

# Best Practices for Achieving Business Continuity using Oracle Applications and Oracle Database Technology

*An Oracle White Paper*  
*September, 2005*

**NOTE:**

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Best Practices for Achieving Business Continuity using Oracle Applications and Oracle Database Technology

Note.....	2
Business Continuity for Oracle Applications .....	4
What Causes Downtime? .....	5
Unplanned Downtime: Computer Failures .....	6
Redundancy in the Middle Tier .....	7
Forms and SSA .....	7
Concurrent Managers .....	7
Managing the Middle Tier Grid .....	9
Redundancy in the Database Tier.....	9
Unplanned Downtime: Data Failures.....	10
Storage Failures .....	11
Automated Storage Management.....	11
Human Error / Security .....	11
Guarding against human error/system breaches.....	12
Recovering from human error.....	13
Data Corruption.....	15
H.A.R.D.....	15
Flash Backup and Recovery.....	15
Site Failures.....	16
Data Guard Redo Apply .....	16
Combining Physical Standby and Flashback Database .....	17
Other Components.....	17
Planned Downtime .....	18
Administrative / Data Changes .....	18
System Changes.....	19
Tech Stack Maintenance.....	19
Database Upgrades .....	20
Applications Patching and Upgrades .....	20
Oracle GSIAP Implementation.....	22
Middle Tiers.....	22
Concurrent Managers.....	23
Database Tier.....	24
Load Direction.....	24
Disaster Recovery .....	25
Patching the Environment .....	26
Conclusion.....	27

# Best Practices for Achieving Business Continuity using Oracle Applications and Oracle Database Technology

## **BUSINESS CONTINUITY FOR ORACLE APPLICATIONS**

Oracle recommends as a best practice implementing the Oracle Applications in a “global single database” configuration – one database holding all the data for all the application modules, to retrieve and analyze information as a coherent whole, and to simplify the overall operational structure of the enterprise. This has a side-effect of making the environment truly mission critical – all users, in all departments and all divisions, need this application to be running to do their jobs.

What happens when your users cannot access the Oracle Applications? For many areas of the company, productive work simply grinds to a halt. If you were able to plan the outage, the impact can be mitigated, but whether planned or unplanned the cost to the organization is significant.

Given the significance of the cost, there is great benefit to avoiding service outages. This drives you to create a Business Continuity plan – a complete set of solutions for resuming service within a bounded period of time after a partial or complete interruption, covering every aspect of the business – from utilities and communications to facilities to press relations, HR, and of course core IT service delivery. For core IT service delivery, you need to first identify critical business functions and the systems that serve them, then determine the activities or incidents that can disrupt those services. This paper focuses on these technical aspects of Business Continuity Planning for an Oracle Applications environment – architecting to protect those core services.

Avoiding IT service outages involves understanding the details of the infrastructure that delivers the service to the user community, determining where failure or maintenance of a component will result in failure to deliver the service, and architecting the environment to provide resilience at these points of failure.

Oracle provides a wide variety of solutions to provide resilient services within an infrastructure. Oracle has published a best practices blueprint called Maximum Availability Architecture (MAA) [1] – a fully integrated and proven blueprint for highly available systems. This paper applies MAA concepts and principles to the Oracle Applications, so show how to achieve the highest possible availability for this mission critical system. The versions addressed are Oracle Database 10g (10.1.0.4), and Oracle Applications 11.5.9 and 11.5.10 (the versions certified for

Oracle Database 10g). Please refer to [2] and [3] for documentation on Oracle Database 10g and Oracle Applications 11.5.10 respectively.

This paper is written for technical architects and administrators of an Oracle Applications implementation, and has information relevant to applications administrators, database administrators, system administrators, and network administrators.

The paper has five main sections:

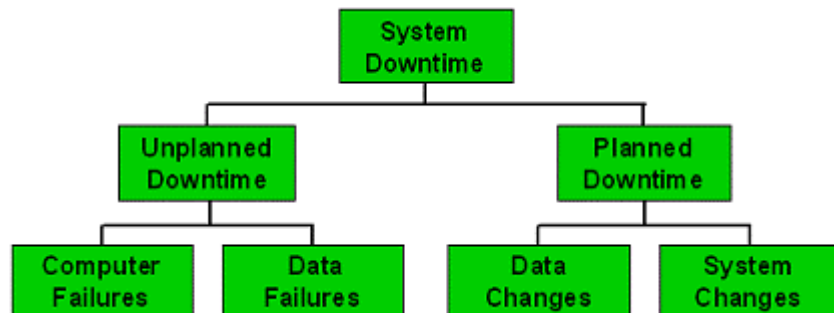
- Unplanned downtime – Computer failures: best practices for avoiding outages due to hardware issues
- Unplanned downtime – Data failures: best practices for keeping the data itself protected
- Planned downtime: best practices for reducing planned maintenance outages
- Making it Work: thoughts on keeping the environment highly available after it's implemented
- Oracle GSIAP Implementation: a description of our internal global single database environment, to serve as a case study.

Please check our Maximum Availability Architecture web site for the most current version of this paper:

<http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm>

## WHAT CAUSES DOWNTIME?

The challenge of designing a highly available IT infrastructure boils down to identifying and addressing all possible causes of downtime. Figure 1 classifies causes of outages into two primary categories – planned and unplanned downtime. While the same technology can sometimes be used to protect against each type, it's important to consider them separately to ensure complete coverage.

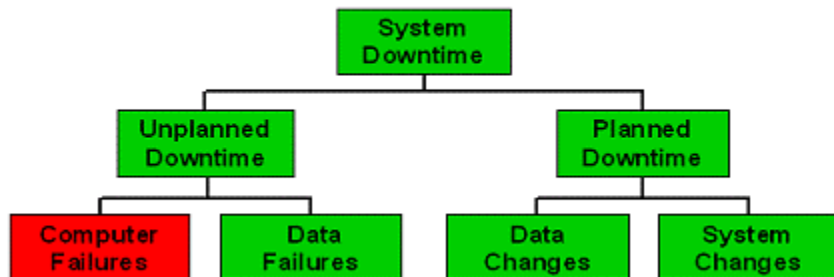


Unplanned downtime is generally caused by failure of part of the infrastructure required to deliver the service, and covers many areas – power, building services, networks, computers, storage, and the data itself. It is generally addressed by providing redundancy of the service, and mechanisms for switching between the redundant components in as transparent a manner as possible.

Planned downtime results from system and data changes that must be applied to the production system's infrastructure – configuration changes, data maintenance, patches, upgrades, and component replacement.

### **UNPLANNED DOWNTIME: COMPUTER FAILURES**

“Computer,” or server, failure is in general provided via local redundancy, then by a combination of properly protecting the resources and by directing or balancing the load across the resources – in essence, the Grid. The ultimate target is a large configuration of standardized, commodity-priced components for servers (especially middle tier) and disk.



For Oracle Applications, this translates to multiple middle tiers, well-configured DMZ and load balancing for the middle tiers, and Real Application Clusters (RAC) in the database tier. With this configuration, you can realize

- Flexibility in deploying hardware where it is needed
- Cost-effective solutions for increasing the availability of your system
- Less risk in capacity planning efforts, as major server swap-outs are no longer needed as business needs change
- Fault tolerance for server failures
- Management and isolation of workloads as needed according to the processing calendar

## **Redundancy in the Middle Tier**

Best practices for deploying the Oracle Applications middle tiers are to have many relatively small servers with very fast CPUs, plenty of memory, shared storage, and a hardware load balancer in front to scatter UI connection requests.

### **Forms and SSA**

There are two basic UI types in the Oracle Applications – Forms, and the Self-Service Applications (SSA) coded in the Oracle Applications Framework. Load distribution for Forms can be accomplished either via the Forms Metrics server or by configuring communications via the Forms servlet processes, then using the same distribution architecture as for SSA. Oracle recommends using the latter approach for consistency and ease of management.

Incoming connection requests are routed to web nodes based on rules in the router. This routing decision is made only once per session, as the user session stays with one node once started – if “sticks” to that node. The routing rules define how to balance requests across active nodes, how to deal with non-responsive nodes and with newly available nodes, and “stickiness”.

The Self-Service Apps use shared connections in a connection pool on the middle tier. To best manage resources on the middle tiers and in the database, sessions should be made to “time out” after a few minutes of inactivity. The system can be set to “passivate” a session before timing it out, thus allowing it to resume on a different middle tier when the user returns. Several SSA flows offer a “Save for Later” button, for the user to save their work.

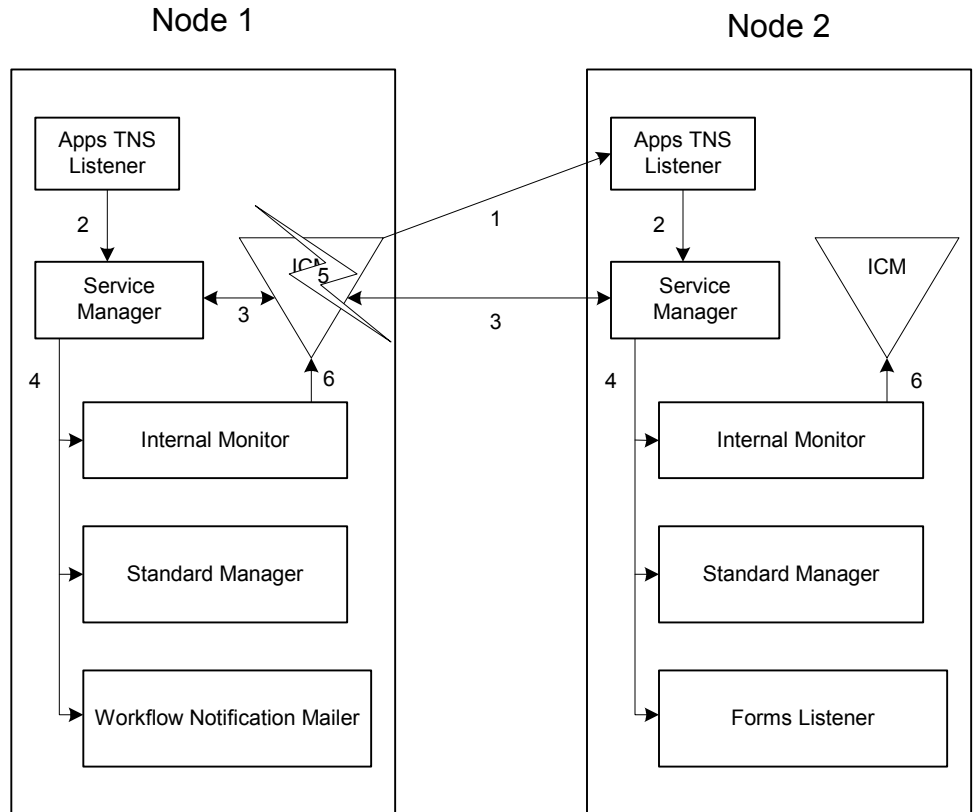
Forms sessions are also affected by timeouts. If a timeout is triggered, the user is prompted to re-authenticate. If that is done successfully, the Forms session continues where work left off. If re-authentication fails, the Forms session is terminated and work done since the last commit is lost.

For the middle tier, failover is most easily configured via the hardware load balancer. In the future, a mechanism similar to passivation may be able to provide fault tolerance at the middle tier level for Self-Service applications, but for now the users must re-establish their sessions on the node to which the load balancer has sent their new connection request.

### **Concurrent Managers**

Concurrent Managers (CM) can be deployed on the same type of hardware used for the other middle tiers, again using multiple small servers for this load. Load direction and failover for the Concurrent Managers are handled by mechanisms built within the CM framework. Generic Service Management (GSM) provides a fault tolerant processing framework, with primary and secondary middle tier nodes defined for each processing queue, the Internal Concurrent Manager (ICM) starting / restarting processes on each available server according to your configuration, and Internal Manager (IM) processes on every box monitoring to be sure the ICM is

alive. Please see the Oracle Applications System Administrator's Guide – Maintenance for detailed configuration information.



The ICM starts the rest of the services on each middle tier node, as defined in your configurations. A DBMS lock is created for each process. The lock is only released when the process's dedicated session process exits, and is used to verify the process is still "alive". It is important to set "SQL Net dead connection detection" to ensure failed processes are detected as quickly as possible.

The ICM starts Service Managers on each CM node, using the APPS TNS listener on remote nodes (1 and 2 in the diagram above). The ICM communicates with the Service Managers (3) to spawn and terminate the rest of the services (4), and manage GSM services locally. To manage GSM services, it calls control cartridges (basically object libraries) to start, stop, ping, suspend, resume, and verify the service.

If a middle tier fails, the ICM will start its queues on the middle tier defined as a secondary server. When the primary returns to service, the ICM will "fail" the queues back to the primary as jobs on the secondary end. During this time, jobs for the queue could be running on either system.



An Internal Monitor (IM) can be placed on any CM node. The Internal Monitors check to be sure the ICM is running. If it fails (5 in the above diagram), every IM will start an ICM (6). The “first” one will take over the ICM duties; the rest will notice a replacement is running and will terminate.

Configuring load direction starts with determining which workload should run on which servers. Oracle recommends at least a two-tier definition of Concurrent Manager queues, with one set of queues defining the “mission critical” programs that, at some point in the processing calendar, are the highest priority to finish their work, and the other set(s) of queues for “normal” processing. This framework can be used to define the number of workers running for each type of load, which can be dynamically changed according to the load being experienced on the servers. It can also be used to manage throughput if part of your grid fails, and you must selectively fail work over to another server.

### **Managing the Middle Tier Grid**

Deploying many small boxes in the middle tier could lead to complexity and longer planned maintenance outages unless you create a ‘farm’ mentality within the grid. Ease of management can be attained with a cookie-cutter approach to deployment – all servers alike, all servers configured the same at the OS level, all at the same patch level, and all share one copy of the Applications and tech stack code. Then replacing a failed unit is simple, defining a monitoring framework is simplified, and patching is done only once no matter how many servers are in the farm.

Sharing the Oracle Applications and tech stack code in the middle tiers is accomplished with an NFS mounted or network attached storage (NAS) solution. To avoid creating a new single point of failure, this solution should be redundant in a mission critical production environment, with a simple and well-tested failover process in place.

Depending on the size of your implementation, you may wish to create “server pools” for each type of work done in the middle tier, so they can be managed and monitored for the specific types of issues that may arise depending on the service offered.

### **Redundancy in the Database Tier**

Providing protection against server failure in the database tier is again done with physical redundancy, with Real Application Clusters (RAC) transparently providing access to your Oracle Applications database from several database servers. As in the middle tier, this gives the ability to continue to provide services to the users if one of the database servers fails, and gives a simple path for adding capacity in the database tier as the business grows.

Oracle recommends load for the UI programs be balanced across all available database instances for each connection type. The Oracle Applications configuration tool AutoConfig will generate a load balanced TNS entry for your

RAC nodes. Point to it via the Applications profile options “Tools ORACLE\_HOME TWO\_TASK” and “iAS ORACLE\_HOME TWO\_TASK”. Using this method, server-side load balancing is provided (the database listeners manage load distribution), and new requests are directed to surviving nodes if a database server dies.

Note: Some customers choose to direct UI sessions to specific database servers based on the application module being run, at the Application or Responsibility level, using profile options. If this method is used, failover is accomplished by changing the node name in the profile option to one of the surviving nodes.

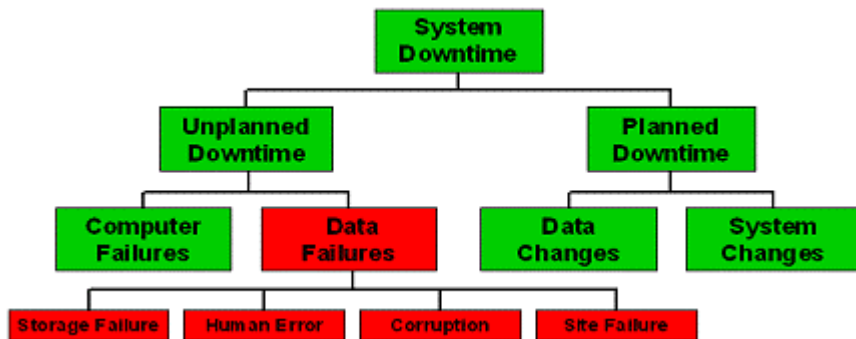
Concurrent Manager connections must be directed to specific database instances. All Concurrent Managers running on a given middle tier node must point to one specific database instance. Of course, more than one CM middle tier can point to a given database instance.

While most Concurrent Manager processes are asynchronous batch jobs, some are synchronous request servers whose database sessions must be running on every database server processing certain types of Forms sessions. These “Transaction Managers” create a unique requirement for at least a one-to-one mapping of Concurrent Manager middle tier nodes to database servers, including provision for continuing to provide database-level request processing if a middle tier fails.

Finally, you may wish to have more direct control of Concurrent Manager failover, routing the queues to instances chosen at the time of the outage based on current processing load. For instance, failing extra work over to an instance currently handling biweekly payroll processing may not be the best decision. To handle database instance failure in a custom manner, turn off “instance failure” for PCP by setting the profile option Concurrent: PCP Instance Check to OFF.

## **UNPLANNED DOWNTIME: DATA FAILURES**

A data failure is the damage, corruption, or complete loss of the Oracle Applications database. Causes may again be simply hardware failure, but can be more complex, resulting from human error, corruption from a fault in the tech stack, or complete site failure.



## Storage Failures

Over the last twenty years, disk capacity has grown by three orders of magnitude. In the early 1980's, a 200MB disk was state of the art. Today, a 200GB disk is common, and there's no end in sight for further growth. As capacity is growing, cost per megabyte has fallen dramatically. "Cheap disk" is now competitive with tape for operational backups. Disk-based backups offer the luxury of being online – available all the time, and with random access capabilities. This has allowed Oracle to rethink our backup and recovery strategies, which has resulted in reducing backup and recovery times that formerly took hours down to minutes. Backup and recovery are covered further under the "Data Corruption" section.

Oracle also provides a solution for disk management for the primary database itself, automating the implementation of storage layout best practices.

## Automated Storage Management

Traditionally, the process of deploying disks for an Oracle Applications database involved (over-)estimating the ultimate disk requirements for the system, mapping out an optimal layout according to intended use and protection mechanisms (e.g., mirroring), configuring it, deploying it, managing hot spots on it, etc. With Oracle Database 10g, Oracle offers the Automatic Storage Management (ASM) feature – a vertically integrated file system and volume manager. ASM "understands" the usage of the various files needed in an Oracle database, and is built to implement the well-tested SAME (stripe and mirror everything) disk layout method. It provides mirroring, spreads files across all available storage for optimal performance, and simplifies management of the disks and files.

ASM is a feature of the Oracle Database 10g database, and has been certified with the Oracle Applications. In particular, it makes the move to RAC simpler as you do not need to manage hundreds of data files on raw devices. See Metalink Note 312731.1 for more information.

## Human Error / Security

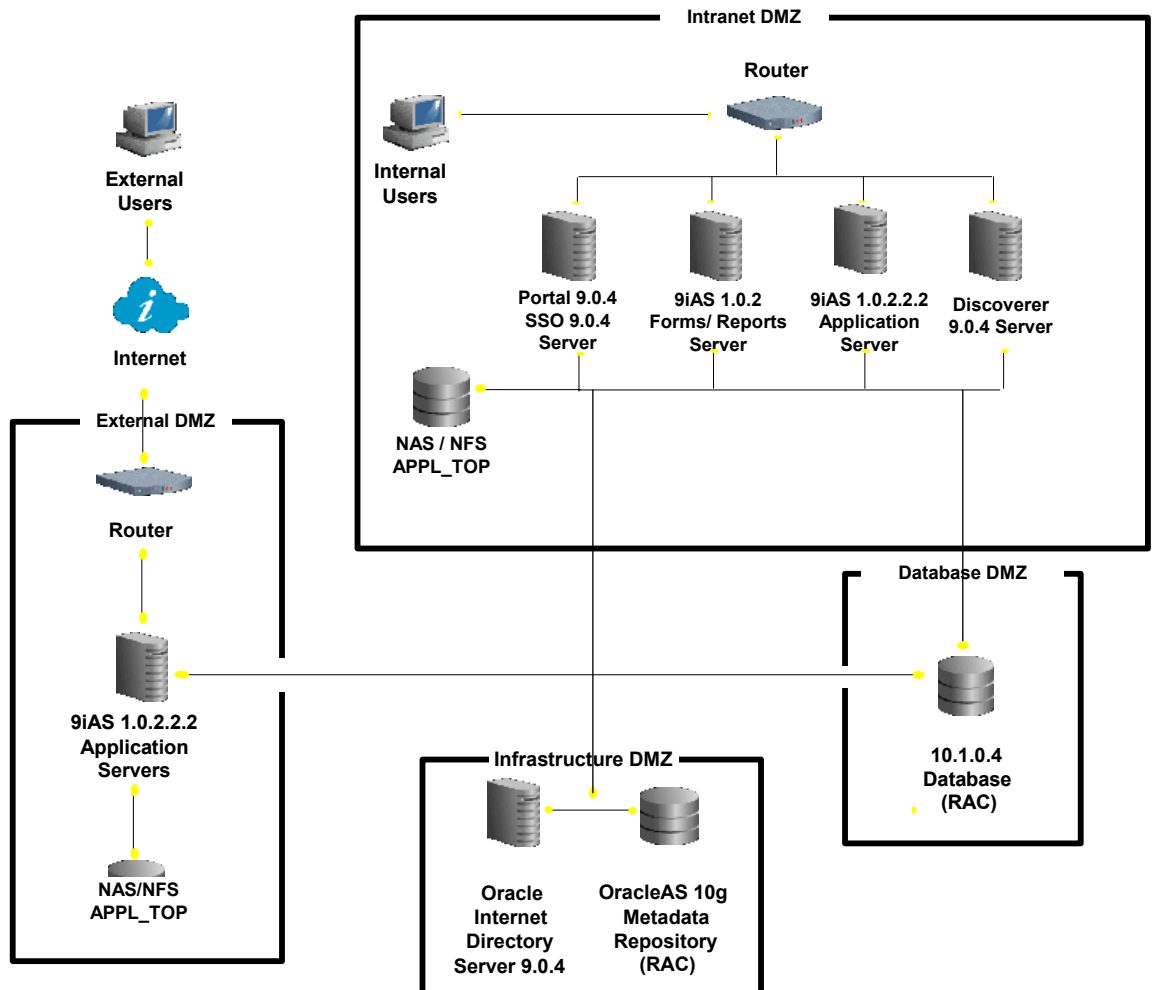
The most challenging unplanned outages can result from human error and security breaches. The best defense is to prevent unauthorized access to the production system, granting the minimum required privileges, and removing unnecessary

access paths. This helps prevent both accidentally performing an activity against a production database that was intended for a test system, and intentionally destructive actions. If it does occur, though, Oracle Database 10g provides some excellent mechanisms for recovering the database.

### Guarding against human error/system breaches

Excellent security practices are key to a successful implementation of the Oracle Applications. General best practices are:

- At each tier, restrict network access to required ports only. External middle tiers, intranet middle tiers, and the database servers should each be in their own network zone, protected by firewall technology. A sample environment is shown below, with the different types of servers each in their own protected LAN segment:



- Database access should be limited to known IP addresses, or the Apps' feature 'valid node checking' should be enabled.
- Operating systems should be hardened to remove potential access paths if a zone is breached.
  - Reduce Unix services to the minimum required for the type of server.
  - Set and manage file permissions correctly.
  - Control access explicitly to the fewest required users. Audit access to critical components, and review the audit trails.
  - Use SSL to encrypt network communications.
  - Control all access to the database server – have none from client desktops. Use terminal servers to perform client-server activities, and limit access.
- Manage passwords – disable unused accounts, change default passwords, ensure users create good user passwords (minimum length, level of complexity), do not share accounts.
- Enable Oracle Applications server security. This feature allows only registered application servers to access the application at the database level, and only a specific list of responsibilities / functionality to execute on those nodes, depending on node type. This is a particular help for securing the software running in the external access DMZ.
- Keep software, especially the administrative toolkits (AD) and AutoConfig, up to date.

See Metalink note 189367.1 for security best practices for the Oracle Applications.

### **Recovering from human error**

When authorized people make mistakes, you need tools to correct the resulting errors. Oracle Applications is now certified on Oracle Database 10g. This makes Flashback technology available for diagnosing and recovering from user errors that corrupt the data.

Flashback can be executed at the row, transaction, table, tablespace, and database levels. Due to the relationships between tables and among code modules in the Oracle Applications, it is generally not supported to restore an Apps table to an earlier point in time without restoring all the related data. Given that, the following set of flashback features will be most useful in an Oracle Applications environment:

- Flashback query – introduced in the Oracle9i database, flashback query allows any SQL statement to be issued “as of” a prior point in time.

- Flashback versions query – retrieves all the different versions of a row across a specified time period. This could be used to isolate a particular transaction ID that created the corruption in the database.
- Using the transaction ID identified above, use flashback transaction query to identify all rows affected by a particular transaction.
- Flashback database – provides a quicker alternative to a traditional point in time recovery. If a serious corruption has occurred and the only recourse is to return the database to a point in time prior to the corruption, it is no longer required to do a full restore and roll forward. Flashback database quickly rewinds the database.
  - If this mechanism is used to restore a database to before a patching exercise starts, the flat files for the Oracle Applications code and tech stacks need to be restored as of the same time period.
  - Note this will only rewind one database, and rewinds the entire database as a whole. If you have implemented an external Oracle Internet Directory (OID) database, you will need to be sure it is in synch with the Applications database after the flashback is complete. If they are configured with bidirectional synchronization between the data stores, this is done automatically. If one or the other is master, manual intervention is required if the master is flashed back.
- Flashback drop – an “undo button” for accidentally dropping database objects. When a user drops a table, Oracle will place it and its dependent objects in a “recycle bin”. Objects in the recycle bin will remain there until they are permanently removed, or until space pressure on the tablespace requires cyclical reuse of the blocks.

While Oracle Applications normally does not support selective restore of database objects, given any updates to the accidentally dropped object which are coordinated with updates to other objects will fail, thoughtful use of this feature to recover from an accidental drop of an object should be fine. Note any tables that have a VPD policy will not be put into the recycle bin.

These features are not specifically configured in an Oracle Applications environment. Configuring them requires:

- Set undo\_retention, and have very large undo tablespaces, for flashback queries.
- Have extra space in individual tablespaces for the recycle bin.
- Configure a flash recovery area for flashback logs.

## **Data Corruption**

It is possible for a database to become corrupted due to a bug or fault in the technology stack. The path a block takes from the Oracle “write” command to disk is complex – from server memory, the block is handled by the IO code on the OS, the file system, the volume manager, the device driver, the host bus adapter, the storage controller and its memory, then finally to the disk drive. Bugs or hardware failures along this path could “flip some bits” in the block, resulting in corrupt data being written to the database or its control structures.

### **H.A.R.D.**

Oracle’s Hardware Assisted Resilient Data (HARD) program is designed to prevent data corruptions before they happen. Offered in partnership with many of the major storage vendors, the data is protected end to end, from the initial write to low-level hardware validation. When HARD is turned on, Oracle adds protection information to database blocks, which is then validated by the storage devices. It eliminates a large class of rare but serious failures previously impossible to prevent.

In the Oracle Database 10g, all file and block types are protected, including database files, online logs, archived logs, and backups. In addition, ASM enables HARD without the customer having to use raw disk devices.

### **Flash Backup and Recovery**

Although rare, multiple failures can render even data mirrored in the storage subsystem unusable. Oracle provides online tools to properly back up a database, to restore a database from a previous backup, and to recover changes to that database either up to current or to an appropriate prior point in time.

Recovery Manager (RMAN) is the tool that manages the backup, restore, and recovery processes for the Oracle database. Oracle Database 10g introduces a new architecture that takes advantage of the disk storage revolution to provide online backups, using a new Flash Recovery Area to hold and manage the backup files. Among many new features is the ability to automatically repair corrupted data blocks while the data file remains online.

The best practices for backing up Oracle Applications stay intact – the database and the tech stack/Applications code file systems should be backed up as of the same basic point in time. Backup sets should be created before and after a major environment maintenance exercise. Backups which need to be retained a number of years should be copied to tape for long-term storage – both the database files and the complete tech stack. Thus, you will still need a tape library system to retain longer-term backups, and you will need to coordinate your RMAN based backups with file system backups written in another tool.

Sizing recommendations for flashback recovery area:

- Two times the size of the maximum number of redo logs generated between backups if only keeping archived redo logs.

- Four times if you will also keep flashback database logs in the flashback recovery area.
- You will need to size the space required for incremental backups based on your experience – these depend on load.
- To keep all the above plus an on disk backup, multiply the size of your database by 3 – that should be room with overhead.

## Site Failures

The discussion so far has covered “local” failures for the production environment. Natural disasters, fire, long-term network or power outages, perhaps even terrorism should be considered – events that lead to a long-term outage of the data center itself. The simplest form of protection against complete data center outage is off-site storage of database and file system backups, along with arrangements for hardware at a remote site in case of disaster. The environment can then be restored and re-configured and brought back online. If planned well, this type of cold site failover should only take a couple of days to implement. It is probable that data will be lost, as the latest archived redo logs are not likely to be included in the off site storage.

## Data Guard Redo Apply

With Oracle Data Guard Redo Apply, an Oracle Applications customer can cut the time required to resume service to an hour or less after a disaster has been declared. A physical standby database used in Redo Apply can also be used to provide availability during major maintenance of the infrastructure in place to provide services – e.g., major network or building wiring maintenance, or a platform upgrade (e.g., to the next version of a vendor’s hardware) – by performing a no-data-loss “switchover” operation.

For disaster recovery protection, you will host your physical standby database in a second data center. In general, this data center should:

- Be physically separate from the primary data center, to protect against local and regional disasters. It is common for a corporation to put its business continuance/DR environment in a data center in a different city than its production data center.
- Have network bandwidth between data centers sufficient to handle peak redo data generation plus, if you choose to synchronize your Concurrent Manager output, synchronization of report log and output files.
- Have reliable and efficient network services to the production data center, to the standby data center, and to the user point of presence.
- Have the same type of servers as the production site, in the desired number for DR protection. The simplest and most flexible configuration will



duplicate the production environment, allowing more freedom to perform switchovers for infrastructure maintenance activities.

To implement Data Guard Redo Apply in an Oracle Applications environment, the following should be done:

- Your environment must be AutoConfig enabled.
- Set `force_logging` at the database level
- Add the standby database server to the list of servers allowed to communicate to the production database
- Implement the database passwordfile.
- Optionally synchronize your concurrent manager “log” and “out” directories, so on failover or switchover your users will not get errors when looking for report output.
- On switchover or failover, update the `fnl_concurrent_requests` table to be able to see files using the new host name.

On patching, you will need to re-sync or re-copy the appropriate file systems. The database will sync itself.

Note Data Guard SQL Apply cannot be used to provide a logical standby database for the Oracle Applications, as there are still several data types used by the Applications that are not supported by SQL Apply, and several activities performed in production that are not propagated via SQL Apply.

### **Combining Physical Standby and Flashback Database**

Before Oracle Database 10g, many customers used a physical standby to protect against user errors and database corruptions by specifying a time lag for redo replay. When this standby was also used to protect against site-level disasters, it led to an increased delay when failing over to the standby, to catch up on redo replay.

This delay is no longer necessary. Flashback Database can be configured in both the primary and the standby database.

- If the entire database must be returned to a prior point in time, it can be used to perform the activity quickly. It can then be done on the standby, and the standby can continue to be used without reinstantiation from the primary.
- Alternatively, the standby could be flashed back to a prior point in time and opened read-only to retrieve lost data. Once the problems are resolved, recovery on the standby can be resumed.

### **Other Components**

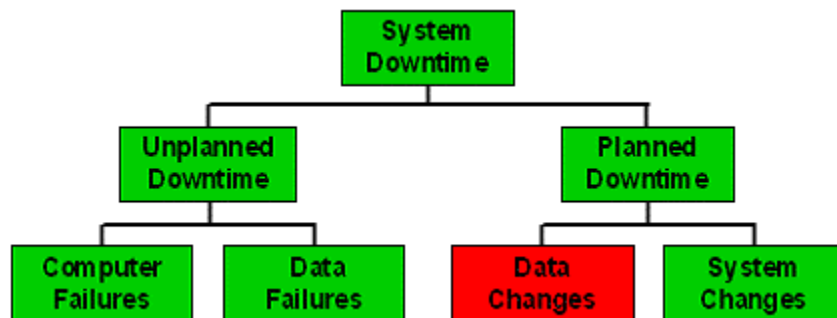
A complete IT service continuity plan must address *each* component involved in delivering service to the end users. The concepts above apply for protection against hardware failure in other areas – you need to provide redundancy and the

ability to fail over across components. Remember to protect the components outside the direct Oracle Applications implementation but required for successful service delivery, include your network services (routers, firewalls, proxy servers, caching servers, DNS services, etc.), single sign-on (OID/SSO), your portal.

## PLANNED DOWNTIME

Running the Oracle Applications involves keeping up to date with various administrative tasks. To ensure best performance, some objects may occasionally need to be rebuilt, or be partitioned. This type of change is covered in Administrative / Data Changes.

In addition, the hardware and software at various levels will need to be patched or upgraded. This is covered under System Changes.



## Administrative / Data Changes

Running the Oracle Applications involves keeping up to date with various administrative tasks. These are in general written in a way that can be done with the system available.

- Keep up with administrative purges – e.g., purging concurrent request and concurrent manager data, purging completed work items, cleaning up workflow control queue data.
- Analyze online. Turn on Oracle Database 10g automatic statistics collection to let the database analyze objects only when needed.

Performance monitoring may identify a hot object, or skewed index. A “busy” table may have fallen behind in purges, and could benefit from rebuilding to reclaim empty space. Rebuilding indexes and tables, shrinking tables and indexes, and partitioning indexes or tables, can be done online with careful planning. As there are points during which the activity will need to briefly but exclusively lock the underlying object, and when cursors accessing the object will be invalidated, it is recommended to do only a small number of these at one time, do them during a time when the underlying object is not under heavy maintenance, and during a time when the system overall is relatively quiet.

## System Changes

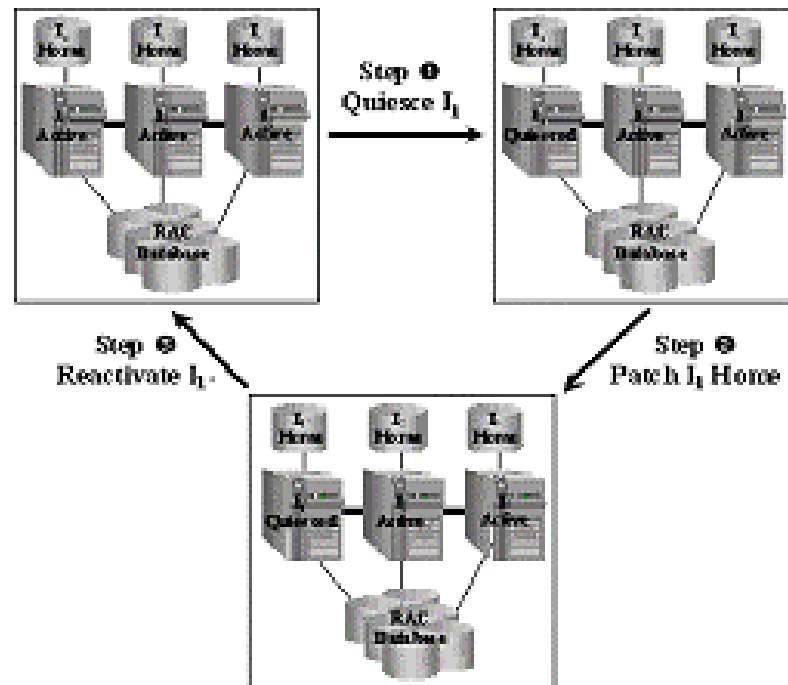
Basic system changes can often be done without disturbing your environment – e.g., most init.ora changes can be made online, SGA memory management is done automatically and online, with ASM disks can be added and load re-balanced with no intervention. Other changes will require more planning.

### Tech Stack Maintenance

The basic mechanism for maintaining availability during planned maintenance of the tech stack under the Oracle Applications is similar to that for protection against system faults – have more than one copy, and try to roll the changes across the copies.

For instance, the operating system for middle tier servers could be upgraded a server at a time. A RAC enabled database environment can sometimes be updated by rolling the changes across RAC instances – this can protect against an OS upgrade, or even a certain database patches marked safe for this process.

The steps for a RAC-based rolling upgrade:



Starting with three nodes actively accessing the shared database, in Step 1 the first instance is quiesced in preparation for being patched. This can be done by “starving” the UI based connections (pointing new connections to other nodes) and by manually failing the Concurrent Manager queues to other nodes. In Step 2, opatch is used to apply the patch to the first instance. In Step 3, the patched instance is brought back up and rejoins the cluster. The administrator “fails back”

the appropriate concurrent manager queues and readjusts the UI configurations to include the instance. The RAC cluster can run in this mixed node for a period of time to be sure the patch corrects the original problem and does not introduce new issues. Then the patch can be rolled across the other instances.

Another solution for major tech stack upgrades (e.g., data server upgrade, OS upgrade, disk frame upgrade) is to take advantage of a Data Guard Physical Standby database. Redo application is stopped, the standby site is upgraded, then redo application is re-started. The new configuration can run as a standby for some time while the environment stabilizes, then the administrator can switch the production environment over to the standby and upgrade the original primary environment.

### **Database Upgrades**

Many Oracle database customers are able to perform a rolling upgrade (e.g., more than a simple patch) of their database software starting with 10.1.0.4, using Data Guard Logical Standby. This is not currently viable with the Oracle Applications database, as the Applications uses data types not supported by Data Guard SQL Apply. Future possibilities for this, using Oracle Streams, are being investigated.

### **Applications Patching and Upgrades**

To reduce planned maintenance outages for the Oracle Applications as much as possible, the following best practices are recommended:

- Proactively maintain the Oracle Applications. Keeping up to date in general makes individual patching exercises as small as possible, as work is not repeated once done.
- Keep AD (Applications DBA) product code up to date. This allows you to take full advantage of improvements in the patching toolkit. This can be done without upgrading the entire suite.
- Keep a patch test system current with production, to be sure your patches are tested against a realistic environment before starting your maintenance window.
- Use AD Merge Patch to consolidate multiple patches into a single patch.
- Apply portions of the patch online where possible. If you have multiple languages installed, merge and apply the US version during downtime, then merge and apply the language patches during uptime. Also, patch Online Help during uptime.
- Study the patch to see if portions can be applied during uptime. For instance, some data migrations in the upgrade to 11.5.10 can be done in two phases – first, a “pre-patch” is applied and the system is brought back up. These data migrations were written to then execute alongside the older software in

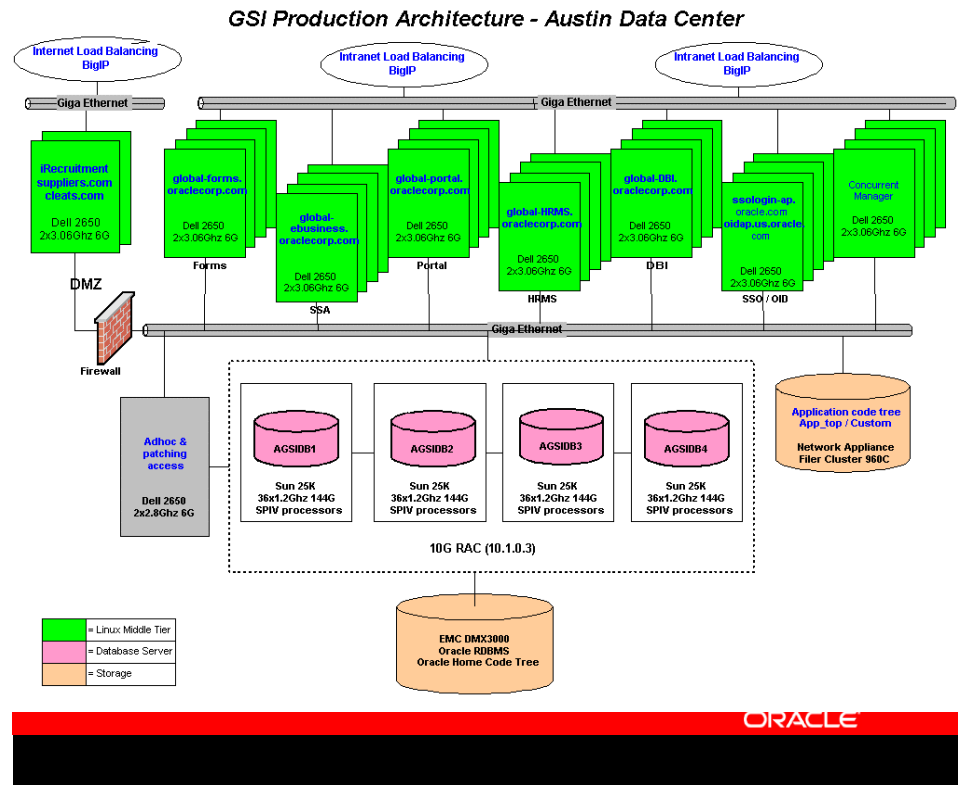
production. During the main upgrade outage, the migration is completed, and the overall downtime reduced.

- Ensure sufficient space for the patch to succeed with no errors – on the APPL\_TOP, in the temp, undo and system tablespaces, and in data and index tablespaces where large data migrations are being performed.
- Use a shared application tier file system (APPL\_TOP, COMMON\_TOP, and Apps tech stacks), so patches need only be applied once.
- Use Distributed AD, to deploy additional middle tiers during the patching exercise.
- Use a staged APPL\_TOP to reduce the actual downtime to only the database update.
- Gather schema statistics, and upload your patch history, during uptime.
- Explore purging any historical data you no longer need before an upgrade.

## ORACLE GSIAP IMPLEMENTATION

Oracle uses the Oracle Applications as their primary business application, and has deployed it in a global single database. All employees, all departments, all divisions, regions, countries – everyone accesses this one environment.

We've included a description of our internal production architecture, to serve as an example deployment, pictured below:



The focus internally at Oracle is to take up new software versions as quickly as possible, thus provide a measure of ‘real-world’ testing, but not to be a model “highly available” environment. Nevertheless, it is a mission critical system and does take advantage of many of the concepts discussed throughout the paper.

### Middle Tiers

All Oracle Applications software runs on 54 identical servers in the middle tiers. The servers (Dell 2650s, each with two 3Ghz CPUs and 6gb memory) run the Linux operating system. BigIP secures the LANs and routes traffic to the middle tiers. Shared storage is used to create one copy of each layer of software running on the environment, per middle tier DMZ:

- Tech stack (the required ORACLE\_HOMEs)
- APPL\_TOP and COMMON\_TOP

Since the servers accessed via the Internet cannot share storage with the servers accessed by employees via the Intranet, due to firewall configuration, this results in two primary copies of each layer of software. Configuration, patches and upgrades are thus simplified, as they need only be done twice (once for the internal servers, once for the external servers). Downtimes can be reduced for the same reason – a copy can be made of the software being patched and the patch applied during uptime, then the server farm bounced to take advantage of the new software. This also allows many systems administration activities to be executed during business hours.

The servers are further divided into sub-farms according to the type of work to be performed – Forms, self-service, portal, SSO/OID, HR self-service, DBI, and concurrent managers. This is not required, but with a server farm this size it becomes easier to isolate the work according to basic behavior, resulting in simpler diagnosing of issues in the environment. It is also simpler to address adding capacity – for instance, properly estimating the number of new servers needed to handle the increased Self-Service Applications traffic that will result from adding several thousand new employees at one time. Finally, monitoring and administrative tasks are somewhat different on each type of server – different problems may arise on Apache versus Forms versus Concurrent Manager middle tier servers, and if a farm of middle tiers needs to be bounced to address an issue or take a patch, that can be isolated.

### **Concurrent Managers**

The same Dell 2x6 servers run the batch Concurrent Manager (CM) software. Approximately 100,000 concurrent requests are processed per day in this environment. To manage this workload, several queues are defined according to the type of the job:

- Critical (approximately 10% of the requests)
- “Immediate” requests (jobs that always take less than 5 minutes to complete)
- “Priority” requests (jobs that on average complete in less than 10 minutes)
- “Standard” requests (those that run longer than 10 minutes but less than 10 hours)
- “Long-running” requests (those that can take over 10 hours)

Concurrent programs are assigned a “request type” according to which queue they should execute in. CM queues are further controlled via the application identifier, with load for each application directed to a specific database instance.

The business owners decide which programs are assigned to the critical queues. Programs are assigned to the appropriate critical queue using “include” specialization rules. There is a critical queue for each module. Available capacity is given first to these queues.

The remaining queues are set up based on the amount of time it takes a job to run, to avoid having quick requests getting “stuck” behind long-running jobs. One of each type is defined for each server in the database cluster. Each queue can run jobs of its own duration or any available request of shorter duration – e.g., the Priority queues can run both Immediate and Priority jobs, and the Standard queues can run all three types. “Long-Running” runs queued requests of any time designation.

The Standard queue is the default queue, and is set up with only “exclude” rules to prevent the Long-Running jobs from running there. Thus they pick up any non-classified concurrent program. The other queues use “include” rules to specify what programs they will manage.

Workshifts are used to manage capacity across the business day. While all queues have some slots available 24 hours a day, the administrators have reduced the capacity assigned to non-critical queues during the hours the system experiences the highest peaks of online activity during our global work day. This is sometimes further adjusted during unusual peak workloads.

Finally, while Parallel Concurrent Processing is configured to run our CM workload across the cluster of database servers, we have not configured automatic failover of the CM queues. If an instance fails, the administrators will decide which queues should be failed over to which of the available instances, depending on the processing calendar / current activities. For instance, if the system is processing the payroll run, the administrators will not fail new load over to the instance managing that work. Other times during the monthly cycle, though, the HR/Payroll server may be the best choice for available capacity.

### **Database Tier**

The database is accessed via a four node RAC cluster of Sun 25Ks. Each box has 36 dual-core 1.2 Ghz processors and 144gb memory. The database itself is held in an EMC DMX3000, and is approximately seven terabytes in size. No, we have never purged any data. The database version is currently 10.1.0.3 with many patches; the most current certified release is 10.1.0.4. Communications between nodes is handled via UDP across a SunCluster private interconnect.

### **Load Direction**

Several years ago, our internal global single instance production database was a cluster of Oracle Parallel Server nodes. At that time, load direction (versus automatic load balancing) was required, because inter-node communication was accomplished by pinging disk. The RAC cache fusion interconnect, introduced in Oracle Database release 9i, removes this requirement, but internally we have not yet switched to scattering our UI load across the database instances.

The impact of this is that we cannot configure automatic failover for our online sessions. If a database instance will be down for a significant amount of time, the



administrators must change the profile options to point to the newly desired node. While this has been scripted, it is in essence a manual step.

It is required to direct Concurrent Manager load to specific database instances. We have split the load across gross functional areas in a manner that generally balances resource consumption across the instances:

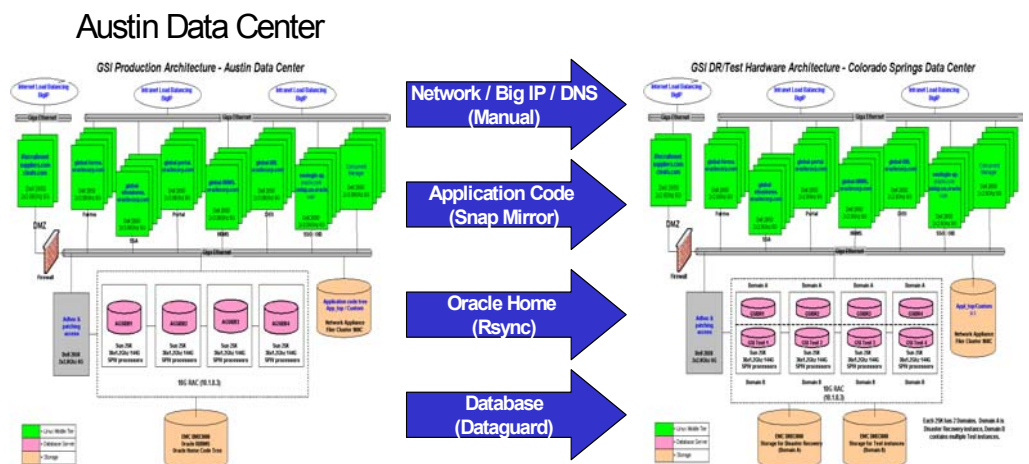
- Node 1: Order Management, Install Base, Shipping, Education, Contracts
- Node 2: Payroll, HR, Benefits, Timecards (self-service), Sales & Marketing
- Node 3: Financials (GL, AP, PO, FA, CM, PN), Self-Service (iProcurement, Expenses), DBI, iStore
- Node 4: AR, Incentive Compensation, Project Accounting, Imaging

CM queue management and failover are discussed above.

## Disaster Recovery

Our production data center is in Austin, TX (the Austin Data Center, or ADC). We have implemented a Data Guard Physical Standby for disaster recovery protection in our Rocky Mountain Data Center (RMDC). This environment is a complete duplicate of the production environment – the same number, type, and capacity servers in each tier. We maintain a four-hour redo application lag, the historical solution for protecting against various logical database corruptions (we are not yet using Flashback technologies in production).

# Disaster Recovery



The duplicate systems in RMDC are deployed as test servers unless and until a true disaster occurs. The production database servers are not configured with domains;

the DR site servers are. These test servers may have thousands of users accessing them as various testing efforts are conducted, so are not used to provide production access during tech stack or other environment upgrades, via a Physical Standby switchover operation. A better configuration for higher availability would allow more frequent switching between sites.

### **Patching the Environment**

With a trim staffing approach, and staying very current with our software, our focus has been more to patch safely than to be a showcase for availability. We schedule a weekly 24-hour planned maintenance window all weekends except those reserved for quarter-end processing.

Both the database and the code must be kept in synch across the sites, but it must also be possible to quickly back out a failed patch. There is a snapshot taken every day of the production code trees (tech stack ORACLE\_HOMEs and APPL\_TOP/COMMON\_TOP) in ADC. There are also two copies in RMDC.

The basic steps we take to patch the environment with a physical standby in place are:

- .
- Shortly before starting the patch, take an EMC BCV backup of the database at RMDC.
- Just before patching starts, take an EMC BCV backup of the production database in ADC.
- Continue with redo transport and apply during the patching exercise.
- After the patch is complete, synchronize one copy of the code trees from the production to the standby site.
- After a few hours, synchronize the second copy of the code trees on the standby site.

To back out a patch:

- On the primary database in ADC: restore from the pre-patch BCV (takes 2-3 hours).
- On the standby database in RMDC: restore from the BCV backup taken a bit earlier, and apply the redo logs generated between the two backups.
- “Snap-mirror” the code trees from the second copy at the standby site back to the primary.
- At the standby site: synchronize the first copy of the code trees from the second copy (which is the older version).

This procedure has been in place for over a year. It has not yet been necessary to execute the “back out” steps.

## **CONCLUSION**

The Oracle Applications meet the criteria of “mission critical.” The basic tools for providing both local and remote failover protection are available for the Oracle Applications via Oracle Database 10g, and can be configured to provide a level of availability appropriate to an organization. Monitor the Maximum Availability Architecture (MAA) Best Practice home page for the latest information:  
<http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm>.



Best Practices for Achieving Business Continuity using Oracle Applications and Oracle Database Technology  
September 2005

Author: Lyn Pratt, Ashish Ray

Contributing Authors: Raji Mani, Andrea Ludtke

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

[oracle.com](http://oracle.com)

Copyright © 2005, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.