topdown
CONSULTING

Extending ODI: Hyperion Automation & Error Trapping

WHITEPAPER

While every attempt has been made to ensure that the information in this document is accurate and complete, some typographical errors or technical inaccuracies may exist. TopDown Consulting does not accept responsibility for any kind of loss resulting from the use of information contained in this document. The information contained in this document is subject to change without notice.

This edition published May 2012.

TABLE OF CONTENTS

About The Author

Opal Alapat is a Services Engineering Manager at TopDown Consulting. She has over 12 years experience working with Oracle Hyperion applications. She has exceptional problem solving, analysis, and architecting skills with a strong technical background and years of financial experience across multiple industries. She specializes in architecture and management of Enterprise Performance Management and data integration solutions, specifically related to the Hyperion version 11 suite.

TopDown Consulting Helps a Major Retailer Build an Automation Solution using Oracle Data Integrator

Executive Summary

One of the keys to creating an efficient (and happy) software support team is the generation of administration processes that can be easily utilized, maintained, and understood. Yet, the complex EPM architecture of today's world makes it difficult to achieve this utopia. Companies that lack a strategy and knowhow (or in some cases, consulting budget) in this area rely on piecemeal administrative processes that require regular manual human intervention, inefficient 24/7 support, and wasted hours chasing down error messages – leading inevitably to frustrated and exhausted employees. Often times, hundreds of steps, scripts, and procedures make up the totality of administration required to maintain an enterprise-wide EPM solution. If a strategy is not executed that fully thinks through how the administrative team left behind will maintain these processes, the result is a stressed and confused support team in reactive mode instead of proactive mode.

"Because of lack of information, processes, and tools, through 2012, more than 35 percent of the top 5,000 global companies will regularly fail to make insightful decisions about significant changes in their business and markets."

—"Gartner Reveals Five Business Intelligence Predictions for 2009 and Beyond"

Gartner, Inc. January 2009

The good news is that there are tools that come bundled with the EPM suite that are both compatible with each of the software components and automatable. Oracle Data Integrator ("ODI") is one such tool that can be extended to create intuitive, push button interfaces that will make life easier. And due to ODI's scalable infrastructure and compatibility with most Oracle EPM toolsets, ODI is also an easy choice for an automation strategy.

This white paper examines the role ODI played on a new EPM implementation at a major retailer. By examining how to ODI can be leveraged beyond its ETL capabilities, companies can become more resourceful with their existing tool sets and move onto a brighter future of support. In this case study, we will explore how ODI was used to create push-button processes needed for an Oracle Hyperion EPM architecture.

"Our experience has shown that with a combination of Oracle Hyperion technology, world-class engineering, and clear understanding of each client's specific needs and objectives, these challenges can be met with certainty and consistency." Juan Porter, founder and president, TopDown Consulting

A Quick Primer on Oracle Data Integrator

In October 2006, a data integration specialist company called Sunopsis was acquired by Oracle. Sunopsis was purchased due to its high performance, cost effective data integration tools. Oracle was impressed by the Sunopsis architecture and began working on plans to fit Sunopsis into their Fusion Middleware family.

Oracle created a strategy to make the Sunopsis tool (then rebranded to "Oracle Data Integrator") compatible with their BI, MDM, and EPM toolsets. Looking for a more strategic and scalable technology than Hyperion Application Link ("HAL"), ODI seemed like a great fit. They then leveraged the API's in the Knowledge Module templates, and, combined with Jython code, created harmony between the toolsets. And so the marriage between ODI and EPM began.

Over the years, ODI has seen some vast improvements in the Knowledge Module templates, and Oracle has added in integration for ERP and other types of 3rd party systems. In 11g, Oracle revamped the user interface for ODI and reengineered the architecture to allow more control through web interfaces. As the EPM software evolved, ODI has also evolved to take advantage of the new features.

Companies still on EPM 9.x versions of Hyperion software have been faced with the challenge of moving past HAL to ODI. HAL provided a lighter footprint than ODI and was very capable of providing simple automation for maintenance processes. But ODI takes all of this a step further, offering scheduled automation, data integration options, and a Service-Oriented Architecture ("SOA"). As companies realize they need to upgrade their EPM software to keep up with current trends and functionality, they are also required to upgrade from HAL to ODI. This has left some wondering what to do with this new tool. While there is no direct upgrade path from HAL to ODI - they are completely different software technologies - with the right skill set and some initial ramp-up training, companies can easily make the transition and feel confident and pragmatic with their decision.

The Client

For more than 70 years, this retailer has been a leader in the world of make-up, skin care, and fragrance. With annual sales of over $8 billion in 2011, and a presence in over 150 countries and territories, this company has been leading the self-image industry all over the world. A family company, this retailer is driven by passion and a vision to bring beauty to all women.

A few years ago, this client launched a large initiative to bridge their global ERP systems into one, unified ledger. Based on this strategy, their software landscape has changed drastically. They introduced Oracle Hyperion into the mix to support their consolidations, planning, reporting, and master data management strategies.

The Situation

This retailer needed an enterprise-wide software architecture that offered greater flexibility, granularity and consistency, and included user-friendly features. This would help them improve insight into account balances, consolidated, global data, and consistent gross margin calculations by distribution channel and product line. The total solution also needed to lay the foundation for future projects, such as planning, forecasting, and detailed allocations.

The ODI automation solution was an afterthought to the original project. Over time, as the project scope expanded and supplementary technologies were added, the automation strategy suddenly became necessary and existent. Instead of needing a point solution to automate a few tasks, routines that updated metadata and data, rebuilt Essbase partitions, ran Oracle stored procedures, kicked off EIS loads, ran Essbase calculations, exported data to flat files, and notified different user groups of completions and failures came into focus. Several options were explored including job automation via Informatica and piecemeal options that utilized batch files and Ctrl-M jobs. However, it was critical that the ownership of these routines lie within the existing Oracle Hyperion support teams, that the technologies integrate well together, and that there was no manual intervention required. Enter ODI, and a winner quickly emerged.

The Solution

As this retailer learned more about their own international business processes, the upstream SAP rollout, and the capabilities of the Oracle toolsets, the scope of the project naturally increased. For example, the core team decided to inflate their chart of accounts and expand to additional dimensions. Both decisions provided more granularity for reporting. In addition, supplementary applications were added for planning, reporting, and allocations.

This client created a vision for automation that incorporated ideals from both the business and IT communities. In addition, adherence to IT protocol was a key to success. Their solution vision for automation included the following traits:

- "Lights Out", "Push Button" interface
- Zero footprint installations
- Business user participation
- On-Demand execution
- Proactive action
- Notifications

Simple, push button interfaces. In a nutshell, the administrator team wanted to ensure that they did not have to get involved in the underlying toolsets when running processes. And honestly speaking… why should they get involved? With everything on an administrator's plate, it makes sense to build an automation architecture that requires as little human involvement as necessary. This criterion was fulfilled by creating ODI packages that strung together processes. These packages were then turned into ODI scenarios that business users could execute.

Zero footprint installations, business user participation, and on-demand execution. Metadata Navigator, the web client for ODI, was the solution for all 3 of these requirements. By using a web interface, no installations were required by the business users needing to execute specific ODI scenarios. This web interface also allowed business user participation, because it did not violate IT protocols. By default, users were not allowed local administrator access to their computers (which is recommended for the desktop ODI client software). Finally, business users were able to run the ODI scenarios on-demand as necessary, since they were not automated through the ODI scheduling tool or any other 3rd party tool.

Proactive action and notifications. When processes completed, it was imperative that the appropriate teams were notified to keep to the busy close cycles. Kick out files from loading processes and failure details were also emailed, when necessary, to head off lengthy investigation time.

The Challenges

A Maintenance Dilemma

An enterprise-wide EPM architecture usually encompasses multiple tool sets with numerous maintenance processes. Common tasks include rebuilding applications from scratch, loading data, updating metadata, rebuilding Essbase partitions, running calculation scripts, running relational procedures, exporting data, etc. These tasks are usually split across the various toolsets with no clear way to integrate them.

This challenge presented itself at this retailer as they rolled out a massive architecture including SAP, Informatica, Oracle, Hyperion Financial Management ("HFM"), Extended Analytics ("EA"), Hyperion Planning, Essbase (both BSO and ASO), Essbase Integration Studio ("EIS"), Data Relationship Management ("DRM"), Financial Data Quality Management ("FDM"), ODI, Financial Reports, and SmartView.

On one of the subprojects related to a new Essbase reporting and allocations system, a critical eye was taken to the maintenance required to support the tools: Oracle database, Essbase, EIS, DRM, Financial Reports, and SmartView. ODI was used to string together the processes, and this automation architecture laid the foundation for the remaining projects and technologies.

One way to string tasks together is to use the command line interface with ODI. The following list names a handful of EPM technologies that contain built-in command line functionality:

- Essbase (via MaxL)
- Essbase Integration Studio ("EIS")
- Essbase Studio
- Planning Outline Load Utility
- Calculation Manager
- Business Rules
- Data Relationship Management ("DRM")

In addition, ODI Scenarios can be called from a command line interface if you would like to schedule them with 3rd party software.

Finally, relational processes can be called through ODI procedures. Oracle stored procedures were called through the ODI procedures in this client's solution, tying in yet one more technology and alleviating the need for manual intervention.

Aimed with knowledge on how to properly integrate the toolsets, anyone can be resourceful with their EPM solution. It just takes some forethought and a bit of strategy.

The Project Challenges

Side note: On this project, ODI 10.1.3.6.x and EPM 11.1.1.3.x were implemented. Therefore, the project challenges listed below may be specific to these particular software versions.

During the course of the project, several challenges presented themselves. They included:

- Client software compatibility
- ODI variables
- ODI security
- Lights out processing

Client Software Compatibility. The ODI server was the primary server for most EPM automation activity. Although there are a few variant installation options when using ODI, this was the route the client was happy with given their specific needs. Due to this configuration, the Essbase, EIS, DRM, and Oracle clients had to be installed to the ODI server so they could be called through the ODI routines. This presented the first challenge.

The environment variable paths were changed during the installation of each client. In particular, ARBORPATH, ISHOME, and PATH were directly affected, creating the client components to stop working. An Oracle support case was logged and after a couple of weeks of trying several path configurations, the proper values were set and documented. This was a minor hiccup for the project, but one that carries with it a good lesson for those wishing to go down this installation path.

ODI Variables. ODI variables can be used on both a global and project level. At this client, they were used at the project level. There are 3 action types that are available: Not Persistent, Latest Value, and Historize. However, the true differences between the 3 action types are not fully documented, adding to confusion on how they should be used in different situations. The challenge faced on this particular project related to:

- How to use the same ODI variable in multiple Scenarios that run concurrently
- How to use the same ODI variable in nested Scenarios that run concurrently

In the first instance, it was discovered that only action type Latest Value would hold a variable value independently of concurrently running Scenarios. Historize and Not Persistent would change the variable value according to what was assigned last chronologically.

In the second instance, it was discovered that no action type would correctly hold a variable value independently of concurrently running nested Scenarios. Careful measure was taken to ensure that nested Scenarios weren't used often.

ODI Security. In ODI 10g, LDAP/Active Directory security is not integrated. Therefore, native ODI security was implemented instead. The documentation supporting the ODI native security set-up created some challenges for the administrator team. Several blogs and the ODI discussion forum on the Oracle Technology Network were used to help work through the specific security details that would allow business users to execute scenarios from specific scenario folders.

Lights Out Processing. "Lights Out" processing was a very important requirement for this project. This term was defined as "no direct interaction with the underlying toolsets". Therefore, the ODI processes had to be created in such a way that interfacing with Oracle and EIS was not necessary except to troubleshoot issues. This was carried out by making the ODI processes completely "push button" – meaning a user could execute a Scenario and all of the independent steps to carry out that routine would be handled seamlessly by ODI. By installing all of the command line tools to the ODI server and ensuring that the account used to run the ODI agent was a network user with access to all required Oracle Hyperion servers, the executions were handled flawlessly.

Tips & Tricks

On every project, knowledge is gained that adds infinitely to the wisdom of the team. Some tips & tricks discovered at this client included:

- Environment variables
- Automation calls
- Checking file size
- MaxL return codes
- Encapsulated design
- Intelligent notifications

Environment Variables. Environment variables are very handy when combined with EPM tools. System environment variables were implemented on each ODI server (Development, QA, and Production) and provided relief for hard-coded constants per environment such as:

- Essbase decryption keys
- Local server drives
- Essbase server names

Automation Calls. Figuring out the exact syntax of the command line calls from the ODIOSCommand or the OS Command ODI objects is important when using multiple tools in your solution. The following are some example calls used in this project's solution (note that environment variables are wrapped in "%"):

- **EIS**

```
cmd.exe /c olapicmd.exe -f%ODIFilesPath%\EISScript\DoorPL\DoorCalc_
MemData_DelResRepZero.cbs
```

- **MaxL (w/ Essbase decryption)**

```
cmd.exe /c essmsh.exe -D %ODIFilesPath%\MaxLScript\DoorPL\DoorCalc_Init.
mxl %Esskey% %ODIFilesPath% %EsbDoorSvr%
```

Relational object calls can be embedded into ODI procedures. Below is an example of how an Oracle stored procedure can be called from an ODI procedure.

- **Oracle stored procedures**

```
BEGIN
FDR_ACT_MON_ST_DATA_POP_DE();
END;
```

Checking File Size. When working with files, sometimes one needs the ability to check a file size to ensure it is greater than 0Kb before taking an action. A very quick and easy way to handle this is through a combination of a batch file and the "KO" ODI return code. A batch file with one line of code: `exit %~z1` will send the size of the file back to ODI. If the file size is larger than 0Kb, a return code of non-zero will be sent – this combined with "KO" will allow a secondary action to be taken based on this result.

MaxL Return Codes. When writing MaxL script, return codes should be used as much as possible, as these will return back to ODI. These should be assigned within the iferror statements - a different code should be attached to different errors. When ODI returns these back to the user within the error notification, it will save precious time hunting down the cause of the tool failure. An example of an iferror section is shown below:

```
/* Login error handler label */
define label ‘LoginError’ ;
spool off;
logout;
set returncode = 900;
exit $returncode;
```

Encapsulated Design. In general, an object-oriented or encapsulated design is better for maintenance overall, as it can dramatically reduce replicated code and objects in an ODI architecture. For instance, imagine how many times you will need to send out email notifications. If you have several dozen processes, do you really want to duplicate the email notification step several dozen times (and then edit all of those when a change is made)? Creating a set of processes that can be recycled as nested Scenarios within other routines is a strategic way to reduce maintenance.

A word of caution - "encapsulated" design should be used minimally with 10g when ODI variables are used. Please see the "ODI Variables" segment within "The Project Challenges" section for more information.

Intelligent Notifications. Most customers have testing environments in addition to their production environment. In an optimal situation, you will have 3 environments: Development, Test, and Production. When you create notifications, do you really want the entire user base to receive a notification if you're running your routines in Development or Test? Ensuring that your notifications are intelligent is easily handled and requires some simple planning and some ODI variables. One way to handle this is by using a different context name in each ODI environment. You can then use ODI variable evaluation logic on the contexts used within each environment to ensure that the appropriate audience receives the notification. There are other methods possible as well – be creative!

Conclusion

The ultimate automation strategy includes a set of maintainable, understandable processes that communicate well and do not require manual intervention. A piecemeal approach is a risky proposition because it wastes valuable time and energy. To make sure your organization continues to meet support demands, it needs a sound strategy and a maintainable architecture to help proactively manage maintenance processes.

Think a few years down the line, 5 maybe even 10—how many processes will your organization have then? As the power of EPM technology is unleashed, additional applications are added to capitalize on the ROI from the initial software investment. What new business, data, and technology requirements will your organization face that demand timely, efficient, and reliable processes?

Results

TopDown Consulting helped a major retailer achieve its goal of creating an automation solution tailored to their specific needs, while also keeping in mind feasibility and maintainability. The project's successes include:

- A successful set of projects and a fruitful collaboration
- A scalable architecture for other, new EPM projects
- A strong foundation for implementing automation tasks for other, new EPM projects
- A favorable view of TopDown, Oracle Hyperion software, and project implementation

Learn More

Learn more about TopDown Consulting's wide array of consulting services for improved value, accountability, performance, and quality. Visit us at www.topdownconsulting.com or call (888) 644-8445.

About TopDown Consulting

About TopDown Consulting, Inc. Founded in 2000, TopDown Consulting is the acknowledged leader in designing, implementing, and deploying EPM solutions. TopDown has the experience, expertise, and proven approach to deliver successful implementations for Global 2000 clients. Our consultants average 20 years of Hyperion and industry experience with a complete range of skill sets, including: solution and system architects, CPAs, finance executives, MBAs, analysts, and project managers. TopDown's proven Project Success Methodology provides a customizable strategic framework for guiding and measuring project initiatives, enabling us to deliver solutions that meet our clients' current and future business and technology needs and their unique corporate culture.

WHITEPAPER

TopDown Consulting, Inc. serves clients nationally and internationally from our San Francisco headquarters.
For more information or to inquire about our services, please contact us.

TOPDOWN CONSULTING, INC.

530 Divisadero Street #310
San Francisco, CA 941177

Phone: (888) 644-8445 (Calls from within US and Canada)
Phone: (617) 576-8445 (Outside US and Canada)

Email: info@topdownconsulting.com
Web: www.topdownconsulting.com