDREN
     QUERY_TYPE_DESCENDANTS
     QUERY_TYPE_BOTTOMLEVEL
     QUERY_TYPE_SIBLINGS
     QUERY_TYPE_SAMELEVEL
     QUERY_TYPE_SAMEGENERATION
     QUERY_TYPE_PARENT
     QUERY_TYPE_DIMENSION
     QUERY_TYPE_NAMEDGENERATION
     QUERY_TYPE_NAMEDLEVEL
     QUERY_TYPE_SEARCH
     QUERY_TYPE_WILDSEARCH
     QUERY_TYPE_USERATTRIBUTE
     QUERY_TYPE_ANCESTORS
     QUERY_TYPE_DTSMEMBERS
     QUERY_TYPE_DIMUSERATTRIBUTES
     QUERY_TYPE_INDEPDIMS
     QUERY_TYPE_INDEPDIMS_DISCRETE
     QUERY_TYPE_INDEPDIMS_CONTINUOUS
```

`QueryReportOptions` is an object containing two other objects, `fieldSelection` and `mbrSelection`.

# Writing Client Programs

➤ To create a client program using Web Services:

1. **Deploy the Web Services `aps.ear` file.**
2. **Determine, using WSDL, which services are available.**
3. **In a development environment (Java or C++), generate client program stubs.**
4. **Using the classes included in these stubs, develop your program.**

# Sample Client Programs

The file `build.xml` in *APS_HOME*`\ws-samples` contains information on how to create, compile, and execute programs. There is also a sample `ws-build.properties` file, which you modify to match your development environment.

The sample program listing below starts an Essbase application, pings the application to determine the round-trip transit time, and stops the application.

```
public static void doAdmin() throws Exception {

 AdminService adminService = new AdminService(new URL("http://localhost:7101/oracle-
essbase-webservices/AdminService?wsdl"), new  QName ("http://
essbase.webservices.oracle", "AdminService"));

IAdminService admin = adminService.getAdminServicePortType(s_securityFeatures);
IAdminService admin = adminService.getAdminServicePortType();
 // Add OWSM/Non-OWSM header information
 // initialize -- build up server, app, db URI  first
 //create URI

               try {
     // start
                    admin.start(uri);
                    System.out.println("[" + server + ":" + app + ":" + db + "]
started");


     // ping
               long pingRes = admin.ping(uri);
               System.out.println("pingRes : " + pingRes);


               // Stop (server:app)
      admin.stop(uri);

       } catch (Exception ex) {
               ex.printStackTrace();
                 }

        }
```

# 6 Setting Up the Sample Programs

In This Chapter



## The Sample Programs

The sample programs provided with Provider Services help you test the software and get you started on developing client programs for Provider Services.

**Note:** The sample programs are available for use with Essbase only.

In order to run the sample programs, you must configure your environment with the following components:

- A Provider Server
- A supported Essbase server with users and sample applications

Unless otherwise noted in this document, the sample programs assume that all necessary software components (Provider Services, Essbase, and the sample programs themselves) are running on the same computer.

The sample programs demonstrate areas of functionality provided through the Essbase JAPI. The sample programs are located in the directory, *EPM_ORACLE_HOME*/common/EssbaseJavaAPI/11.1.2.0/samples/japi. The areas covered by each sample program are summarized in the following list:

- `Allocation.java` shows how to perform allocation on an Essbase ASO cube.
- `BackupAndRestoreDatabase.java` signs onto an Essbase domain, creates an application and database, backs up and then restores the database.
- `BuildDimension.java` adds and removes members from the outline in the active database.
- `CdfCdm.java` shows the usage of CDF and CDM.
- `CellAttributes.java` signs on to Essbase domain, opens a cube view, performs a retrieval, gets the cell attributes, and signs off.

- `Connect.java` demonstrates a simple connection to an Essbase Server.
- `CopyOlapAppAndCube.java` copies an Essbase application and cube within the same Essbase Server; can be extended to perform copying across Essbase Servers.
- `CreateCluster.java` creates a cluster.
- `CreateOlapApp.java` creates OLAP applications.
- `CreateOutline.java` demonstrates creating a cube outline: creates dimensions, members, and other outline elements; verifies the outline; and restructures the database.
- `CubeDataLoad.java` signs onto the Essbase domain, loads data into a cube, and signs off.
- `CubeLocks.java` signs onto the Essbase domain, gets the list of locks held on a cube, and signs off.
- `CustomCalc.java` demonstrates how to perform custom calculation on an Essbase Aggregate Storage(ASO) cube.
- `CustomMessageHandler.java` demonstates how to use the custom message handler.
- `DataQuery.java` demonstrates basic retrieval of data from an Essbase database.
- `DataSource.java` demonstrates the retrieval of Essbase Server information and execution of reports.
- `Domain.java` demonstrates adding, fetching and removing Essbase Servers from the domain of Provider Services.
- `EditOutline.java` signs onto the Essbase domain, connects to an application and database, and performs almost all edit operations on the database outline.
- `ExecuteMaxL.java` logs on to the Analytic Server, executes MaxL statements, then logs off.
- `GetLoginIDRequest.java` signs onto the Essbase domain and retrieves the login ID of the user signing in; and retrieves the list of executing requests from Essbase Server.
- `GetMembers.java` signs on to Essbase domain, performs various metadata operations and signs off.
- `GetOlapUser`, demonstrates fetching of native and external Essbase users.
- `GridDataUpdate.java` demonstrates the retrieving and updating of data in a grid format.
- `GridLockUnlock.java` signs on to an Essbase domain, opens a cube view, performs a lock, retrieval, unlock, and signs off.
- `GridWithUnknownMembers.java` demonstrates how to detect unknown members in data query.
- `HisDrillThrough.java` signs on to an Essbase domain, opens a cube view, performs Oracle Essbase Integration Services drill-through, lists reports, executes reports, and signs off.
- `HisDrillThroughOnRange.java` signs on to an Essbase domain, opens a cube view, performs Oracle Essbase Integration Services drill-through, lists reports, executes reports, and signs off.

- `HybridAnalysis.java` demonstrates how to use the Hybrid Analysis option for data query and metadata operations.
- `KillOwnRequest.java` kills a running request issued to an Essbase application server from the same user session.
- `LinkedObjects.java` signs on to an Essbase domain, opens a cube view, performs a retrieval, performs LRO operations, and signs off.
- `LinkedPartition.java` signs on to an Essbase domain, opens a cube view, performs a retrieval, looks for a linked partition in a cell and signs off.
- `ListAndKillOlapRequests.java` signs on to an Essbase domain, connects to an Essbase Server, lists the requests, kills requests, and signs off.
- `LoadData.java` loads data to a cube.
- `MdxQuery.java` signs on to an Essbase domain, opens a cube view, performs an MDX query execution, retrieves the results, and signs off.
- `MetaData.java` demonstrates retrieval of metadata information from an Essbase database, including member selection.
- `NonUniqueOutline.java` tests an existing Sample Basic outline to verify that it allows unique members only, creates a new outline enabled for duplicate (non-unique) member names, and uses the duplicate member name JAPIs.
- `NonUniqueQueryOutline.java`, queries a outline enabled for duplicate (non-unique) member names. This sample file is intended to test the duplicate member names outline created using the `NonUniqueOutline.java` sample.
- `PropertyViewer.java` gets an application/database object, enumerates its properties, and prints the values.
- `QueryHints.java` signs on to the Essbase domain, performs various query hints-related operations, and signs off.
- `QueryVaryingAttributes.java` signs onto the Essbase and opens a database; performs a member selection; queries for varying attributes; and signs off.
- `ReadOutline.java` signs on to the Essbase domain, reads various items in an outline and signs off.
- `RunReport.java` demonstrates the running of a report from an Essbase database.
- `RunXmlReport.java` signs on to the Essbase domain, runs a report, and signs off.
- `SecurityFilter.java` tests the security filters.
- `SmartListOutline.java` signs onto the Essbase domain, creates a database with MemberType Enabled outline, verifies Smart List outline editing APIs, performs the grid operations, and deletes the outline.
- `SmartListReport.java` signs onto the Essbase demonstrates grid API report specification usages with respect to querying on smart list, date, and formatted string data cell types.
- `SyncCubeReplicas.java` demonstrates the replication of data between two Essbase databases.

- `TimeDimIntelligence.java` shows the time intelligence APIs related to a "date time" dimension.
- `ViewOutlineTree.java` demonstrates the listing of all outline members from an Essbase database outline.

## Configuring Essbase Servers

The sample programs require an Essbase server with the Demo Basic sample application loaded. You must also create the user “system” on theEssbase server.

If you plan to use several Essbase servers in a cluster with the sample programs, you must perform the following procedure for all servers in the cluster.

➤ To configure one or more instances of Essbase server for use with the sample programs:

1 **On the Essbase server, verify that the sample applications Demo Basic and Sample Basic are installed and that the databases have been loaded with data.**

For information about copying Essbase applications, see the *Essbase Database Administrator's Guide*.

2 **Optional:** If you are using more than one server in a cluster, repeat step 1.

## Compiling and Running the Sample Programs

After you have configured the required servers, you can compile and run the sample programs.

Two script files are provided for compiling the sample programs:

- `runsamplesAPS`: for use in a three-tier Java API deployment
- `runsamplesEmbedded`: for use in an embedded Java API deployment

Depending on the deployment option you choose, these scripts must be configured to work with your computer environment. Once the scripts are working in your environment, you can use them as templates for creating compile and run scripts for the other sample programs.

## Configuring the Script Files

To configure the `runsamplesAPS` and `runsamplesEmbedded` script files to work in your computer environment, you must verify that the path and other environment variables in the scripts are set correctly.

**Note:** The procedure in this section applies to both a three-tier or embedded Java API deployment.

➤ To configure the script files to work with your computer environment:

1 **In the *EPM_ORACLE_HOME*/common/EssbaseJavaAPI/11.1.2.0/samples/japi directory, locate the appropriate script.**

If you are deploying the Java API in three-tier mode:

- `runsamplesAPS.cmd` on Windows
- `runsamplesAPS.sh` on UNIX

If you are deploying the Java API in embedded mode:

- `runsamplesEmbedded.cmd` on Windows
- `runsamplesEmbedded.sh` on UNIX

2 **Using a text editor, open the script file you chose in step 1.**

3 **Verify that the APS_HOME variable is set to the location of your Provider Services installation; for example:**

```
set APS_HOME=C:Oracle\Middleware\EPMSystem11R1\common\EssbaseJavaAPI
\11.1.2.0\
```

4 **Verify that the JAVA_HOME variable points to a supported version of the Java Runtime Environment.**

If you did not install the Java Runtime Environment with Provider Services, you must update this variable with a full path to the Java installation; for example,

`set JAVA_HOME="C:\Oracle\Middleware\jdk160_11"` or `C:\Oracle\Middleware\EPMSystem11R1\common\JRE\Sun\1.6.0`

5 **Replace the variable values for SAMPLE_PROG, USER, PASSWORD, OLAP_SERVER, and PROVIDER_URL as necessary.**

These variables are set at the beginning of the script file.

**Tip:** To make running the example programs easier, set up Oracle Essbase Provider Services and Oracle Essbase on your local computer. Then, in Oracle Essbase Administration Services, create a user "system" with a password of "password" and full access to the Sample Basic, Demo Basic, and Demo2 Basic databases. If you set up your computer system in this configuration, you do not need to modify the default settings for the sample client programs.

6 **Save the script file.**

## Compiling and Running Samples

The scripts, `runsamplesAPS` or `runsamplesEmbedded`, compile all the sample programs but runs only one of them. To run the other sample programs, you must create your own scripts or modify the `runsamplesAPS` or `runsamplesEmbedded` script.

You can use the either of the runsamples scripts as a template for new scripts.

The following procedure shows you how to create a version of the runsamplesAPS script to run a different sample program.

➤ To create a version of the runsamples script to run another sample program:

1 **In the *EPM_ORACLE_HOME*\common\EssbaseJavaAPI\11.1.2.0\samples\japi directory, locate the runsamples script ( .cmd on Windows systems, .sh on UNIX systems).**

2 **Open the runsamples script in a text editor.**

3 **In the script file, find the line that begins with `echo Step-2` and replace %SAMPLE_PROG% with the name of a sample program (listed in "The Sample Programs" on page 81) :**

```
echo Step-2: Ready to run "%SAMPLE_PROG%" example ...
pause
"%JAVA_HOME%\bin\java" com.essbase.samples.japi.%SAMPLE_PROG% %USER% %PASSWORD%
%OLAP_SERVER% %PROVIDER_URL%
```

To use the script to run another sample program, substitute the name of the sample program class for MetaData, as shown in the preceding sample.

4 **Save the script file in the *EPM_ORACLE_HOME*\common\EssbaseJavaAPI\11.1.2.0\samples\japi directory.**

Save the file with a .cmd extension on Windows or a .sh extension on UNIX.

5 **Optional: Repeat this procedure to create a separate script for each sample program that you want to run.**

# Index