4

Managing Oracle Transactional Event Queues and Advanced Queuing

These topics discuss how to manage Oracle Transactional Event Queues and Advanced Queuing.

- Oracle Database Advanced Queuing Compatibility Parameters
- Queue Security and Access Control
- Queue Table Export/Import
- Oracle Enterprise Manager Support
- Using Oracle Database Advanced Queuing with XA
- Restrictions on Queue Management
- Managing Propagation

Oracle Database Advanced Queuing Compatibility Parameters

The queues in which buffered messages are stored must be created with compatibility set to 8.1 or higher.

The `compatible` parameter of `init.ora` and the `compatible` parameter of the queue table should be set to 8.1 or higher to use the following features:

- Queue-level access control
- Support for Oracle Real Application Clusters environments
- Rule-based subscribers for publish/subscribe
- Asynchronous notification
- Sender identification
- Separate storage of history management information
- Secure queues

Mixed case (upper and lower case together) queue names, queue table names, and subscriber names are supported if database compatibility is 10.0, but the names must be enclosed in double quote marks. So `abc.efg` means the schema is `ABC` and the name is `EFG`, but `"abc"."efg"` means the schema is `abc` and the name is `efg`.

Queue Security and Access Control

These topics discuss Oracle Database Advanced Queuing queue security and access control.

- Oracle Database Advanced Queuing Security
- Queue Security
- Queue Privileges and Access Control

- OCI Applications and Queue Access
- Security Required for Propagation

Oracle Database Advanced Queuing Security

Configuration information can be managed through procedures in the DBMS_AQADM package.

Initially, only SYS and SYSTEM have execution privilege for the procedures in DBMS_AQADM and DBMS_AQ. Users who have been granted EXECUTE rights to these two packages are able to create, manage, and use queues in their own schemas. The MANAGE_ANY AQ system privilege is used to create and manage queues in other schemas.

Users of the Java Message Service (JMS) API need EXECUTE privileges on DBMS_AQJMS and DBMS_AQIN.

Topics:

- Administrator Role
- User Role
- Access to Oracle Database Advanced Queuing Object Types

See Also:

"Granting Oracle Database Advanced Queuing System Privileges" for more information on AQ system privileges

Administrator Role

The AQ_ADMINISTRATOR_ROLE has all the required privileges to administer queues.

The privileges granted to the role let the grantee:

- Perform any queue administrative operation, including create queues and queue tables on any schema in the database
- Perform enqueue and dequeue operations on any queues in the database
- Access statistics views used for monitoring the queue workload
- Create transformations using DBMS_TRANSFORM
- Run all procedures in DBMS_AQELM
- Run all procedures in DBMS_AQJMS

Note:

- A user does not need `AQ_ADMINISTRATOR_ROLE` to be able to create queues or do any other AQ DDL in his schema. User only needs execute privilege on `DBMS_AQADM` package for the same.
- `AQ_ADMINISTRATOR_ROLE` still will not be able to create AQ objects in `SYS` or `SYSTEM` schema. So the above description that it can create queue tables/queue on any schema excludes `SYS` and `SYSTEM` schema.
- `AQ_ADMINISTRATOR_ROLE` still will not be able to enqueue/dequeue on queues in `SYS` or `SYSTEM` schema.

User Role

You should avoid granting `AQ_USER_ROLE`, because this role does not provide sufficient privileges for enqueuing or dequeuing.

Your database administrator has the option of granting the system privileges `ENQUEUE_ANY` and `DEQUEUE_ANY`, exercising `DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE` and `DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE` directly to a database user, if you want the user to have this level of control.

You as the application developer give rights to a queue by granting and revoking privileges at the object level by exercising `DBMS_AQADM.GRANT_QUEUE_PRIVILEGE` and `DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE`.

As a database user, you do not need any explicit object-level or system-level privileges to enqueue or dequeue to queues in your own schema other than the `EXECUTE` right on `DBMS_AQ`.

Note:

- A user only needs `EXECUTE` privilege on `DBMS_AQ` package to be able to enqueue or dequeue in his schema, if client is OCI or PL/SQL.

  If the client is JDBC or JMS, then added to this you need execute privilege on `DBMS_AQIN` package as well.
- A user does not need `AQ_USER_ROLE` for enqueue or dequeue queues in his schema.

Access to Oracle Database Advanced Queuing Object Types

All internal Oracle Database Advanced Queuing objects are accessible to `PUBLIC`.

Queue Security

Oracle Database Advanced Queuing administrators of Oracle Database can create queues. When you create queues, the default value of the `compatible` parameter in `DBMS_AQADM.CREATE_QUEUE_TABLE` is that of the `compatible` parameter.

To enqueue or dequeue, users need EXECUTE rights on DBMS_AQ and either enqueue or dequeue privileges on target queues, or ENQUEUE_ANY/DEQUEUE_ANY system privileges.

Queue Privileges and Access Control

You can grant or revoke privileges at the object level on queues. You can also grant or revoke various system-level privileges.

Table 4-1 lists all common Oracle Database Advanced Queuing operations and the privileges needed to perform these operations.

Table 4-1 Operations and Required Privileges

Operation(s)	Privileges Required
CREATE/DROP/MONITOR own queues	Must be granted EXECUTE rights on DBMS_AQADM. No other privileges needed.
CREATE/DROP/MONITOR any queues	Must be granted EXECUTE rights on DBMS_AQADM and be granted AQ_ADMINISTRATOR_ROLE by another user who has been granted this role (SYS and SYSTEM are the first granters of AQ_ADMINISTRATOR_ROLE)
ENQUEUE/ DEQUEUE to own queues	Must be granted EXECUTE rights on DBMS_AQ. No other privileges needed.
ENQUEUE/ DEQUEUE to another's queues	Must be granted EXECUTE rights on DBMS_AQ and be granted privileges by the owner using DBMS_AQADM.GRANT_QUEUE_PRIVILEGE.
ENQUEUE/ DEQUEUE to any queues	Must be granted EXECUTE rights on DBMS_AQ and be granted ENQUEUE ANY QUEUE or DEQUEUE ANY QUEUE system privileges by an Oracle Database Advanced Queuing administrator using DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE.

OCI Applications and Queue Access

For an Oracle Call Interface (OCI) application to access a queue, the session user must be granted either the object privilege of the queue he intends to access or the ENQUEUE ANY QUEUE or DEQUEUE ANY QUEUE system privileges.

The EXECUTE right of DBMS_AQ is not checked against the session user's rights.

Security Required for Propagation

Oracle Database Advanced Queuing propagates messages through database links.

The propagation driver dequeues from the source queue as owner of the source queue; hence, no explicit access rights need be granted on the source queue. At the destination, the login user in the database link should either be granted ENQUEUE ANY QUEUE privilege or be granted the right to enqueue to the destination queue. However, if the login user in the database link also owns the queue tables at the destination, then no explicit Oracle Database Advanced Queuing privileges must be granted.

See Also:

"Propagation from Object Queues"

Security Required for AQ Buffered Messages on Oracle RAC

Internally, buffered queues on Oracle RAC may use `dblinks` between instances. Definer's rights packages that enqueue or dequeue into buffered queues on Oracle RAC must grant `INHERIT REMOTE PRIVILEGES` to users of the package.

Queue Table Export/Import

When a queue table is exported, the queue table data and anonymous blocks of PL/SQL code are written to the export dump file. When a queue table is imported, the import utility executes these PL/SQL anonymous blocks to write the metadata to the data dictionary.

Oracle AQ does not export registrations with a user export. All applications that make use of client registrations should take this into account as the client may not be present in the imported database.

Note:

You cannot export or import buffered messages.

If there exists a queue table with the same name in the same schema in the database as in the export dump, then ensure that the database queue table is empty before importing a queue table with queues. Failing to do so has a possibility of ruining the metadata for the imported queue.

Topics:

- Exporting Queue Table Data
- Importing Queue Table Data
- Data Pump Export and Import

Exporting Queue Table Data

The export of queues entails the export of the underlying queue tables and related dictionary tables. Export of queues can also be accomplished at queue-table granularity.

Exporting Queue Tables with Multiple Recipients

For AQ queues, a queue table that supports multiple recipients is associated with the following tables:

- Dequeue index-organized table (IOT)
- Time-management index-organized table
- Subscriber table

- A history IOT

Transactional event queues are associated with the following objects:

- A queue table
- A dequeue table
- A time management table
- An optional exception queue map table
- Indexes for the above tables
- Sequences
- Rules sets and evaluation contexts

These tables are exported automatically during full database mode, user mode and table mode exports. See Export Modes .

Because the metadata tables contain ROWIDs of some rows in the queue table, the import process generates a note about the ROWIDs being made obsolete when importing the metadata tables. This message can be ignored, because the queuing system automatically corrects the obsolete ROWIDs as a part of the import operation. However, if another problem is encountered while doing the import (such as running out of rollback segment space), then you should correct the problem and repeat the import.

Export Modes

Exporting operates in full database mode, user mode, and table mode. Incremental exports on queue tables are not supported.

In full database mode, queue tables, all related tables, system-level grants, and primary and secondary object grants are exported automatically.

In user mode, queue tables, all related tables, and primary object grants are exported automatically. However, doing a user-level export from one schema to another using the `FROMUSER TOUSER` clause is not supported.

In table mode, queue tables, all related tables, and primary object grants are exported automatically. For example, when exporting an AQ multiconsumer queue table, the following tables are automatically exported:

- `AQ$_`*`queue_table`*`_I` (the dequeue IOT)
- `AQ$_`*`queue_table`*`_T` (the time-management IOT)
- `AQ$_`*`queue_table`*`_S` (the subscriber table)
- `AQ$_`*`queue_table`*`_H` (the history IOT)

For transactional event queues, the following tables are automatically exported:

- *`queue_table`*
- `AQ$_`*`queue_name`*`_L` (dequeue table)
- `AQ$_`*`queue_name`*`_T` (time-management table)
- `AQ$_`*`queue_name`* (exception map table)
- `AQ$_`*`queue_name`*`_V` (evaluation context)
- *`queue_name`*`_R` (rule set)

Importing Queue Table Data

Similar to exporting queues, importing queues entails importing the underlying queue tables and related dictionary data. After the queue table data is imported, the import utility executes the PL/SQL anonymous blocks in the dump file to write the metadata to the data dictionary.

Importing Queue Tables with Multiple Recipients

An AQ queue table that supports multiple recipients is associated with the following tables:

- A dequeue IOT
- A time-management IOT
- A subscriber table
- A history IOT

Transactional event queues are associated with the following objects:

- A queue table
- A dequeue log table
- A time management table
- An optional exception queue map table
- Indexes for the above tables
- Sequences
- Rules sets and evaluation contexts

These objects must be imported along with the queue table itself.

Import IGNORE Parameter

You must not import queue data into a queue table that already contains data. The `IGNORE` parameter of the import utility must always be set to `NO` when importing queue tables. If the `IGNORE` parameter is set to `YES`, and the queue table that already exists is compatible with the table definition in the dump file, then the rows are loaded from the dump file into the existing table. At the same time, the old queue table definition is lost and re-created. Queue table definition prior to the import is lost and duplicate rows appear in the queue table.

Data Pump Export and Import

The Data Pump replace and skip modes are supported for queue tables.

In the replace mode an existing queue table is dropped and replaced by the new queue table from the export dump file. In the skip mode, a queue table that already exists is not imported.

The truncate and append modes are not supported for queue tables. The behavior in this case is the same as the replace mode.

See Also:

Oracle Database Utilities for more information on Data Pump Export and Data Pump Import

Oracle Enterprise Manager Support for AQ Queues

Oracle Enterprise Manager supports most of the administrative functions of Oracle Database Advanced Queuing. Oracle Database Advanced Queuing functions are found under the Distributed node in the navigation tree of the Enterprise Manager console.

Functions available through Oracle Enterprise Manager include:

- Using queues as part of the schema manager to view properties
- Creating, starting, stopping, and dropping queues
- Scheduling and unscheduling propagation
- Adding and removing subscribers
- Viewing propagation schedules for all queues in the database
- Viewing errors for all queues in the database
- Viewing the message queue
- Granting and revoking privileges
- Creating, modifying, or removing transformations

Using Oracle Database Advanced Queuing with XA

You must specify "`Objects=T`" in the `xa_open` string if you want to use the Oracle Database Advanced Queuing OCI interface. This forces XA to initialize the client-side cache in Objects mode. You are not required to do this if you plan to use Oracle Database Advanced Queuing through PL/SQL wrappers from OCI or Pro*C.

The large object (LOB) memory management concepts from the Pro* documentation are not relevant for Oracle Database Advanced Queuing raw messages because Oracle Database Advanced Queuing provides a simple RAW buffer abstraction (although they are stored as LOBs).

When using the Oracle Database Advanced Queuing navigation option, you must reset the dequeue position by using the `FIRST_MESSAGE` option if you want to continue dequeuing between services (such as `xa_start` and `xa_end` boundaries). This is because XA cancels the cursor fetch state after an `xa_end`. If you do not reset, then you get an error message stating that the navigation is used out of sequence (ORA-25237).

See Also:

- "Working with Transaction Monitors with Oracle XA" in *Oracle Database Development Guide* for more information on XA
- "Large Objects (LOBs)" in *Pro*C/C++ Programmer's Guide*

Restrictions on Queue Management

These topics discuss restrictions on queue management.

- Subscribers

- DML Not Supported on Queue Tables or Associated IOTs
- Propagation from Object Queues with REF Payload Attributes
- Collection Types in Message Payloads
- Synonyms on Queue Tables and Queues
- Synonyms on Object Types
- Tablespace Point-in-Time Recovery
- Virtual Private Database

Note:

Mixed case (upper and lower case together) queue names, queue table names, and subscriber names are supported if database compatibility is 10.0, but the names must be enclosed in double quote marks. So `abc.efg` means the schema is `ABC` and the name is `EFG`, but `"abc"."efg"` means the schema is `abc` and the name is `efg`.

Subscribers

You cannot have more than 1,000 local subscribers for each queue.

Also, only 32 remote subscribers are allowed for each remote destination database.

DML Not Supported on Queue Tables or Associated IOTs

Oracle Database Advanced Queuing does not support data manipulation language (DML) operations on queue tables or associated index-organized tables (IOTs), if any.

The only supported means of modifying queue tables is through the supplied APIs. Queue tables and IOTs can become inconsistent and therefore effectively ruined, if data manipulation language (DML) operations are performed on them.

Propagation from Object Queues with REF Payload Attributes

Oracle Database Advanced Queuing does not support propagation from object queues that have REF attributes in the payload.

Collection Types in Message Payloads

You cannot construct a message payload using a VARRAY that is not itself contained within an object.

You also cannot currently use a NESTED Table even as an embedded object within a message payload. However, you can create an object type that contains one or more VARRAYs, and create a queue table that is founded on this object type, as shown in Example 4-1.

Example 4-1 Creating Objects Containing VARRAYs

```
CREATE TYPE number_varray AS VARRAY(32) OF NUMBER;
CREATE TYPE embedded_varray AS OBJECT (col1 number_varray);
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE(
  queue_table           =>     'QT',
  queue_payload_type    =>     'embedded_varray');
```

Synonyms on Queue Tables and Queues

No Oracle Database Advanced Queuing PL/SQL calls resolve synonyms on queues and queue tables.

Although you can create synonyms, you should not apply them to the Oracle Database Advanced Queuing interface.

Synonyms on Object Types

If you have created synonyms on object types, you cannot use them in `DBMS_AQADM.CREATE_QUEUE_TABLE`. Error ORA-24015 results.

Tablespace Point-in-Time Recovery

Oracle Database Advanced Queuing currently does not support tablespace point-in-time recovery.

Creating a queue table in a tablespace disables that particular tablespace for point-in-time recovery. Oracle Database Advanced Queuing does support regular point-in-time recovery.

Virtual Private Database

You can use Oracle Database Advanced Queuing with Virtual Private Database by specifying a security policy with Oracle Database Advanced Queuing queue tables.

While dequeuing, use the dequeue condition (`deq_cond`) or the correlation identifier for the policy to be applied. You can use "1=1" as the dequeue condition. If you do not use a dequeue condition or correlation ID, then the dequeue results in an error.

> **Note:**
>
> When a dequeue condition or correlation identifier is used, the order of the messages dequeued is indeterminate, and the sort order of the queue is not honored.

Managing Propagation

These topics discuss managing Oracle Database Advanced Queuing propagation.

- EXECUTE Privileges Required for Propagation
- Propagation from Object Queues
- Optimizing Propagation
- Handling Failures in Propagation

Note:

For propagation to work correctly, the queue `aq$_prop_notify_X` should never be stopped or dropped and the table `aq$_prop_table_X` should never be dropped.

EXECUTE Privileges Required for Propagation

Propagation jobs are owned by `SYS`, but the propagation occurs in the security context of the queue table owner.

Previously propagation jobs were owned by the user scheduling propagation, and propagation occurred in the security context of the user setting up the propagation schedule. The queue table owner must be granted `EXECUTE` privileges on the `DBMS_AQADM` package. Otherwise, the Oracle Database snapshot processes do not propagate and generate trace files with the error identifier `SYS.DBMS_AQADM` not defined. Private database links owned by the queue table owner can be used for propagation. The user name specified in the connection string must have `EXECUTE` access on the `DBMS_AQ` and `DBMS_AQADM` packages on the remote database.

Propagation from Object Queues

Propagation from object queues with `BFILE` objects is supported.

To be able to propagate object queues with `BFILE` objects, the source queue owner must have read privileges on the directory object corresponding to the directory in which the `BFILE` is stored. The database link user must have write privileges on the directory object corresponding to the directory of the `BFILE` at the destination database.

AQ propagation does not support non-final types. Propagation of `BFILE` objects from object queues without specifying a database link is not supported.

See Also:

"CREATE DIRECTORY" in *Oracle Database SQL Language Reference* for more information on directory objects

Optimizing Propagation

AQ propagation jobs are run by the Oracle Scheduler. Propagation may be scheduled in these ways.

- A dedicated schedule in which the propagation runs forever or for a specified duration. This mode provides the lowest propagation latencies.
- A periodic schedule in which the propagation runs periodically for a specified interval. This may be used when propagation can be run in a batched mode.
- An event based system in which propagation is started when there are messages to be propagated. This mode makes more efficient use of available resources, while still providing a fast response time.

The administrator may choose a schedule that best meets the application performance requirements.

Oracle Scheduler will start the required number of job queue processes for the propagation schedules. Since the scheduler optimizes for throughput, if the system is heavily loaded, it may not run some propagation jobs. The resource manager may be used to have better control over the scheduling decisions. In particular, associating propagation jobs with different resource groups can allow for fairness in scheduling which may be important in heavy load situations.

`In setting the number of JOB_QUEUE_PROCESSES`, DBAs should be aware that this number is determined by the number of queues from which the messages must be propagated and the number of destinations (rather than queues) to which messages must be propagated.

A scheduling algorithm handles propagation. The algorithm optimizes available job queue processes and minimizes the time it takes for a message to show up at a destination after it has been enqueued into the source queue, thereby providing near-OLTP action. The algorithm can handle an unlimited number of schedules and various types of failures. While propagation tries to make the optimal use of the available job queue processes, the number of job queue processes to be started also depends on the existence of jobs unrelated to propagation, such as replication jobs. Hence, it is important to use the following guidelines to get the best results from the scheduling algorithm.

The scheduling algorithm uses the job queue processes as follows (for this discussion, an active schedule is one that has a valid current window):

- If the number of active schedules is fewer than half the number of job queue processes, then the number of job queue processes acquired corresponds to the number of active schedules.
- If the number of active schedules is more than half the number of job queue processes, after acquiring half the number of job queue processes, then multiple active schedules are assigned to an acquired job queue process.
- If the system is overloaded (all schedules are busy propagating), depending on availability, then additional job queue processes are acquired up to one fewer than the total number of job queue processes.
- If none of the active schedules handled by a process has messages to be propagated, then that job queue process is released.
- The algorithm performs automatic load balancing by transferring schedules from a heavily loaded process to a lightly load process such that no process is excessively loaded.

Handling Failures in Propagation

The scheduling algorithm has robust support for handling failures. These are the common failures that prevent message propagation.

- Database link failed
- Remote database is not available
- Remote queue does not exist
- Remote queue was not started
- Security violation while trying to enqueue messages into remote queue

Under all these circumstances the appropriate error messages are reported in the `DBA_QUEUE_SCHEDULES` view.

When an error occurs in a schedule, propagation of messages in that schedule is attempted again after a retry period that is a function of the number of failures. After the retries have exceeded a system defined maximum, the schedule is disabled.

If the problem causing the error is fixed and the schedule is enabled, then the error fields that indicate the last error date, time, and message continue to show the error information. These fields are reset only when messages are successfully propagated in that schedule.

See Also:

Troubleshooting Oracle Database Advanced Queuing