

# Using Oracle Streams in an HA Environment

*An Oracle White Paper  
August 2002*

# Using Oracle Streams in an HA Environment

Abstract.....	3
Types of Failures .....	3
Protection from Failures .....	3
Physical Copies.....	4
Logical Copies .....	5
Logical Standby Database .....	5
Streams Replica Database .....	6
When Not to Use Streams .....	7
Application Maintained Copies .....	8
Best Practices for Streams high availability environments .....	8
Architecting Streams for High Availability .....	9
N-way connected configurations .....	9
Hub and Spoke Configurations.....	9
Using Oracle Real Application Clusters with Streams .....	9
Recovering from Failures .....	10
Reestablishing DB Links After a Failover .....	10
Restarting Capture After a Failover.....	11
Restarting Propagation After a Failover .....	11
Restarting Apply After a Failover .....	11
Conclusion.....	12

# Using Oracle Streams in an HA Environment

## ABSTRACT

Architecting a high availability solution requires careful planning and analysis of failure scenarios. Data Guard, in SQL Apply mode, implements a logical standby database in a high availability environment. Since Data Guard is designed for an HA environment, it provides support to handle most failure scenarios. However, some environments may require the flexibility available in Oracle Streams, so they can take advantage of the extended feature set offered by Streams. This paper will discuss some of the scenarios that may require a Streams-based solution, and will then introduce Streams-specific issues that arise in HA environments. Discussed issues will include using RAC, managing database links, and recovery issues. It will conclude with some best practices for deploying Streams in an HA environment, including hardware failover within a cluster, instance failover within a RAC cluster, failover and switchover between replicas, and avoidance of planned downtime.

## TYPES OF FAILURES

One of the true challenges of high availability is anticipating all the failures that may occur in an actual production environment. It is useful to look at two important classifications of failures. The first, instance failure, refers to the scenario where the database instance running on a system can no longer continue, either because it or the system on which it is running has experienced a failure. All data stored in the Oracle SGA is lost. However, data stored in the database files is unaffected by the failure. Oracle9i can recover the instance in a variety of ways, and no data will be lost.

The second type of failure is a data failure. Recovery from this type of failure can be much more difficult. In the case of a data failure, the Oracle data files have either become damaged or unavailable. Examples of data failures include human errors, failed disk subsystems, and disasters. Oracle9i may not be able to automatically recover from these types of failure, as simply restarting the instance cannot recover the data from the lost data files. To protect from this type of failure requires a copy of the database files.

## PROTECTION FROM FAILURES

The best way to protect from instance failure is with a redundant instance. This is usually done using clustering. Clustering can be implemented using cold failover,

where the failed instance is restarted on another system in the cluster, or using Real Application Clusters, where multiple systems in the cluster host instances managing a single database. Real application clusters is the preferred means of protection from instance failure, because it offers the fastest recovery, best scalability, and best hardware utilization. In either case, recovery is quickly initiated by a restarted or surviving instance.

Recovery from data errors requires a second copy of the database. The DBA can make both a physical and a logical copy of the database. Physical copies are block for block identical to the source copy. Logical copies, on the other hand, contain the same information, but it may be stored differently within the database.

## **Physical Copies**

There are three types of physical copies a DBA can make to protect from data failures:

- Database backup
- Mirrored database files
- Physical standby database

The first and most simple physical copy is a database backup. The database backup stores copies of the data files on tape or disk, usually offsite. In the event of failure those copies can be retrieved and restored. Everyone should make backups of their database, even if they are using the more sophisticated techniques described below. However, a backup may not be sufficient to protect from data failures. It may be quite time consuming to restore the data from backup, and any data modifications since the last backup may be lost unless all subsequent redo logs are available.

Another approach to making physical copies of a database is to mirror the data files, redo logs, and archive logs using special storage subsystems or software utilities. These systems simultaneously write to multiple destinations at the same time, instantly creating a physical copy of the data. The mirroring can be implemented locally or remotely. The advantage of mirroring data files remotely is that in the event of a disaster at the primary site, a copy of all your data will be available at the remote site--your backup will always be up to date. The disadvantage of local or remote mirroring is that any errors that are introduced to the primary copy of your data will instantly be introduced to the other copy. For these reasons, remote mirroring is not recommended for protecting database files against corruptions and errors. Rather, Oracle recommends protecting a database using a feature such as Oracle Data Guard, which is designed specifically to protect data stored in an Oracle database.

Creating a physical standby database is the recommended means of creating a physical copy of a production database. Like a backup, a physical standby database is a block for block copy of production database. However it is automatically maintained such that it can be quickly activated with no or very little data loss in the

event of a failure at the primary database. It is recommended that all customers maintain physical copies of data with a physical standby database.

Oracle Data Guard can be used to implement and maintain a physical standby database. It provides advanced remote log writing features that can ensure no data loss in the event of a catastrophic failure at the primary. It also can delay application of updates to the standby database, providing protection from human errors and accidents that can be intercepted before affecting the standby.

## Logical Copies

Creating a logical copy of your production database brings many advantages. However it is recommended that you create a logical copy in addition to a physical copy, not instead of physical copy. Why then would it make sense to also create a logical copy, essentially a logical standby, of your database? Some of the benefits of a logical standby include the following:

- The logical copy can be open while being updated. This makes the logical copy useful for near real time reporting.
- The logical copy can have a different physical layout that is optimized for some other purpose. For example, it can contain additional indexes, improving the performance of reporting applications that utilize the logical copy.
- The logical copy provides better protection from corruptions. Because data is logically captured and applied, it is very unlikely a physical corruption can propagate to the logical standby of the database.

There are three types of logical copies of a production database.:

- Logical standby database
- Streams Replica database
- Application maintained copy

## Logical Standby Database

A logical standby database is a copy of a production database, where the changes are captured from the production database and applied as local updates via SQL to the standby copy. The standby is predominantly read-only, although it supports updates to temporary tables and can be optimized for reporting. Oracle Data Guard in SQL apply mode can create and maintain a logical standby database.

Oracle Data Guard in SQL apply mode is the easiest way to create and manage a logical standby database. When operating in SQL apply mode, Oracle Data Guard utilizes the same underlying components as Oracle Streams to capture database changes, propagate them to subscribing destinations, and apply the changes at the destination. Oracle Data Guard SQL apply mode provides a higher level interface than Streams, tailored for creating, maintaining, and operating a standby database.

It automates many common operations including failover, switchover, and switchback between the production and standby databases.

Data Guard in SQL Apply mode is optimized for high availability. It can remotely write redo logs to the standby site, where changes will be captured from the redo logs and subsequently applied to the local logical standby database. This model ensures zero data loss in the event of a catastrophic failure at the primary site. All the data necessary to bring the logical copy up to date will already be at the standby site before the failure hits.

### **Streams Replica Database**

Like Oracle Data Guard in SQL apply mode, Oracle Streams can capture database changes, propagate them to destinations, and apply the changes at these destinations. Streams is optimized for replicating data. Streams can locally capture changes in the online redo log as it is written, and the captured changes can be propagated asynchronously to replica databases. This optimization can reduce the latency and can enable the replicas to lag the primary database by no more than a few seconds. However, there is the possibility for a small window of data loss in the event of a catastrophic failure, which can be avoided by the use of a companion physical standby database.

Nevertheless, you may choose to use Streams to configure and maintain a logical copy of your production database. Although using Streams may require additional work, it offers increased flexibility that may be required to meet specific business requirements. A logical copy configured and maintained using Streams is called a replica, not a logical standby, because it provides many capabilities that are beyond the scope of the normal definition of a standby database. Some of the requirements that can best be met using an Oracle Streams replica are listed in the following sections.

- **Updates at the Replica Database.** The greatest difference between a replica database and a standby database is that updates made to a replica database will be copied back to the production database (after conflict detection). Applications that must update persistent data can run against the replica, including job queues and reporting applications that log reporting activity. Replica databases also allow local applications to operate autonomously, protecting local applications from WAN failures and reducing latency for database operations.
- **Heterogeneous Platform Support.** The production and the replica need not be running on the exact same platform. This gives more flexibility in using computing assets, and facilitates migration between platforms.
- **Multiple Character Sets.** Streams replicas can use different character sets than the production database. Data is automatically converted from one character set to another before being applied. This ability is extremely

important if you have global operations and you must distribute data in multiple countries.

- **Mining the Online Redo Logs To Minimize Latency.** If the replica is used for near real-time reporting, Streams can lag the production database by no more than a few seconds, providing up-to-date and accurate queries. Changes can be read from the online redo logs as the logs are written, rather than from the redo logs after archiving.
- **Greater Than Ten Copies Of Data.** Streams supports unlimited numbers of replicas. Its flexible routing architecture allows for hub and spoke configurations that can efficiently propagate data to hundreds of replicas. This ability may be important if you must provide autonomous operation to many local offices in your organization.
- **Fast Failover.** Streams replicas can be open to read/write operations at all times. If a primary database fails, Streams replicas are able to instantly resume processing. A small window of data may be left at the primary database, but this data will be automatically applied when the primary recovers. This ability may be important if you value fast recovery time over no lost data. Assuming the primary database can eventually be recovered, the data is only temporarily lost.
- **Single Capture for Multiple Destinations.** In a complex environment, changes need only be captured once. These changes can then be sent to multiple destinations. This ability enables more efficient use of the resources needed to mine the redo logs for changes.
- **Coexistence with Streams.** If you wish to have a logical standby database, but are also using Streams on the primary database, you must use Streams APIs to create and maintain the logical copy of the database. Data Guard SQL apply mode and Streams are not supported on the same database.

#### **When Not to Use Streams**

As mentioned previously, there are scenarios where customers may choose to use Streams to meet some of their high availability requirements. One of the rules of high availability is to keep it simple. Oracle Data Guard is designed for high availability and is easier to implement than a Streams-based high availability solution. Customers who decide to leverage the flexibility offered by Streams must be prepared to invest in the expertise and planning required to make a Streams-based solution robust. This means writing scripts to implement much of the automation and management tools provided out-of-the-box with Oracle Data Guard.

Also, because Streams was designed for data integration, not high availability, it does not provide a zero data loss mode of operation, as does Oracle Data Guard. Customers who cannot afford to lose transactions in the event of a failure should

either use Data Guard rather than Streams, or complement their Streams-based solution with a zero data loss physical standby maintained by Data Guard. Data Guard also provides a delayed apply option to protect from human errors. Again, a complimentary physical standby can provide such protection.

### **Application Maintained Copies**

The best availability can be achieved by designing the maintenance of logical copies of data directly into an application. The application knows what data is valuable and must be immediately moved off-site to guarantee no data loss. It can also synchronously replicate truly critical data, while asynchronously replicating less critical data. Applications maintain copies of data by either synchronously or asynchronously sending data to other applications that manage another logical copy of the data. Synchronous operations are performed using the distributed SQL or remote procedure features of the database. Asynchronous operations are performed using Advanced Queuing. Advanced Queuing is a database integrated message queuing feature built on top of the infrastructure of Oracle Streams.

Although the highest levels of availability can be achieved with application maintained copies of data, great care is required to realize these results. Typically, a great amount of custom development is required. Many of the difficult boundary conditions that have been analyzed and solved with solutions such as Data Guard and Streams replication must be re-analyzed and solved by the custom application developers. In addition, standard solutions like Data Guard and Streams replication undergo stringent testing both by Oracle and its customers. It will take a great deal of effort before a custom-developed solution can exhibit the same degree of maturity. For these reasons, only organizations with substantial patience and expertise should attempt to build a high availability solution with application maintained copies.

## **BEST PRACTICES FOR STREAMS HIGH AVAILABILITY ENVIRONMENTS**

Implementing Streams in a high availability environment will require consideration of failure and recovery scenarios, and the implementation of procedures to ensure Streams continues to capture, propagate, and apply data after a failure. Some of the issues that must be examined include:

- Architecting Streams for High Availability
  - N-way connected configurations
  - Hub and Spoke configurations
  - Using Oracle Real Application Clusters with Streams
- Recovering from Failures
  - Reestablishing DBLinks after a failover



- Restarting Capture after a failover
- Restarting Propagation after a failover
- Restarting Apply after a failover

## **Architecting Streams for High Availability**

When architecting a solution using Streams, it is important to anticipate failures and design availability into the architecture. You must examine every database in the distributed system, and design a recovery plan in case of failure of that database. In some situations, failure of a database affects only services accessing data on that database. In other situations, a failure is multiplied, because it may affect other databases.

### **N-way connected configurations**

N-way connected configurations create a direct communications path between every node in the distributed system. This is the most resilient architecture to failures, as a failure of one node will not prevent any other nodes from operating or communicating. Assuming all data is replicated, services that were using the failed node will be able to connect to surviving replica nodes.

### **Hub and Spoke Configurations**

Although N-way connected configurations provide the best high availability characteristics, they can become unmanageable when the number of nodes becomes large. Hub and Spoke configurations solve this manageability issue by funneling streams from many systems into a hub database, and then to other hub databases, or to other spoke databases. To add a new source or destination you simply connect it to a hub database, rather than establishing connections to every other database.

A hub, however, becomes a very important node in your distributed environment. Should it fail, all communications flowing through the hub will fail. Due to the asynchronous nature of the events propagating through the hub, it can be very difficult to redirect a stream from one hub to another. A better approach is to make the hub resilient to failures.

The same techniques used to make a single database resilient from failures also apply to distributed hub databases. Oracle Corporation recommends Oracle Real Application Clusters to provide protection from instance and node failures. This configuration should be combined with a "no loss" physical standby database, to protect from disasters and data errors. Oracle Corporation does not recommend using a Streams replica as the only means to protect from disasters or data errors.

### **Using Oracle Real Application Clusters with Streams**

Using Oracle Real Application Clusters with Streams introduces some important considerations. Capturing changes from the online redo log as it is written is not

supported with Oracle Real Application Clusters. Rather, changes are captured from the archived redo log. Capturing from the archived redo logs introduces additional latency between the time a change is made at the production database and the time it appears at the replica.

If low latency is important, a cold failover cluster should be used to protect from system failure rather than Oracle Real Application Clusters. A cold failover cluster is not an Oracle Real Application Cluster. Instead, a cold failover cluster uses a secondary node to mount and recover the database when the first node fails.

When running in an Oracle Real Application Clusters cluster, a capture process runs on the instance that owns the queue that is receiving the captured logical change records (LCRs). Job queues should be running on all instances where propagation is enabled. Assuming propagation is enabled for an instance, a propagation job running on that instance will propagate LCRs from any queue owned by that instance to destination queues. An apply process runs on the instance that owns the queue from which the apply process dequeues its events. That may or may not be the same queue on which capture runs.

Any propagation to the database running Oracle Real Application Clusters is made over database links. The database links must be configured to connect to the destination instance that owns the queue that will receive the events.

## **Recovering from Failures**

### **Reestablishing DB Links After a Failover**

It is important to ensure that propagation continues to function after a failure of a destination database instance. A propagation job will retry (with increasing delay between retries) the database link continually after a failure until the connection is reestablished. In the event the database is restarted on the same node, or on a different node in a cold failover cluster, the connection should be reestablished once the database instance is restarted. In some circumstances, the database link may be waiting on a read or write, and will not detect the failure until a lengthy timeout, controlled by the `TCP_KEEPALIVE_INTERVAL` TCP/IP parameter, expires. In such circumstances, the database link should be dropped and re-created to ensure that communication is reestablished quickly.

When an instance in an Oracle Real Application Clusters cluster fails, the instance is recovered by another node in the cluster. Each queue that was previously owned by the failed instance is assigned to a new instance. Any inbound database links must be dropped and reestablished to point to the new instance that owns the destination queue. In a high availability environment, you can prepare scripts that will drop and re-create all necessary database links. Once a failed instance has been recovered, you can execute these scripts so that Streams can resume propagation.

### **Restarting Capture After a Failover**

After a failure and restart of a single node database, or a failure and restart of a database on another node in a cold failover cluster, the capture process will automatically return to the state it was in at the time of the failure. That is, if it was running at the time of the failure, there is no need to restart the capture process.

For a capture process running in an Oracle Real Application Clusters environment, if an instance running the capture process fails, then the queue that receives the captured LCRs will be assigned to another node in the cluster. You must determine which instance now owns the queue used by the capture process by querying the `DBA_QUEUE_TABLES` data dictionary view, and then restart the capture process on that node. If the failed instance is brought back online subsequently, it will not restart the capture process, even though it was running the capture process at the time of failure, because it is no longer the owner of the queue used by the capture process..

### **Restarting Propagation After a Failover**

For events to be propagated from a source queue to a destination queue, a propagation job must run on the instance owning the source queue. In a single node database, or cold failover cluster, propagation will resume when the single database instance is restarted.

When running in a Real Application Cluster environment, care must be taken to ensure the propagation jobs are enabled on the correct instance. A propagation job should run on the instance that owns the source queues from which the propagation job sends events to destination queues. An instance affinity parameter can be specified for the propagation job that will force the job to run on a particular instance. If the instance fails, and the queue ownership migrates to another instance, then the propagation job affinity must be reset to this other instance. Also, for any jobs to run on an instance, the dynamic initialization parameter `JOB_QUEUE_PROCESSES` must be greater than zero for that instance.

### **Restarting Apply After a Failover**

After a failure and restart of a single node database, or a failure and restart of a database on another node in a cold failover cluster, the apply process will automatically return to the state it was in at the time of the failure. That is, if it was running at the time of the failure, then there is no need to restart the apply process.

In an Oracle Real Application Clusters cluster, if an instance hosting the apply process fails, the queue from which the apply process dequeues the events will be assigned to another node in the cluster. You need to determine which instance now owns the queue used by the apply process by querying the `DBA_QUEUE_TABLES` data dictionary view, and then restart the apply process on that node. If the failed instance is brought back online subsequently, it will not

restart the apply process, even though it was running the apply process at the time of failure, because it is no longer the owner of the queue used by the apply process.

## **CONCLUSION**

In general, Data Guard should be used to provide a standby database to protect from data errors and disasters. However, there may be some situations where customers choose to use Streams to create copies of their production data to protect from these incidents. Streams provides a great deal of flexibility that may be necessary to meet the business goals of a particular enterprise. This flexibility, however, must be properly managed. Organizations should fully understand the implications of using Streams in a high availability environment, and be prepared to put in place the processes to support Streams in this environment.



Using Oracle Streams in an HA Environment

June 2002

Author: Bob Thome

Contributing Authors:

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

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

Oracle is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation. All other product and service names mentioned may be trademarks of their respective owners.

Copyright © 2001 Oracle Corporation

All rights reserved.