Contents

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

Chapter 1. Provider Services Overview ..................................................... 9
  Provider Services Introduction ............................................................ 9
  Java API .................................................................................. 10
  Smart View ........................................................................... 12
  XML for Analysis ...................................................................... 12
  Essbase Web Services ................................................................. 12
  EPM System ............................................................................ 13
  Provider Services Documentation ................................................. 13

Chapter 2. Administering Provider Services ............................................... 15
  Administering Provider Services ................................................... 15
    Adding Provider Services ...................................................... 15
    Editing the Authenticating Essbase Server .................................. 16
    Removing Provider Services ................................................... 16
    Connecting to a Stand-alone Essbase Server ............................ 17
    Connecting to Provider Services ............................................... 17
    Monitoring Sessions ............................................................... 18
    Specifying Session Timeout ..................................................... 18
    Specifying Maximum Rows and Columns ................................. 19
    Automatically Deploying Client Upgrades ................................. 19
  Updating References to Rehosted Servers ...................................... 20
  Logging .................................................................................. 21
  Setting TCP/IP Socket Communication ....................................... 21
  Configuring Options in essbase.properties .................................. 21
    Configuring essbase.properties ............................................... 22
  Configurable Options in essbase.properties ................................. 22
  Clustering ............................................................................... 25

Chapter 3. Working with XMLA .............................................................. 27
  Key Features ............................................................................ 27
  Methods .................................................................................. 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover</td>
<td>28</td>
</tr>
<tr>
<td>Execute</td>
<td>31</td>
</tr>
<tr>
<td>XMLA Rowsets</td>
<td>33</td>
</tr>
<tr>
<td>CATALOGS Rowset</td>
<td>33</td>
</tr>
<tr>
<td>MDSCHEMA_CUBES Rowset</td>
<td>35</td>
</tr>
<tr>
<td>MDSCHEMA_DIMENSIONS Rowset</td>
<td>37</td>
</tr>
<tr>
<td>MDSCHEMA_FUNCTIONS Rowset</td>
<td>39</td>
</tr>
<tr>
<td>MDSCHEMA_HIERARCHIES Rowset</td>
<td>41</td>
</tr>
<tr>
<td>MDSCHEMA_MEASURES Rowset</td>
<td>44</td>
</tr>
<tr>
<td>MDSCHEMA_MEMBERS Rowset</td>
<td>47</td>
</tr>
<tr>
<td>MDSCHEMA_PROPERTIES Rowset</td>
<td>50</td>
</tr>
<tr>
<td>MDSCHEMA_SETS Rowset</td>
<td>53</td>
</tr>
<tr>
<td>MDSCHEMA_LEVELS Rowset</td>
<td>54</td>
</tr>
<tr>
<td>DISCOVER_SCHEMA_ROWSETS Rowset</td>
<td>57</td>
</tr>
<tr>
<td>DISCOVER_DATASOURCES Rowset</td>
<td>57</td>
</tr>
<tr>
<td>DISCOVER_PROPERTIES Rowset</td>
<td>57</td>
</tr>
<tr>
<td>DISCOVER_ENUMERATORS Rowset</td>
<td>59</td>
</tr>
<tr>
<td>DISCOVER_KEYWORDS Rowset</td>
<td>61</td>
</tr>
<tr>
<td>DISCOVER_LITERALS Rowset</td>
<td>63</td>
</tr>
<tr>
<td>Flattened Rowset Examples</td>
<td>63</td>
</tr>
<tr>
<td>MDX Examples</td>
<td>63</td>
</tr>
<tr>
<td>XMLA Examples</td>
<td>68</td>
</tr>
</tbody>
</table>

---

Chapter 4. Working with Java API                                      | 71   |
| Key Features                                                          | 71   |
| Embedded JAPI                                                         | 71   |

Chapter 5. Working with Essbase Web Services                           | 75   |
| Key Features                                                          | 75   |
| Deploying Web Services                                                | 75   |
| Datasource                                                            | 76   |
| Administration                                                        | 77   |
| Query                                                                 | 77   |
| Data Queries                                                          | 77   |
| Metadata Queries                                                      | 79   |
| Writing Client Programs                                               | 80   |
| Sample Client Programs                                                | 80   |

Chapter 6. Setting Up the Sample Programs                              | 81   |
| The Sample Programs                                                   | 81   |
Documentation Accessibility

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

Access to Oracle Support

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

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

Figure 1 illustrates the relationship of Provider Services to Essbase, and to its Java API, Smart View, and XMLA clients.
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 is: http://server_name:port/aps/SmartView.

XML for Analysis

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

Figur 4  XMLA System Architecture

Essbase Web Services

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

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

- Datasource
- Administration
- Data and Metadata Query
EPM System

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

Provider Services Documentation

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

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

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

Use Administration Services Console to administer Provider Services:

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

Adding Provider Services

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

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

The provider name is displayed under the Provider Services node.

**Editing the Authenticating Essbase Server**

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

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

To edit the authenticating Essbase Server:

1. From Enterprise View or a custom view, under the Hyperion Provider Services node, select a provider.
2. Right-click and select Edit Authenticating Essbase Server.
   
   A dialog box where you can edit and specify another authenticating Essbase server is displayed.
3. Click OK.

**Removing Provider Services**

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

To remove Provider Services:

1. From Enterprise View or a custom view, under the Hyperion Provider Services node, select a provider.
2. Right-click and select Remove.
3. In Remove Hyperion Provider Services, click Yes.
Connecting to a Stand-alone Essbase Server

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

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

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

➢ To connect to a stand-alone Essbase Server:

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

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

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

Connecting to Provider Services

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

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

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

**Monitoring Sessions**

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

To monitor Provider Services sessions:

1. From Enterprise View or a custom view, under the Hyperion Provider Services node, select a provider.
2. Right-click and select Sessions. The Provider Services Sessions window is displayed:
   - **Session**—Active session ID
   - **Session Type**—Type of request, from stand-alone server or cluster
   - **Mode**—Stand-alone server mode (server) or Analytic Cluster mode (cluster)
   - **User**—The use who generated the request
   - **Analytic Server**—Essbase Server to which the request was made
   - **Application**—Application name
   - **Database**—Database name
   - **Request Time**—Time of request
   - **Request**—Name of current running request, if any
3. To see one user's sessions, select Show sessions for user and select from user lists.
4. To see a session, select Show sessions for type and select JAVA, XMLA, or SMARTVIEW
5. Click Refresh to update the view.

**Specifying Session Timeout**

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

To specify the session timeout limit:

1. From Enterprise View or a custom view, under the Hyperion Provider Services node, select the Provider node.
2. Right-click and select Edit, then Properties.
3. In Hyperion Provider Services Properties, select Settings.
4  In **Idle Session timeout in minutes**, specify how long the session can be inactive before timing out (default is 60). If the session times out, Smart View users must reconnect to Provider Services.

5  Click **Apply**.

6  Click **Close**.

### Specifying Maximum Rows and Columns

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

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

➢ To specify maximum rows and columns:

1  Open Administration Services.

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

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

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

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

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

7  Click **Apply**.

8  Click **Close**.

### Automatically Deploying Client Upgrades

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

➢ To automatically deploy Smart View clients:

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

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

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

4  Select an option:

   - **Force Smart View client to upgrade**—Users must upgrade to continue using Smart View.
Warn Smart View client to upgrade—Informs users of available Smart View upgrade. Users can continue using Smart View clients without upgrading.

Apply Smart View client to upgrade—Enables the administrator to apply new versions of Oracle Hyperion Smart View for Office and inform users without requiring Provider Services restart.

5  Click Apply.

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

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

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

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

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

Updating References to Rehosted Servers

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

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

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

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

1  Navigate to EPM_ORACLE_INSTANCE/bin/upgrades.
2 From a command prompt, run the following script:

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

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

Execute this script once for each reference to be updated.

## Logging

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

### Setting TCP/IP Socket Communication

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

To specify network operation timeout:

1. **Navigate to** `EPM_ORACLE_INSTANCE\bin\essbase.properties`.
2. **Double-click** `essbase.properties` to open the file.
3. **Set** `olap.server.netSocketTryInfinite=false`.
4. **Set** `olap.server.netRetryCount= xxx`, where the total network operation timeout = `olap.server.netRetryCount x olap.server.netSocketTimeOut` in milliseconds.

### Configuring Options in `essbase.properties`

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>olap.server.netConnectRetry</code></td>
<td>The number of attempts a client makes to connect to an Essbase server</td>
</tr>
<tr>
<td><code>olap.server.netDelay</code></td>
<td>The time that the thread waits before attempting another connect against Essbase</td>
</tr>
<tr>
<td><code>olap.server.netRetryCount</code></td>
<td>The number of times an API can retry a unsuccessful network operation</td>
</tr>
<tr>
<td><code>olap.server.netLoopIPAddresses</code></td>
<td>For connections to hosts with both IPv4 and IPv6 network interfaces, enables performance benefit</td>
</tr>
<tr>
<td><code>olap.server.netSocketTimeOut</code></td>
<td>The time that a network operation can be blocked before it times out</td>
</tr>
</tbody>
</table>
### Configuring `essbase.properties`

To edit `essbase.properties`:

1. Navigate to `EPM_ORACLE_INSTANCE\bin\essbase.properties`.
2. Double-click `essbase.properties` to open the file.
3. Enter each setting on a separate line.
   - Semicolon terminators are not required.
4. Save and close `essbase.properties`.
5. Restart the Provider Services server.

### Configurable Options in `essbase.properties`

#### `olap.server.netConnectRetry`

**Description**

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

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

**Syntax**

```
olap.server.netConnectRetry=n
```

**Parameters**

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

**Example**

```
olap.server.netConnectRetry=20
```
**olap.server.netDelay**

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

**Syntax**

```
olap.server.netDelay=n
```

**Parameters**

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

**Example**

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

**olap.server.netRetryCount**

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

**Syntax**

```
olap.server.netRetryCount=n
```

**Parameters**

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

**Example**

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

**olap.server.netLoopIPAddresses**

**Description**
If Provider Services is needed to connect to hosts that have both IPv4 and IPv6 network interfaces enabled but only one is being used, this property can be set to false to get a performance benefit.
When set to false, Provider Services will not loop through all the interfaces while connecting and instead use only the default one returned by host network environment.

**Syntax**

```plaintext
olap.server.netLoopIPAddresses=boolean
```

**Parameters**

True or false - (default is true).

**Sample**

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

**olap.server.netSocketTimeOut**

**Description**

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

**Syntax**

```plaintext
olap.server.netSocketTimeOut=n
```

**Parameters**

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

**Example**

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

**olap.server.netSocketTryInfinite**

**Description**

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

**Syntax**

```plaintext
olap.server.netSocketTryInfinite=boolean
```
Parameters

True or false - (default is true).

Sample

olap.server.netSocketTryInfinite=true

Clustering

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

For information about clustering Provider Services, see “Clustering Web Applications” in Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide.
Key Features

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

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

Key XMLA features:

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

Note: XMLA is available for use with Essbase only.

Methods

The following methods provide a standard way for XML applications to access basic information from the server. Because these methods are invoked using SOAP, they accept input and deliver output in XML. By default, these methods are stateless, so the server context ends at the completion of any command.

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

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

Figure 5  XMLA Architecture

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

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

The XML namespace for these methods is “urn:schemas-microsoft-com:xml-analysis”. Connection information is supplied in each method call with the connection properties.

Discover

The Discover method retrieves information, such as the list of data sources on a server or details about a data source. The data retrieved with the Discover method depends on the values of the parameters passed to it.

Namespace
SOAP Action
"urn:schemas-microsoft-com:xml-analysis:Discover"

Syntax
Discover {
  [in] RequestType As EnumString,
  [in] Restrictions As Restrictions,
  [in] Properties As Properties,
  [out] Result As Rowset
}

Parameters

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

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

<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. (For an example of how these may be published, see “XMLA Implementation Walkthrough” in the XML for Analysis Specification, available on the Hyperion Developer Network.)</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>
</tbody>
</table>
| DISCOVER_LITERALS      | Returns information about literals supported by the data source provider. Schema Rowset Constant Given, a constant that corresponds to one of the schema rowset names defined by OLE DB, such as MDSHEMA_CUBES, returns the OLE DB schema rowset in XML format. Note that providers also may extend OLEDB by providing additional provider-specific schema rowsets. The schema rowsets that tabular data providers (TDP) and multidimensional data providers (MDP) are required to support are listed in the section "DISCOVER_SCHEMA_ROWSETS Rowset."

Restrictions [in]
This parameter, of the Restrictions data type, enables the user to restrict the data returned in
Result. Result columns are defined by the rowset specified in the RequestType parameter. Some
columns of Result can filter the rows returned. For these columns and those that can be restricted, see the rowset tables in “XMLA Rowsets” on page 33. To obtain the restriction information for provider-specific schema rowsets, use the DISCOVER_SCHEMA_ROWSETS request type. This parameter can be empty, but it must be included.

Properties [in]

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

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

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

Result [out]

This required parameter contains the result set returned by the provider as a Rowset object. The columns and content of the result set are specified by the values in the RequestType and Restrictions parameters. The column layout of the returned result set also is determined by the value specified in RequestType. For information about the rowset layouts that correspond to each RequestType value, see “XMLA Rowsets” on page 33.

Example

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

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
  <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <RequestType>MDSCHEMA_CUBES</RequestType>
  <Restrictions>
   <RestrictionList>
    <CATALOG_NAME>Demo</CATALOG_NAME>
   </RestrictionList>
  </Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>
     Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Format>Tabular</Format>
   </PropertyList>
  </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The provider returns the following result to the client:

---

30 Working with XMLA
Execute

The Execute method sends action requests, including those involving data transfer, such as retrieving or updating data on the server, to the server.
Namespace
urn:schemas-microsoft-com:xml-analysis

SOAP Action
"urn:schemas-microsoft-com:xml-analysis:Execute"

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

Parameters
Command [in]
This required parameter is of Command data type and consists of an MDX statement to be executed.

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

The available properties and their values can be obtained by using the DISCOVER_PROPERTIES request type with the Discover method.

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

Result [out]
This parameter contains the Resultset result returned by the provider. The Command parameter and values in the Properties parameter define the shape of the result set. If no shape-defining properties are passed, the XMLA provider may use a default shape. The two result set formats defined by this specification are Tabular and Multidimensional, as specified by the client through the Format property. OLAP data lends itself to the Multidimensional format (although the Tabular format also can be used). A provider may support additional rowset types, and clients aware of the specialized types can request them.

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

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<SOAP-ENV:Body>
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <Command>

32 Working with XMLA
<Statement>
SELECT CrossJoin([Measures].CHILDREN, [Market].CHILDREN)
on columns, [Product].Members on rows
from Sample.Basic
</Statement>
</Command>
Properties
</PropertyList>
<DataSourceInfo>
Provider=Essbase;Data Source=localhost
</DataSourceInfo>
<Catalog>Sample</Catalog>
<Format>Multidimensional</Format>
<AxisFormat>TupleFormat</AxisFormat>
<Content>SchemaData</Content>
</PropertyList>
</Properties>
</Execute>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
The abbreviated response for the preceding method call:

```xml
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">  
<SOAP-ENV:Body>
  <m:ExecuteResponse
xmlns:m="urn:schemas-microsoft-com:xml-analysis">
  <m:return
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <root xmlns="urn:schemas-microsoft-com:xmldatamodel">
      ...<!-The schema for the data goes here. -- >
    </root>
  </m:return>
  </m:ExecuteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### XMLA Rowsets

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

#### CATALOGS Rowset

The CATALOGS rowset identifies the physical attributes associated with catalogs accessible from Analytic Services.
the section called “Flattened Rowset Examples” describes the rowset structure.

Table 2 CATALOGS 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>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/>
      <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:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
            targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:sql="urn:schemas-microsoft-com:xml-sql"
            elementFormDefault="qualified">
            <xsd:element name="root">
              <xsd:complexType>
                <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="row" type="row"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:schema>
        </root>
      </m:return>
    </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
<xsd:element name="CATALOG_NAME" type="xsd:string" sql:field="CATALOG_NAME"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

<row>
  <CATALOG_NAME>Demo</CATALOG_NAME>
</row>

................................More Rows....................>
</root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

MDSCHEMA_CUBES Rowset

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

GUID: MDSCHEMA_CUBES

Table 3 describes the rowset structure.

<table>
<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>“CUBE”</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

<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>
        </RestrictionList>
    </Discover>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
<CATALOG_NAME>Demo</CATALOG_NAME>
</RestrictionList>
</Restrictions>
</Properties>
</PropertyList>
</DataSourceInfo>
</Format>Tabular</Format>
</PropertyList>
</Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

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"
xmllns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmllns: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="row"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="row">
<xsd:sequence maxOccurs="unbounded" minOccurs="0">
<xsd:element name="CATALOG_NAME" type="xsd:string"
sql:field="CATALOG_NAME"/>
<xsd:element name="CUBE_NAME" type="xsd:string"
sql:field="CUBE_NAME"/>
<xsd:element name="CUBE_TYPE" type="xsd:string"
sql:field="CUBE_TYPE"/>
<xsd:element name="LAST_SCHEMA_UPDATE" type="xsd:dateTime"
sql:field="LAST_SCHEMA_UPDATE" minOccurs="0"/>
<xsd:element name="DESCRIPTION" type="xsd:string"
sql:field="DESCRIPTION" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
MDSCHEMA_DIMENSIONS Rowset

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

GUID: MDSCHEMA_DIMENSIONS

Table 4 describes the rowset structure.

<table>
<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_UNIQUESETTINGS</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:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
```
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_DIMENSIONS</RequestType>
  <Restrictions>
    <RestrictionList>
      <CATALOG_NAME>Sample</CATALOG_NAME>
      <CUBE_NAME>Basic</CUBE_NAME>
    </RestrictionList>
  </Restrictions>
  <Properties>
    <PropertyList>
      <DataSourceInfo>Provider=Essbase;Data Source=localhost</DataSourceInfo>
      <Format>Tabular</Format>
    </PropertyList>
  </Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Response Example (truncated)

<?xml version="1.0"?>
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
            <xsd:element name="root" type="xsd:complexType">
              <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="row" type="xsd:row"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </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>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------------------------------</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>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: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>
</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">
<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">
```

Working with XMLA
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>
```
Response Example

<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
      <m:return xsi:type="xsd:string">
        <root>
          <CATALOG_NAME xsi:nil="true"/>
          <CUBE_NAME xsi:nil="true"/>
          <DIMENSION_UNIQUE_NAME xsi:nil="true"/>
          <HIERARCHY_NAME xsi:nil="true"/>
          <HIERARCHY_UNIQUE_NAME xsi:nil="true"/>
          <HIERARCHY_CAPTION xsi:nil="true"/>
          <DIMENSION_TYPE xsi:nil="true"/>
          <HIERARCHY_CARDINALITY xsi:nil="true"/>
          <DEFAULT_MEMBER xsi:nil="true"/>
          <ALL_MEMBER xsi:nil="true"/>
          <DESCRIPTION xsi:nil="true"/>
          <STRUCTURE xsi:nil="true"/>
        </root>
      </m:return>
    </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
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>MEASUREAgregar</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: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/">
<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">
```

XMLA Rowsets 45
Working with XMLA

```xml
<row>
  <CATALOG_NAME>Sample</CATALOG_NAME>
  <CUBE_NAME>Sample.Basic</CUBE_NAME>
  <MEASURE_NAME>Measures</MEASURE_NAME>
  <MEASURE_UNIQUE_NAME>[Measures]</MEASURE_UNIQUE_NAME>
  <MEASURE_CAPTION>Measures</MEASURE_CAPTION>
  <MEASURE_AGGREGATOR>0</MEASURE_AGGREGATOR>
  <DATA_TYPE>5</DATA_TYPE>
  <NUMERIC_PRECISION>0</NUMERIC_PRECISION>
  <NUMERIC_SCALE>0</NUMERIC_SCALE>
  <MEASURE_IS_VISIBLE>true</MEASURE_IS_VISIBLE>
</row>
```

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

Response Example (truncated)

<?xml version="1.0"?>
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
      <m:return xsi:type="xsd:string"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
          <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
            targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:sql="urn:schemas-microsoft-com:xml-sql"
            elementFormDefault="qualified">
            <xsd:element name="root">
              <xsd:complexType>
                <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="row" type="row"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:schema>
        </root>
      </m:return>
    </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
<row>
  <CATALOG_NAME>Sample</CATALOG_NAME>
  <CUBE_NAME>Sample.Basic</CUBE_NAME>
  <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
  <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
  <LEVEL_UNIQUE_NAME>[Year].Levels(2)</LEVEL_UNIQUE_NAME>
  <LEVEL_NUMBER>2</LEVEL_NUMBER>
  <GENERATION_NUMBER>1</GENERATION_NUMBER>
  <MEMBER_ORDINAL>1</MEMBER_ORDINAL>
  <MEMBER_NAME>Jan</MEMBER_NAME>
  <MEMBER_UNIQUE_NAME>[Jan]</MEMBER_UNIQUE_NAME>
  <MEMBER_TYPE>1</MEMBER_TYPE>
  <MEMBER_CAPTION>Jan</MEMBER_CAPTION>
  <CHILDREN_CARDINALITY>0</CHILDREN_CARDINALITY>
</row>
**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>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>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------</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>
    <Properties>
      <PropertyList>
        <DataSourceInfo>
          Provider=Essbase;Data Source=localhost
        </DataSourceInfo>
        <Format>Tabular</Format>
      </PropertyList>
    </Properties>
  </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:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sql="urn:schemas-microsoft-com:xml-sql"
elementFormDefault="qualified">
            <xsd:element name="root">
              <xsd:complexType>
                <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="row" type="row"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>

            <xsd:complexType name="row">
              <xsd:sequence 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_UNIQUE_NAME" type="xsd:string" sql:field="DIMENSION_UNIQUE_NAME"/>
                <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string" sql:field="HIERARCHY_UNIQUE_NAME"/>
                <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string" sql:field="LEVEL_UNIQUE_NAME" minOccurs="0"/>
                <xsd:element name="MEMBER_UNIQUE_NAME" type="xsd:string" sql:field="MEMBER_UNIQUE_NAME" minOccurs="0"/>
                <xsd:element name="PROPERTY_TYPE" type="xsd:short" sql:field="PROPERTY_TYPE" minOccurs="0"/>
                <xsd:element name="PROPERTY_NAME" type="xsd:string" sql:field="PROPERTY_NAME" minOccurs="0"/>
                <xsd:element name="PROPERTY_CAPTION" type="xsd:string" sql:field="PROPERTY_CAPTION" minOccurs="0"/>
                <xsd:element name="DATA_TYPE" type="xsd:unsignedShort" sql:field="DATA_TYPE" minOccurs="0"/>
                <xsd:element name="CHARACTER_MAXIMUM_LENGTH" type="xsd:unsignedInt" sql:field="CHARACTER_MAXIMUM_LENGTH" minOccurs="0"/>
                <xsd:element name="CHARACTER_OCTET_LENGTH" type="xsd:unsignedInt" />
            </xsd:sequence>
        </xsd:complexType>
    </root>
  </m:return>
</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>

--------------------

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 10  MDSCHEMA_SETS 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 11  MDSCHEMA_LEVELS 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>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>Column Name</td>
<td>Essbase Mapping</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</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.microsoft.com/soap/encoding/">
<RequestType>MDSCHEMA_LEVELS</RequestType>
<Restrictions>
<RestrictionList>
<CATALOG_NAME>Sample</CATALOG_NAME>
<CUBE_NAME>Basic</CUBE_NAME>
<DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
</RestrictionList>
</Restrictions>
<Properties>
(PropertyList>
<DataSourceInfo>Provider=Essbase;Data Source=localhost</DataSourceInfo>
<Format>Tabular</Format>
</PropertyList>
</Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Response Example**

```xml
<?xml version="1.0"?>
SOAP-ENV:encodingStyle="http://schemas.microsoft.com/soap/encoding/">
<SOAP-ENV:Body>
<m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
<root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
<xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
xmlns:sql="urn:schemas-microsoft-com:xml-sql" elementFormDefault="qualified">
<xsd:element name="root">
<xsd:complexType>
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<xsd:element name="row" type="row"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

XMLA Rowsets 55
<row>
  <CATALOG_NAME>Sample</CATALOG_NAME>
  <CUBE_NAME>Sample.Basic</CUBE_NAME>
  <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
  <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
  <LEVEL_NAME>[Year].Levels(2)</LEVEL_NAME>
  <LEVEL_UNIQUE_NAME>[Year].Levels(2)</LEVEL_UNIQUE_NAME>
  <LEVEL_CAPTION>[Year].Level 2</LEVEL_CAPTION>
  <LEVEL_NUMBER>2</LEVEL_NUMBER>
  <LEVEL_CARDINALITY>12</LEVEL_CARDINALITY>
  <LEVEL_TYPE>4</LEVEL_TYPE>
  <LEVEL_UNIQUE_SETTINGS>2</LEVEL_UNIQUE_SETTINGS>
  <LEVEL_IS_VISIBLE>true</LEVEL_IS_VISIBLE>
  <ESSBASE_GEN_UNIQUE_NAME>[Year].[Months]</ESSBASE_GEN_UNIQUE_NAME>
  <ESSBASE_GEN_CAPTION>[Year].Months</ESSBASE_GEN_CAPTION>
</row>

.........More Rows.........

</root>
**DISCOVER_SCHEMA_ROWSETS Rowset**

GUID: DISCOVER_SCHEMA_ROWSETS

Table 12 describes the rowset structure.

<table>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Essbase Mapping</strong></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>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Essbase Mapping</strong></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 Analytic 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>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Essbase Mapping</strong></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>
</tbody>
</table>
### Column Name | Essbase Mapping
--- | ---
PropertyAccessType | Access for the property. The value can be Read, Write, or ReadWrite
IsRequired | True if a property is required, false if it is not required
Value | Current value of the property

#### Request Example

```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">
        <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <xsd:element name="root" type="row"/>
        </xsd:schema>
      </m:return>
    </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
DISCOVER_ENUMERATORS Rowset

GUID: DISCOVER_ENUMERATORS

Table 15 describes the 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>ElementDescription</td>
<td>Description of the element</td>
</tr>
<tr>
<td>EnumType</td>
<td>Data type of the Enum values</td>
</tr>
</tbody>
</table>

---

Table 15  DISCOVER_ENUMERATORS Rowset Structure
Request Example

```xml
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
      <RequestType>DISCOVER_ENUMERATORS</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

```xml
<?xml version="1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <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">
          
        </root>
      </m:return>
    </m:DiscoverResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
DISCOVER_KEYWORDS Rowset

GUID: DISCOVER_KEYWORDS

Table 16 describes the rowset structure.

Table 16  DISCOVER_KEYWORDS Rowset Structure

<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

<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>
        <xmla:Rowsets>
          <xmla:Rowset>
            <xmla:ColumnTypes>
              <xmla:complexType name="row">
                <xmla:sequence maxOccurs="unbounded" minOccurs="0">
                  <xmla:element name="EnumName" type="xsd:string" sql:field="EnumName"/>
                  <xmla:element name="EnumDescription" type="xsd:string" sql:field="EnumDescription" minOccurs="0"/>
                  <xmla:element name="ElementName" type="xsd:string" sql:field="ElementName"/>
                  <xmla:element name="ElementDescription" type="xsd:string" sql:field="ElementDescription" minOccurs="0"/>
                  <xmla:element name="ElementValue" type="xsd:string" sql:field="ElementValue" minOccurs="0"/>
                  <xmla:element name="EnumType" type="xsd:string" sql:field="EnumType"/>
                </xmla:sequence>
              </xmla:complexType>
              <xmla:row>
                <EnumName>ProviderType</EnumName>
                <ElementName>TDP</ElementName>
                <EnumType>string</EnumType>
              </xmla:row>
              <!-- More Rows -->
            </xmla:ColumnTypes>
            <xmla:RowsetCollection />
          </xmla:Rowset>
        </xmla:Rowsets>
      </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:Body>
    <m:DiscoverResponse
xmlns:m="urn:schemas-microsoft-com:xml-analysis">
      <m:return xsi:type="xsd:string">
            <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: DBLITERAL_LIKE_PERCENT</td>
</tr>
<tr>
<td>LiteralValue</td>
<td>Contains the literal value</td>
</tr>
<tr>
<td></td>
<td>Example, if LiteralName is DBLITERAL_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</td>
</tr>
<tr>
<td></td>
<td>maximum 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.

```mdx
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.

```sql
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.

```sql
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):

<table>
<thead>
<tr>
<th>[Market]. Levels(1). [MEMBER_CAPTION]</th>
<th>[Product]. [Family]. [MEMBER_CAPTION]</th>
<th>[Profit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>Colas</td>
<td>12656</td>
</tr>
<tr>
<td>East</td>
<td>Root Beer</td>
<td>2534</td>
</tr>
<tr>
<td>East</td>
<td>Cream Soda</td>
<td>2627</td>
</tr>
<tr>
<td>East</td>
<td>Fruit Soda</td>
<td>6344</td>
</tr>
<tr>
<td>East</td>
<td>Diet Drinks</td>
<td>2408</td>
</tr>
<tr>
<td>West</td>
<td>Colas</td>
<td>3549</td>
</tr>
</tbody>
</table>
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):

```
East 1 Colas 1 Cola 2 11129
East 1 Colas 1 Diet Cola 2 1114
East 1 Colas 1 Caffeine Free Cola 2 413
East 1 Root Beer 1 Old Fashioned 2 -2540
East 1 Root Beer 1 Diet Root Beer 2 982
East 1 Root Beer 1 Birch Beer 2 4092
East 1 Cream Soda 1 Dark Cream 2 3233
East 1 Cream Soda 1 Vanilla Cream 2 -918
...
...
...
...
```
<table>
<thead>
<tr>
<th>[Market]. Levels(1). [MEMBER Caption]</th>
<th>[Product]. [Family]. [MEMBER Caption]</th>
<th>[Year]. Levels(1). [MEMBER Caption]</th>
<th>[Profit]</th>
</tr>
</thead>
<tbody>
<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/">
    <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"
```
<table>
<thead>
<tr>
<th>Qtr</th>
<th>Month</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qtr1</td>
<td>Jan</td>
<td>8024.00</td>
</tr>
<tr>
<td>Qtr1</td>
<td>Feb</td>
<td>8346.00</td>
</tr>
<tr>
<td>Qtr1</td>
<td>Mar</td>
<td>8333.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td>Apr</td>
<td>8644.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td></td>
<td>9001.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td></td>
<td>9001.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td></td>
<td>9001.00</td>
</tr>
<tr>
<td>Qtr2</td>
<td></td>
<td>9001.00</td>
</tr>
<tr>
<td>Qtr</td>
<td>Month</td>
<td>Sales</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Qtr2</td>
<td>May</td>
<td>8929.000000</td>
</tr>
<tr>
<td>Qtr2</td>
<td>Jun</td>
<td>9534.000000</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Jul</td>
<td>9878.000000</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Aug</td>
<td>9545.000000</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Sep</td>
<td>8489.000000</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Oct</td>
<td>8653.000000</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Nov</td>
<td>8367.000000</td>
</tr>
<tr>
<td>Qtr3</td>
<td>Dec</td>
<td>8780.000000</td>
</tr>
<tr>
<td>Qtr4</td>
<td>calctest</td>
<td>4.000000</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 Analytic Server
- Easy to deploy and use
- Easy to switch between embedded and three-tier modes

Embedded JAPI

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

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

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

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

   `java -DESS_ES_HOME=<root of Embedded JAPI installation>`
Include the following JAR files, which are necessary for Embedded JAPI to work, in your CLASSPATH:

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

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

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

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

**signOn APIs in lessbase 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
) throws EssException
```

**Parameters:**

- **userName**—The user name. Can be null if password is cssToken and the passwordIsToken flag is true.
- **password**—The user password. Cannot be null. If the passwordIsToken flag is true, this represents the cssToken string.
- **passwordIsToken**—A boolean indicating whether the password is cssToken string.
- **userNameAs**—The user name you want to impersonate. If null, no impersonation occurs.
- **providerUrl**—The URL of the Provider Services servlet (For embedded mode pass “embedded”). In the embedded mode, the JAPI client and provider are in the same process space, and JAPI talks directly to the OLAP server. (No separate provider application needs to be running.)

**Returns:**

The signed on domain.

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

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

Working with Java API
java.lang.String userNameAs,
java.lang.String providerUrl,
java.lang.String olapServerName)
throws EssException

Parameters:

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

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

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

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

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

olapServerName—The host name where Essbase Server is running.

Returns:

The connected Essbase Server instance.
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.

- Stop Provider Services.

To configure Web Services to support OWSM:
Find and open `essbase.properties` in `Middleware_HOME\user_projects\domains\epmsystem1\aps\bin`.

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

Save and close `essbase.properties`.

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 e10 = el.addChildElement(envelope.createName("UserName", "", "http://context.webservices.epm.oracle/"));
e10.setValue(s_userName);
SOAPElement e11 = el.addChildElement(envelope.createName("Password", "", "http://context.webservices.epm.oracle/"));
e11.setValue(s_password);
SOAPElement e12 = el.addChildElement(envelope.createName("IsToken", "", "http://context.webservices.epm.oracle/"));
e12.setValue(Boolean.toString(b_isPwdToken));
SOAPElement e13 = el.addChildElement(envelope.createName("UserNameAs", "", "http://context.webservices.epm.oracle/"));
e13.setValue(userNameAs);
```

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

### Datasource

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

- **GetTypes()**
  
  Input: none  
  Output: `BaseObject[]` (array)

- **Get()**
  
  Input: URI  
  Output: `BaseObject`

- **Create()**
  
  Input: `BaseObject`  
  Output: `BaseObject`

- **Rename()**
  
  Input: URI, String(newName)  
  Output: None

- **Update()**
Input: URI, Base Object (application or cube)
Output: BaseObject

- Delete()
  Input: URI
  Output: None

- List()
  Input: UriType, URI
  Output: BaseObject[]

**Administration**

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

- Start()
  Input: URI (Essbase application or cube)
  Output: Void

- Stop()
  Input: URI (Essbase application or cube)
  Output: Void

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

- Perform()
  Input: MaxL statement
  Output: Two-dimensional string array of MaxL result set

**Query**

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

**Data Queries**

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

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

- **zoomIn()**

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

- **zoomOut()**

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

- **pivot()**

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

Output: Grid

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

- **keepOnly()**

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

- **removeOnly()**

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

- **execute()**

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

- **queryMemberHeader**
  In: URI (Essbase cube), QueryMemberOptions
  Out: Essbase member array

- **queryMember**
  In: URI (Essbase cube), QueryMemberOptions
  Out: Essbase member array

- **queryMemberReport**
  In: URI (Essbase cube), QueryReportOptions
  Out: Essbase member array

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

**QueryOption** fields in QueryMemberOptions:
- QUERY_OPTION_MEMBERSONLY
- QUERY_OPTION_ALIASSONLY
- 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_WILDSROMEARCH
- QUERY_TYPE_USERATTRIBUTE
- QUERY_TYPE_ANCESTORS
- QUERY_TYPE_DTMEMBERS
- 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();
    }
}
```
In This Chapter

The Sample Programs ................................................................. 81
Configuring Essbase Servers ..................................................... 84
Compiling and Running the Sample Programs ............................ 84

The Sample Programs

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

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

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

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

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

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

- Allocation.java shows how to perform allocation on an Essbase ASO cube.
- BackupAndRestoreDatabase.java signs onto an Essbase domain, creates an application and database, backs up and then restores the database.
- BuildDimension.java adds and removes members from the outline in the active database.
- CdfCdm.java shows the usage of CDF and CDM.
- CellAttributes.java signs on to Essbase domain, opens a cube view, performs a retrieval, gets the cell attributes, and signs off.
- Connect.java demonstrates a simple connection to an Essbase Server.
- CopyOlapAppAndCube.java copies an Essbase application and cube within the same Essbase Server; can be extended to perform copying across Essbase Servers.
- CreateCluster.java creates a cluster.
- CreateOlapApp.java creates OLAP applications.
- CreateOutline.java demonstrates creating a cube outline: creates dimensions, members, and other outline elements; verifies the outline; and restructures the database.
- CubeDataLoad.java signs onto the Essbase domain, loads data into a cube, and signs off.
- CubeLocks.java signs onto the Essbase domain, gets the list of locks held on a cube, and signs off.
- CustomCalc.java demonstrates how to perform custom calculation on an Essbase Aggregate Storage(ASO) cube.
- CustomMessageHandler.java 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 Analytic Server, executes MaxL statements, then logs off.
- GetLoginIDRequest.java signs onto the Essbase domain and retrieves the login ID of the user signing in; and retrieves the list of executing requests from Essbase Server.
- GetMembers.java signs on to Essbase domain, performs various metadata operations and signs off.
- GetOlapUser, demonstrates fetching of native and external Essbase users.
- GridDataUpdate.java demonstrates the retrieving and updating of data in a grid format.
- GridLockUnlock.java signs on to an Essbase domain, opens a cube view, performs a lock, retrieval, unlock, and signs off.
- GridWithUnknownMembers.java demonstrates how to detect unknown members in data query.
- HisDrillThrough.java signs on to an Essbase domain, opens a cube view, performs Oracle Essbase Integration Services drill-through, lists reports, executes reports, and signs off.
- HisDrillThroughOnRange.java signs on to an Essbase domain, opens a cube view, performs Oracle Essbase Integration Services drill-through, lists reports, executes reports, and signs off.
- HybridAnalysis.java demonstrates how to use the Hybrid Analysis option for data query and metadata operations.
- KillOwnRequest.java kills a running request issued to an Essbase application server from the same user session.
- LinkedObjects.java signs on to an Essbase domain, opens a cube view, performs a retrieval, performs LRO operations, and signs off.
- LinkedPartition.java signs on to an Essbase domain, opens a cube view, performs a retrieval, looks for a linked partition in a cell and signs off.
- ListAndKillOlapRequests.java signs on to an Essbase domain, connects to an Essbase Server, lists the requests, kills requests, and signs off.
- LoadData.java loads data to a cube.
- MdxQuery.java signs on to an Essbase domain, opens a cube view, performs an MDX query execution, retrieves the results, and signs off.
- MetaData.java demonstrates retrieval of metadata information from an Essbase database, including member selection.
- NonUniqueOutline.java tests an existing Sample Basic outline to verify that it allows unique members only, creates a new outline enabled for duplicate (non-unique) member names, and uses the duplicate member name JAPIs.
- NonUniqueQueryOutline.java, queries a outline enabled for duplicate (non-unique) member names. This sample file is intended to test the duplicate member names outline created using the NonUniqueOutline.java sample.
- PropertyViewer.java gets an application/database object, enumerates its properties, and prints the values.
- QueryHints.java signs on to the Essbase domain, performs various query hints-related operations, and signs off.
- QueryVaryingAttributes.java signs onto the Essbase and opens a database; performs a member selection; queries for varying attributes; and signs off.
- ReadOutline.java signs on to the Essbase domain, reads various items in an outline and signs off.
- RunReport.java demonstrates the running of a report from an Essbase database.
- RunXmlReport.java signs on to the Essbase domain, runs a report, and signs off.
- SecurityFilter.java tests the security filters.
- SmartListOutline.java signs onto the Essbase domain, creates a database with MemberType Enabled outline, verifies Smart List outline editing APIs, performs the grid operations, and deletes the outline.
- SmartListReport.java signs onto the Essbase demonstrates grid API report specification usages with respect to querying on smart list, date, and formatted string data cell types.
- SyncCubeReplicas.java demonstrates the replication of data between two Essbase databases.
- TimeDimIntelligence.java shows the time intelligence APIs related to a "date time" dimension.
- ViewOutlineTree.java demonstrates the listing of all outline members from an Essbase database outline.

## Configuring Essbase Servers

The sample programs require an Essbase server with the Demo Basic sample application loaded. You must also create the user “system” on 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:
   - On the Essbase server, verify that the sample applications Demo Basic and Sample Basic are installed and that the databases have been loaded with data.
     
     For information about copying Essbase applications, see the Essbase Database Administrator’s Guide.
   
   - Optional: If you are using more than one server in a cluster, repeat step 1.

## Compiling and Running the Sample Programs

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

Two script files are provided for compiling the sample programs:

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

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

## Configuring the Script Files

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

**Note:** The procedure in this section applies to both a three-tier or embedded Java API deployment.
To configure the script files to work with your computer environment:

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

   If you are deploying the Java API in three-tier mode:
   - `runsamplesAPS.cmd` on Windows
   - `runsamplesAPS.sh` on UNIX

   If you are deploying the Java API in embedded mode:
   - `runsamplesEmbedded.cmd` on Windows
   - `runsamplesEmbedded.sh` on UNIX

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

3. Verify that the `APS_HOME` variable is set to the location of your Provider Services installation; for example:
   ```
   set APS_HOME=C:\Oracle\Middleware\EPMSystem11R1\common\EssbaseJavaAPI\11.1.2.0\n   ```

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

   If you did not install the Java Runtime Environment with Provider Services, you must update this variable with a full path to the Java installation; for example,
   ```
   set JAVA_HOME=\"C:\Oracle\Middleware\jdk160_11\" or C:\Oracle\Middleware\EPMSystem11R1\common\JRE\Sun\1.6.0
   ```

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

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

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

6. Save the script file.

## Compiling and Running Samples

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

You can use the either of the runsamples scripts as a template for new scripts.
The following procedure shows you how to create a version of the runsamplesAPS script to run a different sample program.

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

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

2. Open the runsamples script in a text editor.

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

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

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

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

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

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

### A
- Administration Services Console, 15
- architecture, 9
- authenticating Essbase Server, 16
- automatic deployment of Smart View client, 19

### C
- clustering, 25

### D
- Discover method, 28
- documentation
  - installation and configuration, 13

### E
- editing
  - authenticating Essbase Server, 16
- EPM system, 13
- Essbase, 16, 17
- essbase.properties file
  - options configurable in, 21
- session time out, specifying, 18
- sessions, monitoring, 18
- Smart View, 12
  
### F
- flattened rowsets, 63

### H
- high availability, 25

### I
- installation, 13

### J
- JAPI. See Java API
  - Java API, 10
    - embedded, 71
    - embedded, three-tier solution, 10
    - embedded, two-tier solution, 10
    - features, 71

### L
- logging, 21

### M
- maximum rows and columns, specifying, 19
- monitoring sessions, 18

### P
- Provider Services
  - adding to Administration Services Console, 15
  - connecting to, 17
  - connecting to a stand-alone Essbase Server, 17
  - removing, 16

### S
- sample programs, 81
- session time out, specifying, 18
- sessions, monitoring, 18
- Smart View, 12

### X
- XML for Analysis, 12
  - examples, 68
  - features, 27
methods, 27
rowsets, 33
XMLA. See XML for Analysis