# Contents

**Documentation Accessibility** ........................................................... 7

**Documentation Feedback** ............................................................. 9

**Chapter 1. Provider Services Overview** ............................................... 11
  Provider Services Introduction ...................................................... 11
  Java API .................................................................................. 12
  Smart View .............................................................................. 14
  XML for Analysis ................................................................. 14
  Essbase Web Services ............................................................. 14
  EPM System ........................................................................... 15

**Chapter 2. Administering Provider Services** ......................................... 17
  Administering Provider Services .................................................. 17
    Adding Provider Services ...................................................... 18
    Editing the Authenticating Essbase Server ............................... 18
    Removing Provider Services .................................................. 18
    Connecting to a Stand-alone Essbase Server .............................. 19
    Connecting to Provider Services ............................................ 20
    Monitoring Sessions ........................................................... 20
    Specifying Session Timeout ............................................... 21
    Specifying Maximum Rows and Columns ................................. 21
    Automatically Deploying Client Upgrades ................................. 22
    Updating References to Rehosted Servers .................................. 22
  Provider Services Logging ......................................................... 23
  Network Error in Essbase Logs .................................................... 23
  Setting TCP/IP Socket Communication ......................................... 23
  Configuring Options in essbase.properties .................................... 24
    Configuring essbase.properties .............................................. 24
    Configurable Options in essbase.properties ............................... 25
  Clustering .............................................................................. 28
### Chapter 3. Working with XMLA

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Features</td>
<td>29</td>
</tr>
<tr>
<td>Methods</td>
<td>29</td>
</tr>
<tr>
<td>DISCOVER</td>
<td>30</td>
</tr>
<tr>
<td>Execute</td>
<td>33</td>
</tr>
<tr>
<td>XMLA Rowsets</td>
<td>35</td>
</tr>
<tr>
<td>CATALOGS Rowset</td>
<td>35</td>
</tr>
<tr>
<td>MDSHEMA_CUBES Rowset</td>
<td>37</td>
</tr>
<tr>
<td>MDSHEMA_DIMENSIONS Rowset</td>
<td>39</td>
</tr>
<tr>
<td>MDSHEMA_FUNCTIONS Rowset</td>
<td>41</td>
</tr>
<tr>
<td>MDSHEMA_HIERARCHIES Rowset</td>
<td>43</td>
</tr>
<tr>
<td>MDSHEMA_MEASURES Rowset</td>
<td>46</td>
</tr>
<tr>
<td>MDSHEMA_MEMBERS Rowset</td>
<td>49</td>
</tr>
<tr>
<td>MDSHEMA_PROPERTIES Rowset</td>
<td>52</td>
</tr>
<tr>
<td>MDSHEMA_SETS Rowset</td>
<td>56</td>
</tr>
<tr>
<td>MDSHEMA_LEVELS Rowset</td>
<td>56</td>
</tr>
<tr>
<td>DISCOVER_SCHEMA_ROWSETS Rowset</td>
<td>59</td>
</tr>
<tr>
<td>DISCOVER_DATASOURCES Rowset</td>
<td>59</td>
</tr>
<tr>
<td>DISCOVER_PROPERTIES Rowset</td>
<td>60</td>
</tr>
<tr>
<td>DISCOVER_ENUMERATORS Rowset</td>
<td>61</td>
</tr>
<tr>
<td>DISCOVER_KEYWORDS Rowset</td>
<td>63</td>
</tr>
<tr>
<td>DISCOVER_LITERALS Rowset</td>
<td>65</td>
</tr>
<tr>
<td>Flattened Rowset Examples</td>
<td>65</td>
</tr>
<tr>
<td>MDX Examples</td>
<td>65</td>
</tr>
<tr>
<td>XMLA Examples</td>
<td>70</td>
</tr>
</tbody>
</table>

### Chapter 4. Working with Java API

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Features</td>
<td>75</td>
</tr>
<tr>
<td>Embedded JAPI</td>
<td>75</td>
</tr>
<tr>
<td>Java API Support for Creating Partitions</td>
<td>77</td>
</tr>
</tbody>
</table>

### Chapter 5. Working with Essbase Web Services

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Features</td>
<td>79</td>
</tr>
<tr>
<td>Deploying Web Services</td>
<td>79</td>
</tr>
<tr>
<td>Datasource</td>
<td>80</td>
</tr>
<tr>
<td>Administration</td>
<td>81</td>
</tr>
<tr>
<td>Query</td>
<td>81</td>
</tr>
<tr>
<td>Data Queries</td>
<td>81</td>
</tr>
<tr>
<td>Metadata Queries</td>
<td>83</td>
</tr>
<tr>
<td>Writing Client Programs</td>
<td>84</td>
</tr>
</tbody>
</table>
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 that have purchased support 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.
Documentation Feedback

Send feedback on this documentation to: epmdoc_ww@oracle.com

Follow EPM Information Development on these social media sites:
LinkedIn - http://www.linkedin.com/groups?gid=3127051&goback=.gmp_3127051
Twitter - http://twitter.com/hyperionepminfo
Facebook - http://www.facebook.com/pages/Hyperion-EPM-Info/102682103112642
Google+ - https://plus.google.com/106915048672979407731/#106915048672979407731/posts
YouTube - http://www.youtube.com/user/OracleEPMWebcasts
Provider Services Overview

In This Chapter

Provider Services Introduction ............................................................................ 11
Java API..................................................................................................... 12
Smart View.................................................................................................. 14
XML for Analysis............................................................................................ 14
Essbase Web Services..................................................................................... 14
EPM System ................................................................................................ 15

Provider Services Introduction

Oracle Hyperion Provider Services is a middle-tier data-source provider to Oracle Essbase for Java API, Oracle 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.
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
You can embed Java API in the middle tier of an application as shown in Figure 3:
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:
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

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
**Note:** The WSDL URLs for Essbase Web Services are disabled by default. Before using Web Services, enable WSDL using the Oracle Enterprise Manager user interface. See the Oracle Enterprise Manager documentation.

**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.
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 Server after logging into Oracle Essbase Administration Services. Provider Services communicates with the designated Essbase server and grants or denies administrator permissions based on your role in that Essbase instance.

Use Administration Services Console to administer Provider Services:
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 18 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 Essbase 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 user who generated the request
- **Essbase Server**—The 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, and then Properties.
4. In Hyperion Provider Services Properties, select Settings.
5. In Maximum number of rows, specify the number of rows to retrieve.
   The default value is 5000.
6. In Maximum number of columns, specify the number of columns to retrieve.
   The default value 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, and 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 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
   <?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 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 the following directory:
   
   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.

Provider Services Logging

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

Network Error in Essbase Logs

When JAPI client programs are not disconnected from Essbase, or when standalone applications shut down abnormally, the following network error may be written to the Essbase log files:

Network error [10054]: Failed to receive data

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 the following directory:
Double-click essbase.properties to open the file.

Set olap.server.netSocketTryInfinite=false.

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>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>olap.server.netConnectRetry</td>
<td>The number of attempts a client makes to connect to an Essbase server.</td>
</tr>
<tr>
<td>olap.server.netDelay</td>
<td>The time that the thread waits before attempting another connect against Essbase.</td>
</tr>
<tr>
<td>olap.server.netRetryCount</td>
<td>The number of times an API can retry a unsuccessful network operation.</td>
</tr>
<tr>
<td>olap.server.netLoopIPAddresses</td>
<td>For connections to hosts with both IPv4 and IPv6 network interfaces, enables performance benefit.</td>
</tr>
<tr>
<td>olap.server.netSocketTimeOut</td>
<td>The time that a network operation can be blocked before it times out.</td>
</tr>
<tr>
<td>olap.server.netSocketTryInfinite</td>
<td>Indicates that the client will keep trying infinitely on a network operation.</td>
</tr>
<tr>
<td>smartview.smartslice.management.grantAnyUser</td>
<td>Enables any user with access to a cube to manage a smartslice.</td>
</tr>
<tr>
<td>olap.server.socketIdleTime</td>
<td>The time that the socket remains open while waiting for next API call from the user session.</td>
</tr>
<tr>
<td>olap.system.socketOptimization</td>
<td>Indicates that the socket optimization option is active.</td>
</tr>
</tbody>
</table>

### Configuring essbase.properties

To edit essbase.properties:

1. Navigate to the following directory:
   
   $EPM_ORACLE_HOME\common\EssbaseJavaAPI\11.1.2.0\bin

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**

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

Causes of connection failure include network congestion, server inaccessibility, and network interruptions.

**Syntax**

```plaintext
clap.server.netConnectRetry=n
```

**Parameters**

- `n`—An integer value. The default value is 3.

**Example**

```plaintext
clap.server.netConnectRetry=20
```

**olap.server.netDelay**

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

**Syntax**

```plaintext
clap.server.netDelay=n
```

**Parameters**

- `n`—An integer value of 100 or greater, expressed in milliseconds. The default value is 200.

**Example**

```plaintext
clap.server.netDelay=300
```

**olap.server.netRetryCount**

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**

```plaintext
clap.server.netRetryCount=n
```

**Parameters**

- `n`—An integer value. The default value is 600 retries.
Example
olap.server.netRetryCount=400

**olap.server.netLoopIPAddresses**
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
- `boolean`—True or false. The default value is true.

Example
olap.server.netLoopIPAddresses=true

**olap.server.netSocketTimeOut**
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`—An integer value of 0 or greater, expressed in milliseconds. The default value is 200.

Example
olap.server.netSocketTimeOut=120000

**olap.server.netSocketTryInfinite**
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
- `boolean`—True or false. The default value is true.
Example
olap.server.netSocketTryInfinite=true

**smartview.smartslice.management.grantAnyUser**

Enables any user with access to a cube to manage a smartslice.

**Syntax**
smartview.smartslice.management.grantAnyUser=boolean

**Parameters**

*boolean*—True or false. The default value is true.

**Example**
smartview.smartslice.management.grantAnyUser=true

**olap.server.socketIdleTime**

The time that the socket remains open while waiting for the next API call from the user session.

**Syntax**
olap.server.socketIdleTime=n

**Parameters**

*n*—An integer value of 0 or greater, expressed in milliseconds. The default value is 300.

**Example**
olap.server.socketIdleTime=300

**olap.system.socketOptimization**

Indicates that the socket optimization option is active.

**Syntax**
olap.system.socketOptimization=boolean

**Parameters**

*boolean*—True or false. The default value is true.

**Example**
olap.system.socketOptimization=true
Clustering

For information about clustering Provider Services and about using Provider Services to cluster Essbase databases, see the Oracle Enterprise Performance Management System Deployment Options Guide.
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.

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**—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.

![XMLA Architecture](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"
Discover (RequestType As EnumString, Restrictions As Restrictions, Properties As Properties, 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 35. Support is required for the following explicitly named RequestType enumeration values.

<table>
<thead>
<tr>
<th>Enumeration value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCOVER_DATASOURCES</td>
<td>Returns a list of XMLA data sources available on the server or Web Service.</td>
</tr>
<tr>
<td>DISCOVER_PROPERTIES</td>
<td>Returns a list of information and values about the requested properties that are supported by the specified data source (provider).</td>
</tr>
<tr>
<td>DISCOVER_SCHEMA_ROWSETS</td>
<td>Returns the names, values, and other information of all supported RequestType enumeration values (including those listed here), and any additional provider-specific enumeration values.</td>
</tr>
<tr>
<td>DISCOVER_ENUMERATORS</td>
<td>Returns a list of names, data types, and enumeration values of enumerators supported by the provider of a specific data source.</td>
</tr>
<tr>
<td>DISCOVER_KEYWORDS</td>
<td>Returns a rowset containing a list of keywords reserved by the provider.</td>
</tr>
<tr>
<td>DISCOVER_LITERALS</td>
<td>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 &quot;DISCOVER_SCHEMA_ROWSETS Rowset.&quot;</td>
</tr>
</tbody>
</table>

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 35. 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 each RequestType value, see “XMLA Rowsets” on page 35.

**Example**

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

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<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>
    <m:Discover xmlns:m="urn:schemas-microsoft-com:xml-analysis"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <m:RequestType>
        MDSCHEMA_CUBES
      </m:RequestType>
      <m:Restrictions>
        <m:RestrictionList>
          <m:CATALOG_NAME>
            Demo
          </m:CATALOG_NAME>
        </m:RestrictionList>
      </m:Restrictions>
      <m:Properties>
        <m:DataSourceInfo>
          Provider=Essbase;Data Source=localhost
        </m:DataSourceInfo>
        <m:Format>
          Tabular
        </m:Format>
      </m:Properties>
    </m:Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The provider returns the following result to the client:

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
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">
        Demo
      </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:

```xml
<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>
            34
          </DataSourceInfo>
        </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/"
  xmlns:m="urn:schemas-microsoft-com:xml-analysis">
  <m:ExecuteResponse
    xmlns:m="urn:schemas-microsoft-com:xml-analysis">
    <m:return
      xmlns:soap="http://schemas.xmlsoap.org/soap/encoding/">
      <root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset">
        ...<!-The schema for the data goes here. -- >
        <xsd:schema
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:xars="urn:schemas-microsoft-com:xars">
          ...<!-The data in MDDataSet format goes here. -- >
        </xsd:schema>
        ...<!-The schema for the data goes here. -- >
      </root>
      </m:return>
    </m:ExecuteResponse>
  </m: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 Essbase.

**GUID: DBSCHEMA_CATALOGS**

the section called “Flattened Rowset Examples” describes the rowset structure.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>CATALOGS Rowset Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Always null</td>
</tr>
</tbody>
</table>

Request Example

```xml
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/>
      </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
<?xml version="1.0"?>
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">
        <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:complexType name="row">
            <xsd:sequence maxOccurs="unbounded">
              <xsd:element name="CATALOG_NAME" type="xsd:string" sql:field="CATALOG_NAME"/>
            </xsd:sequence>
          </xsd:complexType>
        </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>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>CUBE_TYPE</td>
<td>&quot;CUBE&quot;</td>
</tr>
<tr>
<td>LAST_SCHEMA_UPDATE</td>
<td>Time stamp of last outline update</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Database description</td>
</tr>
</tbody>
</table>

Request Example

```xml
<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>
```

37
Response Example

<?xml version="1.0"?>
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"
elementFormDefault="qualified">
<xsd:element name="root">
<xsd:complexType>
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<xsd:element name="row" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

38
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>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>DIMENSION_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>DIMENSION_UNIQUE_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>DIMENSION_CAPTION</td>
<td>Dimension name</td>
</tr>
<tr>
<td>DIMENSION_ORDINAL</td>
<td>Dimension number. First dimension is 1, second is 2, and so on</td>
</tr>
<tr>
<td>DIMENSION_TYPE</td>
<td>If Essbase dimension type is:</td>
</tr>
<tr>
<td></td>
<td>● TIME: MD_DIMTYPE_TIME</td>
</tr>
<tr>
<td></td>
<td>● ACCOUNTS: MD_DIMTYPE_MEASURE</td>
</tr>
<tr>
<td></td>
<td>● ALL OTHER: MD_DIMTYPE_OTHER</td>
</tr>
<tr>
<td>DIMENSION_CARDINALITY</td>
<td>Number of members in the dimension</td>
</tr>
<tr>
<td>DEFAULT_HIERARCHY</td>
<td>Dimension name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Comment added for the dimension</td>
</tr>
<tr>
<td>DIMENSION_UNIQUE_SETTINGS</td>
<td>2</td>
</tr>
<tr>
<td>DIMENSION_IS_VISIBLE</td>
<td>True always</td>
</tr>
</tbody>
</table>

Request Example

```xml
<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>
          39
```

```
<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: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="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="row">
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<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:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</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 5 describes the rowset structure.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION_NAME</td>
<td>Name of the function</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Description of the function</td>
</tr>
<tr>
<td>PARAM_LIST</td>
<td>A comma delimited list of parameters</td>
</tr>
<tr>
<td>RETURN_TYPE</td>
<td>Always 12</td>
</tr>
<tr>
<td>ORIGIN</td>
<td>1 (always:MDX functions)</td>
</tr>
<tr>
<td>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>INTERFACE_NAME</td>
<td>One of the following: Member, Set, Tuple, Numeric, Dimension, Level, Boolean</td>
</tr>
<tr>
<td>OBJECT</td>
<td>One of the following values: Set, Member, Tuple, Level, Hierarchy, Dimension</td>
</tr>
<tr>
<td>HELP_CONTEXT</td>
<td>Help context ID for the function</td>
</tr>
<tr>
<td>CAPTION</td>
<td>Display caption of the function</td>
</tr>
</tbody>
</table>

**Request Example**

```xml
<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
<?xml version="1.0"?>
<SOAP-ENV:Envelope 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:schema="http://www.w3.org/2001/XMLSchema-instance"
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>
        </root>
    </m:return>
</SOAP-ENV:Envelope>
```
MDSCHEMA_HIERARCHIES Rowset

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

GUID: MDSCHEMA_HIERARCHIES

Table 6 describes the rowset structure.
Table 6  MDSCHEMA_HIERARCHIES Rowset Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>DIMENSION_UNIQUE_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>HIERARCHY_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>HIERARCHY_UNIQUE_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>HIERARCHY_CAPTION</td>
<td>Dimension name</td>
</tr>
<tr>
<td>DIMENSION_TYPE</td>
<td>If Essbase dimension type is:</td>
</tr>
<tr>
<td></td>
<td>• TIME: MD_DIMTYPE_TIME</td>
</tr>
<tr>
<td></td>
<td>• ACCOUNTS: MD_DIMTYPE_MEASURE</td>
</tr>
<tr>
<td></td>
<td>• ALL OTHER: MD_DIMTYPE_OTHER</td>
</tr>
<tr>
<td>HIERARCHY_CARDINALITY</td>
<td>Number of members in the dimension</td>
</tr>
<tr>
<td>DEFAULT_MEMBER</td>
<td>Dimension name</td>
</tr>
<tr>
<td>ALL_MEMBER</td>
<td>Dimension name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Dimension comment</td>
</tr>
<tr>
<td>STRUCTURE</td>
<td>MD_STRUCTURE_UNBALANCED(2)</td>
</tr>
<tr>
<td>HIERARCHY_UNIQUE_SETTINGS</td>
<td>2</td>
</tr>
<tr>
<td>HIERARCHY_IS_VISIBLE</td>
<td>True</td>
</tr>
</tbody>
</table>

Request Example

```xml
<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>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Response Example (truncated)

```xml
<?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">
            xmlns:sql="urn:schemas-microsoft-com:xml-sql">
            <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_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"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:schema>
        </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 7 describes the rowset structure.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>MEASURE_NAME</td>
<td>Member names in the Accounts dimension</td>
</tr>
<tr>
<td>MEASURE_UNIQUE_NAME</td>
<td>Above member name</td>
</tr>
<tr>
<td>MEASURE_CAPTION</td>
<td>Above member name</td>
</tr>
<tr>
<td>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>MEASURE_AGGREGATOR</td>
<td>Essbase ADDITION: 1</td>
</tr>
<tr>
<td></td>
<td>Essbase SUBTRACTION: 17</td>
</tr>
<tr>
<td></td>
<td>Essbase MULTIPLICATION: 18</td>
</tr>
<tr>
<td></td>
<td>Essbase DIVISION: 19</td>
</tr>
<tr>
<td></td>
<td>Essbase PERCENT: 20</td>
</tr>
<tr>
<td></td>
<td>Essbase NOOP: 21</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Member comment</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>5</td>
</tr>
<tr>
<td>EXPRESSION</td>
<td>Member formula</td>
</tr>
<tr>
<td>MEASURE_IS_VISIBLE</td>
<td>True</td>
</tr>
</tbody>
</table>

Request Example

```xml
<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
<?xml version="1.0"?>
 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">
```
<table>
<thead>
<tr>
<th>CATALOG_NAME</th>
<th>CUBE_NAME</th>
<th>MEASURE_NAME</th>
<th>MEASURE_UNIQUE_NAME</th>
<th>MEASURE_CAPTION</th>
<th>MEASURE_AGGREGATOR</th>
<th>DATA_TYPE</th>
<th>NUMERIC_PRECISION</th>
<th>NUMERIC_SCALE</th>
<th>DESCRIPTION</th>
<th>MEASURE_IS_VISIBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>Sample.Basic</td>
<td>Measures</td>
<td>[Measures]</td>
<td>Measures</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>
MDSCHEMA_MEMBERS Rowset

The MEMBERS rowset contains information about the available members.

GUID: MDSCHEMA_MEMBERS

Table 8 describes the rowset structure.

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

```xml
<xsd:element name="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>
</xsd:element>
```
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>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>HIERARCHY_UNIQUE_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>LEVEL_UNIQUE_NAME</td>
<td>Dimension name</td>
</tr>
<tr>
<td>PROPERTY_TYPE</td>
<td>1 (MDPROP_MEMBER)</td>
</tr>
<tr>
<td>PROPERTY_NAME</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>● For attribute dimension, the name of the dimension is the name of the property</td>
</tr>
<tr>
<td></td>
<td>● For UDA, the UDA name</td>
</tr>
<tr>
<td></td>
<td>● For aliases, the alias name</td>
</tr>
<tr>
<td>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROPERTY_CAPTION</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• For attribute dimensions, the attribute dimension name</td>
</tr>
<tr>
<td></td>
<td>• For UDA, the UDA name</td>
</tr>
<tr>
<td></td>
<td>• For aliases, the alias name</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>1 (double) – attribute dimension</td>
</tr>
<tr>
<td></td>
<td>2 (boolean) – attribute dimension</td>
</tr>
<tr>
<td></td>
<td>3 (string) – attribute dimension, UDA or alias</td>
</tr>
<tr>
<td></td>
<td>4 (integer) – attribute dimension</td>
</tr>
<tr>
<td>CHARACTER_MAXIMUM_LENGTH</td>
<td>80 (for UDA or an attribute dimension)</td>
</tr>
<tr>
<td></td>
<td>30 (for alias)</td>
</tr>
<tr>
<td>CHARACTER_OCTET_LENGTH</td>
<td>320 (for UDA or an attribute dimension)</td>
</tr>
<tr>
<td></td>
<td>120 (for alias)</td>
</tr>
<tr>
<td>PROPERTY_CONTENT_TYPE</td>
<td>0 (MD_PROPTYPE_REGULAR)</td>
</tr>
<tr>
<td>SQL_COLUMN_NAME</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• For attribute dimensions, the attribute dimension name</td>
</tr>
<tr>
<td></td>
<td>• For UDA, the UDA name</td>
</tr>
<tr>
<td></td>
<td>• For aliases, the alias name</td>
</tr>
<tr>
<td>PROPERTY_ORIGIN</td>
<td>1 (MD_USER_DEFINED)</td>
</tr>
<tr>
<td>PROPERTY_ATTRIBUTE_HIERARCHY_NAME</td>
<td>For attribute dimensions, the attribute dimension name</td>
</tr>
<tr>
<td>PROPERTY_CARDINALITY</td>
<td>ONE (for UDA and aliases)</td>
</tr>
<tr>
<td></td>
<td>MANY (for attribute dimension)</td>
</tr>
<tr>
<td>PROPERTY_IS_VISIBLE</td>
<td>True</td>
</tr>
</tbody>
</table>

**Request Example**

```xml
<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>
    </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Response Example (truncated)

<?xml version="1.0"?>
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  <SOAP-ENV:Body>
        <xsd:complexType name="row">
          <xsd:sequence maxOccurs="unbounded">
            <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"/>
          </xsd:sequence>
        </xsd:complexType>
      </root>
    </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
<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 -->

</m:return>
</m:DiscoverResponse>
**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 10* describes the rowset structure.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>SET_NAME</td>
<td>Name of the set</td>
</tr>
<tr>
<td>SCOPE</td>
<td>Session</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_NAME</td>
<td>Application name</td>
</tr>
<tr>
<td>CUBE_NAME</td>
<td>Database name</td>
</tr>
<tr>
<td>DIMENSION_UNIQUE_NAME</td>
<td>Name of the dimension to which the level belongs</td>
</tr>
<tr>
<td>HIERARCHY_UNIQUE_NAME</td>
<td>Name of the dimension to which the level belongs</td>
</tr>
<tr>
<td>LEVEL_NAME</td>
<td>Unique level name</td>
</tr>
<tr>
<td>LEVEL_UNIQUE_NAME</td>
<td>Unique level name</td>
</tr>
<tr>
<td>LEVEL_CAPTION</td>
<td>Level name</td>
</tr>
<tr>
<td>LEVEL_NUMBER</td>
<td>Level number</td>
</tr>
<tr>
<td>LEVEL_CARDINALITY</td>
<td>Number of members in the level</td>
</tr>
<tr>
<td>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LEVEL_TYPE</td>
<td>MDLEVEL_TYPE_ALL (for dimension level)</td>
</tr>
<tr>
<td></td>
<td>MDLEVEL_TYPE_TIME (for dimension type TIME)</td>
</tr>
<tr>
<td></td>
<td>MDLEVEL_TYPE_REGULAR (for all others)</td>
</tr>
<tr>
<td>LEVEL_UNIQUE_SETTINGS</td>
<td>2 (MDDIMENSIONS_MEMBER_NAME_UNIQUE)</td>
</tr>
<tr>
<td>LEVEL_IS_VISIBLE</td>
<td>True</td>
</tr>
<tr>
<td>ESSBASE_GEN_UNIQUE_NAME</td>
<td>Generation unique name</td>
</tr>
<tr>
<td>ESSBASE_GEN.Caption</td>
<td>Generation caption</td>
</tr>
</tbody>
</table>

**Request Example**

```xml
<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/"
   xsi:type="xsd:string">
   <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
<?xml version="1.0"?>
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string">
     <xsd:rowset xmlns="urn:schemas-microsoft-com:xml-analysis:rowset">
    </xsd:rowset>
   </m:return>
  </m:DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
<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>
DISCOVER_SCHEMA_ROWSETS Rowset

GUID: DISCOVER_SCHEMA_ROWSETS

Table 12 describes the rowset structure.

Table 12  DISCOVER_SCHEMA Rowset Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchemaName</td>
<td>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.</td>
</tr>
<tr>
<td>Restrictions</td>
<td>List of restrictions allowed</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the schema</td>
</tr>
</tbody>
</table>

DISCOVER_DATASOURCES Rowset

GUID: DISCOVER_DATASOURCES

Table 13 describes the rowset structure.

Table 13  DISCOVER_DATASOURCES Rowset Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSourceName</td>
<td>Name of the data source</td>
</tr>
<tr>
<td>DataSourceDescription</td>
<td>Description of the data source</td>
</tr>
<tr>
<td>DataSourceInfo</td>
<td>Provider=Essbase Data Source= name of the Essbase Server</td>
</tr>
<tr>
<td>ProviderName</td>
<td>XMLA for Essbase</td>
</tr>
<tr>
<td>ProviderType</td>
<td>MDP</td>
</tr>
<tr>
<td>AuthenticationMode</td>
<td>Authenticated</td>
</tr>
</tbody>
</table>
DISCOVER_PROPERTIES Rowset

GUID: DISCOVER_PROPERTIES

Table 14 describes the rowset structure.

Table 14  DISCOVER_PROPERTIES Rowset Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyName</td>
<td>Name of the property</td>
</tr>
<tr>
<td>PropertyDescription</td>
<td>Description of the property</td>
</tr>
<tr>
<td>PropertyType</td>
<td>XML data type of the property</td>
</tr>
<tr>
<td>PropertyAccessType</td>
<td>Access for the property. The value can be Read, Write, or ReadWrite</td>
</tr>
<tr>
<td>IsRequired</td>
<td>True if a property is required, false if it is not required</td>
</tr>
<tr>
<td>Value</td>
<td>Current value of the property</td>
</tr>
</tbody>
</table>

Request Example

```xml
<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
<?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"
DISCOVER_ENUMERATORS Rowset

GUID: DISCOVER_ENUMERATORS

Table 15 describes the rowset structure.
Table 15  DISCOVER_ENUMERATORS Rowset Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumName</td>
<td>Name of the enumerator that contains a set of values</td>
</tr>
<tr>
<td>EnumDescription</td>
<td>Description of the enumerator</td>
</tr>
<tr>
<td>ElementName</td>
<td>Name of one of the value elements in the enumerator set</td>
</tr>
<tr>
<td></td>
<td>Example: TDP</td>
</tr>
<tr>
<td>ElementDescription</td>
<td>Description of the element</td>
</tr>
<tr>
<td>EnumType</td>
<td>Data type of the Enum values</td>
</tr>
<tr>
<td>ElementValue</td>
<td>Value of the element</td>
</tr>
<tr>
<td></td>
<td>Example: 01</td>
</tr>
</tbody>
</table>

Request Example

```xml
<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
<?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/">
 <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"
Table 16 describes the rowset structure.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword</td>
<td>A list of keywords reserved by a provider</td>
</tr>
<tr>
<td></td>
<td>Example: AND</td>
</tr>
</tbody>
</table>
Request Example

```xml
<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
<?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">
        <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"
              elementFormDefault="qualified">
            <xsd:element name="root">
              <xsd:complexType>
                <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="row" type="xsd:string" sql:field="Keyword"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:schema>
        </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

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Essbase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiteralName</td>
<td>Name of the literal described in the row</td>
</tr>
<tr>
<td></td>
<td>Example: DLBLITERAL_LIKE_PERCENT</td>
</tr>
<tr>
<td>LiteralValue</td>
<td>Contains the literal value</td>
</tr>
<tr>
<td></td>
<td>Example, if LiteralName is DLBLITERAL_LIKE_PERCENT and the percent character (%)</td>
</tr>
<tr>
<td></td>
<td>is used to match zero or more characters in a LIKE clause, this column’s value</td>
</tr>
<tr>
<td></td>
<td>would be “%.”</td>
</tr>
<tr>
<td>LiteralInvalidChars</td>
<td>Characters, in the literal, that are not valid</td>
</tr>
<tr>
<td></td>
<td>Example: If table names can contain anything other than a numeric character, this</td>
</tr>
<tr>
<td></td>
<td>string would be “0123456789”</td>
</tr>
<tr>
<td>LiteralInvalidStartingChars</td>
<td>Characters that are not valid as the first character of the literal. If the</td>
</tr>
<tr>
<td></td>
<td>literal can start with any valid character, this is null.</td>
</tr>
<tr>
<td>LiteralMaxLength</td>
<td>Maximum number of characters in the literal. If there is no maximum or the maximum</td>
</tr>
<tr>
<td></td>
<td>is unknown, the value is -1.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>[Product].[Family].[MEMBER_CAPTION]</th>
<th>[Profit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>30468</td>
</tr>
<tr>
<td>200</td>
<td>27954</td>
</tr>
<tr>
<td>300</td>
<td>25799</td>
</tr>
<tr>
<td>400</td>
<td>21301</td>
</tr>
<tr>
<td>Diet</td>
<td>28826</td>
</tr>
</tbody>
</table>

**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):

<table>
<thead>
<tr>
<th>[Product].[Family].[MEMBER_CAPTION]</th>
<th>[Product].[SKU].[MEMBER_CAPTION]</th>
<th>[Profit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100-10</td>
<td>22777</td>
</tr>
<tr>
<td>100</td>
<td>100-20</td>
<td>5708</td>
</tr>
<tr>
<td>100</td>
<td>100-30</td>
<td>1983</td>
</tr>
<tr>
<td>200</td>
<td>200-10</td>
<td>7201</td>
</tr>
<tr>
<td>200</td>
<td>200-20</td>
<td>12025</td>
</tr>
<tr>
<td>200</td>
<td>200-30</td>
<td>4636</td>
</tr>
<tr>
<td>200</td>
<td>200-40</td>
<td>4092</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**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):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[100]</td>
<td>1</td>
<td>[100-10]</td>
<td>2</td>
<td>22777</td>
</tr>
<tr>
<td>[100]</td>
<td>1</td>
<td>[100-20]</td>
<td>2</td>
<td>5708</td>
</tr>
<tr>
<td>[100]</td>
<td>1</td>
<td>[100-30]</td>
<td>2</td>
<td>1983</td>
</tr>
<tr>
<td>[200]</td>
<td>1</td>
<td>[200-10]</td>
<td>2</td>
<td>7201</td>
</tr>
<tr>
<td>[200]</td>
<td>1</td>
<td>[200-20]</td>
<td>2</td>
<td>12025</td>
</tr>
<tr>
<td>[200]</td>
<td>1</td>
<td>[200-40]</td>
<td>2</td>
<td>4092</td>
</tr>
<tr>
<td>[300]</td>
<td>1</td>
<td>[300-10]</td>
<td>2</td>
<td>12195</td>
</tr>
<tr>
<td>[300]</td>
<td>1</td>
<td>[300-20]</td>
<td>2</td>
<td>2511</td>
</tr>
<tr>
<td>[300]</td>
<td>1</td>
<td>[300-30]</td>
<td>2</td>
<td>2511</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**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):
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):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>New York</td>
<td>Colas</td>
<td>Cola</td>
<td>3498</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Root Beer</td>
<td>Old Fashioned</td>
<td>-2594</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Root Beer</td>
<td>Birch Beer</td>
<td>3086</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Cream Soda</td>
<td>Dark Cream</td>
<td>2496</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Cream Drinks</td>
<td>Vanilla Cream</td>
<td>-1952</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Fruit Soda</td>
<td>Grape</td>
<td>1329</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Fruit Soda</td>
<td>Orange</td>
<td>1388</td>
</tr>
<tr>
<td>East</td>
<td>New York</td>
<td>Fruit Soda</td>
<td>Strawberry</td>
<td>951</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>1</td>
<td>Colas</td>
<td>1</td>
<td>Cola</td>
<td>2</td>
<td>11129</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Colas</td>
<td>1</td>
<td>Diet Cola</td>
<td>2</td>
<td>1114</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Colas</td>
<td>1</td>
<td>Caffeine Free Cola</td>
<td>2</td>
<td>413</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Root Beer</td>
<td>1</td>
<td>Old Fashioned</td>
<td>2</td>
<td>-2540</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Root Beer</td>
<td>1</td>
<td>Diet Root Beer</td>
<td>2</td>
<td>982</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Root Beer</td>
<td>1</td>
<td>Birch Beer</td>
<td>2</td>
<td>4092</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Cream Soda</td>
<td>1</td>
<td>Dark Cream</td>
<td>2</td>
<td>3233</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>Cream Soda</td>
<td>1</td>
<td>Vanilla Cream</td>
<td>2</td>
<td>-918</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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):
<table>
<thead>
<tr>
<th>[Market]. Levels(1). [MEMBERCaption]</th>
<th>[Product]. [Family]. [MEMBERCaption]</th>
<th>[Year]. Levels(1). [MEMBERCaption]</th>
<th>[Profit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>Colas</td>
<td>Qtr1</td>
<td>2747</td>
</tr>
<tr>
<td>East</td>
<td>Colas</td>
<td>Qtr2</td>
<td>3352</td>
</tr>
<tr>
<td>East</td>
<td>Colas</td>
<td>Qtr3</td>
<td>3740</td>
</tr>
<tr>
<td>East</td>
<td>Colas</td>
<td>Qtr4</td>
<td>2817</td>
</tr>
<tr>
<td>East</td>
<td>Root Beer</td>
<td>Qtr1</td>
<td>562</td>
</tr>
<tr>
<td>East</td>
<td>Root Beer</td>
<td>Qtr2</td>
<td>610</td>
</tr>
<tr>
<td>East</td>
<td>Root Beer</td>
<td>Qtr3</td>
<td>372</td>
</tr>
<tr>
<td>East</td>
<td>Root Beer</td>
<td>Qtr4</td>
<td>990</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**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.

```xml
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
<?xml version="1.0"?>
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-analysis: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:schema>
                  </root>
                </m:return>
              </m:ExecuteResponse>
            </SOAP-ENV:Body>
          </SOAP-ENV:Envelope>
```
<table>
<thead>
<tr>
<th>Quarter</th>
<th>Month</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qtr2</td>
<td>Apr</td>
<td>8644.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td>May</td>
<td>8929.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td>Jun</td>
<td>9534.00</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Jul</td>
<td>9878.00</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Aug</td>
<td>9545.00</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Sep</td>
<td>8489.00</td>
</tr>
<tr>
<td>Qtr4</td>
<td>Oct</td>
<td>8653.00</td>
</tr>
<tr>
<td>Qtr4</td>
<td>Nov</td>
<td>8367.00</td>
</tr>
<tr>
<td>Qtr4</td>
<td>Dec</td>
<td>8780.00</td>
</tr>
<tr>
<td>column1</td>
<td>calctest</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>column3</td>
<td>4.000000</td>
<td></td>
</tr>
</tbody>
</table>
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 Essbase 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 Essbase Servers. No mid-tier Provider Services Server is necessary.

Sample script files are provided to illustrate the use of a JAPI sample in embedded mode and in three-tier Provider Services mode:

- Embedded Java API deployment:
  - Windows: runsamplesEmbedded.cmd
  - UNIX: runsamples
- Three-tier Java API deployment:
  - Windows: runsamplesAPS.cmd
  - UNIX: runsamples
The sample script files are located in the following directory:

```
EPM_ORACLE_HOME/common/EssbaseJavaAPI/11.1.2.0/samples/japi
```

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, you must change the value of the String `providerUrl` in the `signOn` API:

- Embedded JAPI—The string `embedded`
- Three-tier Provider Services JAPI—The URL to the Provider Services instance

The API is the same for embedded JAPI and three-tier JAPI through Provider Services. The difference between the two is the `providerUrl` parameter.

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

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

**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**

```java
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 Essbase directly. (No separate provider application needs to be running.)
- **olapServerName**—The host name where Essbase Server is running.

**Returns:**

The connected Essbase Server instance.

**Java API Support for Creating Partitions**

You can create partitions using the JAPI client. For details on creating, editing, and using partitions, see “Managing Partitions” in the Administration Services online help.
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 Navigate to the following directory:
   
   `EPM_ORACLE_HOME/common/EssbaseJavaAPI/11.1.2.0/bin`

3 Open `essbase.properties`.

4 Set `essbase.webservices.disable.owsm` to `False`.

5 Save and close `essbase.properties`.

6 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:

```java
SOAPElement el = header.addHeaderElement(envelope.createName("Parameters", "", "http://context.webservices.epm.oracle/"));
SOAPElement el0 = el.addChildElement(envelope.createName("UserName", "", "http://context.webservices.epm.oracle/"));
e0.setValue(s_userName);
SOAPElement el1 = el.addChildElement(envelope.createName("Password", "", "http://context.webservices.epm.oracle/"));
e1.setValue(s_password);
SOAPElement el2 = el.addChildElement(envelope.createName("IsToken", "", "http://context.webservices.epm.oracle/"));
e2.setValue(Boolean.toString(b_isPwdToken));
SOAPElement el3 = el.addChildElement(envelope.createName("UserNameAs", "", "http://context.webservices.epm.oracle/"));
e3.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
Administration

You use Administration Web Services to start, monitor, or stop Essbase Server, 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 operation 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_WILDSSEARCH`
- `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.

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

    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();
    }
}
```
About 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 following 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 aggregate storage 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 cube.
- CustomMessageHandler.java demonstrates 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 Essbase 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 the Essbase 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.

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

Subtopics

- Configuring the Script Files
- Compiling and Running Samples

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

Sample script files are provided for compiling the sample programs.

- Embedded Java API deployment:
  - Windows: runsamplesEmbedded.cmd
  - UNIX: runsamples
- Three-tier Java API deployment:
  - Windows: runsamplesAPS.cmd
  - UNIX: runsamples

Sample script files are located in the following directory:

EPM_ORACLE_HOME/common/EssbaseJavaAPI/11.1.2.0/samples/japi
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 sample 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 three-tier and embedded Java API deployments.

To configure a script file to work with your computer environment:

1. Using a text editor, open the appropriate script file.
   
   For the names and locations of the sample script files, see “Compiling and Running the Sample Programs” on page 88.

2. Verify that the `ESS_ES_HOME` variable is set to the location of your Provider Services installation. For example:
   - Windows:
     
     ```bash
     set ESS_ES_HOME=C:\Oracle\Middleware\EPMSYSTEM11R1\common\EssbaseJavaAPI\11.1.2.0
     ```
   - UNIX:
     
     ```bash
     export ESS_ES_HOME=/oracle/Middleware/EPMSYSTEM11R1/common/EssbaseJavaAPI/11.1.2.0
     ```

3. 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 the `JAVA_HOME` variable with a full path to the Java installation. For example:
   - Windows:
     
     ```bash
     set JAVA_HOME=C:\Oracle\Middleware\jdk160_11
     ```
   - UNIX:
     
     ```bash
     export JAVA_HOME=/opt/oracle/Middleware/jdk160_11
     ```

4. 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 Hyperion 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.

5. Save the script file.
Compiling and Running Samples

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

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

1. Using a text editor, open the appropriate script file.
   
   For the names and locations of the sample script files, see “Compiling and Running the Sample Programs” on page 88.

2. 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 “About the Sample Programs” on page 85):  
   
   ```
   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.

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

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