

Fast-Start Failover

Oracle Data Guard 10g Release 2

An Oracle White Paper
September 2005

Fast-Start Failover

Oracle Data Guard 10g Release 2

Fast-Start Failover – an Overview	3
User Experience with Fast-Start Failover	3
Amazon.com And Data Guard Automatic Failover	4
Configuring Fast-Start failover	5
Events That Trigger Fast-Start Failover	7
Fast-Start Failover Following Primary Database Failures.....	7
Fast-Start Failover and Network Disconnects	8
Oracle Test Results	10
Reinstatement after a Fast-Start Failover.....	11
Oracle Best Practices for Fast-Start Failover	11
Primary Configuration	11
Standby Configuration	12
Network Transport.....	12
Observer.....	13
Fast-Start Failover Threshold	13
Monitoring	14
Flashback Database Configuration	14
Configurations with Multiple Standbys	14
Applications well suited to Fast-Start Failover	15
Automatic Client Failover	15
Data Guard Role Transition Events.....	15
DB_ROLE_CHANGE Event.....	16
DB_DOWN Event	16
Application Notification	16
Example: Configuring Automatic Failover	17
Configuring for Transparent Client Failover.....	18
Create a primary specific service	18
Configure Oracle Net Service for External Procedures	18
Create necessary support files for LDAP modification	20
Define a middle tier / application restart script.....	21
Create database DB_ROLE_CHANGE trigger.....	21
Conclusion.....	22
References	23

Fast-Start Failover

Oracle Data Guard 10g Release 2

“Fast-Start Failover takes the DBA off the critical path. Database failover is automatic. Data Guard can now address recovery time objectives measured in seconds.”

—Ranjit Singh Veen,
Manager, Enterprise Systems Management
Fannie Mae

FAST-START FAILOVER – AN OVERVIEW

[Oracle Data Guard](#)[1] is the management, monitoring, and automation software infrastructure that creates, maintains, and monitors one or more standby databases to protect enterprise data from failures, disasters, errors, and corruptions.

Data Guard maintains these standby databases as transaction consistent copies of the production database. These standby databases can be located at remote disaster recovery sites thousands of miles away from the production data center, or they may be located in the same city, same campus, or even in the same building. If the production database becomes unavailable because of a planned or an unplanned outage, Data Guard can switch any standby database to the production role, thus minimizing the downtime associated with the outage, and preventing any data loss.

Fast-start Failover is a new Oracle Data Guard 10g Release 2 feature that provides the ability to automatically, quickly, and reliably fail over to a designated, synchronized standby database in the event of loss of the primary database, without requiring manual intervention to execute the failover.

In addition, following a fast-start failover the old primary database is automatically reconfigured as a new standby database upon reconnection to the configuration. This enables Data Guard to restore disaster protection in the configuration easily, without complex manual steps, improving the robustness of the disaster-recovery features of Data Guard, as well as improving Data Guard manageability.

USER EXPERIENCE WITH FAST-START FAILOVER

Companies universally recognized as leaders in their industries tested Fast-Start Failover prior to its general release. In each case, the business driver generating interest in Fast-Start Failover is a mission critical application where high availability is essential to the business function. For this class of applications, any outage is unacceptable, even in the case of an event that makes an entire data-center unavailable. In such instances the goal is a Recovery Time Objective, or the time that it takes to have applications functioning at a remote standby site, measured in seconds. Fast-Start Failover test participants included Amazon.com, Thomson Legal and Regulatory, Fannie Mae, and Airbus. Quotes referencing their collective

experiences are provided in this paper. A more detailed description of the experience of Amazon.com is provided below.

Amazon.com And Data Guard Automatic Failover

At \$6.92 Billion (US) in sales and over 7800 employees, it is not hard to understand why High Availability and Business Continuity are such hot topics at Amazon.com. Continuous availability of Amazon's customer facing systems is essential. Any disruption in system availability directly impacts Amazon's bottom line.

Historically, Amazon has utilized custom-maintained standby databases to protect business operations and data that would otherwise be impacted by system and site-wide failures. Amazon has built-in automation through custom developed scripts and processes that enable business continuity even in the event of complete system or site failures. Amazon has determined that automation is critical to meeting aggressive service level agreements.

Fast-Start Failover is designed to meet requirements such as Amazon's. Fast-Start Failover quickly and reliably fails over a target standby database to the primary database role without requiring manual intervention to invoke the failover. Continually seeking to enhance their high availability architecture, Amazon evaluated Fast-Start Failover during early beta testing of Oracle Data Guard 10g Release 2.

Amazon determined that Fast-Start Failover did indeed satisfy their High Availability requirements. For the first time, an out-of-the-box solution is available that can eliminate the overhead of supporting and maintaining custom scripts and processes. Amazon has found that:

- Fast-Start Failover reliably executes fast, automatic failover to a standby site without human intervention.
- Fast-Start Failover reduced failover time using Oracle Data Guard from minutes to seconds. Average failover time was 25 seconds based on a primary/standby pair with round-trip network latency of 0.5ms.
- Fast-Start Failover ensured zero data loss. Failover will not commence automatically if the failover target is not synchronized with the primary database.
- Because Fast-Start Failover is based on Data Guard Maximum Availability protection mode, the primary production database is not impacted by network or standby failures.
- After failover the old primary database can be automatically reinstated as a new target standby (assuming that the database can be restarted). Data Guard quickly and automatically resynchronizes the old primary. It no longer needs to be restored from a backup of the new primary.

“The capability of fast, guaranteed zero-data-loss failover with Fast-Start Failover in Oracle Data Guard takes the availability of an Oracle database platform to new levels. Our initial tests running Oracle Database 10g Release 2, show that Fast-Start Failover offers a magnitude of improvement in availability.”

—Rajesh Sheth,
Manager, Database Engineering
Amazon.com.

The remainder of this paper provides an in-depth discussion of Fast-Start Failover, Oracle test results, procedures for automating client failover, and Oracle recommended best practices for configuration and operation.

CONFIGURING FAST-START FAILOVER

Fast-start Failover is used within a Data Guard configuration under the control of the Data Guard Broker. The Data Guard Broker provides centralized management of all resources within a Data Guard configuration. Through its command line interface (DGMGRL), the Data Guard Broker uses single commands to perform the equivalent work of multiple SQL*Plus statements, greatly simplifying the management of a Data Guard configuration. Note that the Data Guard Broker is included with Data Guard and does not require a separate installation.

Fast-Start Failover is configured using DGMGRL or Oracle Enterprise Manager 10g Grid Control. Enterprise Manager provides a GUI interface that interacts with the Data Guard Broker. Not only does this provide a very easy to use interface for monitoring and control, it also enables centralized management of all resources in one or more Data Guard configurations.

Using either DGMGRL or Oracle Enterprise Manager, the administrator configures the three essential participants (figure 1) in a fast-start failover configuration:

- The primary database
- The target standby database
- The Fast-Start Failover Observer

The target standby database will become the new primary database following a fast-start failover (note there can be multiple standby databases in a Data Guard configuration).

The Observer is a separate process incorporated into the DGMGRL client that continuously monitors the primary database and the target standby database for possible failure conditions (ref. Fig. 1). The Observer should be run on a different computer host than that of the primary and target standby databases so that it is not subject to any failures that affect the primary or standby computer hosts.

The underlying rule of fast-start failover is that out of these three participants, whichever two can communicate with each other will determine the outcome of fast-start failover. For example, if the primary database becomes unavailable, the Observer confirms with the target standby database that the primary database is unavailable and that the target standby database is synchronized with the primary database, and if so, initiates a fast-start failover to the target standby database.

Details for configuring & enabling Fast-Start Failover are as follows:

- Begin with a Data Guard configuration that includes a primary database and at least one standby database configured using Maximum Availability protection mode with `LGWR SYNC` Redo Transport Services. Flashback database and Flash Recovery Area must also be enabled on both primary and standby databases.

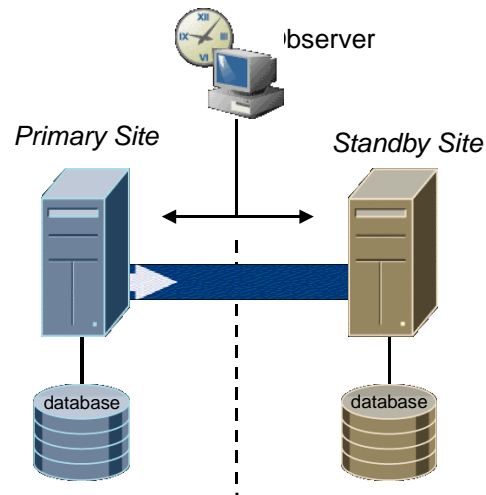


Figure 1 Fast-Start Failover Configuration

- Designate a third system to be the Observer host. It must have the DGMGRL utility installed and have Oracle Net connectivity to both the primary and standby. The Observer should run on a host that is not located in the same datacenter as the primary database, and should be on a different server than the standby database.
- Data Guard Real Time Apply is also recommended to minimize failover time. This will insure that the standby database is up-to-date with the latest redo received from the primary. This eliminates any delay at failover caused by waiting for the standby database to complete the application of received redo.
- Configure `FastStartFailoverTarget` by providing the `DB_UNIQUE_NAME` of the database that is intended to be the new primary.

```
DGMGRL > EDIT DATABASE 'North_Sales' SET  
PROPERTY FastStartFailoverTarget = 'DR_Sales';
```

- Configure the failover threshold which is the number of seconds the Observer attempts to reconnect to the primary database before initiating a failover.

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY  
FastStartFailoverThreshold = 45;
```

- Start

```
DGMGRL> ENABLE FAST_START FAILOVER;  
DGMGRL> START OBSERVER;
```

Following a fast-start failover, the Observer periodically attempts to contact the old primary database. If a reconnection to the old primary database is made, the Observer automatically reinstates the old primary database so that it can be a standby database to the new primary database. At all times through this process, Data Guard insures that no new transactions are allowed in the old primary database prior to assuming a standby role. This quickly restores high availability to the Data Guard configuration.

EVENTS THAT TRIGGER FAST-START FAILOVER

The following database conditions will trigger a fast-start failover:

- Database instance failure (or last instance failure in a RAC configuration)
- Shutdown abort (or shutdown abort of the last instance in a RAC configuration)
- Datafiles taken offline due to I/O errors

The following network conditions will trigger a fast-start failover:

- When both the Observer and the standby database lose their network connection to the primary database, and when the standby database confirms that it is in a “synchronized” state.

The detailed behavior of a fast-start failover and accompanying automatic reinstatement of the original primary database as a new standby database is described below.

Fast-Start Failover Following Primary Database Failures

A fast-start failover may also be initiated in the following cases of primary database failure:

- Instance failures: If a single-instance primary database fails, or if all instances of a RAC primary database fail, the Observer attempts a fast-start failover.
- SHUTDOWN ABORT: If a single-instance primary database or if all instances of a RAC primary database are shut down with the ABORT option, the Observer attempts a fast-start failover. Fast-start failover is not attempted for the other types of database shutdown (NORMAL, IMMEDIATE, TRANSACTIONAL).
- Offline datafiles: If the Observer determines that one or more datafiles in the primary database have been taken offline by the database because of I/O errors, it attempts a fast-start failover. Note that in this instance the FastStartFailoverThreshold is ignored and a failover attempt is immediately triggered.

In all cases, a fast-start failover is not executed unless the Observer and the standby database can agree on the synchronized state of the standby. This insures that a fast-start failover will result in zero data loss.

Fast-Start Failover and Network Disconnects

Fast-start failover may also occur in the event the network links between the primary and standby database, as well as between the primary database and the Observer, get disconnected, and while the connection between the Observer and standby database remains intact. The exact behavior of fast-start failover in this case depends on the order in which the network links around the primary database get disrupted, and whether the Observer and standby database agree that there was complete synchronization with the primary database at the time of failure. Specifically:

- Suppose {primary -> standby} network link fails first. The primary database being in a Maximum Availability configuration, upon this network disconnect, will attempt to move to a RESYNCHRONIZATION protection-level, as reflected in the PROTECTION_LEVEL column in v\$database. The {primary -> Observer} link is still intact, so the Observer agrees to and acknowledges this transition, and the primary database completes the transition to this protection level. The overall fast-start failover state becomes UNSYNCHRONIZED, as reflected in the fs_failover_status column in v\$database of both the primary and standby database (the Observer advises the standby database to move to this state). The primary database continues committing user transactions; redo gets generated and accumulates locally. The Observer does not initiate a fast-start failover in this case because it knows that the fast-start failover state is UNSYNCHRONIZED. If the {primary -> Observer} link fails now, fast-start failover will not occur, and application transactions continue at the primary database.

- Suppose {primary -> Observer} network link fails first. In this case, the `fs_failover_status` and `fs_failover_observer_present` columns in `v$database` will be as follows:

Site	fs failover status	fs failover observer present
Primary	Synchronized	No
Standby	Synchronized	Yes

Fast-start failover does not occur in this scenario. Primary database continues applying transactions and generating redo as well as transmitting redo to the standby database. Now, at this stage, if the {primary -> standby} link fails, the primary database tries to transition protection level from `MAXIMUM AVAILABILITY` to `RESYNCHRONIZATION` – however it cannot communicate with either the Observer or standby database, so this transition does not succeed and the primary database stalls, preventing new transactions from committing. Meanwhile, assuming the {Observer -> standby} link is still intact, the Observer has been asking the standby database if it is ready to failover to which it has responded 'no' up to this point. However, now that the {primary -> standby} link has failed, the standby database will answer 'yes' after `FastStartFailoverThreshold` seconds have passed and Fast-Start Failover will ensue.

- If the {primary -> standby} and {primary -> Observer} network links fail simultaneously, then a fast-start failover will ensue. Following is the sequence of state transitions that occur in this case.
 - Upon losing connection to the primary database, the Observer attempts to reconnect for the number of seconds specified by the Broker property `FastStartFailoverThreshold`. If unsuccessful, it will ask the target standby database if it is ready to failover.
 - If the standby database also has not seen the primary database for `FastStartFailoverThreshold` seconds, then it accepts the Observer's invitation to fail over, transitioning to the primary database role. The `fs_failover_status` column of `v$database` of the standby database will indicate the value `REINSTATE REQUIRED`.
 - Meanwhile the primary database, if still up, stalls because it attempts to transition to an `UNSYNCHRONIZED` fast-start failover state and `RESYNCHRONIZATION` protection-level, but does not get any acknowledgement from the Observer or the standby database in that regard. This old primary database is no longer allowed to commit any transactions and the value `STALLED` is reflected in the `fs_failover_status` column of its

v\$database. This prevents the primary database and the standby database from becoming transactionally-divergent.

- Once the standby database has transitioned to the primary database role, it broadcasts a “*database down*” event on behalf of the old primary database. Oracle 10g Release 2 TAF-enabled OCI applications will respond to this event by immediately disconnecting from the old primary database and reconnecting to the new primary database where they can continue their work. Other middle tier applications can be automatically restarted as part of a role change trigger that occurs as the new primary is opened. For further details on role transition-events, please refer to the section “Automatic Client Failover”, in the section below.

As can be seen from the above discussions, by ensuring that at least two fast-start failover partners agree to major state transitions, conditions such as split-brain scenarios (i.e. a configuration in which there are two divergent “primary databases”) are avoided in a fast-start failover configuration.

ORACLE TEST RESULTS

Oracle tested a Fast-Start Failover configuration comprised of a primary database, standby database, and observer, all running Redhat Linux 3.0. The results of the test are provided in figure 2 below.

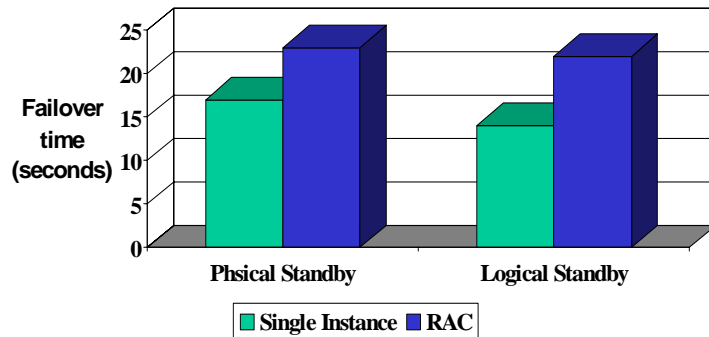


Figure 2 – Fast Start Failover Test Results

The test databases were each 100GB in size. Each host was connected to the next over a Gigabit Network. The workload on the primary database generated 3 MB/second of redo. Both single instance and RAC configurations were tested. Tested configurations included failover to a physical standby database (Redo Apply), and a logical standby database (SQL Apply). In all cases, the failover threshold (or time to detect the failure) was not included in the failover timing

calculation, the test measured only the time required to complete the actual database failover. Total time to complete failover ranged between 10 and 25 seconds, depending upon the configuration.

REINSTATEMENT AFTER A FAST-START FAILOVER

In one of the scenarios described above, the primary database will be stalled if a fast-start failover occurs. If the network connection between the primary database and the Observer is restored, the Observer will automatically reinstate this primary database as a new standby database in the configuration. However, prior to that, the administrator may forcibly shut down the primary database instance and attempt to restart it. In other potentially common cases, the primary database server might have crashed, leading to a fast-start failover. In this case, the administrator may simply attempt to restart the primary database instance. In either case, the administrator may restart the primary database instance using “STARTUP” or “STARTUP MOUNT” through SQL*Plus or DGMGRL.

If the administrator tries to open the old primary database with the `STARTUP` command, Data Guard Broker will not allow it to proceed from the *mount* state to the *open* state until at least one other fast-start failover member agrees to that state transition. Because fast-start failover has already occurred and there is a new primary database in the configuration, the primary database will not get confirmation from either the Observer or target standby database, thus - preventing another potential split-brain scenario. An ORA-16649 error message (“*database must be opened by Data Guard broker when Fast-Start Failover is enabled*”) will be generated. However, because the Observer regains network access to the primary database instance, it will automatically initiate reinstatement of the old primary database into a new standby database in the Data Guard configuration, restoring disaster protection of the new primary database.

If the administrator tries to start the old primary database with the `STARTUP MOUNT` command, no error message will be generated, and the Observer will automatically reinstate the old primary database as a new standby database in the Data Guard configuration.

If the administrator tries to start the old primary database with the `STARTUP NOMOUNT` command, the old primary database will not be mounted and the Broker will not pursue reinstatement until further administrative action is taken (e.g. mounting the old primary database).

ORACLE BEST PRACTICES FOR FAST-START FAILOVER

Primary Configuration

The Data Guard configuration must be set at Maximum Availability protection mode using `LGWR SYNC AFFIRM` as the redo transport mode. This means that redo generated on the primary is synchronously shipped and written to disk on the

standby server. The primary database does not acknowledge the commit to the database client until it receives an acknowledgement that the redo has been written to disk on both primary and standby servers. Maximum Availability insulates the primary database from the impact of network or standby server failures (such failures are automatically detected and the primary database continues processing). However, due to the synchronous nature of redo shipping, there is potential for the performance of the primary database to be impacted by network resources and disk writes on the standby database. For this reason it is very important to follow Oracle best practices for network transport and insure that networks have suitable bandwidth and latency to support synchronous redo shipping.

Standby Configuration

To minimize failover time it is strongly recommended that the standby database be configured with Data Guard Standby Redo Logs and Real Time Apply. This enables the Managed Recovery Process on a physical standby, or the SQL Apply Process on a logical standby, to apply redo to the standby database as it is received, without waiting for a log switch on the primary database. This means the standby database will be completely up-to-date with the primary database. There will be no delay in failover time resulting from the standby database needing to complete the application of redo received from the primary database.

In Data Guard configurations where significant Redo volume is generated at times of peak usage, it may be that default settings are insufficient, and it becomes necessary to further tune the Data Guard processes that apply redo the standby database. In all HA configurations, Oracle recommends users read the comprehensive review of Oracle best practices contained in [Oracle Database 10g. High Availability Architecture and Best Practices](#) [3]. For further drill down into best practices for tuning the Redo Apply process for physical standby reference: [Oracle Database 10g Best Practices, Data Guard Redo Apply and Media Recovery](#) [4]. For similar information for SQL Apply processing on a logical standby, reference: [Data Guard SQL Apply Best Practices in Oracle Database 10g](#) [5].

Network Transport

Tuning operating system parameters that affect Data Guard network throughput can significantly enhance the ability of a network to support a Maximum Availability configuration. These parameters include the TCP Send and Receive buffers and settings that regulate the size of the buffer between the kernel network subsystems and the driver for network interface cards. The importance of tuning for applications with significant workload is clear from testing done by Oracle which demonstrated an order of magnitude improvement in throughput by simply adjusting 3 parameters (see figure 3).

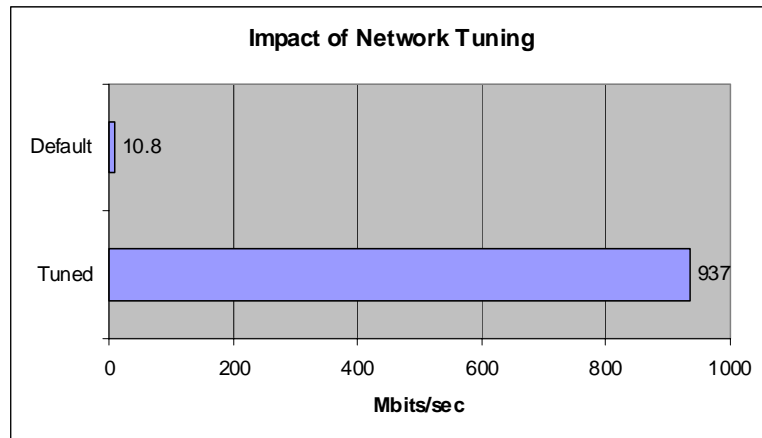


Figure 3 – Impact of Network Tuning
Tuning TCP Send/Receive Buffers, Device Network Queue Size & SDU

For a complete understanding please reference [Data Guard Primary Site and Network Best Practices](#) [2].

Observer

For Disaster Recovery requirements, install the Observer in a location separate from the primary and standby data centers. This prevents any single event from impacting 2 out of 3 of the members of a Fast-Start Failover configuration.

Installing the Observer is simple – all that is required is the Oracle Administrator Client installation. The Observer is also included with the Oracle Database Enterprise or Oracle Personal Edition. An Oracle instance is not required on the Observer system.

Fast-Start Failover Threshold

A Fast-Start Failover occurs when the Observer and the Standby Database both lose contact with the primary database for a period of time that exceeds the Fast-Start Failover Threshold. The correct setting weighs the trade-off between the fastest possible failover (thus minimizing downtime), and triggering unnecessary failovers due to fleeting network irregularities. Recommended settings for the `FastStartFailoverThreshold` are:

- Single instance primary, low latency, reliable network = 10-15 seconds
- Single instance primary, high latency network over WAN = 30-45 seconds
- RAC primary = (RAC miscount + reconfiguration time) + 24-40 seconds

Monitoring

Because a Fast-Start Failover will not occur unless the primary and standby are in a synchronized status (insuring zero data loss), it is important to respond quickly to any event such as a network outage or standby server crash, so that Data Guard can quickly resolve any resulting redo gap and return the standby configuration to a synchronized status.

Enterprise Manager can be used to continually monitor configuration status and automatically notify administrators of events that require attention. Alternatively the state of the configuration can be monitored via the `FS_FAILOVER_STATUS` column of the `V$DATABASE` view. A `SYNCHRONIZED` status means that the primary and standby are in sync and a Fast-Start Failover is possible. `UNSYNCHRONIZED` status means the standby has not received all of the redo generated by the primary database and a Fast-Start Failover cannot occur.

Because a Fast-Start Failover requires two of the three members in the configuration to agree, Observer status should also be monitored, either through the Enterprise Manager GUI, or by monitoring the Observer via the `FS_FAILOVER_OBSERVER_PRESENT` column of the `V$DATABASE` view.

Flashback Database Configuration

Configure Flashback Database on both the primary and the standby databases. Flashback Database is used to automatically reinstate a failed primary as a standby database to the new primary in the event that a failover occurs. This can occur as long as the old primary can be restarted and the failover operation is a zero data loss failover executed by Fast-Start Failover.

Oracle recommends that when used in a Fast-Start Failover context, `DB_FLASHBACK_RETENTION_TARGET` is increased from its default to a setting of 10 minutes. Even this low retention target is a conservative value when used by Fast-Start Failover. Note that if Flashback Database serves the additional function of protection against user error and corruption, then an extended flashback retention period should be set for an amount of time required to achieve these goals.

Configurations with Multiple Standbys

It is not unusual for a Data Guard configuration to include more than one standby database for a single primary database. Often a “local” standby is utilized to achieve zero data loss protection by utilizing a low latency LAN capable of supporting synchronous data protection without incurring the performance overhead of a high latency WAN. The second standby is always geographically remote; 100’s to 1,000’s of miles away from the primary production site. The second standby always utilizes asynchronous data protection to avoid the performance hit of a high latency WAN. Using Fast-Start Failover in such a configuration, the local standby would be designated as the failover target and would be the system, along with the primary production database, that the

Observer would monitor. At failover, Fast Start Failover would transition the local standby into the primary role, and the remote standby would transition seamlessly to the role of standby database to the new primary – affording continuous data and disaster recovery protection while the original primary is repaired.

Applications well suited to Fast-Start Failover

Data Guard is typically used to maintain a hot standby system in a data center remotely located from the primary production site for the purposes of data protection and disaster recovery. Beginning with Oracle9i, Data Guard has enabled zero data loss protection and database failover times that can be completed in minutes assuming an administrator is immediately available to execute the failover.

However, there is a class of applications that are extremely sensitive to downtime. Manufacturing applications where any downtime translates into lost production, trading systems where downtime results in lost business, online web retailers where downtime directly impacts revenue generation, to name a few. These applications cannot tolerate the minutes of downtime that a manual failover may require. They also cannot afford exposure to additional delay if an administrator is not immediately available to execute failover when needed.

Fast-Start Failover eliminates the uncertainty of a process that requires manual intervention and automatically executes a zero data loss failover within seconds of an outage being detected.

AUTOMATIC CLIENT FAILOVER

There are numerous approaches to configuring client failover. Prior to Fast-Start Failover, however, accommodations needed to be made for manual intervention required to execute database failover. Fast-Start Failover eliminates this requirement by making database failover automatic. In addition, Data Guard 10g Release 2 also includes a new `DB_ROLE_CHANGE` system event and `DB_DOWN` event that make it possible to quickly notify clients that a failover has occurred so that they are automatically redirected to the new primary database. These events are described below, along with a sample implementation of automatic client failover.

Data Guard Role Transition Events

In Oracle Database 10g Release 2, Data Guard role changes cause events to be posted in an effort to help the administrator automate post role change tasks and notify applications when a database it is connected to is no longer operating in the primary/production role. Specifically, a system event, `DB_ROLE_CHANGE`, and a FAN (Fast Application Notification) event, `DB_DOWN`, will be posted. This is in

addition to the other HA features already provided by RAC and will be available for the single instance case as well.

DB_ROLE_CHANGE Event

Whenever a database transitions from one role to another, a `DB_ROLE_CHANGE` system event is fired. This is much like the `STARTUP` system event, except it fires only after a role change. Administrators can develop triggers that execute on this event to manage post role change tasks. The event fires when the database opens for the first time after the role transition regardless of its new role, i.e., regardless of whether the role change caused it to open for the first time as a primary database or as a logical standby or as a physical standby, in read-only mode.

The `DB_ROLE_CHANGE` system event may be used to manage/automate post role change tasks. Typical tasks include starting a service / services on the new primary, changing the naming services (e.g. LDAP/OID changes) or connection descriptors so clients will reconnect to the new primary, starting third party applications, adding temporary tablespaces, etc. This is a flexible option allowing the administrator to automate any actions that can be accomplished via database triggers.

DB_DOWN Event

In addition, when a failover operation is coordinated through the Data Guard Broker, a `DB_DOWN` event is posted for the failed primary database. This is a FAN event to notify OCI clients of the failure of the primary database. Further, if the connection was TAF-enabled (Transparent Application Failover), the application could automatically failover to the new primary database.

The `DB_DOWN` event is posted on successful completion of the failover operation and is posted by the database that is operating in the new primary database role on behalf of the old primary. In other words, the event notifies subscribers of this event that the database that used to be operating in the primary role is now down or unavailable. In Oracle Database 10g Release 2, only OCI clients that meet the following requirements are subscribers of this event:

- The environment is created in `OCI_EVENTS` mode.
- The application is linked with a thread library.
- The service to which the application connects is enabled for events.

Application Notification

The services that applications connect to can be enabled for events by enabling the `AQ_HA_NOTIFICATIONS` and `FAILOVER` attributes for the service using the `DBMS_SERVICE.CREATE_SERVICE` or `DBMS_SERVICE.MODIFY_SERVICE` procedures. For further information about these procedures, refer to the PL/SQL Packages and Types Reference manual for Oracle Database 10g Release 2.

When applications that meet the above criteria connect to the primary database, they automatically become subscribers of the HA event notification. This subscription information is registered in the REGS system table.

In the case where the failover target is a physical standby database, TAF enabled applications will receive notification causing them to disconnect from the old primary and reconnect to the new primary database. This notification occurs regardless of whether the new primary database is single instance or RAC. This allows applications to connect to the new primary soon after getting the notification instead of having to wait for the TCP timeout period, as would be the case if it were a site failure. Getting this notification is beneficial even in the RAC case if it was a whole site failure where the RAC monitors do not get a chance to post the notifications.

A caveat: since system tables are not replicated to logical standby databases, if failover were to occur to a logical standby database, applications connected to the failed primary database will not get this notification, and will need to be manually redirected to the new primary database – previously a logical standby.

EXAMPLE: CONFIGURING AUTOMATIC FAILOVER

In previous versions of Oracle if an entire primary site suffered a complete outage (either all primary hosts or network) then ONS (JDBC & ODP) and OCI clients would not be notified of the outage. All clients connected to that cluster would wait for TCP timeout since the nodes of the cluster would not be able to close the TCP sockets. When a failover to the standby occurs, the FAN ONS clients and OCI clients had to be manually restarted in order to break out of this TCP wait.

In Oracle Database 10g Release 2 there is the ability to notify OCI clients that a Data Guard failover has occurred using the DB_DOWN event. Once notified the clients drop the connections to the failed primary and reestablish connections to the new primary. In addition, it is now possible to automate the restart of middle tier applications using the DB_ROLE_CHANGE system event, so that both FAN ONS and non FAN ONS JDBC applications can connect to the new primary. More specifically, when a failover occurs the new primary can be previously configured so that the following takes place automatically:

- Enable primary specific service name(s)
- Update LDAP or other naming methods to point to the new primary host
- Notify FAN OCI clients (OCI apps) that the old primary is down and a reconnection to the new primary should be performed
- Restart any middle tiers so that FAN ONS and non FAN ONS JDBC clients can reconnect to the new primary

Configuring for Transparent Client Failover

The following illustrates the steps needed to configure for transparent client failover in Oracle Database 10g Release 2.

Assumptions:

- Data Guard configuration with Fast-Start Failover enabled
- Clients using LDAP directory naming to resolve service names
- OCI applications meet the following requirements:
 - Initialize the OCI Environment in OCI_EVENTS mode
 - Connect to a service that has notifications enabled (Covered within this document)
 - Link with a thread library

Create a primary specific service

The clients and application should connect to the database using a primary specific service name. This service will migrate to any database that currently holds the primary role. In addition to being specific to the primary role the service must be created with AQ_HA_NOTIFICATIONS set to TRUE to enable proper client notification following a role transition.

1. Using DBMS_SERVICE create a service on the primary database that will be unique to the primary database. Note that enabling AQ_HA_NOTIFICATIONS is required.

```
SQL> exec dbms_service.create_service
(service_name => 'sales',
network_name  => 'sales',
aq_ha_notifications => TRUE,
failover_method => 'BASIC',
failover_type  => 'SELECT',
failover_retries => 180, failover_delay => 1);
```

2. Start the new service on all primary instances:

```
SQL> exec dbms_service.start_service('sales');
```

Configure Oracle Net Service for External Procedures

An external procedure is a procedure called from another program, but written in a different language. In this example it is a PL/SQL program calling one or more C programs that are required for complete client notification. In order to properly

call external programs via a trigger, external procedures must be configured within the Oracle Net components in addition to database libraries and wrapper scripts.

1. On all primary hosts and standby hosts setup network files to make use of external procedures. An example of the changes needed to be made on each host are:

listener.ora:

```
LISTENER =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST =
      halinux03)(PORT = 1521))
    (ADDRESS= (PROTOCOL= IPC)(KEY=external)))
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (ORACLE_HOME = /u01/app/oracle/product/10.2.0)
      (global_dbname=STELLA_DGMGRL)
      (SID_NAME = STELLA1))
    (SID_DESC =
      (SID_NAME = external)
      (ORACLE_HOME = /u01/app/oracle/product/10.2.0)
      (PROGRAM = /u01/app/oracle/product/10.2.0
        /bin/extproc)))
```

tnsnames.ora:

```
extproc_connection_data =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = IPC)(KEY = external))
    (CONNECT_DATA = (SID = external)(SERVER=
      DEDICATED)))
```

For further information reference the [Oracle® Database Net Services Administrator's Guide Chapter 13](#) [6].

2. In order to facilitate calling external programs from a database trigger create a file called shell.c with the following contents:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void sh(char *);
void sh( char *cmd )
{
```

```

        int num;
        num = system(cmd);
    }

```

3. Compile the above program with the following command:

```

$ make -f demo_rdbms.mk extproc_nocallback \
    SHARED_LIBNAME=shell.so OBJS=shell.o

```

4. Move the resulting .so file to \$ORACLE_HOME/bin on both the primary and standby hosts. Note that so file must be in the same location all primary hosts as well as the standby host.

5. Create a library on the primary database.

```

SQL> create library shell_lib is
'/u01/app/oracle/product/10.2.0/bin/shell.so';

```

6. Create a wrapper pl/sql procedure on the primary database:

```

SQL> create or replace procedure shell(cmd IN char)
    as external
        name "sh"
        library shell_lib
        language C
        parameters (cmd string);
/

```

Create necessary support files for LDAP modification

In this example clients connect to the primary database using a service that is only available on the primary database and which is resolved via LDAP directory naming. When a failover occurs and the standby becomes the new primary then the primary specific service in the LDAP directory must be altered to point the new primary host. This can be accomplished via the ldapmodify command which will be included in the DB_ROLE_CHANGE trigger in a later step.

The ldif file should reside in the same directory structure on all hosts. In addition, the value for the host= parameter must be the hostname of the current host or if RAC it must be the value of each primary node VIP addresses. For example:

```

dn: cn=sales,cn=OracleContext,dc=netfl-labsun1,dc=com
changetype: modify
replace: orclNetDescString
orclNetDescString:

```

```
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=halinux03vip)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=halinux04vip)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=sales)))
```

The ldapmodify that will be called from the DB_ROLE_CHANGE trigger will be:

```
ldapmodify -D "cn=orcladmin" -w welcome1 -h netfl-labsun1 -p 3061 -v -f /u01/app/oracle/product/10.2.0/change.ldif
```

Define a middle tier / application restart script

As noted earlier, in a complete primary site outage it is likely FAN ONS clients (as well as non FAN ONS JDBC clients), such as middle tier applications, will not be notified of the primary site failure. In order to break the FAN ONS clients out of the TCP timeout a script is defined which will remotely login to the middle tier hosts and restart the middle tier processes or client application. In this example the script that is created is called "restart_app". Such a script will vary depending upon environment and requirements. A representative example of the logic implemented by such a script is as follows:

```
-----  
Create remote shell  
Source environment file  
    If client application running then  
        Shutdown client application  
        Startup client application  
    Else  
        Startup client application  
Verify client application startup  
-----
```

For a more detailed discussion and examples please refer to a future MAA best practices whitepaper on client failover that will be published on the OTN MAA website.

Create database DB_ROLE_CHANGE trigger

The DB_ROLE_CHANGE trigger is fired whenever the role of the database is changed. In this step we will create a trigger which will do the following:

- Enable the primary specific service name
- Update the service name in the LDAP directory to point to the new primary host

Restart middle tiers or client applications

On the primary create the role change trigger. The trigger should at the least start the primary service name, call ldapmodify to change the net alias in the ldap server, and calls a script which restarts the middle tier or other client applications:

```
SQL> create or replace trigger set_rc_svc after
DB_ROLE_CHANGE on database
declare
    role varchar(30);
begin
    select database_role into role from v$database;
    if role = 'PRIMARY' then
        DBMS_SERVICE.START_SERVICE('sales');
        shell('ldapmodify -D "cn=orcladmin" -w welcome1
-h netfl-labsun1 -p 3061 -v -f
/u01/app/oracle/product/10.2.0/change.ldif');

shell('/u01/app/oracle/product/10.2.0/restart_app');
    else
        DBMS_SERVICE.STOP_SERVICE('sales');
    end if;
end;
```

If the Data Guard configuration is not using Real Time Apply then archive the current log to get the changes to the standby database.

```
SQL> alter system archive log current;
```

CONCLUSION

Oracle Data Guard has evolved over a number of major Oracle releases into the most fully functional disaster recovery solution available for the protection of Oracle data and the high availability of applications that require access to that data regardless of the nature or scale of events that impact the primary production system.

Fast-Start Failover further extends Data Guard's ability to address business continuity requirements. Fast-Start Failover monitors the Data Guard configuration 24x7x365 and executes a failover automatically when specific conditions exist. The automatic nature of Fast-Start Failover avoids delays that can result from human interaction. Automatic failover is also carefully controlled so that any risk of data loss or "split brain" processing of transactions is completely avoided.

"Airbus testing of Data Guard 10g Fast-Start Failover produced impressive results.

Failover executed in less than a minute.

This was much faster than a cold failover using third party cluster technology. With Data Guard, Airbus can achieve continuous data protection and high levels of availability using a standard feature of the Oracle Database."

—Werner Kawollek
Application Management Operations
Airbus Deutschland GmbH

Fast-Start Failover's automatic reinstatement of the original primary following failover will in most cases eliminate the time and effort required for a "manual rebuild" of the original primary database. This makes it easier (and much faster) to execute failovers rather than incur any downtime while administrators troubleshoot failures on the primary production system.

New Data Guard 10g Release 2 Role Transition events provide the added capability to integrate database failover with failover procedures at the middle tier to quickly and automatically redirect clients and applications to the new primary database at the standby location – providing an end-to-end solution for achieving business continuity.

REFERENCES

1. Oracle Data Guard

<http://www.oracle.com/technology/deploy/availability/htdocs/DataGuardOverview.html>

2. Data Guard Primary Site and Network Best Practices

http://www.oracle.com/technology/deploy/availability/pdf/MAA_DG_NetBestPrac.pdf

3. Oracle Database 10g High Availability Architecture and Best Practices

http://download-west.oracle.com/docs/cd/B14117_01/server.101/b10726/toc.htm

4. Oracle Database 10g Best Practices: Data Guard Redo Apply and Media Recovery

http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gRecoveryBestPractices.pdf

5. Data Guard SQL Apply Best Practices in Oracle Database 10g

http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gSQLApplyBestPractices.pdf

6. Oracle Database Net Services Administrator's Guide Chapter 13.

http://download-west.oracle.com/docs/cd/B19306_01/network.102/b14212/advcfg.htm#sthref1188



Fast-Start Failover, Oracle Data Guard 10g Release 2

September 2005

Contributing Authors: Joseph Meeks, Michael T. Smith, Ashish Ray, Sadhana Kyathappala

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

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.