Hyperion Essbase Basics, Coding Standards And Best Practices

Amit Sharma
Hyperion/OBIEE Trainer/Consultant
BISP
Learnhyperion.wordpress.com
aloo_a2@yahoo.com

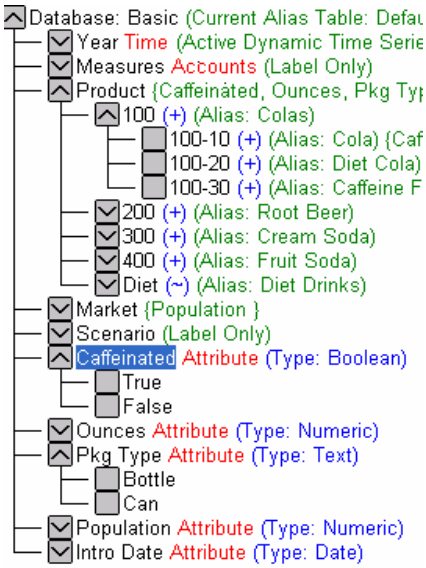
People often ask me what standard one should follow while writing script or creating Essbase artifacts so I consolidated my understanding and documented the standards I follow. This document will help developers and Administrator to follow the standards while executing their day to day activities. The basic idea behind this document is to facilitate the use of Best Practice.

First I would like to explain all the definition, abbreviations and artifacts used in Essbase.

Essbase Objects: Any Essbase file (regardless of how it is created or where it is stored) with the following extensions: .otl, .csc, .rul, .rpt, .pag, .ind, .esm, .tct, .db, .dbb, .ddb, .app, .apb

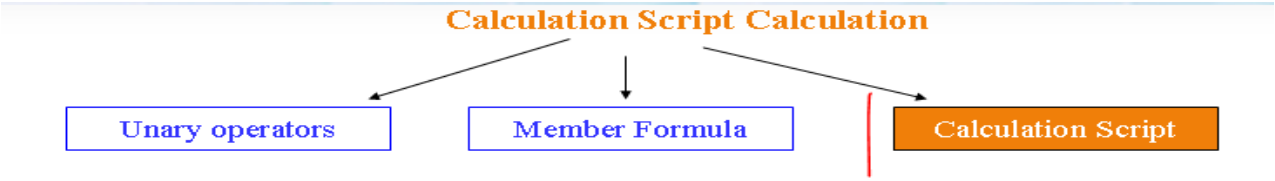
Essbase Outline: including its hierarchies, member formulas, and child to parent consolidations (data rollups).

The dbname.otl file. This information in this file defines the number and properties of all the dimensions in the database. It also indicates the number of members and the members' properties in each dimension. The Outline may also contain business rules for calculations and aggregations. The Outline may only be modified through the Essbase administrative interface (Application Manager), via an Essbase Load Rule, or through the API.



Application Manager (App. Manager or AppMan): The Essbase Administrative interface. Consists of four major components: the Outline Editor, Load Rule Editor, Calc Script Editor, and the Report Script Editor. Additional menu items allow Server, Application, database, and Security Settings to be changed.

Calculation Script:



Calculation script calculation is the anther method of calculation. Using a calculation script, you can choose exactly how to calculate a database

```
IF(EMP_Salary>12000)
        Income_Tax=EMP_Salary*0.03;
EndIF;
```

Calc Script Created to perform more complicated/efficient calculations than the Default. While this object may be created or edited with any text editor, Application Manager provides syntax help and error checking.

Load Rule: Raw data from an ASCII text file, Microsoft Excel, or from an RDBMS database will be loaded into your Essbase cube using Essbase rules file objects. Essbase objects with the .rul extension. These files allow data sourced from either text files or a SQL source to be mapped to populate the appropriate cells in the Essbase database. They are also used to map metadata into the proper hierarchy in the dimensions.

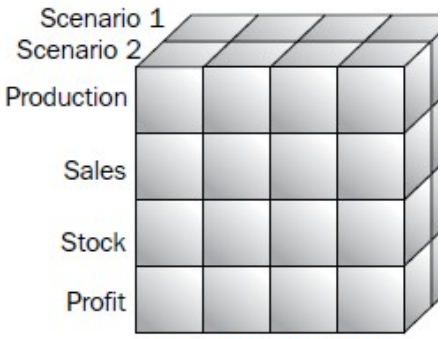
Report Script: Essbase objects with .rpt extensions. These are queries, which extract the desired data from an Essbase database to a flat file, printer, or the screen.

Essbase Agent: The Essbase agent is what you install on your server in order to make it an Essbase analytic server. The Essbase agent is what you use to build, deploy, and maintain your Essbase databases. The executable (essbase.exe) that manages the Essbase OLAP Server. The Agent validates all log-ins, starts and stops Applications, and directs requests for data to the proper Application/database.

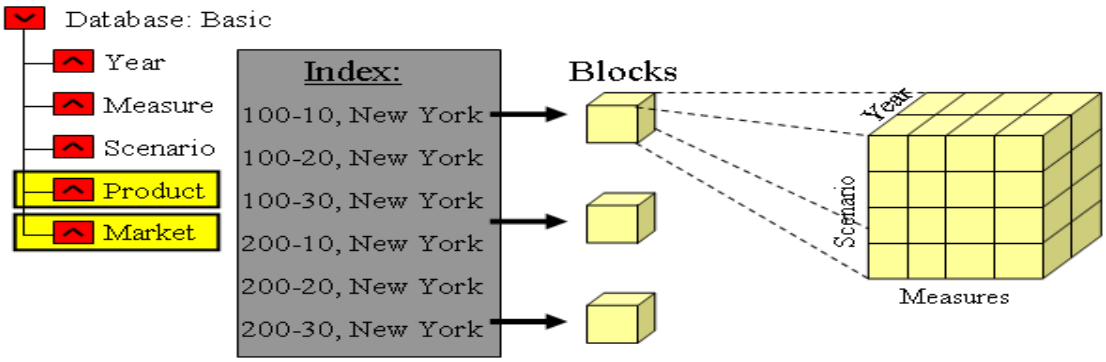
Application: The "package" or "container" that houses Essbase databases. An Essbase database must be created in the context of an Application. The Application has its own directory structure and the database is in a sub-directory. Each Application that has been started launches its own instance of an executable called essvr.exe. This executable allocates memory and other resources for that Application to use to process its database(s).

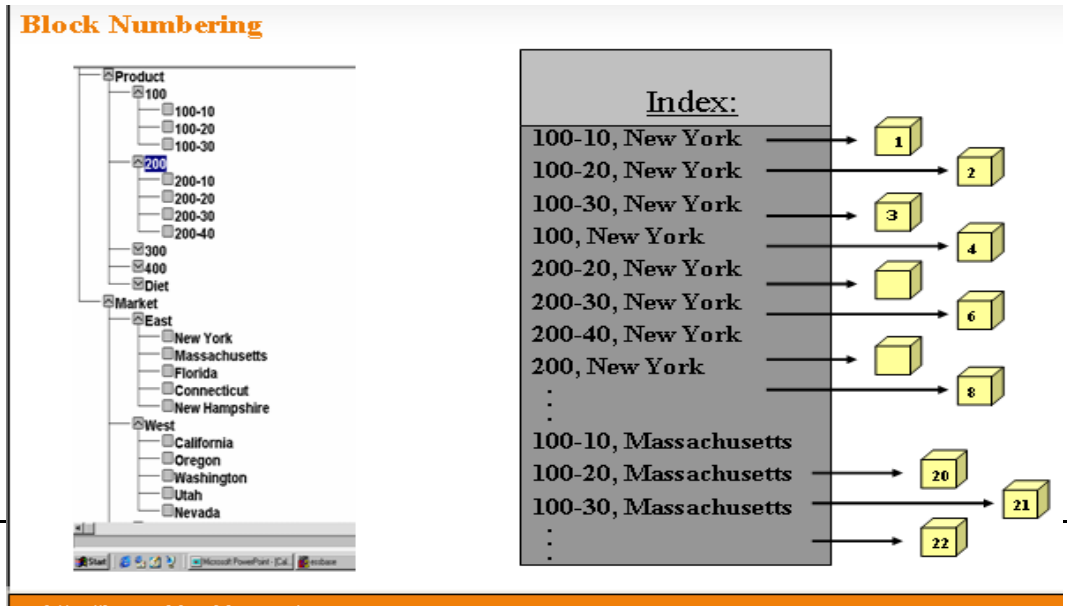
Database: The “package” or “container” that houses Essbase databases. An Essbase database must be created in the context of an Application. The Application has its own directory structure and the database is in a sub-directory. Each Application that has been started launches its own instance of an executable called essvr.exe. This executable allocates memory and other resources for that Application to use to process its database(s).

Datablock: These data blocks are the building blocks of the Essbase cube: These data blocks are the building blocks of the Essbase cube The fundamental unit of storage in Essbase. A multidimensional array, which creates an 8-byte cell for every possible combination of stored members from every Dense dimension in the Outline. Datablocks are stored in the .pag file(s).



Index Entry: Also known as an Index page or simply Index. It is an entry in the .ind file(s) created by every combination of stored members from the Sparse dimensions that actually has been populated. It references the data stored in a single Datablock.





Cube: It can also be defined as the capability of manipulating and analyzing data from multiple perspectives See 'database'. The term cube is mostly used by people whom are absolutely ignorant of how Essbase actually works and will not be mentioned again in the remainder of this document. Relational databases are not well suited for near instantaneous analysis and display of large amounts of data.

ARBORPATH: The path to where Essbase was installed. There is an ARBORPATH on clients as well as the Essbase Servers. The current ARBORPATH on the CRM Essbase Servers is: '/opt/hyperion/essbase'. On a workstation it is usually either 'C:\hyperion\essbase' or 'C:\Program Files\hyperion\essbase.

Skeleton Application: A shell of the completed Application. It should contain a database with the same name as the completed Application. That database's Outline will contain all "hard-coded" member and dimension information. It will also contain all other objects (Calc Scripts, Load Rules, Report Scripts, et al) necessary to process and create the completed Application.

Staging Application: A copy of the Skeleton Application. This is where all Dimension Builds, Data Loads, and Calculations take place. When that processing is done, the Staging App. is copied or re-named to become the new completed, or Production Application for a given time period.

Object Naming Standards:

Application	8 Characters Maximum	May be mixed case. No spaces.	Actual text should be what the Users prefer.
Database	8 Characters Maximum	May be mixed case. No spaces.	Same as Application name or per Users' preferences.
Calc scripts	8 Characters Maximum	All lower case to reduce potential errors when referenced by a script. No spaces.	
Load Rules	8 Characters Maximum	All lower case to reduce potential errors when referenced by a script. No spaces.	If processing a text file, the Load Rule should contain some portion of that text file's name (text file may be longer than 8 characters).
Report Scripts	8 Characters Maximum if located in ARBORPATH	All lower case to reduce potential errors when referenced by a script. No spaces.	
Outlines	By definition will be the same as the Database name.		
ESSCMD Scripts		May be mixed case. No spaces.	Must have an extension of '.scr'.
MaxL Scripts		May be mixed case. No spaces.	Must have an extension of '.msh'.
Skeleton Applications	8 Characters Maximum	May be mixed case. No spaces.	Please begin the name with the characters 'Nw', followed by a 3 to 4 letter lower case abbreviation for the name of the completed Application, followed by 'sk' or 'skl'.

Outline Standards

Contrary to popular belief, comments can be used within the Outline Editor. Comments are required wherever:

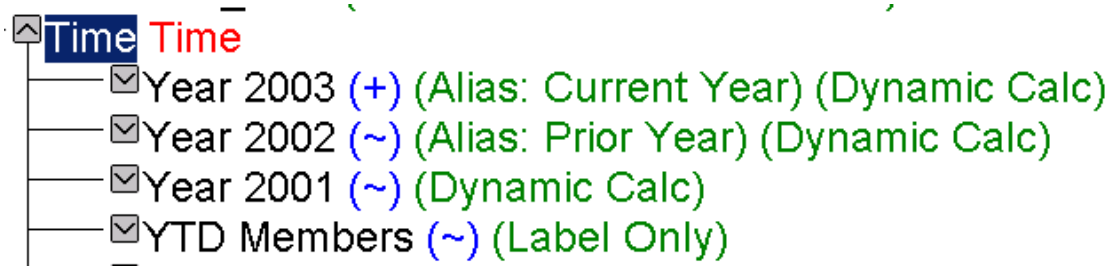
- A potential issue could be created during maintenance unless a certain procedure is followed.
- A Member is calculated by a formula in a Calc Script, not in the Outline.
- The Design requires settings that would appear "wrong" to another Essbase Programmer at first glance.

Creating comments in the Outline Editor:

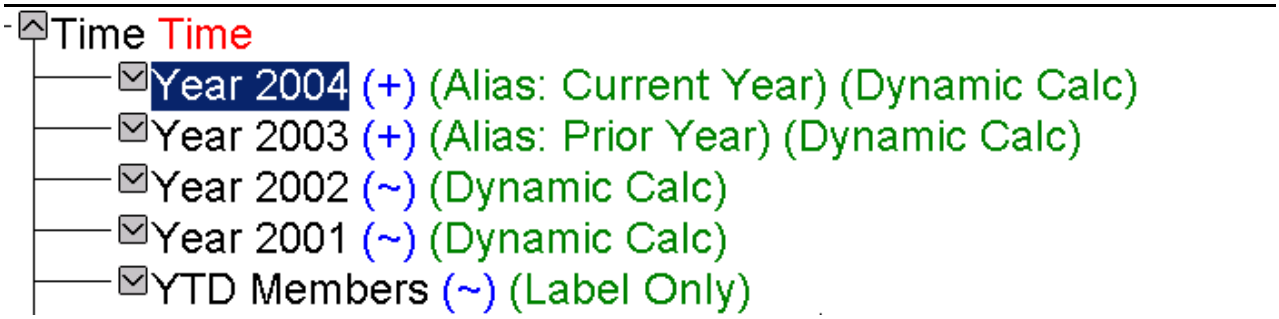
1. Highlight the Member that you wish to attach the comment to.
2. Open the Member/Dimension Properties Dialog Box.
3. Type a concise, brief comment into the 'Comments' field.
4. Essbase will automatically wrap the text within its comment block symbols ('/*' and '*/)

Example of the proper use of comments in the Outline:

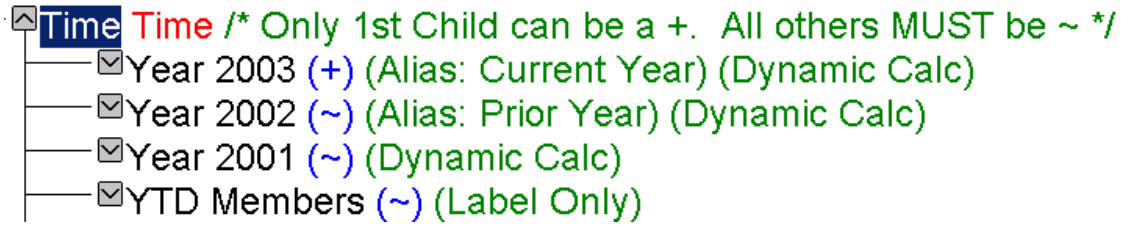
In the example below, Time does not store data. Because only one of its Children has a consolidation symbol (the '+' on 'Year 2003') that results in data being aggregated to it, Essbase has created an "Implied Share". Time is merely a pointer to display 'Year 2003' data when Time is queried. When it becomes necessary to add a new year to the outline, an inexperienced person performing the maintenance could break this Implied Share.



Note that the default consolidation symbol is a '+'. This means that when 'Year 2004' was added, Essbase will now attempt to sum all data for both years together and store it in Time. The two years are dynamic calc's, but Time is not. Time is calculated in Batch. It is no longer an Implies Share because more than one of its Children are aggregating their data to it. Making a stored Member, such as Time in this example, depend upon dynamically calculated Members will either result in terribly long run times or (as in this case) prevent the calculation from even running.



These types of potential problems can be mitigated with some type of simple comments similar to below:



If the person performing the maintenance does not understand the comment, at least it should motivate them to seek out a more experienced Essbase Developer who does. The comments do not show up in the Users' reports. The Member Selection tool built into the Spread Sheet Add-in, does allow a User to view a Member's comments if they choose to. Placing informative comments on key Members of the Outline can help prevent errors, and provides a more accessible place to reference vital information than design documents (which may not exist at all for legacy applications).

Implies Shares

Though the functionality of Implied Shares in Essbase is certainly well intentioned, they can create numerous problems and should be avoided where possible and practical in favor of using Label Only Members.

Conditions where Essbase creates Implied Shares:

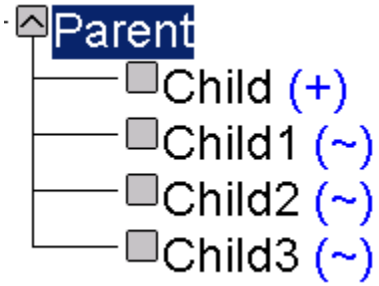
1. Where a Member has only one Child that will aggregate data to it.
2. Where a Label Only Member displays the data from its first Child.

Examples of Condition 1:

If a Member only has one Child, that Parent will always be an Implied Share. Even though it looks like data from Child will added to Parent and stored there, Essbase looks for opportunities like this to keep the database as small as possible. It will realize that Parent will ALWAYS equal Child, so Parent is

implicitly converted to a Shared Member, displaying Child's data. No data is actually stored in Parent:

Even if a Member has multiple Children, if only one of them actually aggregates to the Parent, that Parent will still become an Implied Share, as in this example:



Example of Condition 2:

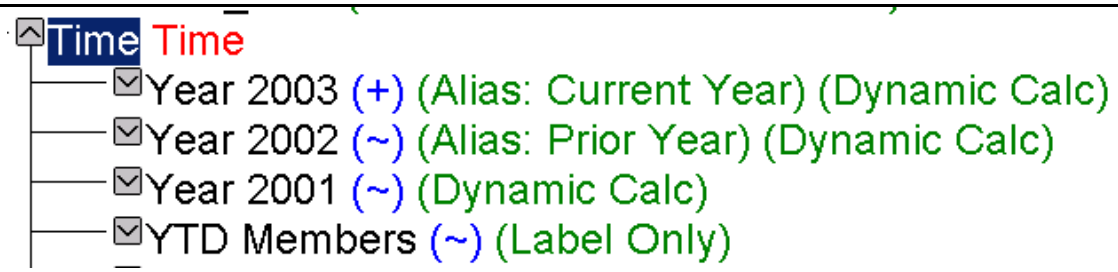
Even though SCENARIO is set to Label Only (i.e. it won't store data), if it is queried it will display the data from its first Child (2003 in this example):

Proper Use of Implied Shares:

Developers must be aware of when an Implied Share will be created and control it themselves to the greatest extent possible.

Improper Use Example:

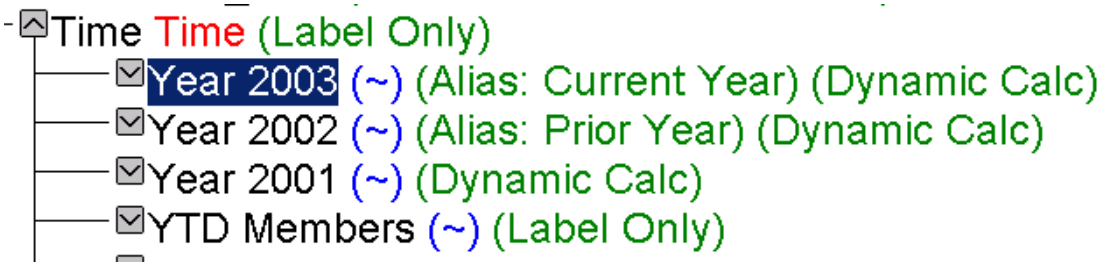
In the following example it was not the Developer's intention for the Member Time to store data. They allowed Essbase to automatically create the Implied Share for them by only setting one Child to aggregate to Time:



This will work, but has a high potential for something to go wrong during maintenance. It is also inefficient, because every time a Batch Calc is performed Essbase must evaluate all the conditions involved before implementing the Implied Share.

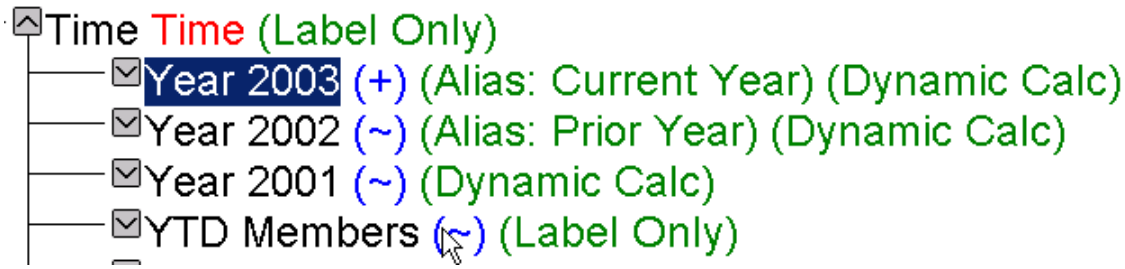
Proper Use Example:

Since it was not intended that Time be stored, EXPLICITLY set it as such with a Label Only setting. There is no need to use a consolidation setting other than '~' for its Children, since we know the first Child will become the source of the data that Time displays.



Avoid the Old School Methodology:

Traditionally, a lot of Essbase Developers have used the Label Only setting in conjunction with setting the Child they want to use as the data for the Implied Share to be a '+'. This is not necessary and is inefficient. It would look like this:



Implied Shares Summary:

- Members that you do not intend to store data in should have a Storage Property other than 'Store Data'. They should be set to 'Label Only', 'Dynamic Calc', etc.

- If the Developer feels it is necessary to assign a Consolidation Property other than a '+' to a Child of a Label Only Member, they must:

    1. Document that they are doing so and why.
    2. Provide proof that what they are attempting works.
    3. Provide comments in the Outline itself, summarizing number 1, above.

Member Formulas in the Outline

- Please follow the same standards as for Calc Scripts, below. The only exception is that for a Formula in the Outline it is not necessary (or desired) to explicitly state the left hand side of the Formula. The left hand side is the Member the Formula is attached to.

Calc Scripts Standards

Comments

- All Calc Scripts must begin with a comment block that contains the following information:
  - The Calc Scripts filename
  - The name of the Application/Database string it calculates.
  - The date it was created/last modified.
  - The name of the creator/modifier.
- Essbase Calc Scripts use the slash star ('/*') to begin a comment block and a star slash ('*/') to end one. Comments may span multiple lines. Essbase does not like "nested" comments and will return an error upon validation and at runtime.
- Comments should be used to explain design decisions, to explain complex aspects of the code, and to clarify the business logic behind complex algorithms.
- Comments should not be used where they are repeating information that is readily understood from the code itself.
- Comments should precede the block of code they clarify. For simple, one-line Calc Script Functions they may be placed directly to the right of the Function, on the same line.
- Please use Standard English grammar, punctuation, and spelling so that the meaning of the comment is clear to someone other than the author.
- Obsolete code should be deleted from the script and not merely commented out.

Vertical Spacing

- Start every statement or Function on a new line.
- Use blank lines to separate logically related lines of code

Example:

```
SET CACHE HIGH;
SET LOCKBLOCK HIGH;
SET AGGMISSG ON;
CALC DIM(ASSOCIATE, PRODUCT, NAICS);
```

- Where a long line of code is broken into several lines for readability, ensure that the breaks in the line are made in logical, readable places.

- Do not begin a line with a comma.

Horizontal Spacing

- Lines should not be longer than can be viewed in the Calc Script Editor without scrolling to the right.

- Indent procedures that follow any Function that limits the focus of the Essbase Calculation Engine. This includes FIX and IF. Further indent nested FIX and IF Statements.

  Example:

```
FIX (“Current Year”)

    FIX (@IDESCENDANTS (“Deposit Products”))

        “January Rolling 12”
        (
        IF (@ISMBR (“Average Balance”);
            “January Rolling 12” = @AVG (SKIPNONE,
                                         "January"->"Current Year",
                                        "December"->"Prior Year",
                                        "November"->"Prior Year",
                                        "October"->"Prior Year",
                                        "September"->"Prior Year",
                                        "August"->"Prior Year",
                                        "July"->"Prior Year",
                                        "June"->"Prior Year",
                                        "May"->"Prior Year",
                                        "April"->"Prior Year",
                                        "March"->"Prior Year",
                                        "February"->"Prior Year");
        ENDIF
        )
        ENDFIX

ENDFIX
```

- Essbase ignores white space, but people don’t. To make the code more readable a space must always follow a comma.

Please DO this:

CALC DIM(PRODUCT, Market, Region);

Please DON'T do this:

CALC DIM(PRODUCT, Market, Region);

Capitalization

- Please type all Essbase Calc Script Functions in all upper case (if you paste them into your script using the Paste Function tool it will do this for you).

- By default, an Essbase outline is not case-sensitive, however please type all Member names referenced in a Calc Script EXACTLY as they appear in the Outline, including mixed case (if you paste them into your script using the Member Selection tool it will do this for you).

Using Aliases

- When referencing Member Names that will "roll over" with time, it is best to use their Alias as opposed to Hard-Coding in the Member Name. For example if it is currently 2004, it is better to reference the Alias "Current Year", instead of the Member Name 2004, so that the script doesn't have to be updated the next year.

Report Scripts

- Please follow the same standards for vertical and horizontal spacing as apply to Calc Scripts.

- Do not hard-code Member names that rollover with time. Use the Alias instead.

Load Rules (Both Data and Dimension Build)

- Do not use embedded SQL. All Load Rules should use as input a flat file. Essbase's SQL interface is not very sophisticated, and using it creates another dependency when other groups need to make changes to the RDBMS environment.

- Any Selection/Rejection criteria MUST be documented.

- Any Replacement criteria MUST be documented.

- Any Field Edits MUST be documented.

- If the Data Values settings are changed to be anything other than the default "Overwrite Existing Values", the developer must have a valid reason for doing so, and it must be documented.

Comments

- All ESSCMD Scripts must begin with a comment block that contains the following information:
    - The filename of the script.
    - The name of the Application/database(s) the script runs against.
    - The date it was created/last modified.
    - The name of the creator/modifier.
- ESSCMD scripts use the colon (':') to indicate a comment. In ESSCMD you can only comment an individual line.
- ESSCMD also uses the colon to indicate a Label used for error handling (with IFERROR). Because of this, please use at least two colons ('::') for a comment to differentiate comments from Labels.
- Comments should be used to explain design decisions and to explain complex aspects of the code.
- Comments should not be used where they are repeating information that is readily understood from the code itself.
- Comments should precede the block of code they clarify.
- Please use Standard English grammar, punctuation, and spelling so that the meaning of the comment is clear to someone other than the author.
- Use comments to break the code into logical sections. For example:

```
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Purpose: Calculate Staging dB
:: filename: /olpp/src/ksh/amit-pc/calc_example.scr
:: Modifications:
::   Created by Amit Sharma on Jan 13, 2010
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

:: Init
OUTPUT 1 "/opt/hyperion/essbase/logs/diam_calc.log";
LOGIN "amit-pc" "userid" "password" "xxxxxx" "xxxxx";
  IFERROR "QUIT_SCRIPT";

:: Execute Calculation
RUNCALC 2 "CalcAll";
  IFERROR "QUIT_SCRIPT";

:: Successful Completion
LOGOUT;
OUTPUT 3;
EXIT;

:: Graceful Exit if an error encountered
  :QUIT_SCRIPT;
    EXIT;
```

- Obsolete code should be deleted from the script and not merely commented out.

Dimension Builds

When doing multiple dimension builds in the same script, please use INCBUILDIM instead of BUILDIM so that only one re-structure is performed for the entire process.

Path References

Where links are available to the 'arbor' ID, please use the links as opposed to hard-coding paths to Filers. This will reduce potential errors and maintenance in the event that the files being referenced are migrated to another Filer.

Logs

- All ESSCMD scripts run in batch must output to a log file for trouble-shooting purposes.
- For ESSCMD scripts utilized for one-off type administrative tasks, log files are optional.

Vertical Spacing

- Start every statement or Function on a new line.
- Use blank lines to separate logically related lines of code
- Where a long line of code is broken into several lines for readability, ensure that the breaks in the line are made in logical, readable places.
- Do not begin a line with a comma.

Horizontal Spacing

- Lines should not be longer than can be viewed on the screen without scrolling to the right (based upon an 800 by 600 pixel display). Typically, a line of code should not be more than about 78 characters long.
- Standard indentation is two to four characters. Tab characters should not be used as they are displayed differently in various editors.

MaxL Scripts

Comments

- All MAXL Scripts must begin with a comment block that contains the following information:
    - The filename of the script.
    - The name of the Application/database(s) the script runs against.
    - The date it was created/last modified.
    - The name of the creator/modifier.
- MAXL scripts use the slash star ('/*') to begin a comment block and a star slash ('*/') to end one. Comments may span multiple lines. MaxL does not like "nested" comments and will fail to compile.
- Comments should be used to explain design decisions and to explain complex aspects of the code.
- Comments should not be used where they are repeating information that is readily understood from the code itself.
- Comments should precede the block of code they clarify.
- Please use Standard English grammar, punctuation, and spelling so that the meaning of the comment is clear to someone other than the author.
- Use comments to break the code into logical sections. For example:

```
/***************************************
*Purpose: Script to Restore Sample Application app
*
*Filename: /olpp/src/ksh/archive/restSample_App.msh
*
*Modifications:
*
*Amit Sharma 10th Jan 2008
***************************************/

/*** init ***/

shell rm '/opt/hyperion/essbase/logs/arch/restapp.log';

spool on to '/opt/hyperion/essbase/logs/arch/restapp.log';

login userid identified by password on cltssp301;

  /** ensure dB to be restored is loaded in memory **/

  alter application 'Sample_App' load database 'Freedom';


/*** Disable connections to dB to be restored ***/

alter application 'Sample_App' disable connects;


/*** Kill active requests  on dB to be restored ***/

alter system kill request on application 'Sample_App';


/*** Logout all users on dB to be restored ***/

alter system logout session on application 'Sample_App';


/*** UnCompress (UnZip) export files ***/

shell gunzip -r '/Net/cltffp13/data1/app/arch/Sample_App';
```

- Obsolete code should be deleted from the script and not merely commented out.

Dimension Builds

When doing multiple dimension builds in the same script, please use ESSCMD until further notice. While the 'import dimensions' command in MaxL is, per documentation, capable of an incremental dimension build it has proven to be buggy as of this writing. Using the 'shell' command in MaxL allows you to call an ESSCMD script from a MaxL script.

Path References

Where links are available to the 'arbor' ID, please use the links as opposed to hard-coding paths to Filers. This will reduce potential errors and maintenance in the event that the files being referenced are migrated to another Filer.

Logs

- All MAXL scripts run in batch must output to a log file for trouble-shooting purposes.
- For MAXL scripts utilized for one-off type administrative tasks, log files are optional.

Vertical Spacing

- Start every statement or Function on a new line.
- Use blank lines to separate logically related lines of code
- Where a long line of code is broken into several lines for readability, ensure that the breaks in the line are made in logical, readable places.
- Do not begin a line with a comma.

Horizontal Spacing

- Lines should not be longer than can be viewed on the screen without scrolling to the right (based upon an 800 by 600 pixel display). Typically, a line of code should not be more than about 78 characters long.
- Standard indentation is two to four characters. Tab characters should not be used as they are displayed differently in various editors.