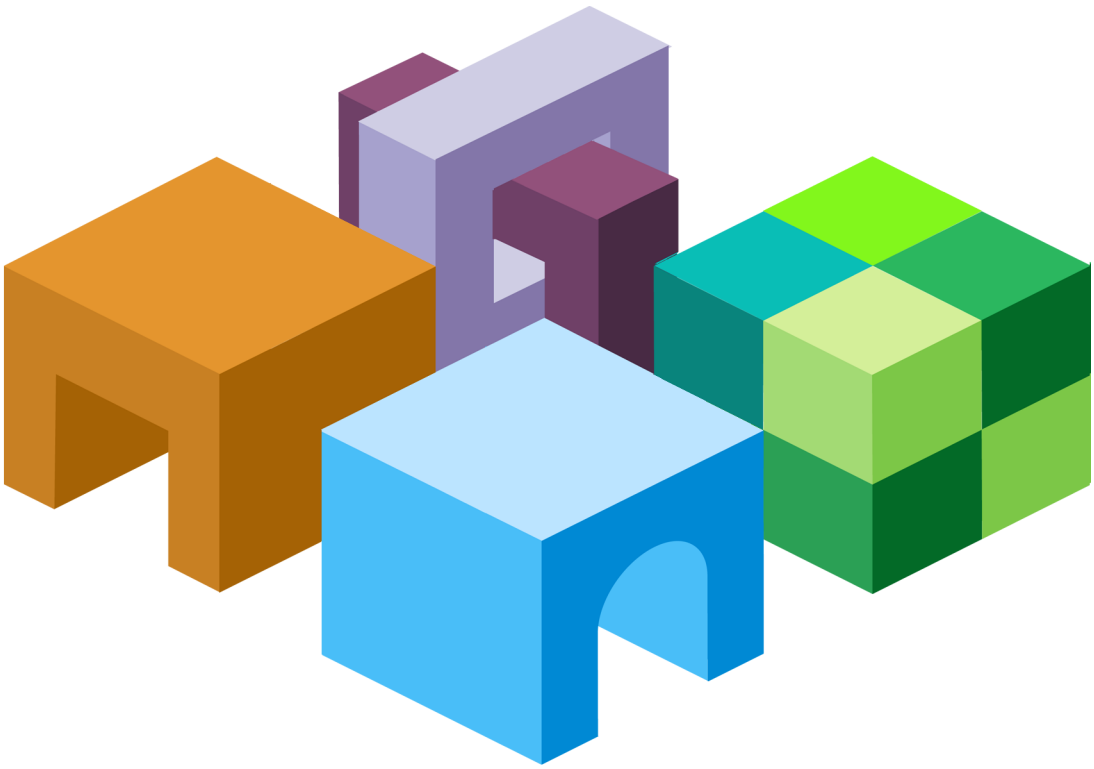

Oracle(R) Hyperion Enterprise Performance Management Architect, Fusion Edition

リリース 11.1.1.3

バッチ・クライアント・ユーザー・ガイド

ORACLE
ENTERPRISE PERFORMANCE MANAGEMENT SYSTEM

Performance Management Architect バッチ・クライアント・ユーザー・ガイド, 11.1.1.3

著者: EPM Information Development Team

目次

1 Performance Management Architectバッチ・クライアントの使用方法

この章の内容

Oracle Hyperion EPM Architect, Fusion Edition バッチ・クライアントによって、データのエクスポート、メタデータのロード、データのロード、計算などのプロセスを組み合せて、通常の夜間または週末のロード・プロセスの処理中に、これらの操作を開始できます。

バッチ・クライアントによる外部スケジューリング・ツールを使用してプロセスを開始できます。バッチ・クライアントを使用すると、次のような多くのタスクを実行できます:

- Performance Management Architect にメタデータをロードする
- 次元およびメジャーのセキュリティに関するプロパティを更新する
- アプリケーションにデータをロードする

バッチ・クライアントは、Performance Management Architect をインストールするときに自動的にインストールされます。バッチ・クライアントは、Windows プラットフォームで実行できます。Performance Management Architect をインストールすると、インストール中に生成されるクラス・パスを設定するためのバッチ・ファイルが自動的に作成されます。

Performance Management Architect バッチ・クライアントは、次の 2 つのモードで実行できます:

- コマンド・ライン・モード
  - コマンドを対話的に入力できる
  - 各コマンドは、複数の行にまたがることができる
  - 複数のコマンド・ステートメントをセミコロン(;)で区切ることができる
  - 単一引用符を使用する必要がある
  - コマンドは即座に実行される
- スクリプト・モード
  - 対話なしで一連のコマンドを実行する

- プログラムの起動時にコマンド・ファイルを指定でき、オプションで結果ログ・ファイルとトレース・ログ・ファイルを指定できる
- サードパーティのスケジューラを使用して、スクリプトの実行をスケジュールできる

バッチ・クライアントの起動

Performance Management Architect バッチ・クライアントは、対話型のコマンド・ライン・モードで実行するか、またはコマンド・ラインで指定するスクリプト・ファイルを実行できます。

➤ バッチ・クライアントを起動するには、スタート、プログラム、Oracle EPM System、Foundation Services、Performance Management Architect、EPMA バッチ・クライアントの起動の順に選択します。

`HYPERION_HOME¥products¥Foundation¥BPMA¥EPMABatchClient` に移動し、対話型コマンド・ラインまたはスクリプトを使用して `epma-batch-client.bat` ファイルを実行することもできます。

たとえば、対話型コマンド・ライン・モードでバッチ・クライアントを起動する場合は、次のようにパラメータなしでバッチ・ファイルを起動します:

```
HYPERION_HOME¥products¥Foundation¥BPMA¥EPMABatchClient¥epma-batch-client.bat
```

スクリプトを実行する場合は、スクリプト・ファイル名に-C オプションを付けて指定する必要があります。すべての追加パラメータは任意です。例:

```
HYPERION_HOME¥products¥Foundation¥BPMA¥EPMABatchClient¥epma-batch-client.bat -CMyScript.txt
```

注： スクリプト・ファイル名とログ・ファイル名は、相対パスで指定できますが、バッチ・ファイルを起動するフォルダではなく、`EPMABatchClient` フォルダに対する相対パスにする必要があります。

コマンド・ライン・オプション

バッチ・クライアントを起動するときは、次の引数を指定できます。

表 1　バッチ・クライアントのコマンド

コマンド	説明
-H	バッチ・クライアントのヘルプを表示します。
-C	実行するスクリプト・ファイルの名前を指定します。 例: `-C'C:¥Scripts¥LightsOut.txt'`
-G	使用する言語を指定します。このパラメータを使用するには、-Gx と入力します。ここで、x は、次の言語コードのいずれかです: ● DA - デンマーク語

コマンド	説明
	● DE - ドイツ語 ● ES - スペイン語 ● FR - フランス語 ● IT - イタリア語 ● JA - 日本語 ● KO - 韓国語 ● PT_BR - ブラジル・ポルトガル語 ● RU - ロシア語 ● SV - スウェーデン語 ● TR - トルコ語 ● ZH_CN - 簡体字中国語 ● ZH_TW - 繁体字中国語 たとえば、フランス語の場合は-Gfr、イタリア語の場合は-Git となります。 -G オプションが指定されていない場合、バッチ・クライアントはオペレーティング・システムの現在のデフォルト言語を使用しようとします。リソース・ファイルが見つからない場合は、デフォルト言語である英語が使用されます。特定の言語のリソース・ファイルで文字列が見つからない場合は、英語版が使用されます。
-T	出力トレース・ファイルを指定します。 例: `-T'C:¥LogFiles¥Trace.log'`
-R	マシンに結果を書き込むファイルの名前を指定します。 例: `-R'C:¥LogFiles¥ScriptResult.log'`
-L	コマンドをログ出力するかどうかを指定します。デフォルト値は 0 です。 ログのコマンドは次のとおりです: Off (デフォルト) =コマンドをログ出力しない On =コマンドをログ出力する 例: `-LOn`
-S	コマンドが失敗すると、スクリプトの実行を停止します。デフォルト値は 1(TRUE)です。 エラーが発生した場合は停止する: `TRUE` エラーが発生しても実行を継続する: `FALSE` 例: `-SFalse` (エラーが発生しても実行を継続します。)
-U	Performance Management Architect にログインするために使用するユーザー名を指定します。 例: `-U'Admin'`
-P	Performance Management Architect にログインするために使用するパスワードを指定します。 例: `-Ppassword`
-V	スクリプト確認のオン/オフを切り換えます。検証をオンにすると、実行前にスクリプトの構文エラーがチェックされます。エラーがあると、スクリプトの実行は停止します。「Off」を指定すると、実行前に検証は行われません。「On」を指定すると、実

コマンド	説明
	行前にスクリプトが検証されます。デフォルト値「On」で、実行前にスクリプトが検証されます。たとえば、-VOff を指定すると、実行前にスクリプトは検証されません。
-0	検証のみです。これを指定すると、バッチ・クライアントはスクリプトを検証しますが、結果にかかわらず、スクリプトは実行されません。これは、スクリプトに構文エラーがあるかどうかをテストするために使用できます。

注： -S などの On/Off を使用するコマンド・ライン・パラメータでは、0 または 1、Y または N、True または False、On または Off を使用できます。たとえば、-S0、-SY、-SNo、-STrue、-SOff などは、すべて有効です。

次に、コマンド・ライン・オプションの使い方のコード例を示します。

```
epma-batch-client -H
epma-batch-client -C"C:\Hyperion\EPMA\Commands.txt" -T"C:\Hyperion\EPMA
\epma-batch-clientTraceFile.log “
         -R"C:\Hyperion\EPMA\ResultFile.txt"  -LOn -SFalse
         -Uadmin -Ppassword
```

戻りコード

バッチ・クライアントが終了するときは、スクリプトの終了状態に基づいて呼び出しプログラムに結果コードが戻されます。StopOnError の設定に基づいて、次の 2 つのシナリオがあります。

`StopOnError = False (-S0)`

`StopOnError` が FALSE の場合の戻りコードは、一般的な成否を表します。

0 =成功(エラーなし)

-1 =失敗(1 つ以上のエラーが発生)

StopOnError = TRUE (-S1)

`StopOnError` が TRUE の場合、エラーが発生するとバッチ・クライアントは終了し、次の表に基づくコードを戻します。デフォルトでは、StopOnError は TRUE に設定されています。

バッチ・クライアントは、成否に基づいて次のような結果コードを戻します。

コマンド	コマンド・コード	クラス	クラス・コード	戻りコード
正常終了	該当なし	該当なし	該当なし	0
一般エラー	該当なし	該当なし	該当なし	-1
検証エラー	該当なし	該当なし	該当なし	1

コマンド	コマンド・コード	クラス	クラス・コード	戻りコード
解析エラー	該当なし	該当なし	該当なし	100
コマンド・ライン・エラー	該当なし	該当なし	該当なし	4
Copy	15	Application	1	1501
Copy	15	次元	2	1502
Create	1	Application	1	101
Create	1	次元	2	102
Create	1	Member	3	103
Create	1	Association	10	110
Debug	21	該当なし	該当なし	2100
Delete	2	Application	1	201
Delete	2	次元	2	202
Delete	2	Member	3	203
Delete	2	Association	10	210
Detach	16	次元	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	該当なし	0	500
Include	6	次元	2	602
Include	6	Member	3	603
Insert	7	Member	3	703
Login	8	該当なし	0	800
Logout	9	該当なし	0	900

コマンド	コマンド・コード	クラス	クラス・コード	戻りコード
Move	19	Member	3	1903
Option	20	該当なし	0	2000
Quit	10	該当なし	0	1000
Remove	11	次元	2	1102
Remove	11	Member	3	1103
Rename	18	Member	3	1803
Set	12	該当なし	0	1200
Share	17	次元	2	1702
Update	13	Application	1	1301
Update	13	DimensionAssociation	8	1308
Update	13	次元	2	1302
Update	13	Member	3	1303
Variable	14	該当なし	0	1400

DOS のバッチ・ファイルまたは Windows のコマンド・ファイルでは、次のコマンドを実行して、エラーをチェックできます:

```
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
```

ロギング

バッチ・クライアントには、結果ファイルとトレース・ファイルを使用して出力できるいくつかのロギングのレベルがあります。結果ファイルには、コマンドおよびその実行ステータス、エラーまたは警告メッセージの詳細が含まれます。トレース・ファイルには、デバッグに役立つスタック・トレースの詳細が含まれます。

バッチ・クライアントは、ロギングに log4j を使用します。結果ファイルおよびトレース・ファイルの出力をフォーマットできます。たとえば、電子メールなどの新規種類のライターにメッセージを転送するように log4j を構成できます。すべての構成パラメータは、conf ディレクトリの BPMAPlusLog4j.properties ディレクトリに保管されます。(例: HYPERION_HOME¥products¥Foundation¥BPMA ¥EPMABatchClient¥output。)

次に、結果ファイルのコード例を示します。

```
log4j.appender.resultfile=org.apache.log4j.RollingFileAppender
log4j.appender.resultfile.File=${user.dir}/output/epma-batch-
```

```
clientResults.log
log4j.appender.resultfile.MaxFileSize=1024KB
log4j.appender.resultfile.MaxBackupIndex=5
log4j.appender.resultfile.layout=org.apache.log4j.PatternLayout
log4j.appender.resultfile.layout.ConversionPattern=%d EPMABatch: %m%n
```

コマンドが正常終了したかどうかを簡単に判断できるように、次のコマンドについて、関連するジョブ ID およびジョブ ID URL がログに記録されます。

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

2 コマンド・ファイルの構成

この章の内容

コマンド・ファイルは、バッチ・クライアントの入力ファイルで、次の 1 つ以上を含められます:

- コマンド
- 変数、宣言および割当て
- コメント

次に、コマンド・ファイルのコード例を示します。

```
// 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, RemoveChildren)
    Values('M1-1-1', 'A1', 'M1-1', true);

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

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

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

Delete Dimension
    Properties(DimensionName)
```

```
    Values('E1');

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


set ApplicationName = '';

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

quit;
```

コマンド・ファイルのほとんどのコマンドは即座に実行されます。ただし、EXECUTE コマンドは実行に時間がかかり、WaitForCompletion パラメータをサポートします。DIMSYNCRONIZATION を除くすべて実行コマンドはこのパラメータをサポートします。たとえば、WaitForCompletion パラメータを使用して、バッチ・クライアントにコマンドが実行されるまで待機させることもできます。次のコマンドに WaitForCompletion パラメータの例を示します。この場合、管理者は実行時間が比較的長いデータ同期コマンドを実行しています。実行に長い時間を要するその他のコマンドとしては、インポートおよびアプリケーションの配置があります。

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

変数

変数は、定義してスクリプトで呼び出すことができます。変数の特性は次のとおりです:

- 変数には、名前と単一の値のタイプがある
- 変数の値は、任意のデータ型にできる
- 一度定義した変数を複数の場所で使用できる
- コマンド間で変数の値を変更できる
- 変数名には大文字と小文字の区別がある
- var キーワードを使用して変数を定義し、$を使用して変数を参照できる

次に、変数を使用して共有次元を作成するコード例を示します。

```
// 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');
```

次の 3 つのタイプの変数があります:

- ユーザー

  ユーザー変数は、var キーワードを使用して割り当てます。次に例を示します:

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

- システム

  システム変数は、set コマンドを使用して割り当てます。次に例を示します:

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

- オブジェクト

  オブジェクト変数は、set コマンドを使用して割り当てます。次に例を示します:

  ```
  set ApplicationName = 'Comma';

  set dimension=Account;
  ```

コメント

行の先頭に 2 つのスラッシュ(//)を記述すると、スクリプト内の任意の行をコメントにできます。複数の行にわたってコメントにする場合は、各行をコメントにする必要があります。次に例を示します:

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

コマンド

コマンドの構成要素は次のとおりです:

- コマンド動詞
- コマンド・クラス
- パラメータ値の集合またはプロパティ値の集合

パラメータおよびその値は、カンマによって区切ります。サポートされているコマンド動詞は、次のとおりです:

- CREATE
- COPY

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

サポートされているコマンド・クラスは、次のとおりです:

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

ヒント： 次元の root メンバーには、`#root` 定数を使用します。共有ライブラリを宛先にするコマンドのアプリケーション名には、`#shared` 定数を使用します。

スクリプト

スクリプトは、連続して実行できるコマンドの集合です。スクリプトには、任意の順序でコマンドを記述できますが、login のような特定の初期化コマンドは、他のコマンドより前に実行する必要があります。スクリプト内の各コマンドはセミコロン(;)によって区切られます。コマンドには空白文字を含め、複数の行にまたがれます。

一般的に使用されるコマンド

次の項では、一般的に使用されるコマンドの例を示します。

オプション・コマンド

Option コマンドを使用すると、スクリプトの実行中にコマンド・ライン・オプションを動的に変更できます。変更可能なオプションは次のとおりです:

```
StopOnError
option StopOnError = true;

EchoComments
 option EchoComments = true;

LogCommands
 option LogCommands = true;
```

ログイン/ログアウトのコマンド

Login

Performance Management Architect にログインします。

スクリプトを使用してログインする場合は、次のように入力します:

```
Login admin,password;
```

コマンド・ラインを使用してログインする場合は、次のように入力します:

```
Login;
```

Logout

Performance Management Architect からログアウトします。

```
Logout;
```

Quit

バッチ・クライアントを閉じます。

```
Quit;
```

Exit

バッチ・クライアントを閉じます。

```
Exit;
```

コピー・コマンド

アプリケーション

Copy Application コマンドは、アプリケーション・ライブラリで使用可能な"Duplicate As New"コマンドと同じです。

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

ApplicationName - 既存のアプリケーションの名前です。

CopyApplicationToName - 新規複製されたアプリケーションの名前です。

ApplicationDescription - アプリケーションの説明です。

ApplicationType - アプリケーションのタイプです。有効な値は次のとおりです: 汎用、連結、収益性、Enterprise Analytics または Essbase Analytics。

次元

共有ライブラリ内、アプリケーション内または共有ライブラリとアプリケーションの間で次元をコピーする方法を提供します。あるアプリケーションから別のアプリケーションに次元を直接コピーできません。

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

ApplicationName - 既存のアプリケーションの名前です。

DimensionName - 既存の次元の名前です。

TargetDimensionName - ターゲット次元の名前です。

TargetDimensionDescription - ターゲット次元の説明です。

destApplicationName - 宛先アプリケーションの名前です。

作成コマンド

アプリケーション

指定された名前で空のアプリケーションを新規作成します。

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

ApplicationName - アプリケーションの有効な名前を含む文字列です。

ApplicationDescription - アプリケーションの有効な名前を含む文字列です。

ApplicationType - 次の値をサポートします:

- 汎用
- 連結
- Planning
- Enterprise Analytics
- Essbase Analytics
- 収益性

次元

指定されたアプリケーションまたは共有ライブラリに空の次元を新規作成します。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリに次元を作成する場合は、`#Shared` を使用します。

DimensionName - 次元の有効な名前です。

DimensionType - 次元タイプは、次のいずれかを使用できます:

- 勘定科目
- 別名
- AllocationType
- 属性
- ConsolidationMethod
- 国
- 通貨
- エンティティ
- 汎用
- ICP

- メジャー
- シナリオ
- SecurityClass
- スマート・リスト
- 時間
- UDA
- 値
- バージョン
- 表示
- 年

メンバー

指定された次元に新規メンバーを作成します。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリに次元を作成する場合は、`#Shared` を使用します。

DimensionName - 既存の次元の名前です。

ParentName - 新たに作成したメンバーを挿入する親の名前です。ツリーのトップ・レベルに新規メンバーを追加する場合は、`#Root` を使用します。

MemberName - 新規メンバーの有効な名前です。

MemberDescription - 新規メンバーの説明です。

関連付けの作成コマンド

2 つの次元の間に関連付けを作成します。基本次元を共有する場合は、関連次元を共有次元にする必要があります。

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

ApplicationName - 既存のアプリケーションの名前です。

DimensionName - 既存の次元の名前です。

AssociatedDimensionName - 関連付ける次元の名前です。

PropertyName - 更新するプロパティの名前です。必要なプロパティをいくつでも記載できますが、プロパティには有効な値を指定する必要があります。

PropertyDescription - コメントまたはプロパティの説明を入力できるオプションのパラメータ。

関連付けの削除コマンド

既存の次元の関連付けを削除します。

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

ApplicationName - 既存のアプリケーションの名前です。

DimensionName - 既存の次元の名前です。

PropertyName - 更新するプロパティの名前です。必要なプロパティをいくつでも記載できますが、プロパティには有効な値を指定する必要があります。

削除コマンド

アプリケーション

指定されたアプリケーションを削除します。

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

ApplicationName - 既存のアプリケーションの名前です。

関連付け

既存の次元の関連付けを削除します。

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

ApplicationName - 既存のアプリケーションの名前です。

DimensionName - 既存の次元の名前です。

PropertyName - 更新するプロパティの名前です。必要なプロパティをいくつでも記載できますが、プロパティには有効な値を指定する必要があります。

次元

指定された次元を削除します。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元を削除する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

メンバー

指定されたメンバーを削除します。メンバーのすべての子を削除することもできます。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを削除する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

ParentName - メンバーを削除する親の名前です。ツリーのトップ・レベルのメンバーを削除する場合は、#Root を使用します。

MemberName - 削除するメンバーの名前です。

RemoveChildren - また、削除されるメンバーの下の子も削除するかどうかを指定します。指定できる値は次のとおりです:

- TRUE
- FALSE

次元の添付解除コマンド

アプリケーション内の共有次元をローカル次元に変換します。

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

ApplicationName - 既存のアプリケーションの名前です。

DimensionName - 既存の次元の名前です。

RetainFilteredStructure - TRUE に設定すると、次元の現在のビューが維持され、除外されたメンバーまたは他のフィルタ・メンバーは次元のローカル・コピーに表示されません。FALSE に設定すると、次元のすべてのメンバーが次元のローカル・コピーに表示されます。指定できる値は次のとおりです:

- TRUE
- FALSE

RetainPropertyOverrides - TRUE に設定すると、すべてのプロパティの上書きが保持されます。それ以外の場合は、次元の共有バージョンからの値が使用されます。指定できる値は次のとおりです:

- TRUE
- FALSE

WaitForCompletion - TRUE 値を指定すると、バッチ・クライアントはジョブが終了するまで待ちます。FALSE 値を指定すると、単にジョブを送信して処理を継続します。指定できる値は次のとおりです:

- TRUE
- FALSE

除外コマンド

メンバー

共有次元からメンバーを除外します。

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

ApplicationName - 既存のアプリケーションの名前です。次元を含めるための目標として#Shared を使用できません。

DimensionName - 共有ライブラリ内の既存の次元の名前で、インクルードする次元です。

ParentName - 除外するメンバーの親の名前です。

MemberName - 除外するメンバーの名前です。

実行コマンド

EXECUTE コマンドを使用して、ジョブを実行できます。

(アプリケーションまたは共有ライブラリへの)インポート

既存のインポート・プロファイルを実行します。

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

ImportType - 実行するインポートの種類です。指定できる値は次のとおりです:

- FlatFile
- InterfaceTables

ProfileName - 既存のインポート・プロファイルの名前です。

FileName - フラット・ファイルのタイプをインポートする場合のインポートするフラット・ファイルの名前です。

WaitForCompletion - TRUE 値を指定すると、バッチ・クライアントはジョブが終了するまで待ちます。FALSE 値を指定すると、単にジョブを送信して処理を継続します。指定できる値は次のとおりです:

- TRUE
- FALSE

共有ライブラリとの次元の同期

共有ライブラリと次元の同期をやり取りします。

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

ApplicationName - 既存のアプリケーションの名前です。次元の同期のアプリケーション名として`#Shared` を使用できません。

DimensionName - アプリケーションの既存の次元の名前です。

SharedDimensionName - 共有ライブラリ内の既存の次元の名前です。

SyncToApp - TRUE 値を指定すると、共有次元はアプリケーションと同期され、FALSE 値を指定すると、アプリケーションが共有ライブラリに同期されます。指定できる値は次のとおりです:

- TRUE
- FALSE

ReplaceMode - TRUE 値を指定すると、次元の同期は置換モードを使用し、FALSE 値を指定すると、マージ・モードを使用します。指定できる値は次のとおりです:

- TRUE
- FALSE

配置

アプリケーションを指定された製品に配置します。

```
Execute Deploy
Parameters(ApplicationName, InstanceName, ApplicationServer,
HubProject, ClearAll, CheckIntegrity, waitforcompletion,
purgeTransactions,  deployOption, escapeValidateRules)
Values('AppName', 'HFM931', 'AppServer', 'HubProj', 'false', 'false',
'true', 'true', ‘AppView’, 'true');
```

ApplicationName - 既存のアプリケーションの名前です。配置のアプリケーション名として`#Shared` を使用できません。

InstanceName - 配置先のインスタンスの名前です。

ApplicationServer - 配置先のアプリケーション・サーバーの名前です。

HubProject - 配置済アプリケーションを追加する Oracle の Hyperion(R) Shared Services プロジェクトです。

PurgeTransactions - トランザクション履歴を削除します。TRUE 値を指定すると履歴が削除され、FALSE 値を指定すると削除されません。指定できる値は次のとおりです:

- TRUE
- FALSE

WaitForCompletion - TRUE 値を指定すると、バッチ・クライアントはジョブが終了するまで待ちます。FALSE 値を指定すると、単にジョブを送信して処理を継続します。指定できる値は次のとおりです:

- TRUE
- FALSE

Planning アプリケーション向け:

datasourceName - 指定した名前でデータ・ソースを作成します。この値は文字列です。

CreateOutline - 初めてアプリケーションを配置する場合は、Essbase アウトラインを作成します。指定できる値は次のとおりです:

- TRUE
- FALSE

RefreshOutline - アプリケーションの構造を変更した後にアプリケーション・データベースをリフレッシュします。指定できる値は次のとおりです:

- TRUE
- FALSE

CreateSecurityFilters - 暗号化されたデータ・ファイルにアクセス権を保管します (`Essbase.sec`)。指定できる値は次のとおりです:

- TRUE
- FALSE

SharedMembersSecurityFilters - アクセス権を共有メンバーに適用します。指定できる値は次のとおりです:

- TRUE
- FALSE

ValidateSecurityFilterLimit - 1 行当たり 64KB の Oracle Essbase セキュリティ・フィルタの限界を超えるセキュリティ・フィルタを識別します。これによって、Oracle Essbase のセキュリティ・フィルタを構築する前に、フィルタ・サイズを検証してサイズの限界を超えないようにできます。指定できる値は次のとおりです:

- TRUE
- FALSE

再配置

アプリケーションを製品サーバーに再配置します。

```
Execute Redeploy
Parameters(ApplicationName, HubProject, ClearAll, CheckIntegrity,
waitforcompletion, purgeTransactions, deployOption, escapeValidateRules)
Values('AppName', 'HubProj', 'false', 'false', 'true', 'true',
‘AppView’, 'true');
```

有効値については、24 ページの「配置」の例を参照してください。

データの同期

既存のデータ同期プロファイルを実行します。

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

DataSynchronizationName - 実行するデータ同期プロファイルの名前です。

DataTransformationOperator - 指定できる値は次のとおりです:

- なし
- '*' (乗算)
- '/' (除算)
- '+' (加算)
- '-' (減算)

DataTransformationValue - データ値を変更するために DataTransformationOperator と連携して使用する値です。

FileName - データ・ソースをポイントするファイル URL です。

UploadFile - データ・ソースとして使用するアップロード・ファイルです。

ValidateOnly - データの同期を実行せずに検証します。

WaitForCompletion - TRUE 値を指定すると、バッチ・クライアントはジョブが終了するまで待ちます。FALSE 値を指定すると、単にジョブを送信して処理を継続します。指定できる値は次のとおりです:

- TRUE
- FALSE

検証

アプリケーションの検証を実行します。StopOnError オプションが TRUE に設定されているかぎり、スクリプトは検証が失敗した場合に終了します。

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

ApplicationName - 既存のアプリケーションの名前です。

ValidateType - 実行する検証のタイプです。たとえば、アプリケーションのみの検証、ビジネス・ルールの検証またはすべて(アプリケーションおよびルール)の検証があります。指定できる値は次のとおりです:

- AppView
- Rules
- All

インクルード・コマンド

次元

共有ライブラリから指定したアプリケーションに既存の次元を追加します。次元は共有次元として追加することも、ローカル次元としてアプリケーションにコピーすることもできます。

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

ApplicationName - 既存のアプリケーションの名前です。次元を含めるための目標として#Shared を使用できません。

DimensionName - 共有ライブラリ内の既存の次元の名前で、インクルードする次元です。

IncludeAsShared - 共有ライブラリのソース次元とのリンクを維持する共有次元としてインクルードする場合は TRUE 値を使用します。共有ライブラリの次元とは別に次元のコピーをインクルードする場合は FALSE 値を使用します。指定できる値は次のとおりです:

- TRUE
- FALSE

挿入コマンド

メンバーの挿入

メンバーのコピーを共有メンバーとして挿入します。Insert Member コマンドを使用できるのは、ローカル次元および共有ライブラリ次元のみです。アプリケーション内の共有次元ではメンバーを挿入できません。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを操作する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

ParentName - メンバーの挿入先の親の名前です。

InsertMemberName - メンバーの挿入先のメンバーの名前です。

MemberToInsertName - 挿入する共有メンバーの名前です。

移動コマンド

メンバーの移動

次元構造内のある場所から別の場所にメンバーを移動します。Move Member コマンドを使用できるのは、ローカル次元および共有ライブラリ次元のみです。アプリケーション内の共有次元のメンバーは移動できません。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを操作する場合は、`#Shared` を使用します。

DimensionName - 既存の次元の名前です。

FromParentName - 移動するメンバーの親の名前です。

MemberName - 移動するメンバーの名前です。

ToParentName - メンバーの移動先の親の名前です。

InsertAfterMember - メンバーの挿入先となる ToParentName に属する子を示します。メンバーを子として挿入する必要があることを示す特別な値'#none'を設定できます。これにより、挿入されるメンバーとその後のすべてのメンバーのソート順が影響を受けます。

名前変更コマンド

メンバー名の変更

メンバーおよびそのメンバーのすべての共有コピーを名前変更します。Rename Member コマンドを使用できるのは、ローカル次元および共有ライブラリ次元のみです。アプリケーション内の共有次元のメンバーは名前変更できません。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを操作する場合は、`#Shared` を使用します。

DimensionName - 既存の次元の名前です。

ParentName - 名前変更するメンバーの親の名前です。

MemberName - 名前変更する既存のメンバーの名前です。

NewMemberName - メンバーの新しい名前です。

除去コマンド

メンバーの除去(アプリケーションまたは共有ライブラリ)

指定した次元からメンバーを除去しますが削除はしません。Remove Member コマンドを使用できるのは、アプリケーション内の共有次元を除去する場合のみです。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを操作する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

ParentName - 除去するメンバーの親の名前です。

MemberName - 除去するメンバーの名前です。

次元の除去

アプリケーションから共有次元を除去します。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを操作する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

Force - TRUE に設定した場合、除去対象の次元がアプリケーション内の他の次元に関連付けられていても除去されます。FALSE に設定した場合、除去対象の次元がアプリケーション内の他の次元に関連付けられていると次元を除去できません。指定できる値は次のとおりです:

- TRUE
- FALSE

次元の共有コマンド

ローカル次元を新しい共有次元に変換するか、既存の次元とマージします。

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

ApplicationName - 既存のアプリケーションの名前です。

DimensionName - 既存の次元の名前です。

ShareAsNew - TRUE に設定すると、SharedDimensionName プロパティおよび MergeAsShared プロパティが無視されます。これは、次元を新規として共有する場合に、これらのプロパティが適用されないためです。指定できる値は次のとおりです:

- TRUE
- FALSE

SharedDimensionName - FALSE に設定する場合は、共有ライブラリで共有する次元の名前を指定する必要があります。指定できる値は次のとおりです:

- TRUE
- FALSE

MergeAsShared - TRUE に設定すると、共有対象の次元がターゲット次元とマージされます。FALSE に設定すると、共有対象の次元によってターゲット次元が置換されます。指定できる値は次のとおりです:

- TRUE
- FALSE

WaitForCompletion - TRUE 値を指定すると、バッチ・クライアントはジョブが終了するまで待ちます。FALSE 値を指定すると、単にジョブを送信して処理を継続します。指定できる値は次のとおりです:

- TRUE
- FALSE

更新コマンド

UPDATE スクリプト・コマンドを使用して、アプリケーション、次元またはメンバーのプロパティ値を変更する場合は、Performance Management Architect のプロパティ・グリッドに表示されるプロパティ・ラベルではなくプロパティ名を使用する必要があります。プロパティ・ラベルとプロパティ名については、『Oracle Hyperion Enterprise Performance Management Architect 管理者ガイド』の付録に説明があります。次に、スクリプトの例を示します:

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

この例では、DimensionName、ParentName および MemberName は、すべて標準のスクリプト・アイテムですが、`AggregationWeight` と `NumDecimalPlaces` は、Oracle Hyperion EPM Architect, Fusion Edition のメンバー・レベルのプロパティです。

アプリケーション

指定されたアプリケーションの 1 つ以上のプロパティを更新します。

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

ApplicationName - 既存のアプリケーションの名前です。#Shared のプロパティ値は更新できません。

PropertyName - 更新するプロパティの名前です。必要なプロパティをいくつでも記載できますが、プロパティには有効な値を指定する必要があります。

次元

指定された次元の 1 つ以上のプロパティを更新します。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元を更新する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

PropertyName - 更新するプロパティの名前です。必要なプロパティをいくつでも記載できますが、プロパティには有効な値を指定する必要があります。

メンバー

指定されたメンバーの 1 つ以上のプロパティを更新します。

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

ApplicationName - 既存のアプリケーションの名前です。共有ライブラリの次元メンバーを更新する場合は、#Shared を使用します。

DimensionName - 既存の次元の名前です。

ParentName - 更新するメンバーの親の名前です。

MemberName - 更新するメンバーの名前です。

PropertyName - 更新するプロパティの名前です。必要なプロパティをいくつでも記載できますが、プロパティには有効な値を指定する必要があります。

次元の関連付け

アプリケーションのタイプおよびアプリケーションに含まれる次元に基づいたすべての標準次元の関連付けをアクティブ化します。

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

ApplicationName - 既存のアプリケーションの名前です。`#Shared` のプロパティ値は更新できません。

ActivateAllForApplication - 指定したアプリケーションに対してすべての関連付けをアクティブ化するかどうかを指定します。指定できる値は次のとおりです:

- TRUE
- FALSE

索引

記号

A - Z

あ行

か行

さ行

た行

は行

ま行

ら行