



# APPLICATION STACK VIRTUALIZATION

WHITE PAPER

# TABLE OF CONTENTS

<b>Problem Overview</b> .....	3
<b>Problem Statement</b> .....	3
<b>Why is Refreshing Hard?</b> .....	4
The Data. ....	4
The Application .....	4
The Configuration. ....	5
<b>Impact</b> .....	5
Limited Environments .....	5
Dated Environments .....	5
Lack of Archived Applications .....	5
<b>Application Refresh with Delphix</b> .....	6
Database Refresh with Delphix .....	6
Data File Refresh with Delphix .....	6
Application Binary Refresh .....	6
Application Configuration .....	7
<b>Customer Example: Delphix for EBS</b> .....	7
dbTier Virtualization .....	7
appsTier Virtualization .....	8
Refresh Process .....	8
<b>Delphix Impact on the Application Stack</b> .....	9

## PROBLEM OVERVIEW

Packaged application development teams frequently operate with limited testing environments due to time and labor constraints. For each application, development and quality assurance need an environment that properly matches the production system: an application server that is running the same version application as in production, that is configured the same way, and has the same data files. Further, proper testing requires a database server that is configured the same as the production database server and is loaded with the proper data in production. Setting up these environments requires cross-functional expertise, expensive hardware, and repetitive, mechanical tasks performed by highly paid employees. While production systems change frequently—in terms of data, configuration, and application version—the copies are rarely refreshed to match these systems. The time, cost, and complexity of fully refreshing these copies prevents frequent refreshes which means that companies are trapped between two bad options—poor test environments or wasted time. Poor test environments arise when developers and QA engineers run their tests on an environment that no longer mimics production. Wasted team efforts come when an inordinate amount of time and energy is spent to keep these systems updated. Furthermore, given the difficulty in setting these systems up in the first place, they are rarely decommissioned, even if they are not being used for active testing. In addition to low quality testing from unrefreshed systems, there is a large management cost associated with maintaining an infrequently use standing inventory of servers and applications.

## RISING BUSINESS RELIANCE ON IT

To maintain their competitive edge, businesses are continually looking for better ways to deliver new, innovative products and services. Frequently the business side of the house outpaces IT, leading to delayed products and services and tension between business and IT. IT increasingly is looked at as the best way to deliver future innovation and IT organizations are in the process of trying to convert from being seen as inhibitors to enablers. Fast, agile IT organizations deliver business innovation.

IT needs to respond to business needs and these business needs frequently require new, upgraded, or modified applications. For IT to properly satisfy these needs, they need to be able to develop, test, and bring new services online at a rapid pace. Proper testing is crucial—while IT is trying to move faster, they also need to instill a culture of quality to ensure that business innovation isn't hampered by poor quality applications or data. The demand for high quality applications drives a frequent IT request—application refresh.

When new modules, applications, or custom code are developed to support new business processes, test systems need to be properly update with the latest application, data, and configuration to make sure that the updates are properly tested. Business quality demands that IT use frequently refreshed systems to develop quality services. Business schedules demand IT turn around new services in an increasingly fast pace. In organizations today these two goals are conflicting as the increase in quality from developing on refreshed systems comes with an associated time cost. While the concept of refreshing a test system with new data, applications, and configuration sounds simple, in practice it takes weeks and is infrequently done due to the effort involved.

# ENVIRONMENT REFRESH CHALLENGES

Providing a test system to application developers that works in a non-production environment, has the relevant application binaries and data (both flat file and database), and proper configuration can be a daunting exercise. Provisioning and refreshing each segment of the application stack requires extensive time and knowledge across application, infrastructure, DBA, and backup teams. Without proper orchestration tools in place, the process of refreshing is highly manual—teams must identify the right files, bundle, copy, and deliver those files, and then run all relevant post-processing. Each of the segments are different and require different treatment.

## The Data

Copying and moving data in an age of extreme data growth is hard. Properly testing a modified application to prove that it will work when deployed in production requires moving all relevant data to the test system—both flat file data and all database data. Together, these can easily measure in the terabytes—affecting production systems while this data is extracted, bogging down the network while it is moved, and requiring significant work to actually deliver and use it. Further, proper testing requires that the different data sets all be synchronized by time. Many applications rely on transactions that span databases and files, meaning that an unsynchronized system can generate false errors or ignore true issues in the data.

Extreme data growth leads to companies frequently provisioning new storage when refreshing a system's data. Provisioning new storage can be a weeks or months long process depending on who is involved, and whether new hardware needs to be purchased. Once the storage for data is in place, databases need to be created and validated with the application, requiring hands-on work from DBA teams.

In practice, this simple concept of copying, moving, and delivering data can take even the world's best IT organization weeks or months. When product teams want to deliver new business products and services in that time frame, IT is at odds with business goals.

## The Application

Packaged applications have grown in complexity and size while cementing their importance to business systems. IT teams move copies of these applications—sometimes hundreds of gigabytes—over the network to refresh the existing application stacks.

Further, teams want to keep historic copies of applications—both for before/after testing purposes and compliance reasons. Refreshing or provisioning a new application stack requires making sure the the application is provisioned at the same point in time as all the data.

When companies choose to store additional copies for compliance or historical testing, they run up against storage, timing, and bandwidth issues. To effectively store applications for compliance or testing purposes, the applications need to be continually copied and stored in a safe location, and the relevant data (files and databases) similarly timestamped and stored. The overhead of maintaining consistency between application binaries and file sets and the sheer volume of data to be moved and stored on a regular basis means that most companies don't do it—leaving them short-handed when it comes to testing and at significant compliance risk.

## The Configuration

While labor-intensive, the process of copying and moving the application and data to refresh the development systems is fairly straightforward. However, just creating a refreshed copy of a full application stack in development that matches production won't satisfy testing purposes. These systems need to match the configuration of production systems in terms of functionality, but have to work in a pre-production space. This means updating the system to work with other non-production systems, non-prod workflows, and so on. This post-processing can result in huge "playbooks" of manual steps that need to be executed to deliver a refreshed environment. Without this lengthy period of system modification, the "refreshed" systems aren't actually useable.

# IMPACT ON APPLICATION DEVELOPMENT

Due to the difficulties and time involved in application refreshes, most companies do limited refreshes, or none at all. Further, new non-prod development systems are rarely provisioned. This leads to several issues.

## Limited Environments

Only a small number of systems are ever provisioned because of the initial (hardware, set-up) cost that goes into the environment and the extremely high operating cost of maintaining and refreshing these environments. This means that development teams are forced to fight over environments, sometimes with dozens of developers trying to use the same environment for application testing. This reduces the integrity of the testing as developers frequently step on top of each other's work. Further, limited environments severely limit parallel testing, leading to protracted development schedules.

## Outdated Environments

Because of the high refresh cost, the dev and test environments are often out of date, with configurations and data that does not match production. To circumvent this issue, developers and testers frequently create their own test data or test systems to validate their code. This additional, non-product work leads to much longer development cycles. Worse, without proper configurations and testing, the likelihood of bugs arising from a mismatch of configuration and data in production is quite high, leading to system downtime and high resolution costs.











## Lack of Archived Applications

With the high cost of storing 100+ gigabyte application copies, many-terabyte databases, and associated configuration information, and ensuring time synchronization between the three, companies opt against storing old copies of their application stack. Testing teams frequently need to compare current application functionality against previous functionality and extract data from previous systems. More crucially, after major systems upgrades or changes, auditors frequently force companies to prove that they are processing and reporting on data the same way they previously were.

When companies opt against storing old applications, they expose themselves to significant risk—the inability to recover old environments for testing mean that historic data and processes are lost forever. The inability to recover old environments for audits and compliance reasons mean that they may fail compliance checks, expose themselves to hefty fines, or be required shut down business to rebuild old environments.

# APPLICATION REFRESH WITH DELPHIX

Much like a hypervisor abstracts compute resources to create virtual machines, the Delphix Agile Data Platform abstracts data from hardware and storage to create virtual application instances, virtual files, and virtual databases. By virtualizing the key components of the application stack—the application binaries, data files, and databases—Delphix can quickly deliver refreshed copies of the entire stack regardless of size or complexity. Like a hypervisor abstracts compute resources to create virtual machines, the Delphix Agile Data Platform abstracts data from hardware and storage to create virtual application instances, virtual files, and virtual databases. By virtualizing the key components of the application stack—the application binaries, data files, and databases—Delphix can quickly deliver refreshed copies of the entire stack regardless of size or complexity.

	Physical Provisioning	Size / Time	DELPHIX® Virtual Provisioning / Refresh	Size / Time
Application	 Manually Tar Manually Move Manually Untar	Hundreds of GB / Days	 Automatic Refresh	Few GB / Minutes
Database	 Manually Backup RMAN Backup	Many TB / Weeks	 Automatic	<1 TB / Minutes
Files	 Manually Tar Manually Move Manually Untar	GB / Days	 Automatic	Few MB / Minutes
Configuration	 Manually Configure Custom Scripts Custom Playbooks	MB / Weeks	 Automatic	MB / Minutes
Version(s)	 Version 1.8.7 11/1/2013 11:45 pm		 Version 1.8.7 Version 1.9.0 Version 2.0.0 Version 2.0.1 6/1/2014	

## Database Refresh with Delphix

Delphix connects to production database systems and creates a compressed, master copy of all production data. After creating this master copy, Delphix receives all change data from the connected databases. This continuous flow of data allows database copies to be provisioned from any point in time. Just as importantly, the Delphix provision or refresh can be done in minutes. Rather than creating a full physical copy of all the data in a database copy for an application stack, Delphix only stores unique data across the copies—meaning that the refresh process can be done quickly and efficiently. With powerful developer controls, data can be refreshed or reset quickly and without an IT request.

Delphix connects to a heterogeneous set of databases, supporting Oracle, Oracle RAC, Oracle Exadata, Microsoft SQL Server, and PostgreSQL, among others. This heterogeneous support allows Delphix to streamline application refresh across portfolios, regardless of the underlying technology stack.

## Data File Refresh with Delphix

Delphix connects to and virtualizes files and file systems, creating a series of snapshots view of files as they change over time. This snapshot period can be configured to allow whatever level of granularity is desired, or timed to align with snapshots of the database and application binaries. These files are heavily compressed and file sets can be refreshed or reset in minutes.

## Application Binary Refresh

Much like databases and files, Delphix can regularly copy, version, and compress application binaries associated with an application stack. By versioning these application binaries alongside the relevant data, Delphix can ensure that the different pieces work in harmony—providing a system that has consistent data and functionality.

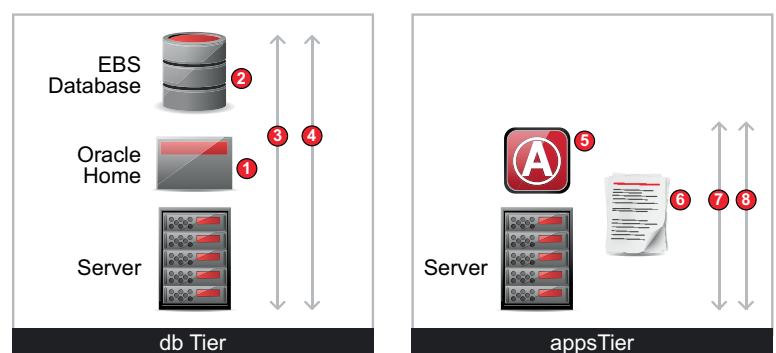
## Application Configuration

In addition to maintaining copies of the relevant applications and data, Delphix provides the ability to do post-processing or configuration on the application stack once it has been provisioned. Delphix's ability to constantly version applications and data allows developers to choose a proper refresh point for the application stack. The ability to deliver the data in binaries in minutes, automatically, eliminates hour or days of waiting time and significant infrastructure, app dev, and DBA team work. The ability to script configuration after data-delivery automates the final pieces of application refresh. Delphix provides the ability to integrate post-deployment scripts that can change configurations, variables, or pointers to additional systems.

With the ability to quickly version, efficiently deliver, and automatically configure data and applications, developers can refresh systems in minutes. Further, as testing is often destructive to the data, Delphix allows full application stack reset, returning the full application to a pristine state without any infrastructure team involvement.

# CUSTOMER EXAMPLE: DELPHIX FOR EBS

Oracle EBS is a specific instance in which the previously described problem arises. Oracle EBS is widespread in enterprises—used extensively for ERP and CRM functions. As with most large enterprise systems, there is a constant series of updates and modifications performed by the EBS development team, leading to the outlined problems.



**DELPHIX®**

1. Provision dbTechStack
2. Provision database
3. Run post-deployment script
4. Run custom scripts
5. Provision appsTier binaries
6. Provision appsTier files
7. Run post-deployment script
8. Run custom scripts

Delphix can quickly and effectively refresh Oracle EBS. This capability automates the data refresh, application binary refresh, and the post-deployment configuration that need to occur to provide fast and easy refreshes of EBS systems for development and testing purposes.

Delphix delivers refreshed EBS instances by automating the following data, application, and configuration steps.

### dbTier Virtualization

“dbTier” is the Oracle EBS term for everything that is related to the database server—including Oracle Home (dbTechStack), the EBS database itself (which is routinely tens of terabytes or more), the listener, amongst other components.

1. **EBS Database:** Delphix creates a master copy of the EBS database. This master copy is a highly compressed, and is constantly updated with changes from the source database as they occur.
2. **Oracle Home Files:** Delphix creates a compressed, master copy of the Oracle Home files—the software that actually runs the database and monitors for changes as they occur.

### appsTier Virtualization

“appsTier” is the Oracle EBS term for everything that runs on the application server.

1. **Binaries:** Delphix creates a compressed master copy of the application binaries and configuration files. These change relatively infrequently (when patches are applied, for example) but are monitored for changes at regular intervals. Binaries and files are copied on the same schedule, ensuring consistency.

### Refresh Process

Delphix provisions lightweight virtual copies of all the collected data (application, files, and databases) to target EBS systems. These virtual copies are extremely efficient (generally less than 10% of the non-virtualized file size) and can be provisioned in minutes, automatically. The target systems are the non-production systems that are refreshed to support on-going development and testing. To refresh or provision application environments, Delphix executes the following tasks:

1. **dbTechStack Provisioning:** Delphix provisions a virtual copy of the dbTechStack to the target database server. This provides all the proper components for running Oracle EBS Database Server.
2. **Run Post-Deployment Script:** Delphix runs `adcfgclone.pl` to configure the Oracle Home instance on the database server.
3. **Starts Database Listener**

### Large Retailer Eliminates Year+ Employee Work

A large retailer uses Delphix to accelerate EBS refresh by 50%. The customer has 15 EBS instances used for development and test and has to refresh the environments one weekly. The customer historically used Delphix to refresh their EBS databases, but still was doing 8 hours of manual work to refresh the application, and apply their post-processing work.

- With Delphix’s automated EBS refresh in place, the customer was able to reduce their application refresh time from 8 hours to 4 hours, freeing up over 3,100 hours annually.
- Physical EBS instances and Virtual EBS instances perform at the same level.
- Refresh of the database and application (not including the post-processing steps) performed in under 45 minutes.
- Delphix application virtualization, eliminates 97% of storage requirements for EBS instances.



4. **Run Custom Scripts:** Depending on a customer's environment, there may be additional configuration steps that are required on the database server. After the first three steps, custom configuration scripts can be called by Delphix.
5. **Provision Database:** When the Oracle Home is provisioned and configured, a virtual copy of the database (all the custom data) is provisioned to the database server. At this point, the database server update is complete.
6. **Provision appsTier binaries and files:** Delphix provisions a time synchronized virtual copy of the application binaries and configuration files to the application server.
7. **Run Post-Deployment Script:** Delphix runs `adcfgclone.pl` to configure the application instance on the application server.
8. **Starts Application**
9. **Run Custom Scripts:** Depending on a customer's environment, there may be additional configuration steps that are required on the application server. After the previous three steps, custom configuration scripts can be called by Delphix. At this point, the application server is completely configured.

When a developer wishes to refresh an EBS application with Delphix, they select their instance in Delphix and click the refresh button. This refresh will automatically handle all of the above steps and fully refresh the instance in a matter of minutes. If the developer or tester simply wants to reset the instance (return to the last clean state), they use the reset button to just clear all new data.

## DELPHIX IMPACT ON THE APPLICATION STACK

By virtualizing the entire application stack, Delphix-powered teams can deliver business results faster, at higher quality, and with lower risk. Business is dependant upon IT to deliver required applications and services, and these applications and services are dependant upon timely and quality refreshes. The Delphix Agile Data Platform enables business through fast, effective refresh, allowing:

**Agile IT:** Delphix allows IT teams to become more agile—delivering value faster and more frequently by eliminating wait time associated with refreshing.

**High Quality Services:** Delphix enables application development teams to deliver higher quality projects. By allowing more frequent refresh and simple reset for application stacks, testing teams can properly validate upgrades, new modules, and new workflows.

**Control Compliance:** Delphix introduces application archiving, which allows IT teams to cheaply and easily store copies of entire application stacks, providing easy recovery if needed for compliance purposes.



## **Application Stack Virtualization**

May 2014

You can find the most up-to-date technical documentation at:

<http://www.delphix.com/support>

The Delphix Website also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

[help@delphix.com](mailto:help@delphix.com)

### **Delphix Corp.**

275 Middlefield Road, Suite 50

Menlo Park, CA 94025

[www.delphix.com](http://www.delphix.com)

© 2014 Delphix Corp. All rights reserved. Specifications subject to change without notice. All information in this data sheet is strictly confidential.

Please do not copy or distribute to other parties.