

# Oracle Streams

*An Oracle White Paper  
October 2002*

Executive Overview .....	3
Introduction .....	3
Oracle Streams Overview.....	4
Capture .....	5
Staging .....	5
Propagation .....	6
Transformations .....	6
Consumption .....	6
Default Apply.....	7
User-Defined Function Apply.....	7
Explicit Dequeue.....	7
Directed Networks .....	7
Subsetting and Migration.....	8
Instantiation.....	8
Heterogeneous Support .....	8
Oracle to Non-Oracle Capture/Apply.....	8
Non-Oracle to Oracle Capture/Apply.....	9
Interoperating with Non-Oracle Message Queuing Systems.....	10
Deployment for Specific Markets .....	10
Message Queuing with Oracle Advanced Queuing.....	10
Event Management with Oracle Streams.....	11
Event Notification with Oracle Streams .....	12
Replication with Oracle Streams.....	13
Data Warehouse Loading with Oracle Streams .....	14
Oracle9i Database Release 2 Information Sharing Features.....	15
Conclusion.....	17

## **EXECUTIVE OVERVIEW**

Traditionally, information sharing has meant users and applications pulling information from their databases using a variety of special purpose technologies. New business models require a more unified approach—one that automatically finds relevant information and shares it with those who need it. It must be flexible enough to adapt to changing business requirements and minimize the tradeoffs of special purpose technologies. What's needed is a single solution that meets all information sharing needs.

In Oracle9i Database Release 2, Oracle introduces a new unified information sharing feature, Oracle Streams. Oracle Streams satisfies most data movement, transaction propagation and event management needs with a single solution. It provides the capabilities needed to build and operate distributed enterprises and applications, data warehouses, and high availability solutions. Developers and DBAs can utilize all the capabilities of Oracle Streams at the same time. If their needs change, they can simply implement a new capability of Oracle Streams without sacrificing existing capabilities.

## **INTRODUCTION**

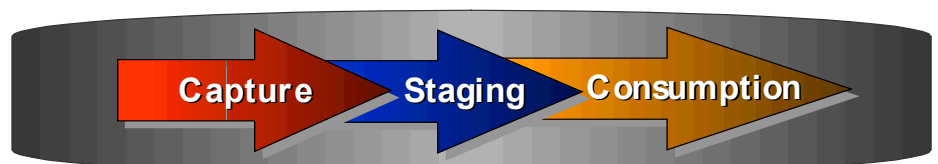
An important feature of any database management system is the ability to share information among multiple databases and applications. Traditionally, this has meant users and applications pulling information from the database using various overlapping technologies. Today, new efficiencies and business models require a more comprehensive and automatic approach. This approach must automatically determine what information is relevant and share that information with those who need it. It must also be a universal solution that adapts to changing business requirements and minimizes the tradeoffs of single purpose solutions. This active sharing of information includes capturing and managing events and transactions in the database, including, but not limited to, DML, and propagating these events to other databases and applications. Information Sharing is important for data and application integration, replication, data warehouse loading, and other applications. However, the variety of information sharing options can be overwhelming. Some solutions are targeted for different purposes, and appear incompatible with the other choices. IT departments find no solution meets all their needs, and fear becoming limited by the solution they have implemented should their needs change. They want a single solution that meets all their information sharing needs.

Oracle9i Database provides a variety of information sharing features, including Advanced Queuing, Advanced Replication, and Synchronous Change Data Capture.. These features provide targeted yet overlapping functionality, and are based on different underlying technology. With Oracle9i Database Release 2, Oracle provides a comprehensive set of information integration sharing features based Oracle Streams. Oracle Streams enables entire new classes of applications. Best of all, Oracle Streams satisfies most data movement, transaction propagation and event management needs with a single solution. Should needs change, Oracle Streams will adapt to meet the new requirements.

## **ORACLE STREAMS OVERVIEW**

Oracle Streams enables the propagation of data, transactions and events in a data stream either within a database, or from one database to another. The stream routes published information to subscribed destinations. The result is a new feature that provides greater functionality and flexibility than traditional solutions for capturing and managing events, and sharing the events with other databases and applications. It allows IT departments to break the cycle of trading off one solution for another. Oracle Streams provides the capabilities needed to build and operate distributed enterprises and applications, data warehouses, and high availability solutions. Developers and DBAs can utilize all the capabilities of Oracle Streams at the same time. If their needs change, they can simply implement a new capability of Oracle Streams without sacrificing existing capabilities.

Oracle Streams provides a set of elements. Using these elements, users control what information is put into a stream, how the stream flows or is routed from node to node, what happens to events in the stream as they flow into each node, and how the stream terminates. By specifying the configuration of the elements acting on the stream, a user can address specific requirements. To simplify deployment of Oracle Streams, Oracle provides applications specifically configured for specific markets.



The architecture of Oracle Streams is very flexible. As shown in the above diagram, Streams contains three basic elements.

- Capture

- Staging
- Consumption

### **Capture**

Oracle Streams supports capture of events (database changes, and application generated messages) into the staging area. These events are captured in two ways. With implicit capture, the server captures DML and DDL events at a source database. Explicit capture allows applications to explicitly generate events and place them in the staging area.

One of Oracle Streams' distinguishing features is support for log-based change capture. Capturing changes directly from the redo log files minimizes the overhead on the source system. Log-based capture leverages the fact that changes made to tables are logged to guarantee recoverability in the event of a crash or media failure. Oracle9i Database can read, analyze, and interpret redo information, which contains information about the history of activity on a database. Oracle9i Database Release 2 can mine the information and deliver change data to the capture process. The database provides supplemental logging to log additional information into the redo stream, such as primary key columns, to facilitate the delivery of this information. The capture process retrieves change data extracted from the redo log, formats it into a Logical Change Record (LCR), and places it in a staging area for further processing. The capture process can intelligently filter LCRs based upon defined rules. Thus, only changes to desired objects are captured.

Oracle Streams supports hot mining the online redo log, as well as mining archived log files. In the case of hot mining, the redo stream is mined for change data at the same time it is written, reducing the latency of capture.

User applications can explicitly enqueue messages representing events into the staging area. These messages can be formatted as LCRs, which will allow them to be consumed by the apply engine, or they can be formatted for consumption by another user application using an explicit dequeue. User applications can also enqueue messages into queue tables, that will be captured by the apply process and published in the staging area as an LCR.

### **Staging**

Once captured, events are placed in a staging area. The staging area is a queue that provides a service to store and manage captured events. Changes to database tables are formatted as LCRs, and then stored in a staging area until subscribers consume them. LCR staging provides a holding area with security, as well as auditing and tracking of LCR data.

Subscribers examine the contents of the staging area and determine whether or not they have an interest in the message representing that event. A subscriber can either be a user application, another staging area, usually on another system, or the default apply process. The subscriber can optionally evaluate a set of rules to determine

whether or not the message meets the criteria set forth in the subscription. If so, then the message will be consumed by the subscriber.

If the subscriber is a user application, then the application will dequeue the message from the staging area in order to consume the message. If the subscriber is another staging area, then the message will be propagated to that staging area. If the subscriber is the default apply process, then it will be dequeued and consumed by the apply process.

### **Propagation**

Events in the staging area may be optionally propagated to other staging areas in the same database, or in other remote databases. To simplify network routing and reduce WAN traffic, events need not be sent to all databases and applications. Rather, they can be directed through staging areas on one or more systems until they reach the subscribing system. Not all systems need subscribe to the events, providing flexibility in what events are applied at a particular system. A single staging area can stage events from multiple databases, simplifying setup and configuration.

### **Transformations**

A transformation is a change in the form of an object participating in capture and apply or a change in the data it holds. Transformations can include changing the datatype representation of a particular column in a table at a particular site, adding a column to a table at one site only, or including a subset of the data in a table at a particular site.

The transformation can be represented by a PL/SQL function that takes the source data type as input and returns an object of the target data type. A transformation can be specified during enqueue, to transform the message to the correct type before inserting it into the staging area. It can also be specified for propagation, which may be useful for subsetting data before it is sent to a remote site. Finally, it can be specified at dequeue or local apply, which can be useful for formatting a message in a manner appropriate for a specific destination.

### **Consumption**

Messages in a staging area are consumed by the apply engine, where the changes they represent are applied to a database, or they are consumed by an application. Oracle Streams includes a flexible apply engine, that allows use of a standard or custom apply function. This enables data to be transformed when necessary. Support for explicit dequeue allows application developers to use Oracle Streams to notify applications of changes to data, while still leveraging the change capture and propagation features of Oracle Streams.

### **Default Apply**

The default apply engine applies DML changes, DDL changes, and user-supplied LCRs. When the destination database is an Oracle database, the apply engine runs locally on the system hosting the Oracle database.

The default apply engine will detect conflicts where the destination row has been changed and does not contain the expected values. If a conflict is detected, then a resolution routine may be invoked.

The apply engine consists of an apply coordinator, one fetch slave, and one or more apply slaves. The fetch slave assembles transactions for Oracle-generated LCRs. The apply coordinator performs transaction dependency and DML level dependency scheduling to maximize concurrency. The apply slaves actually apply the changes.

Since the apply coordinator and apply slaves are typically in the same Oracle instance, there is no network roundtrip involved in dependency scheduling. The apply engine can invoke user-supplied functions to apply the changes.

### **User-Defined Function Apply**

The apply engine can pass the LCR to a user-defined function. This provides the greatest amount of flexibility in processing the LCR. A typical application of a user-defined function would be to reformat the data represented by the LCR before applying it to a local table, for example, field format, object name and column name mapping transformations. A user-defined function could also be used to perform column subsetting, or to update other objects that may not be present in the source database.

### **Explicit Dequeue**

User applications can explicitly dequeue LCRs or other messages from the receiving staging area. This allows a user application to efficiently access the data in a Streams' staging area. Streams can send notifications to registered PL/SQL or OCI functions, giving the applications an alternative to polling for new messages. Of course, applications can still poll, or even wait, for new subscribed messages in the staging area to become available.

### **Directed Networks**

Administrators have a great deal of flexibility to use in specifying the routing of the streams. By using the rules-based publish and subscribe capabilities of the staging area queues, they can choose which changes are propagated to each destination database, and can specify the route messages will traverse on their way to a destination. For example, an event may propagate via a hub database that does not actually apply the event.

It can be important to differentiate between changes that originate at different sites. Any changes made to a database will be reflected in the redo log and can thus be

captured by the capture process. However, in certain cases it will be necessary to filter out changes that originated at another site and were made by the apply process, to prevent their being placed in the staging area and cycling back to the originating site. In other cases, it will be important to capture all changes, including those that originate at other sites.

### **Subsetting and Migration**

The Oracle Streams apply engine supports subsetting of objects, so a destination need only subscribe to a subset of the data at the source. Consider a typical distributed scenario involving HQ databases and branch office databases. The branch office database only needs the data relevant for that branch. A branch office would therefore only subscribe to transactions affecting that branch. Other branches would receive different subsets.

Streams also supports migration of data from one subset to another. Subscriptions are content-based, and should that content change, the subscriptions may change. This may require data to be deleted from one subset database, and inserted into another. Imagine a customer who is located in the north branch's territory, but relocates to the south branch's territory. Streams would automatically migrate their information from the north branch to the south branch.

### **Instantiation**

Instantiation is the process of creating a copy of objects to be replicated at a destination site, and preparing to capture changes beginning with the appropriate SCN in order to ensure no changes are lost. As copies of objects are created using export/import, the system will note the SCN of the copied data, and will begin applying any changes that committed after the object was copied.

### **Heterogeneous Support**

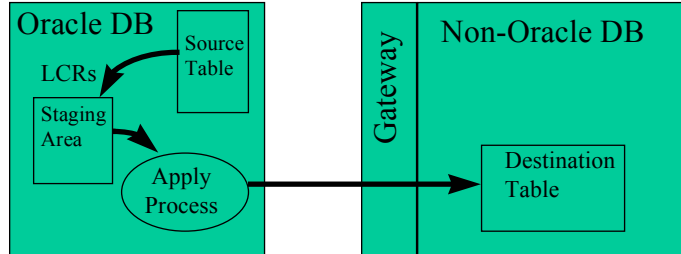
Oracle Streams is an open information sharing solution. Each element supports industry standard languages and standards. Streams supports capture and apply from Oracle to non-Oracle systems. Changes can be applied to a non-Oracle system via a transparent gateway. Streams also includes an API to allow non-Oracle data sources to easily submit or receive change records, allowing for heterogeneous data movement in both directions. In addition, messages can be sent to and received from other message queuing systems such as MQ Series and Tibco via the Message Gateway.

### **Oracle to Non-Oracle Capture/Apply**

Heterogeneous capture and apply is capture and apply between Oracle and non-Oracle databases. Some unique challenges are encountered when configuring Oracle Streams for Oracle to non-Oracle capture/apply.

### **Propagating Changes in an Oracle to Non-Oracle Environment**

To implement capture and apply of DML changes from an Oracle source to a non-Oracle destination, an Oracle system functions as a proxy and carries out some of the steps that would normally be done by the non-Oracle system. The Oracle system then communicates with the non-Oracle system via a transparent gateway.



*Oracle to Non-Oracle Capture/Apply*

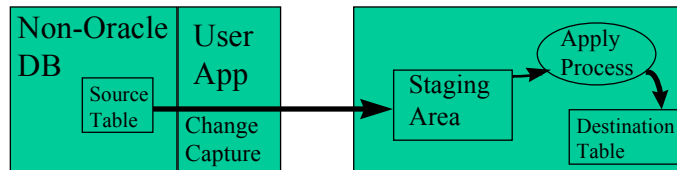
When propagating a database update to a non-Oracle system, there is no staging area at the destination site to receive the changes and no apply engine to dequeue the changes and apply them. Instead, the changes are dequeued in an Oracle database itself and local apply applies the changes to a non-Oracle system across a network connection through gateway. The apply engine can perform a default apply, or can call a user-defined function to implement the apply.

### **Instantiation in an Oracle to Non-Oracle Environment**

Instantiation using Oracle's Export/Import utilities is not feasible with non-Oracle systems. Instantiation to non-Oracle systems can be accomplished by selecting data from the local copy of the table and inserting the data, row by row, into the remote table.

### **Non-Oracle to Oracle Capture/Apply**

Users who want to propagate changes from a non-Oracle database to an Oracle database must write an application. This application will capture the changes made to the non-Oracle database by reading from transaction logs or by using triggers. The application must assemble and order the transactions to convert them to an LCR form. Then, it will use the published Streams OCI or PL/SQL APIs to enqueue the LCRs to the target Oracle database staging area.



*Non-Oracle to Oracle Capture/Apply*

### **Interoperating with Non-Oracle Message Queuing Systems**

The Oracle Messaging Gateway provides interoperability between Oracle Streams and non-Oracle message queuing systems such as MQ Series and Tibco Rendezvous. This allows developers to easily capture updates made to data in the Oracle database, and propagate those events to other applications, regardless of where they are running or what communications technology is common in the environment. The Messaging Gateway also supports propagating messages from MQ Series and Tibco into Oracle Streams, allowing the same third-party applications to submit events via MQ Series and Tibco to applications built in the Oracle Streams environment.

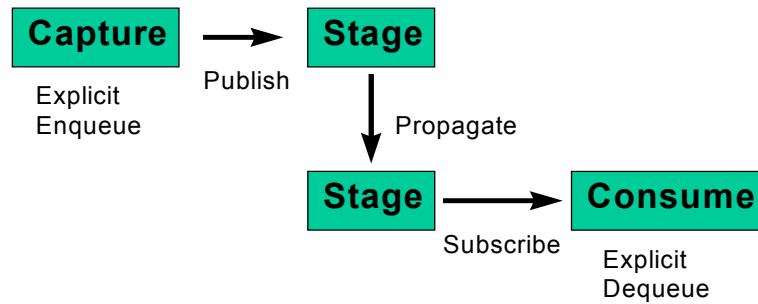
### **DEPLOYMENT FOR SPECIFIC MARKETS**

Oracle Streams satisfies the information sharing requirements of a variety of markets. To simplify deployment of Oracle Streams in these various markets, Oracle provides tools and high level features that facilitate using Streams to provide the specific capabilities required in each market. Oracle Advanced Queuing, the database-integrated feature providing message queuing and event management capabilities, is now built upon Oracle Streams. In addition, Oracle9i Database includes tools to help users build event notification, replication and data warehouse loading solutions using Oracle Streams. Of course, anyone can utilize the full power of Oracle Streams, and create configurations that seemingly span multiple markets, enabling new classes of applications. In addition, most deployments and their associated meta-data are compatible. For example, a system configured to load a data warehouse can easily be extended to enable bi-directional replication—a complete reconfiguration is not required.

### **Message Queuing with Oracle Advanced Queuing**

Oracle Advanced Queuing, a feature built on Oracle Streams, allows user applications to enqueue messages into the staging area, propagate messages to subscribing staging areas, notify user applications that messages are ready for consumption, and dequeue messages at the destination. It supports all the standard features of message queuing systems including multi-consumer queues, publish and subscribe, content-based routing, Internet propagation, transformations, and gateways to other messaging subsystems. Advanced Queuing

also supports notifications to user applications, merging the near real-time benefits of a push model with the scalability and manageability benefits of a pull model.

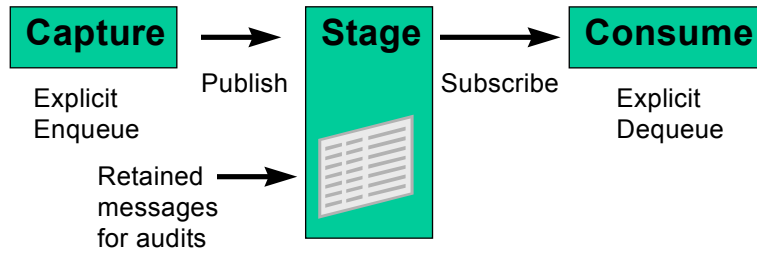


#### *Message Queuing Between Two Databases*

Advanced Queuing exposes those features of Oracle Streams useful for building message queuing applications. An administrator would create a staging area at the source database, which behaves as a queue. Applications can then explicitly enqueue messages into the source database. Applications can use strongly typed queues, as has been characteristic of the queues in previous releases of the database, or can use AnyData queues, which are new with Oracle Streams. AnyData queues can hold messages of different types, so it is possible to enqueue different types of messages into a single staging area. Subscribing applications can directly dequeue messages from the staging area. If the application is remote, a staging area may be created in a remote database that will subscribe to messages published in the source staging area. The destination application can then dequeue from the remote staging area. Alternatively, the destination application can directly dequeue from the source staging area using a variety of standard protocols.

#### **Event Management with Oracle Streams**

Business events are valuable communications between applications or organizations (internal and external). They require the highest degree of reliability, integrity, and security. Oracle Streams message queues are stored in the Oracle database, and are retained after consumption. This allows Advanced Queuing to be used as a business event management system. Advanced Queuing stores all messages in the database in a transactional manner, where they can be automatically audited and tracked. This audit trail can be used to extract intelligence about the business operations.

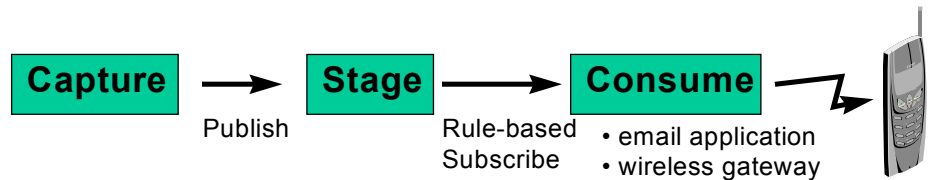


*Event Management with Streams using Advanced Queuing*

The key to using Advanced Queuing as a business event management system is retention in the staging area. As with message queuing, a staging area is created to stage business events that are explicitly enqueued by applications. Business events can be consumed locally via an application explicitly dequeuing the event, or they can be propagated to other staging areas and then consumed by applications. All staging areas will retain messages, even after they are consumed, providing an audit trail critical for managing business events.

**Event Notification with Oracle Streams**

Information overload makes it difficult to find the data that’s important. A new class of applications is used to monitor data and notify interested parties of relevant data. Oracle Streams supports event notification, where changes to an underlying database are captured and then sent to other applications or devices that subscribed to that change event. Streams includes a scalable and sophisticated rules engine that can evaluate captured changes and forward them on to the appropriate subscribers.



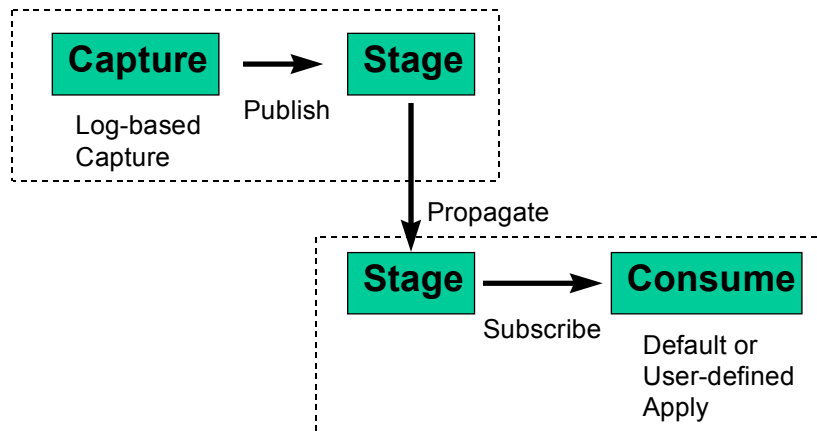
*Event Notification with Streams*

Configuring Streams for Event Notification requires setting up the capture process to capture the events of interest, be they DML, DDL, or implicitly enqueued messages. Those events will then be published and consumed by subscribing applications. Rules can be used to restrict consumption to events of interest. The consuming application can then take action, which may include an email notification, or passing the message to a wireless gateway for transmission to a cell phone or pager.

## Replication with Oracle Streams

Oracle Streams can efficiently capture changes made to a system, and replicate those changes to one or more other systems. Changes are captured using Oracle Streams' implicit change capture mechanism. Changes are propagated to one or more remote databases, where they are applied using the standard apply function. The remote databases are fully open for read/write, and need not be identical copies of the source database. Because the remote database can be updated by other means, the apply mechanism will detect conflicts before changes are applied. These conflicts can also be automatically resolved using built-in or custom resolution mechanisms. Replication can be n-way. Updates at any remote database can be published and propagated to other databases on the network.

Like Advanced Replication, Oracle Streams supports updateable materialized views, or snapshots, to maintain point-in-time copies of data. They can be defined to contain a full copy of a master table or a defined subset of the rows in the master table that satisfy a value-based selection criterion.

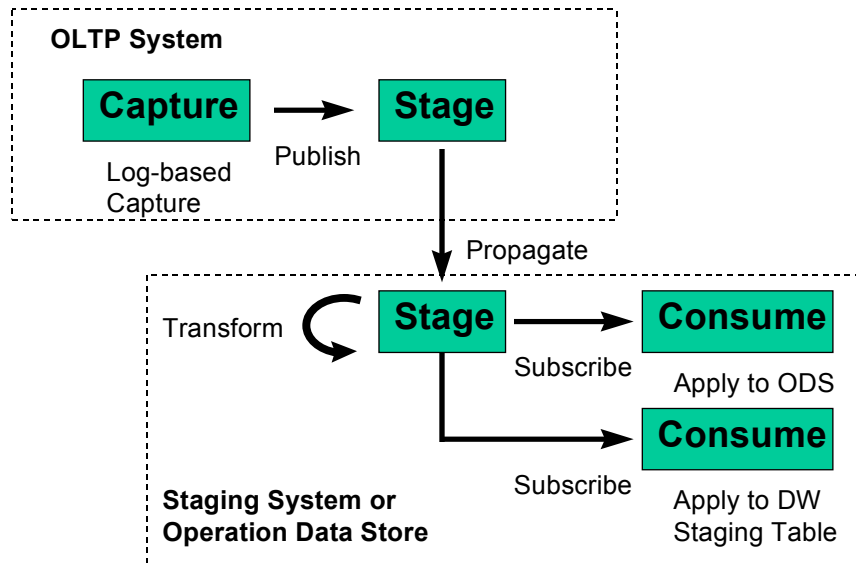


*Two node Replication with Streams*

Configuring Streams for replication begins with specifying the object to be replicated. Events representing DML and DDL changes to these objects will then be captured from the redo log, and published in the local staging area. While in the staging area, they may be transformed to match the format of the destination system. The events are propagated to one or more destination staging areas. The route these events travel is configurable. The apply engine at the destination will subscribe to events in the destination staging area (can be content-based—not all events need be consumed). The default apply engine will apply the changes as they are, and detect and optionally resolve any conflicts that may occur. The apply engine can also pass the event to a user-defined function to apply the changes in a custom manner.

## Data Warehouse Loading with Oracle Streams

One of the most critical tasks in creating and maintaining a data warehouse is refreshing existing data, or adding new data from the operational databases. Oracle Streams can capture changes made to a production system, and send those changes to a staging database or directly to a data warehouse or operational data store. Oracle Streams' log-based capture eliminates unnecessary overhead from the production systems. Support for data transformations and user-defined apply functions allows the necessary flexibility to reformat data or update warehouse specific data fields as data is loaded.



*Loading a DW or Operation Data Store*

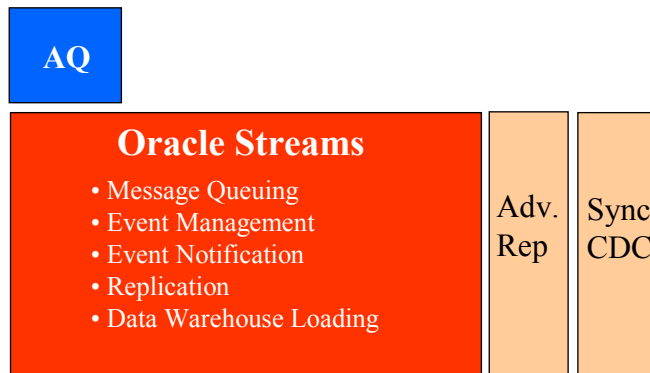
Configuring Streams to load a data warehouse or an operational data store (ODS) is similar to setting up replication. The operational data store, in fact, is likely a replica of a subset of the database. DML would be extracted using implicit log-based change capture, and would be continually streamed to the ODS. Some items may be transformed to a different format for consistency with other data stores. The default apply engine would likely be used to load the updates, and conflict resolution is probably unnecessary as the ODS is read-only. In addition, additional indexes and materialized (or regular) views may be part of the operational data store.

A data warehouse is typically loaded using specialized extraction, translation, and loading (ETL) tools. These usually operate off a staging table, and transform and load data in a bulk operation. Streams role in loading a data warehouse, therefore, is to load the staging table.

## ORACLE9I DATABASE RELEASE 2 INFORMATION SHARING FEATURES

In addition to Oracle Streams and its extended features, Oracle9i Database Release 2 offers the following information sharing features—Advanced Replication, and Change Data Capture. These features were introduced in prior releases of the database and were designed to meet the requirements of specific markets, such as warehouse loading, data protection, and replication.

In Oracle9i Database Release 2, Oracle Streams is the preferred means of providing enterprise replication, messaging, and business event management. Oracle Streams provides almost all the capabilities previously provided by Advanced Replication. Because Advanced Replication supports some features and APIs not provided by Oracle Streams, it remains an independent feature of Oracle9i Database for compatibility. In addition, Synchronous Change Data Capture will continue to exist for backward compatibility as a feature to capture changes to load data warehouses.



IT departments must choose the feature that best meets their needs. In most cases, this is a simple task, as Oracle Streams and those features built on Oracle Streams are the preferred features for information sharing. The following table summarizes the positioning of these features.

Market	Feature Name	Description	Target Customer
Message Queuing	Oracle Advanced Queuing (built on top of Streams)	Full-featured MOM support	<ul style="list-style-type: none"> <li>Application developers building applications utilizing message-based communication</li> <li>Application can run in different location</li> <li>Application can run on different platforms</li> <li>Application can run at different time</li> <li>Applications have different message layout</li> </ul>
Business Event Management	Oracle Advanced Queuing (built on top of Streams)	Business event management system	<ul style="list-style-type: none"> <li>Application developers building event-based applications</li> <li>Communication data need to be retained for automatic documentation</li> <li>Communication data are highly structured - require type support</li> <li>Communication data need to be queried for audits and analysis</li> <li>Users need to subscribe to data based on content</li> </ul>
Event Notification	Oracle Streams	Sending notifications to user applications	<ul style="list-style-type: none"> <li>Customers writing applications that react to changes made to a production database</li> <li>Need near real-time notifications</li> <li>No ability to modify the original application that made the update to send a notification</li> <li>Need to evaluate complex rules to determine if and where to send a notification</li> </ul>
Data Replication	Oracle Streams	Premier feature for replicating data within a distributed enterprise	<ul style="list-style-type: none"> <li>Customers building sophisticated high transaction rate distributed database environments</li> <li>There are a large number of participating servers</li> <li>There are large distances between servers</li> <li>Data needs to be exchanged with non-Oracle servers, or heterogeneous formats and data models</li> <li>Only subsets of data need be replicated to some servers</li> <li>Some participating servers are nearing peak load and need to offload change capture to another system</li> <li>Some change capture sources or destinations are applications, not databases</li> <li>Different Oracle releases or platforms</li> </ul>
	Advanced Replication	Replication of exact copies of objects using a point-to-point infrastructure.	<ul style="list-style-type: none"> <li>Departments, or those needing department-to-department replication.</li> <li>Replicated objects are logically identical on all participating servers</li> <li>There is a small number of active servers</li> <li>Replicating with older (pre Oracle9i Database Release 2 databases)</li> </ul>

<b>Data Warehouse Loading</b>	<b>Oracle Streams</b>	Loading a data warehouse or operational data store	<ul style="list-style-type: none"> <li>• Customers who need to capture changes from a production system and send them to a data warehouse staging area or operational data store.</li> <li>• Need to capture and stage in near real time</li> <li>• Need to minimize effect or access to production system</li> <li>• Need to transform data before loading</li> <li>• Need to capture all updates—including intermediate changes</li> </ul>
	<b>Synchronous Change Data Capture</b>	Feature that synchronously captures primary keys of changed records and loads DW staging tables	<ul style="list-style-type: none"> <li>• Customers who need to capture changes from a production system and send them to a data warehouse staging area</li> <li>• Need backward compatibility with Oracle9i Database (9.0.1) feature</li> <li>• Want simple solution for small light-weight OLTP system</li> </ul>

### CONCLUSION

Oracle9i Database Release 2 provides many powerful features to share information between databases, users, and applications. In addition to established features like Advanced Queuing, Advanced Replication, and Synchronous Change Data Capture, Oracle Streams can now be used to satisfy the most demanding information sharing requirements using a common infrastructure. Complex distributed environments will benefit from a single solution to simplify their information sharing solutions. Simple distributed environments will benefit in the knowledge they can expand their environment as their needs change without having to learn and integrate new products. Developers and administrators will spend less time wrestling with their tools, and more time providing solutions.

# ORACLE

Oracle Streams

October 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.