Oracle® Hyperion Enterprise Performance Management Architect

Batch Client User's Guide

Release 11.1.2.4.000

Updated: July 2015

ORACLE®

Performance Management Architect Batch Client User's Guide, 11.1.2.4.000

Authors: EPM Information Development Team

Contents

Documentation Accessibility

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

Access to Oracle Support

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

Documentation Feedback

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

Follow EPM Information Development on these social media sites:

LinkedIn - http://www.linkedin.com/groups?gid=3127051&goback=.gmp_3127051

Twitter - http://twitter.com/hyperionepminfo

Facebook - http://www.facebook.com/pages/Hyperion-EPM-Info/102682103112642

Google+ - https://plus.google.com/106915048672979407731/#106915048672979407731/posts

YouTube - http://www.youtube.com/user/OracleEPMWebcasts

1 Using Performance Management Architect Batch Client

In This Chapter

Oracle Hyperion EPM Architect Batch Client enables you to combine processes such as data export, metadata loads, data loads, and calculations and kick off these operations during your normal nightly or weekly load process.

The batch client enables you to kickoff processes using external scheduling tools. You can use the batch client to perform many tasks, including:

- Load metadata into Performance Management Architect
- Update security related properties on dimensions and measures
- Load data to applications

The batch client is installed automatically when you install Performance Management Architect. You can run the batch client on Windows platforms. When you install Performance Management Architect, a batch file is automatically created to setup the class paths that are generated during installation.

You can run the Performance Management Architect Batch Client in two modes:

- Command line mode
  - You can enter commands interactively
  - Each command can span multiple lines
  - Command statements are terminated by a semi colon delimiter ';'
  - Must use single quotes
  - Commands are executed immediately
- Script mode
  - Executes a series of commands without your interaction
  - You can specify a command file and optionally a result log file at program startup
  - You can schedule the execution of script using a third party scheduler

Launching Batch Client

You can run the Performance Management Architect Batch Client in interactive command line mode or execute a script file specified on the command line.

To launch the Batch Client, select **Start**, **Programs**, **Oracle EPM System**, **Foundation Services**, **Performance Management Architect**, **Start EPMA Batch Client**.

You can also navigate to `EPM_ORACLE_HOME\products\Foundation\BPMA\EPMABatchClient` and launch the `epma-batch-client.bat` file using an interactive command line or script. (In a LINUX or UNIX environment, navigate to `EPM_ORACLE_HOME/products/Foundation/BPMA/EPMABatchClient` and launch the `epma-batch-client.sh` file.)

For example, to launch the batch client in interactive command mode, launch the batch file with no parameters:

`EPM_ORACLE_HOME\products\Foundation\BPMA\EPMABatchClient\epma-batch-client.bat`.

When launching a script, the script file name must be specified with the –C option. All additional parameters are optional. For example: `EPM_ORACLE_HOME\products\Foundation\BPMA\EPMABatchClient\epma-batch-client.bat –CMyScript.txt`

Note: You can specify the script file name and any log file names as relative paths, however the paths must be relative to the `EPMABatchClient` folder, not the folder in which the batch file was launched.

Command Line Options

You can specify the following arguments when you start the batch client.

Table 1 Batch Client Commands

Command	Description
-H	Displays batch client help.
-C	Specifies the name of the script file to execute. For example: `–C'C:\Scripts\LightsOut.txt'`

Command	Description
-G	Specifies the language to use. To use this parameter, enter –G*x* where *x* is one of the following language codes: ● DA—Danish ● DE—German ● ES—Spanish ● FR—French ● IT—Italian ● JA—Japanese ● KO—Korean ● PT_BR—Brazilian Portuguese ● RU—Russian ● SV—Swedish ● TR—Turkish ● ZH_CN—Simplified Chinese ● ZH_TW—Traditional Chinese For example, –Gfr for French or –Git for Italian. If the –G option is not specified the Batch Client attempts to use the current default language from the operating system. If a resource file is not found the Batch Client will use the default, English. If a string is not found in a specific language resource file, the English version will be used.
-R	Specifies the name of the file to write the results to on your machine. For example: `-R'C:\LogFiles\ScriptResult.log'`
-L	Specifies if commands are logged. The default value is 0. Log Commands include: Off (Default) = Do not log commands On = Log Commands For example: `-LOn`
-S	Stops execution of the script if a command fails. The default value is 1 (true). Stop if an error occurs: `True` Continue on Error: `False` For example: `-SFalse`, continues if an error occurs.
-U	Specifies the user name used to login to Performance Management Architect. For example: `-U'Admin'`
-P	Specifies the password used to login to Performance Management Architect. For example: `-Ppassword`
-V	Turns script verification on or off. If Validate is on, the script is checked for syntax errors before it is run and the script will abort if any errors are found. The value 'Off' does not validate before execution and 'On' validates the script before executing. The default value is On and validates the script before executing. For example, `-VOff` does not validate the script before execution.

Command	Description
-O	Validate Only. If specified, the Batch Client validates the script but, pass or fail, it will not run the script. This can be used to test a script for syntax errors without running it.
-N	Specifies the character encoding to use for the output displayed on the console. For example: -Ncp866 For Windows environments, the default is the locale's code page (which can be determined using the `chcp` command). For UNIX environments, the default is `UTF-8`. **Note:** For Japanese Windows environments, you must set the code page to MS932. For example, `-NMS932`.

Note: For On/Off command line parameters, such as –S, the following values are acceptable: 0 or 1, Y or N, Yes or No, True or False, On or Off. For example, these values are all valid: -S0, -SY, -SNo, -STrue, -SOff.

The following excerpt shows an example of how command line options are used.

```
epma-batch-client -H
epma-batch-client -C"C:\EPM_ORACLE_INSTANCE\EPMA\Commands.txt"
         -R"C:\EPM_ORACLE_INSTANCE\EPMA\ResultFile.txt"  -LOn -SFalse
         -Uadmin -Ppassword
```

Return Codes

When the Batch Client exits it returns a result code to the calling program based on the exit condition of the script. There are two possible scenarios based on the StopOnError setting.

`StopOnError = False (-S0)`

If `StopOnError` is false the return code indicates general success or failure.

0 = success (no errors)

-1 = failure (one or more errors occurred)

StopOnError = True (-S1)

When `StopOnError` is true the Batch Client exit when an error occurs and returns a code based on the following table. By default, StopOnError is set to True.

The batch client returns the following result codes in the event of success or failure.

Command	Command Code	Class	Class Code	Return Code
Success	N/A	N/A	N/A	0
General Error	N/A	N/A	N/A	-1
Validation Error	N/A	N/A	N/A	1
Parse Error	N/A	N/A	N/A	100

Command	Command Code	Class	Class Code	Return Code
Command Line Error	N/A	N/A	N/A	4
Copy	15	Application	1	1501
Copy	15	Dimension	2	1502
Create	1	Application	1	101
Create	1	Dimension	2	102
Create	1	Member	3	103
Create	1	Association	10	110
Debug	21	N/A	N/A	2100
Delete	2	Application	1	201
Delete	2	Dimension	2	202
Delete	2	Member	3	203
Delete	2	Association	10	210
Detach	16	Dimension	2	1602
Exclude	3	Member	3	303
Execute	4	DataSynchronization	4	404
Execute	4	Deploy	5	405
Execute	4	DimensionSynchronization	6	406
Execute	4	Import	7	407
Execute	4	Redeploy	9	409
Execute	4	Validate	12	412
Exit	5	N/A	0	500
Include	6	Dimension	2	602
Include	6	Member	3	603
Insert	7	Member	3	703
Login	8	N/A	0	800
Logout	9	N/A	0	900
Move	19	Member	3	1903
Option	20	N/A	0	2000

Command	Command Code	Class	Class Code	Return Code
Quit	10	N/A	0	1000
Remove	11	Dimension	2	1102
Remove	11	Member	3	1103
Rename	18	Member	3	1803
Set	12	N/A	0	1200
Share	17	Dimension	2	1702
Update	13	Application	1	1301
Update	13	DimensionAssociation	8	1308
Update	13	Dimension	2	1302
Update	13	Member	3	1303
Variable	14	N/A	0	1400

In a DOS batch file or Windows command file the error can be checked as follows:

```
Call epma-batch-client.bat .\scripts\MyScript.txt
IF ERRORLEVEL 0 goto ON_SUCCESS
IF ERRORLEVEL 100 goto PARSE_ERROR
If ERRORLEVEL 101 goto APP_CREATE_FAILED
```

Logging

The batch client provides several levels of logging through result and trace files. Result files contain details of the commands you enter in the command window and their execution status, error, or warning messages. Trace files contain a detail stack trace that is useful for debugging purposes.

The batch client uses Oracle Diagnostics Logging (ODL) for logging. All of the configuration parameters are stored in `logger.xml` in the `conf` directory. For example, `EPM_ORACLE_INSTANCE\products\Foundation\BPMA\EPMABatchClient\output`.

The following excerpt shows an example of a `logger.xml` file.

```
<logging_configuration>
    <log_handlers>
       <log_handler name="traceFile" class="oracle.core.ojdl.logging.ODLHandlerFactory">
        <property name="path" value="${user.dir}/output/EPMABatchClientTrace.log"/>
        <property name="maxFileSize" value="1000000"/>
        <property name="maxLogSize" value="5000000"/>
        <property name="useSourceClassAndMethod" value="true"/>
        <property name="encoding" value="UTF-8"/>
    </log_handler>
     <log_handler name="console"
                       level="ALL"
                       class="java.util.logging.ConsoleHandler"
```

```
                    formatter="com.hyperion.bpma.logger.GenericFormatter"/>
     <log_handler name="resultsFile"
                    class="com.hyperion.bpma.logger.GenericFileHandler"
                    formatter="com.hyperion.bpma.logger.GenericFormatter"
                    level="ALL"/>
    </log_handlers>
     <loggers>
        <logger name="BatchClient.trace" level="ALL" useParentHandlers="false">
            <handler name="traceFile"/>
        </logger>
         <logger name="BatchClient.console" level="INFO" useParentHandlers="false">
            <handler name="console"/>
        </logger>
         <logger name="BatchClient.results" level="INFO" useParentHandlers="false">
            <handler name="resultsFile"/>
        </logger>
    </loggers>
</logging_configuration>
```

To help you determine if the command was successful, the associated JobID and the JobID URL is logged for the following commands:

- Execute Deploy
- Execute ReDeploy
- Execute Validate
- Execute DataSynchronization
- Execute Import
- Copy Application
- Detach Dimension
- Share Dimension

Note: You have the option to use standard Formatters available in Java 1.6. For example, you can use `java.util.logging.SimpleFormatter` instead of `com.hyperion.bpma.logger.GenericFormatter`.

2 Configuring Command Files

In This Chapter

A command file is an input file for the batch client. It can contain one or more of the following:

- Commands
- Variables, declarations, and assignments
- Comments

The following excerpt is an example of a command file.

```
// Test Script
set bpmaserverurl=http://localhost/hyperion-bpma-server;
set workspaceurl=http://localhost:19000/workspace;

login admin,password;

set ApplicationName = 'Sample';

// Delete some members
Delete Member
    Properties(MemberName, DimensionName, ParentName, DeleteAllDescendants)
    Values('M1-1-1', 'A1', 'M1-1', true);

Delete Member
    Properties(MemberName, DimensionName, ParentName, DeleteAllDescendants)
    Values('M1', 'A1', '#root', false);

Delete Member
    Properties(MemberName, DimensionName, ParentName, DeleteAllDescendants)
    Values('M1', 'A1', '#root', true);

Delete Dimension
    Properties(DimensionName)
    Values('A1');

Delete Dimension
```

```
    Properties(DimensionName)
    Values('E1');

Delete Application
    Properties(ApplicationName, WaitForCompletion)
    Values('TestApp1');


set ApplicationName = '';

// Delete shared dims
Delete Dimension
    Properties(DimensionName)
    Values('S1');

quit;
```

Most of the commands in a command file execute immediately. However, EXECUTE commands can take a long time to execute and support a `WaitForCompletion` parameter. All execute commands support this parameter, except for DIMSYNCRONIZATION. For example, you can use a `WaitForCompletion` parameter to force the batch client to wait for command execution. The following command shows an example of the `WaitForCompletion` parameter. In this case, the administrator is executing a data synchronization command, which can take longer to run. Other commands that take longer to run include imports and application deployments.

```
execute datasynchronization
     parameters(DataSynchronizationName, DataTransformationOperator,
DataTransformationValue, FileName,ValidateOnly, WaitForCompletion)
     values('CommaSync3', '*', '1.2345', '', 'false', 'true');
```

Variables

You can define variables and also call them in scripts. Characteristics of variables include:

- Variables have a name and a single value type
- The value of a variable can be any data type
- You can define a variable once and use it in multiple places
- You can change the value of a variable between commands
- Variable names are case sensitive
- You can use the keyword, `var` to define a variable, and refer to the variable using `$`

The following excerpt shows an example of using variables to create a shared dimension.

```
// Create Shared Dimension Script
set bpmaserverurl=http://localhost/hyperion-bpma-server;
set workspaceurl=http://localhost:19000/workspace;

login admin,password;
var DimType='Scenario';
// Create a shared dimension
create Dimension
```

```
Properties(DimensionName, DimensionDescription, DimensionType)
Values('S1','New Scenario','$DimType');
```

There are three types of variables:

- User

  User variables are assigned using the keyword, `var`. For example:

  ```
  var variable1 = 'abc';
  ```

- System

  System variables are assigned using the `set` command. For example:

  ```
  set bpmaserverurl='http://localhost/hyperion-bpma-server';
  set workspaceurl='http://localhost:19000/workspace';
  ```

- Object

  Object variables are assigned using the `set` command. For example:

  ```
  set ApplicationName = 'Comma';

  set dimension=Account;
  ```

Comments

You can comment out any line in a script by using two forward slashes `//` at the beginning of the line. If a line spans into multiple lines, you must comment out each line separately. For example:

```
//execute datasynchronization
//  parameters(DataSynchronizationName, DataTransformationOperator,
DataTransformationValue, FileName,
//     ValidateOnly, WaitForCompletion)
//  values('CommaSync3', '*', '1.2345', '', 'false', 'true');
```

Commands

A command consists of a:

- Command verb
- Command class
- Parameter–Value collection or a Property-Value collection

Parameters and values are separated by commas. Supported command verbs include:

- CREATE
- COPY
- DEBUG

- DELETE
- DETACH
- EXCLUDE
- EXECUTE
- EXIT
- INCLUDE
- INSERT
- LOGIN
- LOGOUT
- MOVE
- OPTION
- QUIT
- REMOVE
- RENAME
- SHARE
- UPDATE
- SET
- VARIABLE

Supported command classes include:

- APPLICATION
- ASSOCIATION
- DIMENSION
- DIMENSIONASSOCIATION
- DIMSYNCHRONIZATION
- MEMBER
- IMPORT
- DEPLOY
- DATASYNCHRONIZATION
- REDEPLOY
- VALIDATE

Tip: Use the constant `#root` for the root member of a dimension. Use the constant `#shared` for the application name for commands targeted at the Shared Library.

Scripts

Scripts are a collection of commands that can be executed sequentially. Scripts can have commands in any sequence, however, certain initialization commands like `login` must execute before others. Each command in a script is separated by a semi colon ';'. Commands can contain white spaces and span multiple lines.

Commonly Used Commands

Subtopics

The following sections provide examples of commonly used commands.

Option Commands

You can use the Option command to dynamically modify command line options during the execution of a script. You can change the following options:

```
StopOnError
option StopOnError = true;

EchoComments
 option EchoComments = true;

LogCommands
 option LogCommands = true;
```

Login / Logout Commands

Subtopics

- login
- Logout
- Quit
- Exit

login

Logs into Performance Management Architect.

If you log in using a script, enter:

```
Login admin,password;
```

If you log in using a command line, enter:

```
Login;
```

You can save the password in an encrypted file and log in using this encrypted file. You can use either of these two methods to encrypt the password:

- To encrypt the password through a command line argument:

  ```
  ..\EPMABatchClient>epma-batch-client.bat -Ac:/some.txt -Xpassword
  ```

  where:

  - -X specifies the password that needs to be encrypted.
  - -A (optional) specifies the path/file name in which the encrypted password is saved. If you do not use this parameter, a file with the name `password.txt` is created in the location where the Batch Client utility resides. If you do use this parameter, you should specify it before -X, because the order in which these parameters are passed is predefined.
- To encrypt the password through a script file:

  ```
  Encrypt <password>,<path>;
  ```

  where:

  - <password> refers to the password that needs to be encrypted.
  - <path> (optional) refers to the path/file where the encrypted password is saved. The order of the parameters should be as shown in the example, because the order in which the parameters are passed is predefined.

There are two options for logging in using the encrypted password file:

- To log in through a command line argument:

  ```
  ..\EPMABatchClient>epma-batch-client.bat -CplanDeploy.txt -Uadmin-Fc:\pass.txt
  ```

  where:

- -F specifies the path of the file containing the encrypted password. The order of the parameters should be as shown in the example, because the order in which the parameters are passed is predefined.
- -P need not be used to specify the password.

- To log in through a script file:

  ```
  login <username>, F<path of file containing encrypted password>;
  ```

  Instead of specifying the password directly, you can use `"-F<path>"` to specify that the password needs to be taken from the encrypted file.

Login Through a Proxy

You can use the SET command and define the following variables to login through a proxy.

```
set ProxyHost='http://localhost/myProxy';
set ProxyPort='8080';
set ProxyUsername='myProxyUser';
set ProxyPassword='myProxyPassword';
```

ProxyHost —The proxy host name.

ProxyPort—The proxy port number.

ProxyUsername—The user name to be authenticated.

ProxyPassword—The user password.

Login Using Single Sign On (SSO) such as Oracle Access Manager

You have two options to login using single sign on:

- Basic/Digest authentication

  For Basic/Digest authentication, set the SSOType to Basic as follows:

  ```
  set SSOType='Basic';
  ```

- Form based authentication

  For form based authentication, set the SSOType to Form as follows:

  ```
  set SSOType='Form';
  ```

  If you are using Form based authentication, you must define the field names for where the username and password should be posted as follows:

  ```
  set SSOFormUsernameField='userid';
  set SSOFormPasswordField='password';
  ```

  Note: The default field name is 'userid' and 'password'.

Logout

Logs out of Performance Management Architect.

```
Logout;
```

Quit

Closes the batch client.

```
Quit;
```

Exit

Closes the batch client.

```
Exit;
```

Copy Commands

Subtopics

- Application
- Dimension

Application

The Copy Application command is the same as the "Duplicate As New" command available in the Application Library.

```
Copy Application
Properties ( ApplicationName, CopyApplicationToName, ApplicationDescription,
ApplicationType)
Values( 'Comma', 'CommaCopy', 'Copied App Desc', 'Consolidation');
```

ApplicationName—The name of an existing application.

CopyApplicationToName—The name of the new duplicated application.

ApplicationDescription—The application description.

ApplicationType—Type of application. Valid values are: Generic, Consolidation, Planning, Profitability - Standard, Profitability - Detailed, Essbase ASO, or Essbase BSO.

Dimension

Provides a way to copy dimensions within the Shared Library, within an application, or between the Shared Library and an application. You cannot copy a dimension directly from one application to another.

```
Copy Dimension
Properties(ApplicationName, DimensionName, TargetDimensionName,
TargetDimensionDescription,
destApplicationName)
    Values('#Shared', 'Scenario','CopyScenario', 'Copy of Scenario Dim', Comma');
```

ApplicationName—The name of an existing application.

DimensionName—The name of an existing dimension.

TargetDimensionName—The name of the target dimension.

TargetDimensionDescription—The description of the target dimension.

destApplicationName—The name of the destination application.

Create Commands

Subtopics

Application

Creates a new, empty application with the specified name.

```
Create Application
Properties(ApplicationName, ApplicationDescription, ApplicationType)
Values('Comma', 'Description for Comma', 'Consolidation');
```

Application Name—A string containing a valid name for the application.

ApplicationDescription—A string containing a valid name for the application.

ApplicationType—Supports the following values:

- Generic
- Consolidation
- Planning
- Enterprise Analytics

  For Essbase ASO applications
- Essbase Analytics

  For Essbase BSO applications
- Profitability
- Profitability - Detailed

Dimension

Creates a new, empty dimension in the specified application or in the Shared Library.

```
Create Dimension
Pproperties(ApplicationName, DimensionName, DimensionDescription,
DimensionType)
Values('Comma','Test_Account','Test Account','Account');
```

ApplicationName—The name of an existing application. Use `#Shared` to create a dimension in the Shared Library.

DimensionName—A valid name for the dimension.

DimensionType—Dimension Type can be any one of the following:

- Account
- Alias
- AllocationType
- Attribute
- ConsolidationMethod
- Country
- Currency
- Entity
- Generic
- ICP
- Measure
- MeasuresDetailed
- Scenario
- SecurityClass
- SmartList
- Time
- UDA
- Value
- Version
- View
- Year

Member

Creates a new member in the specified dimension.

```
Create Member
Properties(ApplicationName, DimensionName, ParentName, MemberName, MemberDescription)
Values('Comma','Member_Dim', '#root', 'TestMember1', 'Description for TestMember1');
```

ApplicationName—The name of an existing application. Use `#Shared` to create a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

ParentName—The name of the parent under which to insert the newly created member. Use `#Root` to add a new member at the top level of the tree.

MemberName—A valid name for the new member.

MemberDescription—A description for the new member.

Create Association Commands

Creates an association between two dimensions. If the base dimension is shared the associated dimension must be a shared dimension.

```
Create Association
    Properties(ApplicationName, DimensionName, AssociatedDimensionName,
PropertyName,PropertyDescription)
    Values('Comma', 'Scenario', 'AttribDim', 'AttribProp', 'Attrib Prop Desc');
```

ApplicationName—The name of an existing application.

DimensionName—The name of an existing dimension.

AssociatedDimensionName—The name of the dimension that you are associating with.

PropertyName—The name of the property to update. (You can list as many properties as desired, however, you must include a valid value for the property.

PropertyDescription—An optional parameter that you can use to enter a comment or property description.

Delete Association Commands

Deletes an existing dimension association.

```
Delete Aassociation
    Properties(ApplicationName, DimensionName, PropertyName)
    Values('Comma', 'Scenario', 'SecurityClass');
```

ApplicationName—The name of an existing application.

DimensionName—The name of an existing dimension.

PropertyName—The name of the property to update. (You can list as many properties as desired, however, you must include a valid value for the property.)

Delete Commands

Subtopics

Application

Deletes the specified application.

```
Delete Application
Properties(ApplicationName, WaitForCompletion)
Values('Comma');
```

ApplicationName—The name of an existing application.

WaitForCompletion—If set to true, the Batch Client waits for the job to finish. If set to false, the Batch Client submits the job and continues. The default value is false. Allowed values:

- True
- False

Association

Deletes an existing dimension association.

```
Delete Aassociation
    Properties(ApplicationName, DimensionName, PropertyName)
    Values('Comma', 'Scenario', 'SecurityClass');
```

ApplicationName—The name of an existing application.

DimensionName—The name of an existing dimension.

PropertyName—The name of the property to update. (You can list as many properties as desired, however, you must include a valid value for the property.)

Dimension

Deletes the specified dimension.

```
Delete Dimension
Properties(ApplicationName, DimensionName)
Values('Comma','C_Scenario');
```

ApplicationName—The name of an existing application. Use `#Shared` to delete a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

Member

Deletes the specified member and optionally all of the member's descendants.

```
Delete Member
Properties(ApplicationName, DimensionName, ParentName, MemberName, DeleteAllDescendants)
Values('Comma','C_Account','#root','TestMember1','false');
```

ApplicationName—The name of an existing application. Use `#Shared` to delete a member from a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

ParentName—The name of the parent under which to delete the member. Use `#Root` when deleting a member at the top level of the tree.

MemberName—The name of the member to delete.

DeleteAllDescendants—Indicates if any descendants under the member being deleted should also be deleted. Allowed values:

- True
- False

Detach Dimension Commands

Converts a shared dimension in an application to a local dimension.

```
Detach Dimension
    Properties(ApplicationName, DimensionName, RetainFilteredStructure,
RetainPropertyOverrides,waitForCompletion)
    Values('Comma', 'Period', 'true', 'true', 'true');
```

ApplicationName—The name of an existing application.

DimensionName—The name of an existing dimension.

RetainFilteredStructure—If set to true, the current view of the dimension is kept and any excluded or otherwise filter members will not be present in the local copy of the dimension. If set to false, all of the members of the dimension will be present in the local copy of the dimension. Allowed values:

- True
- False

RetainPropertyOverrides—If set to true, all of the property overrides will be maintained, otherwise the current values from the shared version of the dimension will be used. Allowed values:

- True
- False

WaitForCompletion—If set to true, the Batch Client waits for the job to finish. If set to false, the Batch Client submits the job and continues. Allowed values:

- True
- False

Exclude Commands

Member

Filters out a member from a shared dimension.

```
Exclude Member
    Properties(ApplicationName, DimensionName, ParentName, MemberName)
    Values('Comma', 'Period', '#root', 'P1');
```

ApplicationName—The name of an existing application. Use cannot use `#Shared` as the target for excluding a dimension.

DimensionName—The name of an existing dimension in the Shared Library to exclude.

ParentName—Name of the Parent of the member to exclude.

MemberName—Name of the member to exclude.

Execute Commands

Subtopics

You can use EXECUTE commands to run jobs.

Import (into an application or Shared Library)

Executes an existing Import Profile.

For flat files, the syntax is:

```
Execute Import
Parameters(importtype, profilename, filename, waitforcompletion)Values('flatfile',
'Comma', '.\AppFiles\CommaApp.ads', 'true');
```

For interface tables, the syntax is:

```
Execute Import
Parameters(importtype, profilename, waitforcompletion)
Values('InterfaceTables', 'intapp', 'true');
```

For Data Relationship Management, the syntax is:

```
Execute Import
Parameters(importtype, profilename, waitforcompletion)
Values('DRM', 'DRM Profile', 'true');
```

ImportType—The type of import to perform. Allowed values:

- FlatFile
- InterfaceTables
- DRM (Data Relationship Management)

ProfileName—The name of an existing import profile.

FileName—The name of the flat file to import, if doing a flat file type import.

Note: If you are importing an interface table or a Data Relationship Management file, you do not use the FileName parameter; it is only used for flat file imports.

WaitForCompletion—If set to true, the Batch Client waits for the job to finish. If set to false, the Batch Client submits the job and continues. Allowed values:

- True
- False

Dimension Synchronization (to or from the Shared Library)

Synchronizes a dimension to or from the Shared Library

```
Execute Dimensionsynchronization
Parameters(SourceApplicationName, DestApplicationName, DestDimensionName, ReplaceMode)
Values('appName', 'DestAppName', 'DimName', 'true');
```

ApplicationName—The name of an existing application. You cannot use `#Shared` as the application name for a dimension synchronization.

DimensionName—The name of an existing dimension in the application.

SharedDimensionName—The name of an existing dimension in the Shared Library.

SyncToApp—A value of true indicates the shared dimension is synchronized to the application, while a value of false indicates that application is synchronized to the Shared Library. Allowed values:

- True
- False

ReplaceMode—A value of true indicates that the dimension synchronization will use Replace mode, while a value of false indicates that it will use Merge mode. Allowed values:

- True
- False

Deployment

Deploys an application to a specified product.

```
Execute Deploy
Parameters(ApplicationName, InstanceName, ApplicationServer, HubProject, ClearAll,
CheckIntegrity, waitforcompletion, deployOption, Notes)
Values('AppName', 'HFM931', 'localhost:1423', 'Default Application Group', 'false',
'false', 'true', 'true', 'AppView', 'deploy comments sample');
```

ApplicationName—The name of an existing application. You cannot use `#Shared` as the application name for a deployment.

InstanceName—Name of the instance to deploy to.

Note: The instance name is required and should be the same as the name of the logical web application (LWA) defined in the registry. There may be multiple logical web application names; you can use any of these names. However, after you deploy the application, you must use the logical web application name used when you deployed the application.

ApplicationServer—Name of the application server to deploy to.

HubProject—Oracle Hyperion Shared Services project to add the deployed application to.

ClearAll—Deletes all dimension members in the application database and any corresponding data, journals, and intercompany transactions. (Available for Consolidation applications only.) Allowed values:

- True
- False

CheckIntegrity—Checks the metadata against the data to ensure integrity. (Available for Consolidation applications only.) Allowed values:

- True
- False

deployOption—Specifies the type of deployment that you want to perform. The default value is `AppView`, which deploys the application only.

WaitForCompletion—If set to True, the Batch Client waits for the job to finish. If set to False, the Batch Client submits the job and continues. Allowed values:

- True
- False

For Planning applications:

datasourceName—Creates a data source with the name that you specify. This value is a string.

CreateOutline—Creates the Essbase outline if you are deploying the application for the first time. Allowed values:

- True
- False

RefreshOutline—Refreshes the application database after changing the application structure. Allowed values:

- True
- False

CreateSecurityFilters—Stores access permissions in an encrypted data file (`Essbase.sec`). Allowed values:

- True
- False

SharedMembersSecurityFilters—Applies access permissions to shared members. Allowed values:

- True
- False

ValidateSecurityFilterLimit—Identifies security filters that exceed the Oracle Essbase security filter limit of 64 KB per row. This validates filter size to ensure it does not exceed the size limit before building Oracle Essbase security filters. Allowed values:

- True
- False

Notes—Optionally stores any comments or notes that you may want to add regarding the deployment.

Redeployment

Redeploys an application to the product server.

```
Execute Redeploy
Parameters(ApplicationName, InstanceName, HubProject, ClearAll, CheckIntegrity,
waitforcompletion, deployOption, escapeValidateRules, Notes)
Values('AppName', 'HubProj', 'false', 'false', 'true', 'true', 'AppView', 'true',
'Comments on redeployment');
```

See "Deployment" on page 32 example for valid values.

When you redeploy an application, the `deployOption` allowed values differ from deployment for Oracle Hyperion Planning and Oracle Hyperion Financial Management applications. Allowed values:

- ApplicationName—The name of an existing application. You cannot use `#Shared` as the application name for a deployment.

  InstanceName—Name of the instance to deploy to.

HubProject—Oracle Hyperion Shared Services project to add the deployed application to.

ClearAll—Deletes all dimension members in the application database and any corresponding data, journals, and intercompany transactions. (Available for Consolidation applications only.) Allowed values:

- True
- False

CheckIntegrity—Checks the metadata against the data to ensure integrity. (Available for Consolidation applications only.) Allowed values:

- True
- False

WaitForCompletion—If set to True, the Batch Client waits for the job to finish. If set to False, the Batch Client submits the job and continues. Allowed values:

- True
- False

- deployOption—Specifies the type of deployment that you want to perform. The default value is `AppView`, which deploys the application only.
- escapeValidateRules—If set to True, rules deploy without being validated. If set to False, rules deploy after being validated. Allowed values:
  - True
  - False
- Notes—Optionally stores any comments or notes that you may want to add regarding the deployment.
- AppView—Deploy the application only. (This setting is the default.)
- Rules—Deploy calculation rules.
- All—Deploy calculation rules and the application.

For all other application types, the only value allowed is `AppView`.

Data Synchronization

Executes an existing data synchronization profile.

```
Execute DataSynchronization
Parameters(DataSynchronizationName, DataTransformationOperator,
DataTransformationValue, FileName, ValidateOnly, WaitForCompletion)
Values('CommaSync3', '*', '1.2345', '', 'false', 'true');
```

DataSynchronizationName—Name of the Data Synchronization profile to execute.

DataTransformationOperator—Valid values are:

- None

- '*' (Multiply)
- '/' (Divide)
- '+' (Add)
- '-' (Subtract)

DataTransformationValue—Value to use in conjunction with the DataTransformationOperator to modify the data values.

FileName—If the synchronization uses an external source file for the source of the synchronization, the location of the external source file. The location must be a URL location to which the Web server has access. The file path should be "file:////<machine name>/<folder name>/<file name> " if the file is on a remote computer, and "file:///C:/CvgCompLd_JobTtl.txt'" if the file is on a computer where the Data Synchronization service is running.

ValidateOnly—Validates the data synchronization without executing it.

WaitForCompletion—If set to true, the Batch Client waits for the job to finish. If set to false, the Batch Client submits the job and continues. Allowed values:

- True
- False

In addition to the above parameters, there are also data load option parameters. Data load option parameters are:

- Dynamic
- Retrieved from the registry
- Parameter names differ based on the application type
- All parameters are case sensitive, unlike the other parameters
- All values are case sensitive

For Planning, there is one parameter: `dataSyncLoadOptionHpMode`. Allowed values:

- ADD
- SUBTRACT
- OVERWRITE

For Essbase ASO, there is one parameter: `dataSyncLoadOptionAsoMode`. Allowed values:

- ADD
- SUBTRACT
- OVERWRITE

For Essbase BSO, there is one parameter: `dataSyncLoadOptionBsoMode`. Allowed values:

- ADD

- SUBTRACT
- OVERWRITE

For Planning, Essbase ASO, and Essbase BSO applications, the default value is `OVERWRITE`.

For Consolidation, there are two parameters:

- `dataSyncLoadOptionHfmMode`—Allowed values:
  - merge
  - replace
  - accumulate

  The default value is `merge`.
- `dataSyncLoadOptionHfmAccummulateInFile`—Allowed values:
  - True
  - False

  The default is `false`.

Note: Profitability and Profitability - Detailed applications do not support data load options.

Validation

Runs an application validation. As long as the StopOnError option is set true the script will end if the validation fails.

```
Execute Validate
Parameters(ApplicationName, ValidateType)
      Values('Comma1', 'All');
```

ApplicationName—Name of existing application.

ValidateType—Type of validation that you want to perform. For example, validate the application only, business rules, or all (application and rules). Allowed values:

- AppView
- Rules
- All

Include Commands

Dimension

Adds an existing dimension from the Shared Library to the specified application. The dimension can be added as a shared dimension or copied to the application as a local dimension.

```
Include Dimension
Properties(DimensionName, IncludeAsShared)
Values('C_Alias', 'true');
```

ApplicationName—The name of an existing application. Use cannot use `#Shared` as the target for including a dimension.

DimensionName—The name of an existing dimension in the Shared Library to include.

IncludeAsShared—Use a value of true to include the dimension as a shared dimension that is still linked to the source dimension in the Shared Library. Use a value of false to include a copy of the dimension that is separate from the dimension in the Shared Library. Allowed values:

- True
- False

Insert Commands

Insert Member

Inserts a copy of a member as a shared member. You can only use the Insert Member command on local dimensions and Shared Library dimensions. You cannot insert members in a shared dimension in an application.

```
Insert Member Properties(DimensionName, ParentName, InsertMemberName, MemberToInsertName)
Values('Account', 'Par1', 'Mem1', 'Mem2');
```

ApplicationName—The name of an existing application. Use `#Shared` to work with a member in a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

ParentName—Name of the Parent of the member to insert under.

InsertMemberName—Name of the member to insert the member under.

MemberToInsertName—Name of the shared member to be inserted.

Move Commands

Move Member

Moves a member from one location in the dimension structure to another. You can only use the Move Member command on local dimensions and Shared Library dimensions. You cannot move members in a shared dimension in an application.

```
Move Member
Properties(ApplicationName, DimensionName, FromParentName,
MemberName,ToParentName,InsertAfterMemberName)
Values('SampleApp', 'Period', '#root', 'r1', 'P2', '#none');
```

ApplicationName—The name of an existing application. Use `#Shared` to work with a member in a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

FromParentName—Name of the parent of the member that you want to move.

MemberName—Name of the member that you want to move.

ToParentName—Name of the parent to move the member under.

InsertAfterMember—Indicates which child, under ToParentName, the member should be inserted after. This can be set to the special value '#none' indicating the member should be inserted as child one. This will affect the sort order for the member being inserted and all members after it.

Rename Commands

Rename Member

Renames a member and all shared copies of the member. You can only use the Rename Member command on local dimensions and Shared Library dimensions. You cannot rename members in a shared dimension in an application.

```
Rename Member
    Properties(ApplicationName, DimensionName, ParentName, MemberName, NewMemberName)
    Values('Comma', 'Account', '#root', 'M2', 'M2REN');
```

ApplicationName—The name of an existing application. Use `#Shared` to work with a member in a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

ParentName—Name of the Parent of the member that you want to rename.

MemberName—Name of the existing member to rename.

NewMemberName—The new name for the member.

Remove Commands

Subtopics

Remove Member (Application or Shared Library)

Removes a member from the specified dimension but does not delete it. You can only use the Remove Member command to remove a shared dimension in an application.

```
Remove Member
Properties(DimensionName, ParentName, MemberName)
Values('Account', 'Mem1', 'Mem2');
```

ApplicationName—The name of an existing application. Use `#Shared` to work with a member in a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

ParentName—Name of the Parent of the member to be removed.

MemberName—Name of the member to remove.

Remove Dimension

Removes a shared dimension from an application.

```
Remove Dimension
    Properties(ApplicationName, DimensionName, Force)
    Values('Comma', 'Period', 'true');
```

ApplicationName—The name of an existing application. Use `#Shared` to work with a member in a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

Force—If set to true, the dimension is removed even if it is currently associated with other dimensions in the application. If set to false, the remove will fail if the dimension is associated with other dimensions in the application. Allowed values:

- True
- False

Share Dimension Commands

Converts a local dimension to either a new shared dimension or merges it with an existing dimension.

```
Share Dimension
    Properties(ApplicationName, DimensionName, ShareAsNew, SharedDimensionName,
MergeAsShared,
         WaitForCompletion)
    Values('Comma', 'Entity', 'false', 'ShareEntity', 'true', 'true');
```

ApplicationName—The name of an existing application.

DimensionName—The name of an existing dimension.

ShareAsNew—If set to true, the SharedDimensionName and MergeAsShared properties are ignored as they do not apply when sharing a dimension as new. Allowed values:

- True
- False

SharedDimensionName—If set to false, you must supply the name of the dimension in the Shared library to share. Allowed values:

- True
- False

MergeAsShared—If set to true, the dimension being shared will be merged with the target dimension. If set to false, the target dimension will be replaced by the dimension being shared. Allowed values:

- True
- False

WaitForCompletion—If set to true, the Batch Client waits for the job to finish. If set to false, the Batch Client submits the job and continues. Allowed values:

- True
- False

Update Commands

Subtopics

- Application
- Dimension
- Member
- Dimension Association

If you use the UPDATE script commands to modify application, dimension, or member property values, you must use the *Property Name* and not the Property Label displayed in the Performance Management Architect Property Grid. Property labels and names are documented in the appendixes of the *Oracle Hyperion Enterprise Performance Management Architect Administrator's Guide*. The following script shows an example:

```
Update Member
Properties(DimensionName, ParentName, MemberName, AggregationWeight, NumDecimalPlaces)
Values('ScenarioDim', '#root', 'Member1', '3', '2');
```

In this example DimensionName, ParentName and MemberName are all standard script items, however, `AggregationWeight` and `NumDecimalPlaces` are Oracle Hyperion EPM Architect member level properties.

Application

Updates one or more properties of the specified application.

```
Update Application
Properties(ApplicationName, ValidationAccount)
Values('Comma', 'Validation');
```

ApplicationName—The name of an existing application. You cannot update property values on `#Shared`.

PropertyName—The name of the property to update. (You can list as many properties as desired, however, you must include a valid value for the property.)

Dimension

Updates one or more properties of the specified dimension.

```
Update Dimension
Properties(ApplicationName, DimensionName, PropertyName)
Values('Comma', 'C_Entity', 'Validation');
```

ApplicationName—The name of an existing application. Use `#Shared` to update a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

PropertyName—The name of the property to update. (You can list as many properties as desired, however, you must include a valid value for the property.)

Member

Updates one or more properties of the specified member.

```
Update Member
Properties(ApplicationName, DimensionName, ParentName, MemberName, ValidationAccount)
Values('Comma', 'C_Entity', 'E1', 'E1-1', 'Validation');
```

ApplicationName—The name of an existing application. Use `#Shared` to update a member in a dimension in the Shared Library.

DimensionName—The name of an existing dimension.

ParentName—Name of the Parent of the member to update.

MemberName—Name of the member to update.

PropertyName—The name of the property to update. (You can list as many properties as desired, however, you must include a valid value for the property.)

Dimension Association

Activates all of the standard dimension associations based on the application type and what dimensions are in the application.

```
Update Dimensionassociation
Properties(activateallforapplication) Values('true');
```

ApplicationName—The name of an existing application. You cannot update property values on `#Shared`.

ActivateAllForApplication—Indicates if all associations should be activated for the specified application. Allowed values:

- True
- False