The Complete Guide To HYPERION INFRASTRUCTURE

US-Analytics

PICTURE RESULTS.

When it comes to on-prem tools, infrastructure is probably the most important topic. Ensuring that your tool is tuned, configured, and optimized is key to getting the best possible ROI.

However, infrastructure is a complex topic, and it's easy to miss steps that will make your tool run smoother. That's why we've compiled this ebook with the help of our infrastructure experts. (Unfortunately, it's nearly impossible to put decades of experience into one ebook — so, if you have any further questions, simply **drop us a line**.)

We'll cover the good, the bad, and the ugly when it comes to Oracle EPM infrastructure. You'll learn best practices, mistakes to avoid, and advanced tutorials and tips to help you optimize your various Hyperion tools. Here's what to expect:

BEST PRACTICES

Basic Performance Tuning and Troubleshooting

Performance tuning and troubleshooting is no one's *favorite* topic, but it's one of the most *important* topics when it comes to Oracle EPM. No one wants a system that lags or — even worse — crashes during an integral time.

In this section, we'll cover some tips and tricks for EPM tuning and troubleshooting so you can make sure your Oracle EPM tools are always working, especially during those vital times.

Knowing Your Performance Drivers

Before you can even think about performance tuning, you need to know what your environment relies on to perform — your key performance drivers. The factors that play a vital role in your environment's performance include:

- Hardware
- Oracle or third-party software tuning
- Business process usage
- Business application design

Hardware

Your hardware's performance depends on several factors, including:

- Speed of your processor
- Number of servers
- RAM
- Network speed

Oracle or third-party software tuning

Oracle or third-party software tuning encompasses the tuning of the other software you need to actually run your EPM products. This might include your webserver, appserver, browser, or relational databases.

Business Process Usage

Business process usage simply means how your EPM tool is being used. You'll need to consider things like your number of users, the types of tasks they're performing, and how many concurrent tasks are being performed. Your performance could be majorly impacted by multiple users performing large tasks at the same time, so it's important to take note of how (and how often) tasks are being performed.

Business Application Design

When looking at your business application design you need to be careful about size, making sure that your application isn't so large that it affects functionality. You should also look at the structure and product functionality while designing your applications. Are you including any functionality that won't be used? Or is the functionality based on your specific business requirements?

Performance Tuning with Server Directives

There are several Oracle HTTP server directives that will help you improve the performance of your EPM system. We'll cover them in this section. *Note: These parameters are set in ohs configuration file, i.e. httpd.conf file.*

Timeout 300

Timeout tells you the amount of time your webserver takes and keeps the connection open before sending a requested file or completing tasks. For the best performance, you should set it to 300. If your connection is idle for about 5 minutes, it will close that connection and you'll need to open a new connection to your webserver.

KeepAlive On

KeepAlive allows you to have a persistent connection to webserver without closing those connections. For better performance, you should set it to "on," and then set the number of requests per connection.

MaxKeepAliveRequests 0

If you have *KeepAlive* set to On, then you should set *MaxKeepAliveRequests* to 0. *MaxKeepAliveRequests*sets the number of requests allowed per connection when *KeepAlive* is set to “on.” When you set *MaxKeepAliveRequests* to 0, you’re allowed an unlimited number of requests, which will help you achieve the best performance.

KeepAliveTimeout 15

KeepAliveTimeout tells you how much time (in seconds) Apache will wait for a subsequent request before closing the connection. The average time for best performance is 15 seconds.

Preventing & Troubleshooting Login Latency

If you’re still on a version of Hyperion that uses Workspace, this is the section for you. Here are a few tricks that you can use now to troubleshoot login latency in workspace or avoid it if it hasn’t happened yet:

1. You should set your default log level to <Warn> rather than <Debug>.
2. You can improve your login performance if you decrease the number of nestings within a group membership.
3. You should only retrieve your required users or groups. You can use filters to retrieve the required users.
4. The number of users who have access to your EPM system should be limited as much as possible.
5. Use native groups to provision external users rather than external groups.
6. The search order for your user directory should be set so that the user directory with the maximum number of users is specified first.

Keeping Up with HFM and Essbase Performance Tuning

Just like a car, you know your Hyperion products need regular tuning — but there’s not an odometer to tell you exactly when to do it.

Hyperion products install under the assumption you’re going to tune it, and it’s a necessity for regular users to fully optimize your system's performance.

So, when should you revisit performance tuning? In most cases, once a month, but it depends on how often you’re using an application or module and what you’re using it for.

This section will give you an overview of when you should conduct performance tuning based on what system you’re using, and what could happen if you don’t keep up with regular tuning.

Overall Environment

Any time you’re making a large change to your overall environment, you should conduct performance tuning.

This seems like a no-brainer, like when you add a new application or module. But there are other times when performance tuning is needed that might not be so obvious, such as using Oracle Data Integrator (ODI) after it went relatively unused.

If performance tuning isn’t performed regularly or during a “big” change, you could blow out or crash your module.

Just like your overall environment, your Hyperion Planning system could crash if you don’t perform regular tuning.

Essbase

Fortunately, Essbase doesn’t have to update very often. Any time you make a large change, like increasing the amount of Essbase data or changing the app design, you should consider performance tuning.

If Essbase needs tuning, it could cause slowdowns. However, Essbase doesn’t require very much tuning. If you’re experiencing poor Essbase performance, like slowdowns, it’s probably unrelated to tuning, signaling a problem or change made to your server.

Hyperion Financial Management

With HFM, if you don’t complete regular tuning, you could face extreme slowdowns. Something that usually takes seconds could take up to a minute or more.

These items that require regular tune-ups can slow your system, create instability and crashing:

- Log rotation
- Clearing invalid records
- Database maintenance
- Truncating appropriate tables in the database

HFM tuning is also extremely complex with various types of tuning, such as memory and cache tuning.

As your system grows and you're having more data loaded, your history size increases which changes the amount of data you’re dealing with — any time you increase the amount of data you should conduct performance tuning.

Additionally, you should always tune after creating or deleting HFM applications that weren’t present during your last tune-up.

COMMON MISTAKES

Worst Practices in Planning and Essbase

You've probably read your fair share of "best practices" guides for maintaining Hyperion applications. But what about "worst practices"— the most common culprits of slowdowns and even crashes? Below are six infrastructure mistakes we frequently see organizations make with their Planning and Essbase applications and easy ways to avoid them.

Using HsGetValue too often

There's nothing wrong with using HsGetValue to retrieve data. However, this function should be used sparingly and shouldn't be the main way you retrieve data from a Smart View spreadsheet. Each time you use the HsGetValue and you have cells with no data, it writes an error to your Essbase log. Each empty cell creates three lines of data, which could potentially cause you to fill up your Essbase log and corrupt your drive.

Logging in as the admin user during busy times

There's a reason why admins don't complete tasks like performance tuning during busy hours. Users might log in with an admin role because they need full permissions, but what they don't realize is Essbase doesn't apply filters for the admin users. Without a filter applied, Essbase will add significantly more data, and doing this during busy times will cause freezing for other users.
If a user needs full permissions within the Planning app, they should be made a "hype admin" user.

Lack of performance tuning

In Hyperion Planning, most of your performance tuning involves your database connections. The default max connections to Essbase or the database is 10, which can be a problem when you have more than 20 users trying to pull a lot of data at one time. You'll end up with users trying to click through web forms and having to wait.

Failing to check the logs

Your logs will be full of a lot of meaningless information, like people typing in their password incorrectly. But logs also contain meaningful information, like calculation times so you can see if they're getting better or worse. Your logs might even reveal warning signs that you need to increase your connections or your max size. Checking the logs is a good way to find a problem before it becomes catastrophic, i.e. crashing during busy times.

Failing to repair Essbase.cfg

In almost every 11.1.2 version, there's a flaw in the default Essbase config file that people frequently forget to fix or don't even realize is there. In the config file, the quotes aren't closed. This causes Essbase to assume the entire cfg file after the comment is merely a comment and doesn't apply to anything.

The fix is simple:

1) Navigate to 'D:\Oracle\Middleware\user_projects\epmsystem2\EssbaseServer\essbaseserver1\bin'
2) Right-click 'essbase.cfg', click 'Edit'
3) Locate the following section:

```
;BPM_ORACLEBI_DriverDescriptor "Oracle BI Server"

BPM_MySQL_DriverDescriptor "DataDirect 7.0 MySQL Wire Protocol



AuthenticationModule CSS


AGENTPORT 1423
```

4) Add closing quotes in the line in bold, as below:

```
;BPM_ORACLEBI_DriverDescriptor "Oracle BI Server"

BPM_MySQL_DriverDescriptor "DataDirect 7.0 MySQL Wire Protocol”


AuthenticationModule CSS

AGENTPORT 1423
```

5) Navigate ‘File > Exit’

6) Click ‘Yes’ to save changes

Not archiving your logs

Your logs can get really big and slow down your operating system as well as your Hyperion product. About once a month, you should zip up all your logs and save them for historical purposes. This proactive task will ensure your operating system and applications continue to perform properly.

Worst Practices in HFM

Whether you've been using Hyperion Financial Management (HFM) from its advent or you're new to the Hyperion product, you probably haven't seen a guide on the things you shouldn't do to HFM. These worst practices are just as important to know as the "best practices," so you can ensure you're not hurting HFM's performance.

Not tuning on a per app basis

Older versions of HFM required global performance tuning, but now most recent versions enable you to tune on a per app basis. Many customers have multiple HFM apps as well as historical apps, and tuning globally won't give each app enough resources to function. When you tune on a per app basis, you can change the settings for each app — ensuring apps that do consolidating or calculating get more tuning than your historical applications.

Not changing expansion size for the database in SQL Server

If you're using SQL Server, the default expansion size is one megabyte. When you set up your database and have one gigabyte of data, the SQL Server database will pause, add one megabyte, and resume. This will happen over and over and destroy your HFM performance.
Many tuning guides don't mention having to expand SQL Server, and our consultants see this problem a lot. The solution is easy — set your SQL Server database to 1,500 megabytes so you always have some free space. If you're using SQL Server and having HFM performance issues, this may solve all your problems.

Failing to check the logs

Your logs will be full of a lot of meaningless information, like people typing in their password incorrectly. But logs also contain meaningful information, like calculation times so you can see if they're getting better or worse. Your logs might even reveal warning signs that you need to increase your connections or your max size. Checking the logs is a good way to find a problem before it becomes catastrophic, i.e. crashing during busy times.

Using Consolidate All

Using the Consolidate All with Data option is one of the worst things you can do in HFM. If you have empty cells, they will fill up with zeroes — meaning if your spreadsheet is 10 percent full, Consolidate All will make it 100 percent full. Now you're calculating 10 times more data with no option to remove zeroes. To get rid of all the zeroes, you can do exports, delete your app, and rebuild it. The best option is to remove the Consolidate All with Data option in your security settings.

Worst Practices in Security

When it comes to Hyperion security, there are quite a few things you can mess up. You might already know the best practices, but in this section, we'll cover what you shouldn't do when it comes to Hyperion security.

Applying security directly to a user (other than the admin)

This is one of those "work smart, not hard" situations. If you apply security directly to the user, you're losing all those settings if that employee leaves your organization. When a replacement arrives, you'll once again have to figure out all their permissions and security settings. There's really no reason for all that work that's probably keeping you from a few other pressing tasks.

What you need to do is use groups. Make a group for every type of position — it'll be much easier to query and maintain. When someone inevitably leaves, their replacement can easily be dropped into the group and — voila! — they have the proper permissions.

Having too many or too few groups

Now that we've clarified that you should be using groups, you need to know how to use them the *right way.* We've seen some companies with thousands of groups because they didn't write up their requirements and build their security structure properly.

You should design the makeup of your groups around your organization's structure or, more specifically, the structure of your finance department. The easiest way to do this is starting from the bottom up, so you'll want to make a group for READ access. All the finance employees should be able to read all the numbers, but you'll want to limit who can write. From there you'll create a WRITE group that has a portion of the members in the READ group.

Applying role *and* access to every.single.group

When you need to separate segments of your users into basic users versus managers/power users, it's a bad idea to apply both role (READ/WRITE) and access (by entity, by region, by cost center, etc.) to a group. Instead, create access groups (by entity, by region, by cost center, etc.) as usual and then create a few groups based on user types. Each user will belong to an access group and a role group. When the forecast cycle opens, all role groups may be set to WRITE. After the deadline, switch the basic users to READ while your managers/power users keep WRITE access. All users can still see data, but managers are the only ones able to make changes.

Provisioning multiple applications to your groups

Provisioning multiple applications to a few groups may seem like a quick way to get your user base access to all systems necessary, but, as applications are added or phased out, this may cause heartburn down the road — and leave room for a messy trail of “what have I done here?”

Having groups provisioned to a single application is the cleanest way to go, and your maintenance will be much easier to manage — you’ll thank yourself later!

Not keeping Hyperion security documentation up to date

When the system is new, nice, and shiny, it’s easy to take the documentation from your implementation partner, check off the box as complete, and file it away. You know what groups (native or active directory) have been set up and what provisioning is applied to where. You know how to add users where it’s appropriate.

Then, the application development group builds a new application with slightly different security requirements. While you are implementing those changes, don’t skip the documentation step. Your successors will thank you when they get the call from audit for the security documentation on who has access and how it is managed.

Essbase 11.1.2.x — Changing Essbase ODL Logging Levels

ODL (Oracle Diagnostic Logging) information is recorded for both the Essbase agent and the Essbase application server. The level of the logging that is recorded can be amended, if required.

These are the two ODL logs for Essbase:

- EssbaseAgentODLLogger is for the Essbase agent. This writes to the ESSBASE_ODL.log in **MIDDLEWARE_HOME/user_projects/epmsystem1/diagnostics/logs/Essbase/Essbase**

- DefSvrLogger is for the Essbase application server (ESSSVR). This writes to the <appname>_ODL.log in **MIDDLEWARE_HOME/user_projects/epmsystem1/diagnostics/logs/Essbase/Essbase/app/<appname>**

The level of logging for these Essbase ODL logs is controlled by the logging.xml file.

The logging.xml for Essbase is stored here:

```
EPM_ORACLE_INSTANCE/EssbaseServer/essbaseserver1/bin/logging.xml
```

The log level can be changed from the default of ‘TRACE:1’ to another setting e.g. ‘ERROR:1’. The lower the log level, the less information will be recorded. The higher the log level, the more information will be recorded. The log levels, and information about what is recorded at each level, are here:

```
http://docs.oracle.com/cd/E40248_01/epm.1112/epm_troubleshooting/ch03s04s01.html
```

To change the logging levels, edit the logging.xml file, and find the “<loggers>” section.
This is an example of a default logging.xml file.

```
<loggers>
<logger name='EssbaseAgentODLLogger'
level='TRACE:1' useParentHandlers='false'>
<handler name='agenthandler'/>
```

```
</logger>
<logger name='DefSvrLogger'
level='TRACE:1' useParentHandlers='false'>
<handler name='serverhandler'/>
</logger>
```

To change the logging level, change the ‘TRACE:1’ setting to the required setting (e.g. ‘ERROR:1’).

After changing the logging.xml, restart the Essbase service for the configuration change to take effect.

Disabling Implied Share for an Essbase Application

Implied share is the default behavior in Essbase where a data relationship is shared explicitly between parent and child members in the outline. Implied share occurs when a parent has a single child or a parent has only one child of many children that consolidates to the parent in an Essbase outline.

If you need to disable all implied shares in a given outline, there is an Essbase configuration setting that can be set in the Essbase.cfg file. As part of the supported steps, it is recommended to rebuild the outline after the ***IMPLIED_SHARE <ApplicationName> FALSE*** setting is in place. In this section, we'll help field where in certain situations rebuilding the outline may be challenging. There is a way to set an existing outline to have implied share disabled, however, there are risks associated with these steps and no guarantee the metadata and data implied share setting are in sync. If there are any issues after following the "unsupported" steps, Oracle Support will require that the supported steps be followed first.

The Essbase Technical Reference Guide has the supported "best practice" steps to ensure the metadata and data implied share setting are in sync. See the Essbase Technical Reference Guide:

http://docs.oracle.com/cd/E40248_01/epm.1112/essbase_tech_ref/frameset.htm?impliedshare.html

Supported Steps (per the Essbase Technical Reference Guide)

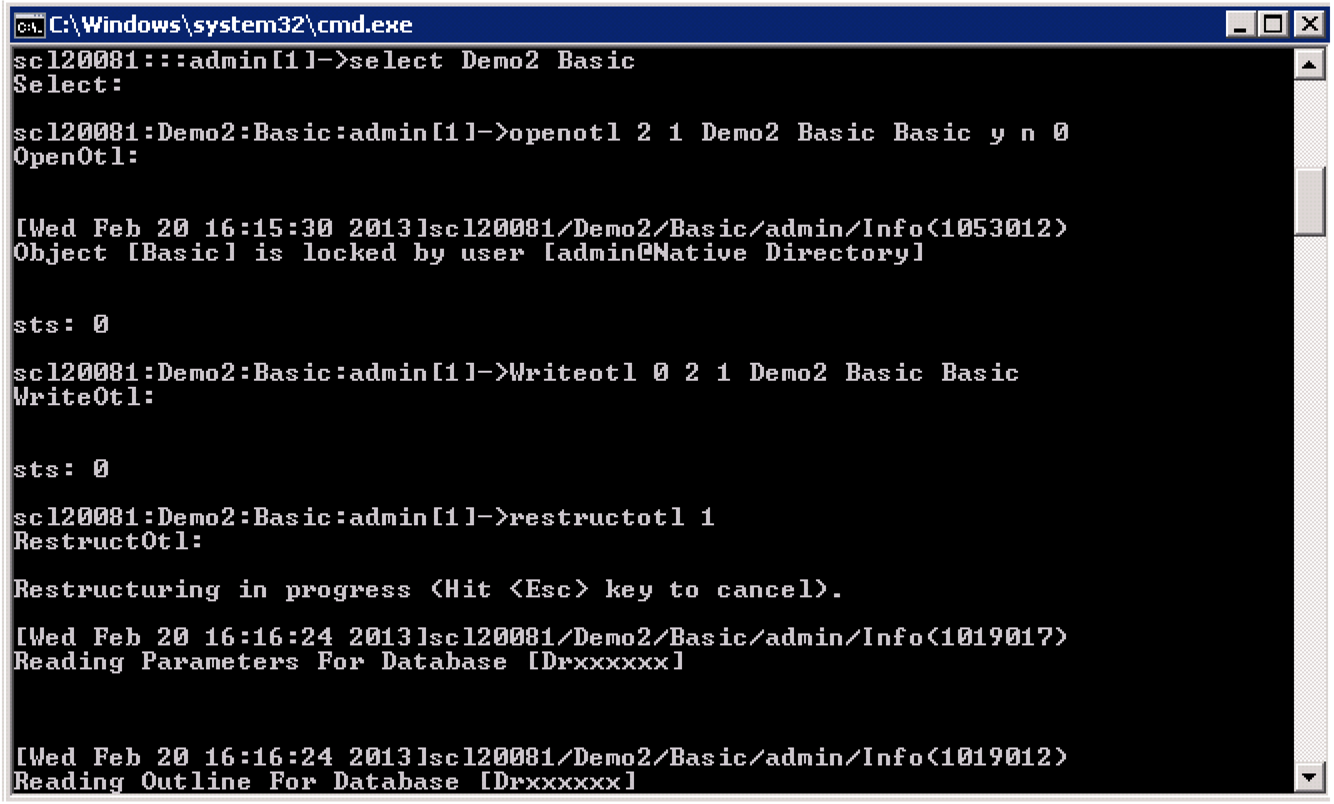
The following steps must be performed any time the IMPLIED_SHARE setting is changed in essbase.cfg:

1. Add IMPLIED_SHARE FALSE to essbase.cfg
2. Restart Essbase Server
3. Create a new application and database, with the IMPLIED_SHARE setting in place
4. Rebuild the outline, with the IMPLIED_SHARE setting in place
5. Reload the data
6. Run aggregation or calculation scripts
7. Restart the application

Unsupported Steps:

If rebuilding the outline is not possible (step #4 in the Essbase Technical Reference Guide) you can disable implied share for the outline using the ESSCMDQ Utility.

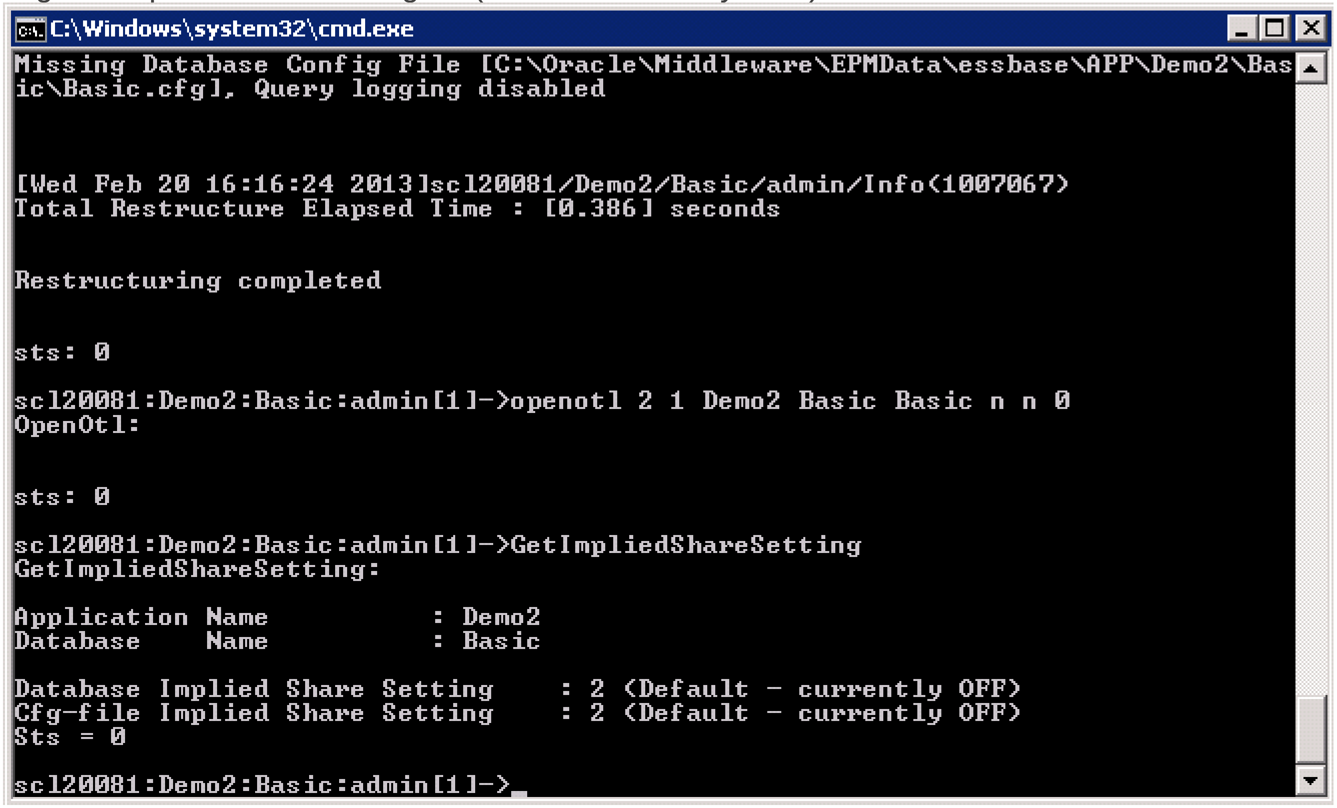
1. Download and install ESSCMDQ from the following URL. Follow installation instructions on the website for correct version and OS of the ESSCMDQ utility:
http://www.oracle.com/technetwork/middleware/bi-foundation/esscmdq-sampleapps-093105.html
2. Follow steps 1 – 3 in the Essbase Technical Reference Guide
3. In EAS, open the existing outline and save the outline to the newly created Application and Database
4. In EAS, unlock outline after closing
5. Start ESSCMDQ Utility to modify outline to accept the Essbase CFG setting
Login <HostNode> <UserId> <Password> ***enter***
Select <AppName> <DbName> ***enter***
Note: You can enter the following commands in one line or hit enter for each prompt command
Openotl 2 1 <AppName> <DbName> <OTLname> y n 0 ***enter***
Writeotl 0 2 1 <AppName> <DbName> <OTLname> ***enter***
enter
Restructotl 1 enter
Exit enter


C:\Windows\system32\cmd.exe

```
scl20081:::admin[1]->select Demo2 Basic
Select:

scl20081:Demo2:Basic:admin[1]->openotl 2 1 Demo2 Basic Basic y n 0
OpenOtl:


[Wed Feb 20 16:15:30 2013]scl20081/Demo2/Basic/admin/Info(1053012)
Object [Basic] is locked by user [admin@Native Directory]


sts: 0

scl20081:Demo2:Basic:admin[1]->Writeotl 0 2 1 Demo2 Basic Basic
WriteOtl:


sts: 0

scl20081:Demo2:Basic:admin[1]->restructotl 1
RestructOtl:

Restructuring in progress (Hit <Esc> key to cancel).

[Wed Feb 20 16:16:24 2013]scl20081/Demo2/Basic/admin/Info(1019017)
Reading Parameters For Database [Drxxxxxx]



[Wed Feb 20 16:16:24 2013]scl20081/Demo2/Basic/admin/Info(1019012)
Reading Outline For Database [Drxxxxxx]
```

6. You can use the ESSCMDQ Utility to check that the outline accepted the Essbase CFG setting
Login <HostNode> <UserId> <Password> enter
Select <AppName> <DbName> enter
Openotl 2 1 <AppName> <DbName> <OTLname> n n 0 enter
GetImpliedShareSetting enter
Note: If IMPLIED_SHARE FALSE setting took effect, you should see the following GetImpliedShareSetting:

Application Name : Demo2
Database Name : Basic
Database Implied Share Setting : 2 (Default - currently OFF)
Cfg-file Implied Share Setting : 2 (Default - currently OFF)



C:\Windows\system32\cmd.exe

```
Missing Database Config File [C:\Oracle\Middleware\EPMData\essbase\APP\Demo2\Bas
ic\Basic.cfg], Query logging disabled


[Wed Feb 20 16:16:24 2013]scl20081/Demo2/Basic/admin/Info(1007067)
Total Restructure Elapsed Time : [0.386] seconds


Restructuring completed


sts: 0

scl20081:Demo2:Basic:admin[1]->openotl 2 1 Demo2 Basic Basic n n 0
OpenOtl:


sts: 0

scl20081:Demo2:Basic:admin[1]->GetImpliedShareSetting
GetImpliedShareSetting:

Application Name           : Demo2
Database    Name           : Basic

Database Implied Share Setting    : 2 (Default - currently OFF)
Cfg-file Implied Share Setting    : 2 (Default - currently OFF)
Sts = 0

scl20081:Demo2:Basic:admin[1]->
```

Exit enter

7. Continue following steps 5 – 7 in the Essbase Technical Reference Guide

EPM Client Cert Authentication

If you are planning to use client cert authentication against EPM, there are a few additional elements to consider on top of the documentation:

"***SSLVerifyclient required|optional***" in Oracle Http Server "OHS".

This ssl parameter is, amongst other tasks, using trusted certificates in OHS wallet to filter out client certificates in the digital certificates browser prompt. That is, if you have client certificates signed by a trusted root that is not in OHS wallet, then these certificates won't show up in the browser prompt to pick the certificate.

- HYPLOGIN header sent by OHS through the following entry:

RequestHeader set HYPLOGIN "%{SSL_CLIENT_CERT}e"

OHS HYPLOGIN header value turns out to be like this (note the question marks):

?----BEGIN CERTIFICATE---- MII....
-----BEGIN CERTIFICATE---- ?

When using the java certificate factory ((X509Certificate)CertificateFactory.getInstance("X.509").generateCertificate) to get the DN, you need to remove these unneeded question marks in your CSS custom login class.

- It is good practice to add

RequestHeader set HYPLOGIN ""

at the top of your VirtualHost, while the SSL_CLIENT_CERT header value will be set in your location entry.

Calculations Using Dates Stored in Planning

Planning allows users to enter date values in Planning. For example, a start date can be entered as 11/01/2013 if the format is MM/DD/YYYY. Although the values are entered in date format, it is possible to calculate values based on dates entered.

Essbase stores values having a date format as numeric values. In the example above the “Start Date” 11/01/2013 is stored in Essbase as the value 20131101. If the “End Date” is 06/30/2014, you can calculate the number of months between the dates using Essbase functions. The following shows how to calculate the number of months between the “Start Date” and “End Date” using the @ROUND function:

1. Start with (@ROUND (“End Date”,-4) - @ROUND (“Start Date”,-4))/10000*12 – This step will calculate the number of months between the “End Date” year and the “Start Date” year. The result of this step will be (20140000 – 20130000)/10000 *12 or 12.

2. Add (@ROUND (“End Date”,-2) - @ROUND (“End Date”,-4))/100 – This step will calculate the number months between the start of the “End Date” year and the “End Date”. The result of the step will be (20140600 – 20140000)/100 or 6.

3. Subtract (@ROUND (“Start Date”,-2) - @ROUND (“Start Date”,-4))/100 – This step will calculate the number months between the start of the “Start Date” year and the “Start Date”. The result of this step will be (20131100 – 20130000)/100 or 11.

Combining the three steps into a single formula results in 12 + 6 – 11 or 7 months. A similar formula can also be written using the @INT or @TRUNCATE Essbase functions.

Error Connecting to Planning Application from Smart View

When the OU of a user is changed in the external provider, users can potentially have issues connecting to Planning applications from Smart View. However, users can still log into Planning just not Smart View.

Below are the error messages found in Smart View, Essbase and Shared Services when the OU of a user is changed in the external provider and not synced with Essbase security:

- Planning
  user connecting to Smartview error:

"Cannot open cube
view. Essbase error 1054060. Essbase failed to select application
<appname> because user is not completely provisioned by Planning"

- Essbase
  application log error:

[Tue Mar 15 12:16:22
2011]Local/ESSBASE0///Error(1054060)
Essbase failed to select application <application name> because
<user@provider> Directory is not completely provisioned by Planning

- SharedServices_Security_Client
  log error:

[….@
Failed to get user with identity @ ldap://GUID=xxxxxxxxxxxxxx?USER. User not
available in configured user directories….]

When the OU of a user is changed in the external provider, perform the following steps to sync the external provider changes to Essbase security:

1. Export the Essbase security from Essbase Administration Services (EAS) Console, which can be found ***\Oracle\Middleware\user_projects\epmsystem1\EssbaseServer\essbaseserver1\bin***

2. In the exported security file, locate each OU user that is having Smart View login issues

3. Perform the following steps to remove impacted users from Essbase security and then refresh from Planning:
   - Login to MaxL as an admin user
   - Execute command: ***display user <username>;***
     - Can find each user name in the security dump file

    - Display user command will give the full user name with provider ***<username@provider>*** needed for the next command to drop
- Execute command: ***drop user <username@provider>*** from security_file;
- Run another security dump to be sure users were removed from Essbase
- Run a Planning security refresh to sync the user back to Essbase

This should correct the Planning application connection from Smart View login issue for each user.

Using the @CURRMBR Function

Essbase provides a suite of calculation functions to facilitate the definition and development of complex member formula calculations. The @CURRMBR calculation function is especially useful with scoping and managing complex calculations. The @CURRMBR (*dimension)* function is used to return the name of the member currently being calculated in the specified dimension.

When using the @CURRMBR calculation function, take into consideration if the calculation is dense or sparse. The @CURRMBR calculation function is expected to be slower when used on a dense dimension since the function calculates at the block level, not the cell level. As a result, the function will calculate all of the dense dimension members, even if a particular dense dimension member is not present in the query. Conversely, performance is not an issue when using the @CURRMBR calculation function on a sparse dimension since the block corresponds to only one sparse dimension member.

In addition, the @CURRMBR function on a dense dimension could produce unexpected results or errors when used in conjunction with other functions such as @CONCATENATE. For example, if a dynamically calculated member in a query contains the formula "Actual" >@MEMBER(@CONCATENATE(@NAME (@CURRMBR ("Account")),"_Total")) and the Account dimension is sparse, the query will run without error as long as the account in the query generates a valid outline member. However, if the Account dimension is dense, the query will result in the following error even if the account in the query generates a valid outline member:

Error executing formula for [*member name*] (line 0): attempt to cross a null @ member in function [@X]

The error is produced because the @CURRMBR function calculates at the block level and therefore calculates on all accounts in the dense block. Not all accounts within the block will generate a valid outline member, resulting in the error message above.

Essbase BSO Parallel Calculation and Calculator Cache

Essbase BSO parallel calculation utilizes calculator cache on a per thread basis. This post is to show the relationship between using parallel calculation and calculator cache.

For calc parallel and calculator (calc) cache:

- Calc Parallel specifies the *number of threads per calculation execution*
- Calc Cache is memory used *per thread*

If your calculations contain, for example, CALCPARALLEL 9 and 5 users run this script, then (9 threads * 5 users) = 45 concurrent threads will be used (assuming 45 threads are available and the content of the calculation does not force serial calculation). If threads are not available, the calculations will wait until threads are available.

If your CALCCACHEHIGH is set to 2 MB (as an example) then every thread will use 2 MB i.e. (9 threads * 5 users) = 45 concurrent threads * 2 MB calc cache per thread, would mean that up to 90 MB of memory would be used for these 5 concurrent calculations.

Although, in some circumstances, it may be possible to increase these settings, it is important to be aware of this memory usage and thread availability when considering use of calc parallel and calc cache with applications, especially with applications that have high user concurrency.

Testing should be performed to prove that performance improvement outweighs the additional memory requirements.

Zero Based Budgeting (ZBB) Considerations within Hyperion Planning

Zero based budgeting (ZBB) applications are becoming increasingly popular as a way to develop a budget, especially for lower growth organizations that are interested in cutting costs. The most significant difference between ZBB applications and traditional budget applications are the level of details captured. Where traditional budgeting applications plan an expense item directly or using a price X quantity formula, ZBB applications will plan every line item related to that expense. For example, when budgeting supply expenses, a ZBB application will include values for each detailed line item, including pencils, pens, paper clips, etc. Given the level of detail required for ZBB applications, careful consideration needs to be taken in order to have optimal performance within Hyperion Planning.

The following questions need to be considered before designing a ZBB application within Hyperion Planning:

- Does the additional ZBB detail require control over how the data is entered?
  - If yes, then add a separate line item dimension.
  - If no, then supporting detail functionality within Planning may be sufficient.

- Does the detail have a direct relationship to an account?
  - If yes, then smart lists within the account dimension could be leveraged.

- Is the ZBB detail relevant for the current budget? The application should include only those line items needed for the current budget and remaining line items should be purged.

- Do all accounts require additional ZBB detail?
  - If no, consider having a separate ZBB plan type to store line item information for the account subset.

As indicated above, ZBB applications tend to require a separate line item dimension for storing the additional detail required. In addition, ZBB applications use smart lists extensively to capture non-numeric driver values used to evaluate the proposed budget items. The following lists the design characteristics of a ZBB application within Hyperion Planning:

- ZBB applications require planning at a line item detail. For non-ZBB applications, supporting detail is usually sufficient. ZBB applications, however, need more control over how the line item detail is entered, making a sparse Line Item dimension necessary.

- The number of line items for each account, which are known as packages in ZBB applications, will vary. The supplies account or package might have five line items while the travel account or package might have 10 line items. As a result, the Account dimension, which is typically called the Package dimension in ZBB, will need to be sparse to account for the differences in the amount of line item detail for a particular sub-account or sub-package.

- ZBB applications typically store multiple drivers, both numeric and non-numeric, for a particular sub-package to provide sufficient justification for each budget line item. Since the bulk of the ZBB values are calculated using driver values, the drivers are separated from the sparse Package dimension and placed in a dense Driver dimension to ensure the driver calculations are dense calculations. The Driver dimension is also typically marked with the Accounts property member tag.

- The design of the ZBB application assumes that all sub-package calculations are self-contained and have no dependencies on other sub-packages. If the design does show sub-package dependencies, then the customer's definition of a sub-package does not follow ZBB best practices and should be reviewed.

As described above, ZBB applications, unlike traditional budgeting applications, tend to place accounts (known as packages in ZBB applications) and account drivers in separate dimensions where the account members are stored in a sparse dimension and the account drivers are stored in a dense dimension marked with the Accounts property member tag. This design characteristic and the extensive use of smart lists to store non-numeric account driver values have the following performance impacts:

- The main impact of the ZBB design from a PBCS and Planning perspective is that the Driver dimension stores the drivers across all sub-packages, and the drivers used for a particular sub-package is typically unique. Therefore, the number of drivers populated for a particular sub-package will be a small subset of the total number of drivers. As a result, a given block of cells will always be sparsely populated for a ZBB application, and performance could suffer by having PBCS and Essbase process more data cells than is necessary.

- Another impact is on reporting ZBB data. The non-numeric drivers are stored in the Driver dimension as smart lists. A ZBB application can potentially have more than 50 smart lists, and each of these smart list items is typically queried in a standard or ad hoc report. If each smart list is mapped to a dimension in an ASO reporting database, the ASO database could potentially have more than 60 dimensions. However, the smart list reporting is typically mutually exclusive where a particular report will typically contain members of only one smart list dimension. Users do not typically perform analyses across multiple smart list dimensions.

As described above, ZBB applications within Hyperion Planning will tend to have sparsely populated blocks as well as large numbers of smart lists that will be mapped to a dimension in an ASO reporting plan type. Careful consideration will need to be given when designing ZBB applications to minimize the impact of these potential performance items.

EPM 11.1.2.x Essbase — DATAEXPORT Calc Command and CALCLOCKBLOCK/LOCKBLOCK

All dynamic member calculations, whether in an outline member formula or a business rule or calc script, will be calculated using the CALCLOCKBLOCK DEFAULT setting.

When running a DATAEXPORT calc script, it is possible to use SET LOCKBLOCK within the script. However, it may not be possible for this setting to be used for the entire export.

With DATAEXPORTOPTIONS "DataExportDynamicCalc ON”, only stored blocks will use the SET LOCKBLOCK setting in the script (e.g. Set LOCKBLOCK HIGH), if set. Any dynamic content will always be exported using the CALCLOCKBLOCK DEFAULT setting.

With DATAEXPORTOPTIONS "DataExportDynamicCalc OFF”; only stored blocks will be exported (i.e. dynamically calculated content will be excluded in the export) and, therefore, any SET LOCKBLOCK setting in the script (e.g. Set LOCKBLOCK HIGH) will be used for the entire export.

EPM 11.1.2.x — Planning/PBCS Best Practices for BSO Business Rule Optimization

In this section, we'll look at best practices for Business Rule optimization for Planning and PBCS models.

It will cover items which can be edited within the script only, i.e. syntax.

Environment Setting

Below are recommended Calculation Commands, which are the elements that instruct the business rule how to execute the calculations.

- **SET UPDATECALC OFF** turns off intelligent calculation, which is best practice for business rules which use cross dimensional operators and where there may be multiple users accessing the same data block combinations. Using **SET UPDATECALC OFF** is considered best practice for Planning/PBCS applications where multiple users write to the database. If intelligent calculation is used, ensure it is producing expected results.

- **SET AGGMISSG OFF** should be set for Planning/PBCS designs where versions are Standard Target or where non leaf data regions are used and cannot be excluded during your calculation. If the Planning/PBCS design has versions setup as Standard Bottom Up, then data is loaded at level0, where **SET AGGMISSG ON** will benefit.

- **SET NOTICE** and **SET MSG SUMMARY** should only be used in development environment for individual calculation analysis. These calc commands should be removed once in production and/or after analysis is completed.

- **SET EMPTYMEMBERSETS ON** should be used when Run Time Prompts (RTP) are included in FIX statements for Planning/PBCS models so that empty sets are not calculated. Prevents a calculation from running on all members when the set is empty.

Optimization/Performance

- Use templates in Calc Manager business rules to avoid repeating sections of code and make best use of RTP.
- Review dependencies on dynamic calc members within each calculation. If possible, change the calculation to avoid including repeated use of dynamic calc or remove the dynamic calc.
- Use FIX and IF to focus your calculation to ensure only data required is calculated.

- Avoid the creation of a 0 (zero) data result (unless you want to see a zero e.g inventory levels). This will create a block which will be included in all FIX/IF statements and will be calculated. Remove zeros on data load, if possible, or prevent their creation in business rules.
- Many rules have a check to see if, for example: IF ((“Budget”==#missing) OR (“Budget” ==0)). IF (“Budget”+1==1) or IF (Budget/Budget ==#missing) will give the same check but avoids the use of Boolean logic within the IF statement.
- Where possible, perform dense calculations, which do not create blocks, before sparse calculations, which do create blocks. If you need to perform aggregations (e.g to get a total for an allocation calculation), ensure that you only aggregate the section of the data that is required for that allocation calculation.
- Minimize the passes on a database where possible.
- Avoid recursive formulas. Excessive recursion can create environment performance issues as well as adding a requirement to clear values to produce consistent results.

FIX Statements

- FIX statements are used to focus the business rule i.e. to keep the # blocks being calculated to be as low as possible i.e. only calculate blocks that are needed.
- FIX is recommended for sparse dimensions because, when used on sparse, it reduces the # blocks that are required for the calc.
- Ensure calculations are done on level 0 of all dimensions when versions are bottomup in Planning/PBCS models.
- Use outer FIX statements on sparse dimensions with inner IF statements on dense dimensions where possible.
- All FIX statements should include members from ALL dimensions, except dimensions that are within the calculation. If a dimension is excluded, all members from that dimension will be included and it is likely that this is not required.
- Nest FIX statements where possible to reduce the number of passes on the database. Each full FIX requires a pass on the database. For example, use an outer FIX for Version, Scenario, and/or any other dimension selections that are static throughout the business rule.
- For Planning/PBCS business rules associated with web forms, leverage the selected page and POV members in FIX statements to reduce the number of blocks calculated.

IF Statements

- IF can be used in member formula. FIX cannot.
- IF should be used within FIX statements to reduce the #blocks that need to be accessed. IF brings all blocks within the FIX into memory.
- Use outer FIX statements on sparse dimensions with inner IF statements on dense dimensions where possible.
- Use ELSE instead of a combination of NOT and ELSEIF where possible to avoid unnecessary analysis of member values during the calculation. However, if an ELSE is not required, it is not necessary.
- Order IF statements, if possible, where the most number of cases hit the first IF in the block. Use NOT within the IF to ensure this, if applicable. See blog https://blogs.oracle.com/pa/entry/essbase_11_1_2_optimisation for more information on using NOT in IF statements.

- Review the Calc Member Block choice. A sparse member without dynamic calc dependencies would be a better choice.

Only Calculate Blocks Required

- For Planning/PBCS models, use RTP to ensure that only the data required is included in the business rule.
- Only aggregate/calculate data that is required at each stage of the calculation to ensure you keep the number of blocks included in the calculation as low as possible for as long as possible.

Level of Calculations

- For Planning/PBCS models, ensure calculations are done on level 0 of all dimensions when versions are bottomup.
- For Planning/PBCS models, aggregations should only be included in the BSO Plan Type if required for the approval process. Other aggregations should be moved to the ASO Plan Type.
- Try and keep the number of blocks included in your calculations to be as low as possible, for as long as possible.

Syntax

- Always use @LEVMBRSrather than @RELATIVEif used on the entire dimension.
- Use @CHILDRENinstead of @RELATIVE, if applicable.
- Use @REMOVE and @LEVMBRS if you only want to exclude some members from a FIX.

Block vs Cell Mode

- Using block mode, where cells are grouped within the block and simultaneously calculated, is generally faster but data dependencies must be carefully considered e.g. SalesYTD = CurMth + PriorMth would have to be calculated in cell mode so that each month is calculated in the order of the outline.
- Using cell mode, each cell is calculated sequentially in the order of the dense dimensions in the outline, is generally slower than block mode.
- Use @CALCMODE to manually control block vs. cell mode.
- Use debug mode application logs to verify calc mode. If a calculation is performed in block mode, no message will appear. A log message will be shown where calculations are performed in cell mode.

BottomUp vs TopDown

- Add calculation function @CALCMODE(BOTTOMUP); or calculation command SET FRMLBOTTOMUP to calculate existing blocks only (BOTTOMUP) instead of potential blocks (TOPDOWN).

- TOPDOWN calculations will calculate all potential data blocks with the member. For this reason, it is important that any irrelevant members within sparse dimensions are removed.
- Thoroughly test calculations using BOTTOMUP to ensure that blocks are created correctly when using @CALCMODE.
- Ensure testing is completed with clearing data and re-running the calculation to ensure all blocks are created correctly, especially when using BOTTOMUP.
- Use debug mode application logs to verify calcmode. If a calculation is performed BOTTOMUP, no message will appear. A log message will be shown where calculations are performed TOPDOWN.

Create Blocks

- Blocks, generally, will be created on the following actions:
- Data load
- DATACOPY
- Sparse calculations e.g. AGG or SparseMember = X * X/X;
- A sparse calculation is triggered:
- Where sparse members are on the left of the =
- Where the formula is within a sparse calc member block e.g. "Budget"("Sales" = "Sales"->"Actual" * 0.95;) where Scenario is sparse and Measures are dense.
- Creating blocks can be carried out using the calculation commands **SET CREATEBLOCKONEQ, SET CREATENONMISSINGBLK** or the calculation function @**CREATEBLOCK**. It is recommended that if these settings are required that they are used very sparingly and within a tight FIX statement. Test to see if it is possible to avoid the use of these statements by changing the type of calculation being performed.
- Block creation is a design related topic. Where there is an issue, it is important to prove that this is a block creation issue before using these calculation commands or calculation function. Submit a 0 into the target block and re-run the calc to prove a block creation issue.

Aggregations

- A sparse dimension aggregation should be ordered starting with the dimension that creates the fewest blocks to the one that creates the most blocks in order to keep the number of blocks as low as possible for as long as possible.
- In Planning/PBCS models, end user business rules should not aggregate entire sparse dimensions
- In Planning/PBCS models, any aggregations required for reporting only should be moved to the ASO Plan Type.
- n
  Planning/PBCS models, only aggregate data that is required for the Planning approval process.
- AGG vs CALC DIM Calculation Commands
- CALC DIM will execute any member formula
- CALC DIM will aggregate dense or sparse dimensions.
- AGG performs aggregations based on outline structure.
- AGG does NOT execute member formula.
- AGG will only aggregate sparse dimensions.
- Test both AGG and CALC DIM as performance can differ depending on levels of aggregation involved in the calculation.

- Exclude dimensions with dynamic calc on upper levels from all aggregations.
- Only aggregate data that is required.

SET CALCPARALLEL / FIXPARALLEL Calculation Commands

- For Planning/PBCS models i.e. multi-user applications with potential for rules running concurrently, it is best practice for end user business rules to be run in serial mode.
- Only use SET CALCPARALLEL around full sparse dimension aggregations in batch calculations.
- Parallel calculation is not recommended on small scripts (for example, less than 10 or 20 seconds) as the overhead of creating parallelism may outweigh the benefit.
- When used, always test SET CALCPARALLEL to ensure that it does give a benefit. Sometimes serial calculation or calculations with lower levels of parallelism can give better results.
- Test to determine if FIXPARALLEL would provide better results than SET CALCPARALLEL? Use Calc Manager debug mode to view logs to review.
- Always consider user concurrency when using SET CALCPARALLEL or FIXPARALLEL.

Debug Methodology for Developing Business Rules

- Always create a unit test i.e. a small subset of data where you know the source data and the expected results and can easily follow the calculation through manually for this set of data.
- Always use Calc Manager debug mode, or application logs, to view calculation entries to help with any debug during development.
- Ensure all data that is required is present (e.g. if a total is required for allocations) and has been pre-calculated prior to the calculation taking place.
- If the script is long, start debug at the top of the script and work down. An issue further up a script may be creating an issue further down. Debug section by section to ensure that all data is created correctly. Check that later sections of script do not overwrite earlier sections etc.
- Use debug mode (or application log) to ensure that each section of script is calculated sequentially where required.
- Always clear data and reload (i.e. do not use a clear script) when testing any business rule in order to ensure that all blocks are created successfully.
- Always test re-running a script to ensure that the syntax creates correct results for input data and/or populated data.
- Always test all data results with more than one data set.
- Where user input may change data sets e.g. allocations, also test data changing from #missing to a value and from a value to #missing, to ensure that previous calculated results are removed (if required) in second and subsequent runs e.g. if a user inputs against Product A,B, C in their first calculation run and then Product B, C, D (and not A) in their second, is the allocation result for Product A correctly removed in the second run.

Using HFM Java API

Many who use 11.1.2.4 have started looking into automating common tasks to further increase the user experience and lower maintenance efforts. Using the Java API, common HFM tasks can be automated easily. This post will provide first steps to log into HFM and run a Consolidation using the Java API.

Some may remember, that one of the changes was in regards to HFM components (e.g. HFM Web, HFM application server) communicating with each other. In previous releases Microsoft DCOM was used, which was replaced by a more common communication system that can be used across operating systems. This is the reason why the old Visual Basic or Visual Basic Script does not work anymore.

Oracle published a “Developer and Customization Guide” as part of the “Oracle Enterprise Performance Management System Documentation”. While this documentation even describes how to setup Oracle JDeveloper to get started with the HFM Java API, this section will cover a small example as well as the information on how to setup the environment to run HFM API programs without setting up JDeveloper on the server.

Let’s look into the different sections of the code ignoring the import section at the top. After reading the System Properties into variables a verification is done to check, if all required Java System properties and command line options exist. Otherwise the program exists after showing usage instructions.

```
public class runConsol{
    // Read System properties into variables
    static String EPM_ORACLE_INSTANCE =
System.getProperty("EPM_ORACLE_INSTANCE");
    static String HFM_CLUSTER = System.getProperty("HFM_CLUSTER");
    static String HFM_APPLICATION = System.getProperty("HFM_APPLICATION");
    static private String ssoToken;
    static private SessionOM sessionOM;
    public static void main(String[] args) throws HFMException {
        // Check if all System properties and arguments are set
        if(args.length != 4 || EPM_ORACLE_INSTANCE == null || HFM_CLUSTER
== null || HFM_APPLICATION == null || (!args[2].equals("1") &&
!args[2].equals("2") && !args[2].equals("3") ) ) {
                System.out.println("Usage: java -
DEPM_ORACLE_INSTANCE=<pathToOracleEPMInstance> -
DHFM_CLUSTER=<HFMClustername> -DHFM_APPLICATION=<HFMApplicationname>
runConsol <USER> <PASSWORD> <ConsolidationType> <POV>");
                System.out.println("  With <ConsolidationType>: 1 -
Consolidate (Impacted)  2 - Consolidate All with Data  3 - Consolidate
All");
                System.exit(0);
        }
```

User name, password, the consolidation type and the point of view, which are command line arguments are stored into variables:

```
        String username = args[0];
        String password = args[1];
        int consolidationType = Integer.parseInt(args[2]);
        WEBOMDATAGRIDTASKMASKENUM consolidationTypeString =
WEBOMDATAGRIDTASKMASKENUM.WEBOM_DATAGRID_TASK_CONSOLIDATE;
        String pov = args[3];
        switch ( consolidationType ) {
                case 1: consolidationTypeString =
WEBOMDATAGRIDTASKMASKENUM.WEBOM_DATAGRID_TASK_CONSOLIDATE;
                        break;
                case 2: consolidationTypeString =
WEBOMDATAGRIDTASKMASKENUM.WEBOM_DATAGRID_TASK_CONSOLIDATEALLWITHDATA;
                        break;
                case 3: consolidationTypeString =
WEBOMDATAGRIDTASKMASKENUM.WEBOM_DATAGRID_TASK_CONSOLIDATEALL;
                        break;
        }
```

Afterwards an Oracle EPM Session is created:

```
        try {
            Map context = new HashMap(5);
            CSSSystem system = CSSSystem.getInstance(context,
EPM_ORACLE_INSTANCE);
            CSSAPIIF css = system.getCSSAPI();
            CSSUserIF user = css.authenticate(context, username,
password);
            ssoToken = user.getToken();
        } catch (CSSException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

This is followed by the creation of a session in HFM:

```
        try {
                sessionOM = new SessionOM();
                SessionInfo session = sessionOM.createSession(ssoToken,
Locale.ENGLISH, HFM_CLUSTER, HFM_APPLICATION);
```

And finally the launch of the Consolidation:

```
                DataOM dataOM = new DataOM(session);
                List<String> povs = new ArrayList<String>(1);
                povs.add(pov);
                System.out.println("Running consolidation with POV: " +
pov);
                ServerTaskInfo taskInfo =
dataOM.executeServerTask(consolidationTypeString, povs);
```

Obviously, we need to close our session after we are done:

```
                sessionOM.closeSession(session);
        } catch (HFMException e) {
                System.out.println(e.getMessage());
        }
```

Now we have a rough idea what the code does, but how can we run the program?

Before even running the Java compiler, the environment needs to be setup, so the compiler as well as the Java Virtual Machine can find all required libraries and environmental information. Here are scripts for Unix and Windows environments, which can be used to setup the environment.

Download the script to setup the environment on UNIX.
Download the script to setup the environment on Windows.

Make sure you change the ORACLE_MIDDLEWARE_HOME variable in the first line according to the path in your environment.

After the environment is setup, Java bytecode can be generated running the compiler. The syntax is slightly different on Windows and Unix.

Windows: `javac -cp %EPM_JAVA_CLASSPATH% runConsol.java`
UNIX: `javac -cp $EPM_JAVA_CLASSPATH runConsol.java`

Now you can launch a Consolidation using the HFM Java API.

Windows:

```
java -cp %EPM_JAVA_CLASSPATH% -
DEPM_ORACLE_INSTANCE=<PATH_TO_EPM_ORACLE_INSTANCE> -
DHFM_CLUSTER=<HFM_CLUSTER_NAME> -
DHFM_APPLICATION=<HFM_APPLICATION_NAME> runConsol <USER> <PASSWORD> 1
"S#<SCENARIO>.Y#<YEAR>.P#<PERIOD>.E#<ENTITY>"
```

UNIX:

```
java -cp $EPM_JAVA_CLASSPATH -
DEPM_ORACLE_INSTANCE=<PATH_TO_EPM_ORACLE_INSTANCE> -
DHFM_CLUSTER=<HFM_CLUSTER_NAME> -
DHFM_APPLICATION=<HFM_APPLICATION_NAME> runConsol <USER> <PASSWORD> 1
"S#<SCENARIO>.Y#<YEAR>.P#<PERIOD>.E#<ENTITY>"
```

Make sure you run the above command on a single line without any line breaks.

While the Consolidation is running the progress can be followed in the applications “Running Tasks” screen.

About US-Analytics

US-Analytics is a full-service consulting firm specialized in Oracle Enterprise Performance Management and Business Intelligence solutions. Applying decades of experience along with advanced degrees and certifications, our team of functional and technical experts have helped hundreds of the nation's largest and brightest companies bridge the gap between business goals and IT deliverables.

To ensure end-to-end coverage of your technology, we provide a complete range of services: process and advisory, infrastructure, implementations, upgrades and migrations, training, and managed services.

Learn more at **www.us-analytics.com**.

Dallas, Texas
600 East Las Colinas Blvd.
Suite 2222
Irving, TX 75039

Houston, Texas
2500 CityWest Blvd.
Suite 300
Houston, TX 77042

info@us-analytics.com
877.828.USAS