27
DBMS_AQADM

The DBMS_AQADM package provides procedures to manage Oracle Database Advanced Queuing (AQ) configuration and administration information.

> **See Also:**
>
> - *Oracle Database Advanced Queuing User's Guide*
> - Oracle Database Advanced Queuing (AQ) Types for information about the TYPEs to use with DBMS_AQADM

This chapter contains the following topics:

DBMS_AQADM Security Model

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 DBMS_AQ and DBMS_AQADM 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 and can be granted and revoked through DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE and DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE respectively. Starting from Oracle Database 12*c* Release 2, MANAGE_ANY privilege will not allow access to SYS owned queues by users other than SYS.

User Roles

The 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.

The application developer gives 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. Starting from Oracle Database 12*c* Release 2, ENQUEUE_ANY and DEQUEUE_ANY privileges will not allow access to SYS owned queues by users other than SYS.

> **See Also:**
>
> - DBMS_AQ.
> - *Oracle Database Advanced Queuing User's Guide* for more information on queue privileges and access control.

Security 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 username specified in the connection string must have `EXECUTE` access on the `DBMS_AQ` and `DBMS_AQADM` packages on the remote database.

> **See Also:**
>
> *Oracle Database Advanced Queuing User's Guide* for more information on security required for propagation.

Queue Table Migration

The `MIGRATE_QUEUE_TABLE` procedure requires that the `EXECUTE` privilege on `DBMS_AQADM` be granted to the queue table owner, who is probably an ordinary queue user. If you do not want ordinary queue users to be able to create and drop queues and queue tables, add and delete subscribers, and so forth, then you must revoke the `EXECUTE` privilege as soon as the migration is done.

> **See Also:**
>
> - "MIGRATE_QUEUE_TABLE Procedure."
> - *Oracle Database Advanced Queuing User's Guide* for more information on granting Oracle Database Advanced Queuing system privileges.

DBMS_AQADM Constants

When using enumerated constants, such as `INFINITE`, `TRANSACTIONAL`, or `NORMAL_QUEUE`, the symbol must be specified with the scope of the packages defining it. All types associated with the administrative interfaces must be prepended with `DBMS_AQADM`. For example: `DBMS_AQADM.NORMAL_QUEUE`.

Table 27-1 Enumerated Types in the Administrative Interface

Parameter	Options
retention	0, 1, 2...INFINITE
message_grouping	TRANSACTIONAL, NONE
queue_type	NORMAL_QUEUE, EXCEPTION_QUEUE, NON_PERSISTENT_QUEUE

> **See Also:**
>
> For more information on the Java classes and data structures used in both DBMS_AQ and DBMS_AQADM, see the DBMS_AQ package.

DBMS_AQADM Subprogram Groups

This section lists and describes the DBMS_AQADM subprogram groups.

This DBMS_AQADM package is made up of the following subprogram groups:

- Queue Table Subprograms
- Privilege Subprograms
- Queue Subprograms
- Subscriber Subprograms
- OKafka Subprograms
- Propagation Subprograms
- Miscellaneous Subprograms
- Oracle Database Advanced Queuing Agent Subprograms
- Alias Subprograms

DBMS_AQADM Queue Table Subprograms

This section lists and describes the DBMS_AQADM Queue Table subprograms.

Table 27-2 Queue Table Subprograms

Subprograms	Description
ALTER_QUEUE_TABLE Procedure	Alters the existing properties of a queue table
CREATE_QUEUE_TABLE Procedure	Creates a queue table for messages of a predefined type
DROP_QUEUE_TABLE Procedure	Drops an existing queue table
ENABLE_JMS_TYPES Procedure	A precondition for the enqueue of JMS types and XML types
MIGRATE_QUEUE_TABLE Procedure	Upgrades an 8.0-compatible queue table to an 8.1-compatible or higher queue table, or downgrades an 8.1-compatible or higher queue table to an 8.0-compatible queue table

Table 27-2 (Cont.) Queue Table Subprograms

Subprograms	Description
MOVE_QUEUE_TABLE Procedure	Moves the AQ queue table within the same tablespace (local redefinition) or to a different tablespace (non-local redefinition)
PURGE_QUEUE_TABLE Procedure	Purges messages from queue tables

DBMS_AQADM Privilege Subprograms

This sections lists and describes the DBMS_AQADM Privilege subprograms.

Table 27-3 Privilege Subprograms

Subprograms	Description
GRANT_QUEUE_PRIVILEGE Procedure	Grants privileges on a queue to users and roles
GRANT_SYSTEM_PRIVILEGE Procedure	Grants Oracle Database Advanced Queuing system privileges to users and roles
REVOKE_QUEUE_PRIVILEGE Procedure	Revokes privileges on a queue from users and roles
REVOKE_SYSTEM_PRIVILEGE Procedure	Revokes Oracle Database Advanced Queuing system privileges from users and roles

DBMS_AQADM Queue Subprograms

This sections lists and describes the DBMS_AQADM Queue subprograms.

Table 27-4 Queue Subprograms

Subprograms	Description
ALTER_QUEUE Procedure	Alters existing properties of a queue
CREATE_NP_QUEUE Procedure	Creates a nonpersistent `RAW` queue
CREATE_QUEUE Procedure	Creates a queue in the specified queue table
CREATE_SHARDED_QUEUE Procedure	Creates a queue and its queue table for a sharded queue all together.
DROP_SHARDED_QUEUE Procedure	Drops an existing sharded queue from the database queuing system
ALTER_SHARDED_QUEUE Procedure	Alters an sharded queue in the database queuing system
CREATE_EXCEPTION_QUEUE Procedure	Creates an exception queue for a sharded queue
DROP_QUEUE Procedure	Drops an existing queue
QUEUE_SUBSCRIBERS Function	Returns the subscribers to an 8.0-compatible multiconsumer queue in the PL/SQL index by table collection type `DBMS_AQADM.AQ$_subscriber_list_t`
START_QUEUE Procedure	Enables the specified queue for enqueuing or dequeuing

Table 27-4 (Cont.) Queue Subprograms

Subprograms	Description
STOP_QUEUE Procedure	Disables enqueuing or dequeuing on the specified queue

DBMS_AQADM Subscriber Subprograms

This sections lists and describes the DBMS_AQADM Subscriber subprograms.

Table 27-5 Subscriber Subprograms

Subprograms	Description
ADD_SUBSCRIBER Procedure	Adds a default subscriber to a queue
ALTER_SUBSCRIBER Procedure	Alters existing properties of a subscriber to a specified queue
REMOVE_SUBSCRIBER Procedure	Removes a default subscriber from a queue

DBMS_AQADM OKafka Subprograms

This section lists and describes the DBMS_AQADM OKafka subprograms.

Table 27-6 OKafka Subprograms

Subprograms	Description
CREATE_DATABASE_KAFKA_TOPIC Procedure	Creates an OKafka Topic. OKafka topic is a multi-consumer transactional event queue specifically design to be used with OKafka application.
DROP_DATABASE_KAFKA_TOPIC Procedure	Drops an existing OKafka topic. OKafka topic is a multi-consumer transactional event queue specifically design to be used with OKafka application.

DBMS_AQADM Propagation Subprograms

This section lists and describes the DBMS_AQADM propagation subprograms.

Table 27-7 Propagation Subprograms

Subprograms	Description
ALTER_PROPAGATION_SCHEDULE Procedure	Alters parameters for a propagation schedule
DISABLE_PROPAGATION_SCHEDULE Procedure	Disables a propagation schedule
ENABLE_PROPAGATION_SCHEDULE Procedure	Enables a previously disabled propagation schedule
SCHEDULE_PROPAGATION Procedure	Schedules propagation of messages from a queue to a destination identified by a specific database link
UNSCHEDULE_PROPAGATION Procedure	Unschedules previously scheduled propagation of messages from a queue to a destination identified by a specific database link

Table 27-7 (Cont.) Propagation Subprograms

Subprograms	Description
VERIFY_QUEUE_TYPES Procedure	Verifies that the source and destination queues have identical types

DBMS_AQADM Miscellaneous Subprograms

This section lists and describes the DBMS_AQADM miscellaneous subprograms.

Table 27-8 Miscellaneous Subprograms

Subprograms	Description
GET_QUEUE_PARAMETER Procedure	Used to get different parameters for sharded queues at queue or database level.
GET_MAX_STREAMS_POOL Procedure	Retrieves the value of Oracle Database Advanced Queuing maximum streams pool memory limit
GET_MIN_STREAMS_POOL Procedure	Retrieves the value of Oracle Database Advanced Queuing minimum streams pool memory limit
GET_WATERMARK Procedure	Retrieves the value of watermark set by the `SET_WATERMARK` Procedure
SET_QUEUE_PARAMETER Procedure	Used to set different parameters for sharded queues at queue or database level.
SET_MAX_STREAMS_POOL Procedure	Used for Oracle Database Advanced Queuing to specify and limit maximum streams pool memory use
SET_MIN_STREAMS_POOL Procedure	Used for Oracle Database Advanced Queuing to specify and limit minimum streams pool memory use
SET_WATERMARK Procedure	Used for Oracle Database Advanced Queuing notification to specify and limit memory use
UNSET_QUEUE_PARAMETER Procedure	Used to unset different parameters for sharded queues at queue or database level.

DBMS_AQADM Agent Subprograms

This section lists and describes the DBMS_AQADM agent subprograms.

Table 27-9 Oracle Streams AQ Agent Subprograms

Subprograms	Description
ALTER_AQ_AGENT Procedure	Alters an agent registered for Oracle Database Advanced Queuing Internet access, and an Oracle Database Advanced Queuing agent that accesses secure queues
CREATE_AQ_AGENT Procedure	Registers an agent for Oracle Database Advanced Queuing Internet access using HTTP/SMTP protocols, and creates an Oracle Database Advanced Queuing agent to access secure queues
DISABLE_DB_ACCESS Procedure	Revokes the privileges of a specific database user from an Oracle Database Advanced Queuing Internet agent

Table 27-9 (Cont.) Oracle Streams AQ Agent Subprograms

Subprograms	Description
DROP_AQ_AGENT Procedure	Drops an agent that was previously registered for Oracle Database Advanced Queuing Internet access
ENABLE_DB_ACCESS Procedure	Grants an Oracle Database Advanced Queuing Internet agent the privileges of a specific database user

DBMS_AQADM Alias Subprograms

This section lists and describes the DBMS_AQADM alias subprograms.

Table 27-10 Alias Subprograms

Subprograms	Description
ADD_ALIAS_TO_LDAP Procedure	Creates an alias for a queue, agent, or a JMS ConnectionFactory in LDAP
DEL_ALIAS_FROM_LDAP Procedure	Drops an alias for a queue, agent, or JMS ConnectionFactory in LDAP

Summary of DBMS_AQADM Subprograms

This section lists and describes the DBMS_AQADM package subprograms.

Table 27-11 DBMS_AQADM Package Subprograms

Subprograms	Description
ADD_ALIAS_TO_LDAP Procedure	Creates an alias for a queue, agent, or a JMS ConnectionFactory in LDAP
ADD_SUBSCRIBER Procedure	Adds a default subscriber to a queue
ALTER_AQ_AGENT Procedure	Alters an agent registered for Oracle Database Advanced Queuing Internet access, and an Oracle Database Advanced Queuing agent that accesses secure queues
ALTER_PROPAGATION_SCHEDULE Procedure	Alters parameters for a propagation schedule
ALTER_QUEUE Procedure	Alters existing properties of a queue
ALTER_QUEUE_TABLE Procedure	Alters the existing properties of a queue table
ALTER_SHARDED_QUEUE Procedure	Provides user the ability to alter the `cache_hint` and comment for the sharded queue *Starting with Oracle Database 20c, sharded queues are deprecated and will be desupported in a futur release. Use Transactional Event Queues(TEQ) instead.*
ALTER_SUBSCRIBER Procedure	Alters existing properties of a subscriber to a specified queue
ALTER_TRANSACTIONAL_EVENT_QUEUE Procedure	Provides user the ability to alter the `cache_hint` and comment for the TEQ queue

Table 27-11 (Cont.) DBMS_AQADM Package Subprograms

Subprograms	Description
CREATE_AQ_AGENT Procedure	Registers an agent for Oracle Database Advanced Queuing Internet access using HTTP/SMTP protocols, and creates an Oracle Database Advanced Queuing agent to access secure queues
CREATE_NP_QUEUE Procedure	Creates a nonpersistent `RAW` queue
CREATE_QUEUE Procedure	Creates a queue in the specified queue table
CREATE_SHARDED_QUEUE Procedure	Creates a queue and its queue table for a sharded queue all together. *Starting with Oracle Database 20c, sharded queues are deprecated and will be desupported in a futur release. Use Transactional Event Queues(TEQ) instead.*
CREATE_EXCEPTION_QUEUE Procedure	Creates an exception queue. *Starting with Oracle Database 20c, sharded queues are deprecated and will be desupported in a futur release. Use Transactional Event Queues(TEQ) instead.*
CREATE_EQ EXCEPTION_QUEUE Procedure	Creates an exception queue.
CREATE_TRANSACTIONAL_EVENT_QUEUE Procedure	Creates a queue and its queue table for a Transactional Event Queue (TEQ).
CREATE_QUEUE_TABLE Procedure	Creates a queue table for messages of a predefined type
DEL_ALIAS_FROM_LDAP Procedure	Drops an alias for a queue, agent, or JMS ConnectionFactory in LDAP
DISABLE_DB_ACCESS Procedure	Revokes the privileges of a specific database user from an Oracle Database Advanced Queuing Internet agent
DISABLE_PROPAGATION_SCHEDULE Procedure	Disables a propagation schedule
DROP_AQ_AGENT Procedure	Drops an agent that was previously registered for Oracle Database Advanced Queuing Internet access
DROP_QUEUE Procedure	Drops an existing queue
DROP_SHARDED_QUEUE Procedure	Drops an existing sharded queue from the database queuing system *Starting with Oracle Database 20c, sharded queues are deprecated and will be desupported in a futur release. Use Transactional Event Queues(TEQ) instead.*
DROP_TRANSACTIONAL_EVENT_QUEUE Procedure	Drops an existing TEQ queue from the database queuing system
DROP_QUEUE_TABLE Procedure	Drops an existing queue table
ENABLE_DB_ACCESS Procedure	Grants an Oracle Database Advanced Queuing Internet agent the privileges of a specific database user
ENABLE_JMS_TYPES Procedure	A precondition for the enqueue of JMS types and XML types

Table 27-11 (Cont.) DBMS_AQADM Package Subprograms

Subprograms	Description
ENABLE_PROPAGATION_SCHEDULE Procedure	Enables a previously disabled propagation schedule
GET_MAX_STREAMS_POOL Procedure	Retrieves the value of Oracle Database Advanced Queuing maximum streams pool memory limit
GET_MIN_STREAMS_POOL Procedure	Retrieves the value of Oracle Database Advanced Queuing minimum streams pool memory limit
GET_QUEUE_PARAMETER Procedure	Used to get different parameters for sharded queues at queue or database level.
GET_WATERMARK Procedure	Retrieves the value of watermark set by the `SET_WATERMARK` Procedure
GRANT_QUEUE_PRIVILEGE Procedure	Grants privileges on a queue to users and roles
GRANT_SYSTEM_PRIVILEGE Procedure	Grants Oracle Database Advanced Queuing system privileges to users and roles
MIGRATE_QUEUE_TABLE Procedure	Upgrades an 8.0-compatible queue table to an 8.1-compatible or higher queue table, or downgrades an 8.1-compatible or higher queue table to an 8.0-compatible queue table
PURGE_QUEUE_TABLE Procedure	Purges messages from queue tables
QUEUE_SUBSCRIBERS Function	Returns the subscribers to an 8.0-compatible multiconsumer queue in the PL/SQL index by table collection type `DBMS_AQADM.AQ$_subscriber_list_t`
REMOVE_SUBSCRIBER Procedure	Removes a default subscriber from a queue
REVOKE_QUEUE_PRIVILEGE Procedure	Revokes privileges on a queue from users and roles
REVOKE_SYSTEM_PRIVILEGE Procedure	Revokes Oracle Database Advanced Queuing system privileges from users and roles
SCHEDULE_PROPAGATION Procedure	Schedules propagation of messages from a queue to a destination identified by a specific database link
SET_QUEUE_PARAMETER Procedure	Used to set different parameters for sharded queues at queue or database level.
SET_MAX_STREAMS_POOL Procedure	Used for Oracle Database Advanced Queuing to specify and limit maximum streams pool memory use
SET_MIN_STREAMS_POOL Procedure	Used for Oracle Database Advanced Queuing to specify and limit minimum streams pool memory use
SET_WATERMARK Procedure	Used for Oracle Database Advanced Queuing notification to specify and limit memory use
START_QUEUE Procedure	Enables the specified queue for enqueuing or dequeuing
STOP_QUEUE Procedure	Disables enqueuing or dequeuing on the specified queue
UNSCHEDULE_PROPAGATION Procedure	Unschedules previously scheduled propagation of messages from a queue to a destination identified by a specific database link

Table 27-11 (Cont.) DBMS_AQADM Package Subprograms

Subprograms	Description
UNSET_QUEUE_PARAMETER Procedure	Used to unset different parameters for sharded queues at queue or database level.
VERIFY_QUEUE_TYPES Procedure	Verifies that the source and destination queues have identical types

ADD_ALIAS_TO_LDAP Procedure

This procedure creates an alias for a queue, agent, or a JMS ConnectionFactory in LDAP. The alias will be placed directly under the database server's distinguished name in LDAP hierarchy.

Syntax

```
DBMS_AQADM.ADD_ALIAS_TO_LDAP(
   alias          IN VARCHAR2,
   obj_location   IN VARCHAR2);
```

Parameters

Table 27-12 ADD_ALIAS_TO_LDAP Procedure Parameters

Parameter	Description
`alias`	Name of the alias. Example: `west_shipping`.
`obj_location`	The distinguished name of the object (queue, agent or connection factory) to which `alias` refers.

Usage Notes

This method can be used to create aliases for queues, agents, and JMS ConnectionFactory objects. These object must exist before the alias is created. These aliases can be used for JNDI lookup in JMS and Oracle Database Advanced Queuing Internet access.

ADD_SUBSCRIBER Procedure

This procedure adds a default subscriber to a queue.

Syntax

```
DBMS_AQADM.ADD_SUBSCRIBER (
   queue_name      IN    VARCHAR2,
   subscriber      IN    sys.aq$_agent,
   rule            IN    VARCHAR2 DEFAULT NULL,
   transformation  IN    VARCHAR2 DEFAULT NULL
   queue_to_queue  IN    BOOLEAN DEFAULT FALSE,
   delivery_mode   IN    PLS_INTEGER DEFAULT DBMS_AQADM.PERSISTENT);
```

Parameters

Table 27-13 ADD_SUBSCRIBER Procedure Parameters

Parameter	Description
queue_name	Name of the queue.
subscriber	Agent on whose behalf the subscription is being defined.
rule	A conditional expression based on the message properties, the message data properties and PL/SQL functions. A rule is specified as a Boolean expression using syntax similar to the WHERE clause of a SQL query. This Boolean expression can include conditions on message properties, user data properties (object payloads only), and PL/SQL or SQL functions (as specified in the where clause of a SQL query). Currently supported message properties are priority and corrid. To specify rules on a message payload (object payload), use attributes of the object type in clauses. You must prefix each attribute with tab.user_data as a qualifier to indicate the specific column of the queue table that stores the payload. The rule parameter cannot exceed 4000 characters.
transformation	Specifies a transformation that will be applied when this subscriber dequeues the message. The source type of the transformation must match the type of the queue. If the subscriber is remote, then the transformation is applied before propagation to the remote queue.
queue_to_queue	If TRUE, propagation is from queue-to-queue.
delivery_mode	The administrator may specify one of DBMS_AQADM.PERSISTENT, DBMS_AQADM.BUFFERED, or DBMS_AQADM.PERSISTENT_OR_BUFFERED for the delivery mode of the messages the subscriber is interested in. This parameter will not be modifiable by ALTER_SUBSCRIBER.

Usage Notes

A program can enqueue messages to a specific list of recipients or to the default list of subscribers. This operation only succeeds on queues that allow multiple consumers. This operation takes effect immediately, and the containing transaction is committed. Enqueue requests that are executed after the completion of this call will reflect the new behavior.

Any string within the rule must be quoted:

```
rule   => 'PRIORITY <= 3 AND CORRID =  ''FROM JAPAN'''
```

Note that these are all single quotation marks.

ALTER_AQ_AGENT Procedure

This procedure alters an agent registered for Oracle Database Advanced Queuing Internet access. It is also used to alter an Oracle Database Advanced Queuing agent that accesses secure queues.

Syntax

```
DBMS_AQADM.ALTER_AQ_AGENT (
  agent_name                IN VARCHAR2,
  certificate_location      IN VARCHAR2 DEFAULT NULL,
  enable_http               IN BOOLEAN DEFAULT FALSE,
```

```
  enable_smtp              IN BOOLEAN DEFAULT FALSE,
  enable_anyp              IN BOOLEAN DEFAULT FALSE )
```

Parameters

Table 27-14 ALTER_AQ_AGENT Procedure Parameters

Parameter	Description
agent_name	Specifies the username of the Oracle Database Advanced Queuing Internet agent.
certification_location	Agent's certificate location in LDAP (default is NULL). If the agent is allowed to access Oracle Database Advanced Queuing through SMTP, then its certificate must be registered in LDAP. For access through HTTP, the certificate location is not required.
enable_http	TRUE means the agent can access Oracle Database Advanced Queuing through HTTP. FALSE means the agent cannot access Oracle Database Advanced Queuing through HTTP.
enable_smtp	TRUE means the agent can access Oracle Database Advanced Queuing through SMTP (e-mail). FALSE means the agent cannot access Oracle Database Advanced Queuing through SMTP.
enable_anyp	TRUE means the agent can access Oracle Database Advanced Queuing through any protocol (HTTP or SMTP).

ALTER_PROPAGATION_SCHEDULE Procedure

This procedure alters parameters for a propagation schedule.

Syntax

```
DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE (
   queue_name           IN    VARCHAR2,
   destination          IN    VARCHAR2 DEFAULT NULL,
   duration             IN    NUMBER   DEFAULT NULL,
   next_time            IN    VARCHAR2 DEFAULT NULL,
   latency              IN    NUMBER   DEFAULT 60,
   destination_queue    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-15 ALTER_PROPAGATION_SCHEDULE Procedure Parameters

Parameter	Description
queue_name	Name of the source queue whose messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the user.
destination	Destination database link. Messages in the source queue for recipients at this destination are propagated. If it is NULL, then the destination is the local database and messages are propagated to other queues in the local database. The length of this field is currently limited to 128 bytes, and if the name is not fully qualified, then the default domain name is used.
duration	Duration of the propagation window in seconds. A NULL value means the propagation window is forever or until the propagation is unscheduled.

Table 27-15 (Cont.) ALTER_PROPAGATION_SCHEDULE Procedure Parameters

Parameter	Description
next_time	Date function to compute the start of the next propagation window from the end of the current window. If this value is NULL, then propagation is stopped at the end of the current window. For example, to start the window at the same time every day, next_time should be specified as SYSDATE + 1 - duration/86400.
latency	Maximum wait, in seconds, in the propagation window for a message to be propagated after it is enqueued. The default value is 60. Caution: if latency is not specified for this call, then latency will over-write any existing value with the default value. For example, if the latency is 60 seconds and there are no messages to be propagated during the propagation window, then messages from that queue for the destination are not propagated for at least 60 more seconds. It will be at least 60 seconds before the queue will be checked again for messages to be propagated for the specified destination. If the latency is 600, then the queue will not be checked for 10 minutes and if the latency is 0, then a job queue process will be waiting for messages to be enqueued for the destination and as soon as a message is enqueued it will be propagated.
destination_queue	Name of the target queue to which messages are to be propagated in the form of a dblink

ALTER_QUEUE Procedure

This procedure alters existing properties of a queue. The parameters max_retries, retention_time, and retry_delay are not supported for nonpersistent queues.

Syntax

```
DBMS_AQADM.ALTER_QUEUE (
   queue_name         IN    VARCHAR2,
   max_retries        IN    NUMBER   DEFAULT NULL,
   retry_delay        IN    NUMBER   DEFAULT NULL,
   retention_time     IN    NUMBER   DEFAULT NULL,
   auto_commit        IN    BOOLEAN  DEFAULT TRUE,
   comment            IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-16 ALTER_QUEUE Procedure Parameters

Parameter	Description
queue_name	Name of the queue that is to be altered
max_retries	Limits the number of times a dequeue with REMOVE mode can be attempted on a message. The maximum value of max_retries is 2**31 -1. A message is moved to an exception queue if RETRY_COUNT is greater than MAX_RETRIES. RETRY_COUNT is incremented when the application issues a rollback after executing the dequeue. If a dequeue transaction fails because the server process dies (including ALTER SYSTEM KILL SESSION) or SHUTDOWN ABORT on the instance, then RETRY_COUNT is not incremented. Note that max_retries is supported for all single consumer queues and 8.1-compatible or higher multiconsumer queues but not for 8.0-compatible multiconsumer queues.

Table 27-16 (Cont.) ALTER_QUEUE Procedure Parameters

Parameter	Description
retry_delay	Delay time in seconds before this message is scheduled for processing again after an application rollback. The default is NULL, which means that the value will not be altered. Note that retry_delay is supported for single consumer queues and 8.1-compatible or higher multiconsumer queues but not for 8.0-compatible multiconsumer queues.
retention_time	Retention time in seconds for which a message is retained in the queue table after being dequeued. The default is NULL, which means that the value will not be altered.
auto_commit	TRUE causes the current transaction, if any, to commit before the ALTER_QUEUE operation is carried out. The ALTER_QUEUE operation become persistent when the call returns. This is the default. FALSE means the operation is part of the current transaction and becomes persistent only when the caller enters a commit. Caution: This parameter has been deprecated.
comment	User-specified description of the queue. This user comment is added to the queue catalog. The default value is NULL, which means that the value will not be changed.

ALTER_QUEUE_TABLE Procedure

This procedure alters the existing properties of a queue table.

Syntax

```
DBMS_AQADM.ALTER_QUEUE_TABLE (
   queue_table          IN   VARCHAR2,
   comment              IN   VARCHAR2       DEFAULT NULL,
   primary_instance     IN   BINARY_INTEGER DEFAULT NULL,
   secondary_instance   IN   BINARY_INTEGER DEFAULT NULL,
   replication_mode     IN   BINARY_INTEGER DEFAULT NULL);
```

Parameters

Table 27-17 ALTER_QUEUE_TABLE Procedure Parameters

Parameter	Description
queue_table	Name of a queue table to be created.
comment	Modifies the user-specified description of the queue table. This user comment is added to the queue catalog. The default value is NULL which means that the value will not be changed.
primary_instance	This is the primary owner of the queue table. Queue monitor scheduling and propagation for the queues in the queue table will be done in this instance. The default value is NULL, which means that the current value will not be changed.
secondary_instance	The queue table fails over to the secondary instance if the primary instance is not available. The default value is NULL, which means that the current value will not be changed.

Table 27-17 (Cont.) ALTER_QUEUE_TABLE Procedure Parameters

Parameter	Description
replication_mode	DBMS_AQADM.REPLICATION_MODE if queue is being altered to be in the Replication Mode. DBMS_AQADM.NONE if queue is being altered to not be replicated. The default value is NULL.

ALTER_SHARDED_QUEUE Procedure

This procedure provides user the ability to alter a sharded queue.

> **See Also:**
>
> *Oracle® Database Advanced Queuing User's Guide* for information about sharded queues

Syntax

```
PROCEDURE ALTER_SHARDED_QUEUE(
    queue_name             IN VARCHAR2,
    max_retries            IN NUMBER         DEFAULT NULL,
    comment                IN VARCHAR2       DEFAULT NULL,
    queue_properties       IN QUEUE_PROPS_T  DEFAULT NULL,
    replication_mode       IN BINARY_INTEGER DEFAULT NULL);
```

Parameters

Table 27-18 ALTER_SHARDED_QUEUE Procedure Parameters

Parameter	Description
queue_name	This parameter specifies the name of the sharded queue. A maximum of 128 characters are allowed.
max_retries	The maximum number of retries allowed.
comment	The comment of the queue.
queue_properties	Properties such as Normal or Exception Queue, Retry delay, retention time, sort list and cache hint. Refer to QUEUE_PROPS_T Typefor more information about queue_properties.
replication_mode	Reserved for future use. DBMS_AQADM.REPLICATION_MODE if queue is being altered to be in the Replication Mode or else DBMS_AQADM.NONE. Default is NULL.

ALTER_SUBSCRIBER Procedure

This procedure alters existing properties of a subscriber to a specified queue. Only the rule can be altered.

Syntax

```
DBMS_AQADM.ALTER_SUBSCRIBER (
   queue_name     IN    VARCHAR2,
   subscriber     IN    sys.aq$_agent,
   rule           IN    VARCHAR2
   transformation IN    VARCHAR2);
```

Parameters

Table 27-19 ALTER_SUBSCRIBER Procedure Parameters

Parameter	Description
`queue_name`	Name of the queue.
`subscriber`	Agent on whose behalf the subscription is being altered. See "AQ__AGENT Type".
`rule`	A conditional expression based on the message properties, the message data properties and PL/SQL functions. The rule parameter cannot exceed 4000 characters. To eliminate the rule, set the rule parameter to `NULL`.
`transformation`	Specifies a transformation that will be applied when this subscriber dequeues the message. The source type of the transformation must match the type of the queue. If the subscriber is remote, then the transformation is applied before propagation to the remote queue.

Usage Notes

This procedure alters both the rule and the transformation for the subscriber. If you want to retain the existing value for either of them, you must specify its old value. The current values for rule and transformation for a subscriber can be obtained from the *schema*.AQ$*queue_table*_R and *schema*.AQ$*queue_table*_S views.

ALTER_TRANSACTIONAL_EVENT_QUEUE Procedure

This procedure provides user the ability to alter a TEQ queue.

> **See Also:**
>
> *Oracle® Database Advanced Queuing User's Guide* for information about TEQ queues

Syntax

```
PROCEDURE ALTER_TRANSACTIONAL_EVENT_QUEUE(
    queue_name              IN VARCHAR2,
    max_retries             IN NUMBER         DEFAULT NULL,
    comment                 IN VARCHAR2       DEFAULT NULL,
```

```
    queue_properties       IN QUEUE_PROPS_T  DEFAULT NULL,
    replication_mode       IN BINARY_INTEGER DEFAULT NULL);
```

Parameters

Table 27-20 ALTER_TRANSACTIONAL_EVENT_QUEUE Procedure Parameters

Parameter	Description
queue_name	This parameter specifies the name of the TEQ queue. A maximum of 128 characters are allowed.
max_retries	The maximum number of retries allowed.
comment	The comment of the queue.
queue_properties	Properties such as Normal or Exception Queue, Retry delay, retention time, sort list and cache hint. Refer to QUEUE_PROPS_T Typefor more information about queue_properties.
replication_mode	Reserved for future use. DBMS_AQADM.REPLICATION_MODE if queue is being altered to be in the Replication Mode or else DBMS_AQADM.NONE. Default is NULL.

CREATE_AQ_AGENT Procedure

This procedure registers an agent for Oracle Database Advanced Queuing Internet access using HTTP/SMTP protocols. It is also used to create an Oracle Database Advanced Queuing agent to access secure queues.

Syntax

```
DBMS_AQADM.CREATE_AQ_AGENT (
  agent_name                IN VARCHAR2,
  certificate_location      IN VARCHAR2 DEFAULT NULL,
  enable_http               IN BOOLEAN DEFAULT FALSE,
  enable_smtp               IN BOOLEAN DEFAULT FALSE,
  enable_anyp               IN BOOLEAN DEFAULT FALSE )
```

Parameters

Table 27-21 CREATE_AQ_AGENT Procedure Parameters

Parameter	Description
agent_name	Specifies the username of the Oracle Database Advanced Queuing Internet agent.
certification_location	Agent's certificate location in LDAP (default is NULL). If the agent is allowed to access Oracle Database Advanced Queuing through SMTP, then its certificate must be registered in LDAP. For access through HTTP, the certificate location is not required.
enable_http	TRUE means the agent can access Oracle Database Advanced Queuing through HTTP. FALSE means the agent cannot access Oracle Database Advanced Queuing through HTTP.
enable_smtp	TRUE means the agent can access Oracle Database Advanced Queuing through SMTP (e-mail). FALSE means the agent cannot access Oracle Database Advanced Queuing through SMTP.

Table 27-21 (Cont.) CREATE_AQ_AGENT Procedure Parameters

Parameter	Description
enable_anyp	TRUE means the agent can access Oracle Database Advanced Queuing through any protocol (HTTP or SMTP).

Usage Notes

The SYS.AQ$INTERNET_USERS view has a list of all Oracle Database Advanced Queuing Internet agents.

CREATE_DATABASE_KAFKA_TOPIC Procedure

This procedure create an OKafka Topic. OKafka Topic is a multi-consumer transactional event queue specifically design to be used with OKafka application.

This procedure will create a transactional event queue underneath which should only be access by OKafka programming interfaces documented at Kafka APIs for Oracle Transactional Event Queues.

Syntax

```
PROCEDURE CREATE_DATABASE_KAFKA_TOPIC(
            topicname IN VARCHAR2,
            partition_num IN NUMBER  Default 1,
            retentiontime IN NUMBER Default 7*24*3600,
            partition_assignment_mode IN NUMBER DEFAULT 2,
            replication_mode IN BINARY_INTEGER DEFAULT SYS.DBMS_AQADM.NONE)
```

Parameters

Table 27-22 CREATE_DATABASE_KAFKA_TOPIC Procedure Parameters

Parameter	Description
topicname	Specifies the name of the topic that is to be created. The name must be unique within a schema and must follow object name guidelines in Oracle SQL Reserved Words and Keywords with regard to reserved characters.
partition_num	Specifies the number of maximum partitions this topic can have
retentiontime	Specifies the retention time for this topic in seconds
partition_assignment_mode	Partition assignment strategy for this topic. Value for this parameter should be 2.
replication_mode	DBMS_AQADM.REPLICATION_MODE if queue is being created in the Replication Mode or else DBMS_AQADM.NONE. Default is DBMS_AQADM.NONE.

CREATE_NP_QUEUE Procedure

This procedure creates a nonpersistent `RAW` queue.

> **Note:**
>
> Nonpersistent queues are deprecated as of Release 10gR2. Oracle recommends using buffered messaging.

Syntax

```
DBMS_AQADM.CREATE_NP_QUEUE (
   queue_name              IN        VARCHAR2,
   multiple_consumers      IN        BOOLEAN  DEFAULT FALSE,
   comment                 IN        VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-23 CREATE_NP_QUEUE Procedure Parameters

Parameter	Description
`queue_name`	Name of the nonpersistent queue that is to be created. The name must be unique within a schema and must follow object name guidelines in *Oracle Database SQL Language Reference.*
`multiple_consumers`	`FALSE` means queues created in the table can only have one consumer for each message. This is the default. `TRUE` means queues created in the table can have multiple consumers for each message. Note that this parameter is distinguished at the queue level, because a nonpersistent queue does not inherit this characteristic from any user-created queue table.
`comment`	User-specified description of the queue. This user comment is added to the queue catalog.

Usage Notes

The queue may be either single-consumer or multiconsumer queue. All queue names must be unique within a schema. The queues are created in a 8.1-compatible or higher system-created queue table (`AQ$_MEM_SC` or `AQ$_MEM_MC`) in the same schema as that specified by the queue name.

If the queue name does not specify a schema name, the queue is created in the login user's schema. After a queue is created with `CREATE_NP_QUEUE`, it can be enabled by calling `START_QUEUE`. By default, the queue is created with both enqueue and dequeue disabled.

You cannot dequeue from a nonpersistent queue. The only way to retrieve a message from a nonpersistent queue is by using the OCI notification mechanism. You cannot invoke the `LISTEN` call on a nonpersistent queue.

CREATE_QUEUE Procedure

This procedure creates a queue in the specified queue table.

Syntax

```
DBMS_AQADM.CREATE_QUEUE (
   queue_name          IN       VARCHAR2,
   queue_table         IN       VARCHAR2,
   queue_type          IN       BINARY_INTEGER DEFAULT NORMAL_QUEUE,
   max_retries         IN       NUMBER         DEFAULT NULL,
   retry_delay         IN       NUMBER         DEFAULT 0,
   retention_time      IN       NUMBER         DEFAULT 0,
   dependency_tracking IN       BOOLEAN        DEFAULT FALSE,
   comment             IN       VARCHAR2       DEFAULT NULL,
   auto_commit         IN       BOOLEAN        DEFAULT TRUE);
```

Parameters

Table 27-24 CREATE_QUEUE Procedure Parameters

Parameter	Description
queue_name	Name of the queue that is to be created. The name must be unique within a schema and must follow object name guidelines in *Oracle Database SQL Language Reference* with regard to reserved characters.
queue_table	Name of the queue table that will contain the queue.
queue_type	Specifies whether the queue being created is an exception queue or a normal queue. NORMAL_QUEUE means the queue is a normal queue. This is the default. EXCEPTION_QUEUE means it is an exception queue. Only the dequeue operation is allowed on the exception queue.
max_retries	Limits the number of times a dequeue with the REMOVE mode can be attempted on a message. The maximum value of max_retries is 2**31 -1. A message is moved to an exception queue if RETRY_COUNT is greater than MAX_RETRIES. RETRY_COUNT is incremented when the application issues a rollback after executing the dequeue. If a dequeue transaction fails because the server process dies (including ALTER SYSTEM KILL SESSION) or SHUTDOWN ABORT on the instance, then RETRY_COUNT is not incremented. Note that max_retries is supported for all single consumer queues and 8.1-compatible or higher multiconsumer queues but not for 8.0-compatible multiconsumer queues.
retry_delay	Delay time, in seconds, before this message is scheduled for processing again after an application rollback. The default is 0, which means the message can be retried as soon as possible. This parameter has no effect if max_retries is set to 0. Note that retry_delay is supported for single consumer queues and 8.1-compatible or higher multiconsumer queues but not for 8.0-compatible multiconsumer queues.
retention_time	Number of seconds for which a message is retained in the queue table after being dequeued from the queue. INFINITE means the message is retained forever. NUMBER is the number of seconds for which to retain the messages. The default is 0, no retention.

Table 27-24 (Cont.) CREATE_QUEUE Procedure Parameters

Parameter	Description
dependency_tracking	Reserved for future use. FALSE is the default. TRUE is not permitted in this release.
comment	User-specified description of the queue. This user comment is added to the queue catalog.
auto_commit	TRUE causes the current transaction, if any, to commit before the CREATE_QUEUE operation is carried out. The CREATE_QUEUE operation becomes persistent when the call returns. This is the default. FALSE means the operation is part of the current transaction and becomes persistent only when the caller enters a commit. Caution: This parameter has been deprecated.

Usage Notes

All queue names must be unique within a schema. After a queue is created with CREATE_QUEUE, it can be enabled by calling START_QUEUE. By default, the queue is created with both enqueue and dequeue disabled.

CREATE_QUEUE_TABLE Procedure

This procedure creates a queue table for messages of a predefined type.

Syntax

```
DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table          IN      VARCHAR2,
   queue_payload_type   IN      VARCHAR2,
   storage_clause       IN      VARCHAR2        DEFAULT NULL,
   sort_list            IN      VARCHAR2        DEFAULT NULL,
   multiple_consumers   IN      BOOLEAN         DEFAULT FALSE,
   message_grouping     IN      BINARY_INTEGER  DEFAULT NONE,
   comment              IN      VARCHAR2        DEFAULT NULL,
   auto_commit          IN      BOOLEAN         DEFAULT TRUE,
   primary_instance     IN      BINARY_INTEGER  DEFAULT 0,
   secondary_instance   IN      BINARY_INTEGER  DEFAULT 0,
   compatible           IN      VARCHAR2        DEFAULT NULL,
   secure               IN      BOOLEAN         DEFAULT FALSE
   replication_mode     IN      BINARY_INTEGER  DEFAULT NONE);
```

Parameters

Table 27-25 CREATE_QUEUE_TABLE Procedure Parameters

Parameter	Description
queue_table	Name of a queue table to be created
queue_payload_type	Type of the user data stored. See Type Name inDBMS_AQ Data Types for valid values for this parameter.

Table 27-25 (Cont.) CREATE_QUEUE_TABLE Procedure Parameters

Parameter	Description
storage_clause	Storage parameter. The storage parameter is included in the CREATE TABLE statement when the queue table is created. The storage_clause argument can take any text that can be used in a standard CREATE TABLE storage_clause argument.The storage parameter can be made up of any combinations of the following parameters: PCTFREE, PCTUSED, INITRANS, MAXTRANS, TABLESPACE, LOB, and a table storage clause. If a tablespace is not specified here, then the queue table and all its related objects are created in the default user tablespace. If a tablespace is specified here, then the queue table and all its related objects are created in the tablespace specified in the storage clause. See *Oracle Database SQL Language Reference* for the usage of these parameters.
sort_list	The columns to be used as the sort key in ascending order. This parameter has the following format: '*sort_column_1*,*sort_column_2*' The allowed column names are priority, enq_time, and commit_time.If both columns are specified, then *sort_column_1* defines the most significant order. After a queue table is created with a specific ordering mechanism, all queues in the queue table inherit the same defaults. The order of a queue table cannot be altered after the queue table has been created. If no sort list is specified, then all the queues in this queue table are sorted by the enqueue time in ascending order. This order is equivalent to FIFO order. Even with the default ordering defined, a dequeuer is allowed to choose a message to dequeue by specifying its msgid or correlation. msgid, correlation, and sequence_deviation take precedence over the default dequeueing order, if they are specified. When commit_time is specified for the sort_list parameter the resulting queue table uses commit-time ordering. See also "Priority and Ordering of Messages" in *Oracle Database Advanced Queuing User's Guide* for information about message ordering in Oracle Database Advanced Queuing.
multiple_consumers	FALSE means queues created in the table can only have one consumer for each message. This is the default. TRUE means queues created in the table can have multiple consumers for each message.
message_grouping	Message grouping behavior for queues created in the table. NONE means each message is treated individually. TRANSACTIONAL means messages enqueued as part of one transaction are considered part of the same group and can be dequeued as a group of related messages.
comment	User-specified description of the queue table. This user comment is added to the queue catalog.
auto_commit	TRUE causes the current transaction, if any, to commit before the CREATE_QUEUE_TABLE operation is carried out. The CREATE_QUEUE_TABLE operation becomes persistent when the call returns. This is the default. FALSE means the operation is part of the current transaction and becomes persistent only when the caller enters a commit. Note: This parameter has been deprecated.

Table 27-25 (Cont.) CREATE_QUEUE_TABLE Procedure Parameters

Parameter	Description
primary_instance	The primary owner of the queue table. Queue monitor scheduling and propagation for the queues in the queue table are done in this instance. The default value for primary instance is 0, which means queue monitor scheduling and propagation will be done in any available instance.
secondary_instance	The queue table fails over to the secondary instance if the primary instance is not available. The default value is 0, which means that the queue table will fail over to any available instance.
compatible	The lowest database version with which the queue is compatible. Currently the possible values are either 8.0, 8.1, or 10.0. If the database is in 10.1-compatible mode, the default value is 10.0. If the database is in 8.1-compatible or 9.2-compatible mode, the default value is 8.1. If the database is in 8.0 compatible mode, the default value is 8.0.
secure	This parameter must be set to TRUE if you want to use the queue table for secure queues. Secure queues are queues for which AQ agents must be associated explicitly with one or more database users who can perform queue operations, such as enqueue and dequeue. The owner of a secure queue can perform all queue operations on the queue, but other users cannot perform queue operations on a secure queue, unless they are configured as secure queue users.
replication_mode	DBMS_AQADM.REPLICATION_MODE if queue is being created in the Replication Mode or else DBMS_AQADM.NONE. Default is DBMS_AQADM.NONE.

Usage Notes

The sort keys for dequeue ordering, if any, must be defined at table creation time. The following objects are created at this time:

- aq$_*queue_table_name*_e, a default exception queue associated with the queue table
- aq$*queue_table_name*, a read-only view, which is used by Oracle Database Advanced Queuing applications for querying queue data
- aq$_*queue_table_name*_t, an index (or an index organized table (IOT) in the case of multiple consumer queues) for the queue monitor operations
- aq$_*queue_table_name*_i, an index (or an index organized table in the case of multiple consumer queues) for dequeue operations

For 8.1-compatible or higher queue tables, the following index-organized tables are created:

- aq$_*queue_table_name*_s, a table for storing information about the subscribers
- aq$_*queue_table_name*_r, a table for storing information about rules on subscriptions

aq$_*queue_table_name*_h, an index-organized table for storing the dequeue history data

CLOB, BLOB, and BFILE are valid attributes for Oracle Database Advanced Queuing object type payloads. However, only CLOB and BLOB can be propagated using Oracle Database Advanced Queuing propagation in Oracle8*i* release 8.1.5 or later. See the *Oracle Database Advanced Queuing User's Guide* for more information.

The default value of the compatible parameter depends on the database compatibility mode in the init.ora. If the database is in 10.1-compatible mode, the default value is 10.0. If the

database is in 8.1-compatible or 9.2-compatible mode, the default value is 8.1. If the database is in 8.0 compatible mode, the default value is 8.0

You can specify and modify the `primary_instance` and `secondary_instance` only in 8.1-compatible or higher mode. You cannot specify a secondary instance unless there is a primary instance.

CREATE_SHARDED_QUEUE Procedure

The `CREATE_SHARDED_QUEUE` API creates a queue and its queue table as appropriate for a sharded queue. This API cannot be used to create unsharded queues. Sharded queues must be created using this single integrated API that will automatically set AQ properties as needed

Starting with Oracle Database 20c, the `CREATE_SHARDED_QUEUE` procedure is deprecated and will be desupported in a future release. Use the CREATE_TRANSACTIONAL_EVENT_QUEUE Procedure procedure instead.

Sharded queues may be either a single consumer or a multi-consumer queue.

Syntax

```
PROCEDURE CREATE_SHARDED_QUEUE (
    queue_name              IN VARCHAR2,
    storage_clause          IN VARCHAR2       DEFAULT NULL,
    multiple_consumers      IN BOOLEAN        DEFAULT FALSE,
    max_retries             IN NUMBER         DEFAULT NULL,
    comment                 IN VARCHAR2       DEFAULT NULL,
    queue_payload_type      IN VARCHAR2       DEFAULT JMS_TYPE,
    queue_properties        IN QUEUE_PROPS_T  DEFAULT NULL,
    replication_mode        IN BINARY_INTEGER DEFAULT NONE,
    queue_type         IN BINARY_INTEGER DEFAULT SHARDED_QUEUE,
    squeue_ver              IN BINARY_INTEGER DEFAULT SQ);
```

Parameters

Table 27-26 CREATE_SHARDED_QUEUE Procedure Parameters

Parameter	Description
`queue_name`	This required parameter specifies the name of the new queue. Maximum of 128 characters allowed.
`storage_clause`	The storage parameter is included in the `CREATE TABLE` statement when the queue table is created. The `storage_clause` argument can take any text that can be used in a standard `CREATE TABLE storage_clause` argument. The storage parameter can be made up of any combinations of the following parameters: `PCTFREE`, `PCTUSED`, `INITRANS`, `MAXTRANS`, `TABLESPACE`, `LOB`, and a table storage clause. If a tablespace is not specified here, then the queue table and all its related objects are created in the default user tablespace. If a tablespace is specified here, then the queue table and all its related objects are created in the tablespace specified in the storage clause. See *Oracle Database SQL Language Reference* for the usage of these parameters.
`multiple_consumers`	`FALSE` means queues can only have one consumer for each message. This is the default. `TRUE` means queues created in the table can have multiple consumers for each message.

Table 27-26 (Cont.) CREATE_SHARDED_QUEUE Procedure Parameters

Parameter	Description
max_retries	This optional parameter limits the number of times that a dequeue can reattempted on a message after a failure. The maximum value of max_retries is 2**31 -1. After the retry limit has been exceeded, the message will be purged from the queue. RETRY_COUNT is incremented when the application issues a rollback after executing the dequeue. If a dequeue transaction fails because the server process dies (including ALTER SYSTEM KILL SESSION) or SHUTDOWN ABORT on the instance, then RETRY_COUNT is not incremented.
comment	This optional parameter is a user-specified description of the queue table. This user comment is added to the queue catalog.
queue_payload_type	Payload can be RAW, JSON, DBMS_AQADM.JMS_TYPE, or an object type. Default is DBMS_AQADM.JMS_TYPE. See DBMS_AQ Data Types.
queue_properties	Properties such as Normal or Exception Queue, Retry delay, retention time, sort list and cache hint. Refer to QUEUE_PROPS_T Type for more information about queue_properties.
replication_mode	Reserved for future use. DBMS_AQADM.REPLICATION_MODE if Queue is being created in the Replication Mode or else DBMS_AQADM.NONE. Default is DBMS_AQADM.NONE.
queue_kind	This parameter specifies the queue type. The following types are: CLASSIC_QUEUE CONSTANT BINARY_INTEGER := 0; SHARDED_QUEUE CONSTANT BINARY_INTEGER := 1; TRANSACTIONAL_EVENT_QUEUE CONSTANT BINARY_INTEGER := 2; The default value is SHARDED_QUEUE.
squeue_ver	The sharded queue phase/version. SQ CONSTANT BINARY_INTEGER := 0; TEQ CONSTANT BINARY_INTEGER := 1;

CREATE_EXCEPTION_QUEUE Procedure

The CREATE_EXCEPTION_QUEUE API creates an exception queue for a sharded queue.

Syntax

```
PROCEDURE CREATE_EXCEPTION_QUEUE(
    sharded_queue_name     IN VARCHAR2,
    exception_queue_name   IN VARCHAR2 DEFAULT NULL
    );
```

Parameters

Table 27-27 CREATE_EXCEPTION_QUEUE Procedure Parameters

Parameter	Description
sharded_queue_name	The name of the sharded queue.

Table 27-27 (Cont.) CREATE_EXCEPTION_QUEUE Procedure Parameters

Parameter	Description
exception_queue_name	The name of the exception queue.

CREATE_EQ EXCEPTION_QUEUE Procedure

The CREATE_EQ_EXCEPTION_QUEUE API creates an exception queue for a TEQ queue.

Syntax

```
PROCEDURE CREATE_EQ_EXCEPTION_QUEUE(
    teq_queue_name     IN VARCHAR2,
    exception_queue_name   IN VARCHAR2 DEFAULT NULL
    );
```

Parameters

Table 27-28 CREATE_EQ_EXCEPTION_QUEUE Procedure Parameters

Parameter	Description
teq_queue_name	The name of the TEQ queue.
exception_queue_name	The name of the exception queue.

CREATE_TRANSACTIONAL_EVENT_QUEUE Procedure

The CREATE_TRANSACTIONAL_EVENT_QUEUE API creates a queue and its queue table as appropriate for a Transactional Event Queue (TEQ). This API cannot be used to create AQ queues. TEQs must be created using this single integrated API that will automatically set AQ properties as needed.

TEQs may be either a single consumer or a multi-consumer queue.

Syntax

```
PROCEDURE CREATE_TRANSACTIONAL_EVENT_QUEUE (
    queue_name             IN VARCHAR2,
    storage_clause         IN VARCHAR2       DEFAULT NULL,
    multiple_consumers     IN BOOLEAN        DEFAULT FALSE,
    max_retries            IN NUMBER         DEFAULT NULL,
    comment                IN VARCHAR2       DEFAULT NULL,
    queue_payload_type     IN VARCHAR2       DEFAULT JMS_TYPE,
    queue_properties       IN QUEUE_PROPS_T  DEFAULT NULL,
    replication_mode       IN BINARY_INTEGER DEFAULT NONE);
```

Parameters

Table 27-29 CREATE_TRANSACTIONAL_EVENT_QUEUE Procedure Parameters

Parameter	Description
queue_name	This required parameter specifies the name of the new queue. Maximum of 128 characters allowed.

Table 27-29 (Cont.) CREATE_TRANSACTIONAL_EVENT_QUEUE Procedure Parameters

Parameter	Description
storage_clause	The storage parameter is included in the CREATE TABLE statement when the queue table is created. The storage_clause argument can take any text that can be used in a standard CREATE TABLE storage_clause argument. The storage parameter can be made up of any combinations of the following parameters: PCTFREE, PCTUSED, INITRANS, MAXTRANS, TABLESPACE, LOB, and a table storage clause. If a tablespace is not specified here, then the queue table and all its related objects are created in the default user tablespace. If a tablespace is specified here, then the queue table and all its related objects are created in the tablespace specified in the storage clause. See *Oracle Database SQL Language Reference* for the usage of these parameters.
multiple_consumers	FALSE means queues can only have one consumer for each message. This is the default. TRUE means queues created in the table can have multiple consumers for each message.
max_retries	This optional parameter limits the number of times that a dequeue can reattempted on a message after a failure. The maximum value of max_retries is 2**31 -1. After the retry limit has been exceeded, the message will be purged from the queue. RETRY_COUNT is incremented when the application issues a rollback after executing the dequeue. If a dequeue transaction fails because the server process dies (including ALTER SYSTEM KILL SESSION) or SHUTDOWN ABORT on the instance, then RETRY_COUNT is not incremented.
comment	This optional parameter is a user-specified description of the queue table. This user comment is added to the queue catalog.
queue_payload_type	Payload can be RAW, JSON, DBMS_AQADM.JMS_TYPE, or an object type. Default is DBMS_AQADM.JMS_TYPE. See DBMS_AQ Data Types.
queue_properties	Properties such as Normal or Exception Queue, Retry delay, retention time, sort list and cache hint. Refer to QUEUE_PROPS_T Type for more information about queue_properties.
replication_mode	Reserved for future use. DBMS_AQADM.REPLICATION_MODE if Queue is being created in the Replication Mode or else DBMS_AQADM.NONE. Default is DBMS_AQADM.NONE.

DEL_ALIAS_FROM_LDAP Procedure

This procedure drops an alias for a queue, agent, or JMS ConnectionFactory in LDAP.

Syntax

```
DBMS_AQADM.DEL_ALIAS_FROM_LDAP(
   alias IN VARCHAR2);
```

Parameters

Table 27-30 DEL_ALIAS_FROM_LDAP Procedure Parameters

Parameter	Description
alias	The alias to be removed.

DISABLE_DB_ACCESS Procedure

This procedure revokes the privileges of a specific database user from an Oracle Database Advanced Queuing Internet agent.

Syntax

```
DBMS_AQADM.DISABLE_DB_ACCESS (
  agent_name                IN VARCHAR2,
  db_username               IN VARCHAR2)
```

Parameters

Table 27-31 DISABLE_DB_ACCESS Procedure Parameters

Parameter	Description
agent_name	Specifies the username of the Oracle Database Advanced Queuing Internet agent.
db_username	Specifies the database user whose privileges are to be revoked from the Oracle Database Advanced Queuing Internet agent.

Usage Notes

The Oracle Database Advanced Queuing Internet agent should have been previously granted those privileges using the ENABLE_DB_ACCESS Procedure.

DISABLE_PROPAGATION_SCHEDULE Procedure

This procedure disables a propagation schedule.

Syntax

```
DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE (
   queue_name            IN   VARCHAR2,
   destination           IN   VARCHAR2 DEFAULT NULL,
   destination_queue  IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-32 DISABLE_PROPAGATION_SCHEDULE Procedure Parameters

Parameter	Description
queue_name	Name of the source queue whose messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the user.

Table 27-32 (Cont.) DISABLE_PROPAGATION_SCHEDULE Procedure Parameters

Parameter	Description
destination	Destination database link. Messages in the source queue for recipients at this destination are propagated. If it is NULL, then the destination is the local database and messages are propagated to other queues in the local database. The length of this field is currently limited to 128 bytes, and if the name is not fully qualified, then the default domain name is used.
destination_queue	Name of the target queue to which messages are to be propagated in the form of a dblink

DROP_AQ_AGENT Procedure

This procedure drops an agent that was previously registered for Oracle Database Advanced Queuing Internet access.

Syntax

```
DBMS_AQADM.DROP_AQ_AGENT (
  agent_name                IN VARCHAR2)
```

Parameters

Table 27-33 DROP_AQ_AGENT Procedure Parameters

Parameter	Description
agent_name	Specifies the username of the Oracle Database Advanced Queuing Internet agent

DROP_DATABASE_KAFKA_TOPIC Procedure

This procedure drops an existing OKafka topic. OKafka Topic is a multi-consumer transactional event queue specifically design to be used with OKafka application.

Syntax

```
PROCEDURE DROP_DATABASE_KAFKA_TOPIC(
            topicname IN VARCHAR2)
```

Parameters

Table 27-34 DROP_DATABASE_KAFKA_TOPIC Procedure Parameters

Parameter	Description
topicname	Specifies the name of the topic that is to be dropped. The name must be unique within a schema and must follow object name guidelines in Oracle SQL Reserved Words and Keywords with regard to reserved characters.

DROP_QUEUE Procedure

This procedure drops an existing queue.

Syntax

```
DBMS_AQADM.DROP_QUEUE (
   queue_name         IN    VARCHAR2,
   auto_commit        IN    BOOLEAN DEFAULT TRUE);
```

Parameters

Table 27-35 DROP_QUEUE Procedure Parameters

Parameter	Description
queue_name	Name of the queue that is to be dropped.
auto_commit	TRUE causes the current transaction, if any, to commit before the DROP_QUEUE operation is carried out. The DROP_QUEUE operation becomes persistent when the call returns. This is the default. FALSE means the operation is part of the current transaction and becomes persistent only when the caller enters a commit. Caution: This parameter has been deprecated.

Usage Notes

DROP_QUEUE is not allowed unless STOP_QUEUE has been called to disable the queue for both enqueuing and dequeuing. All the queue data is deleted as part of the drop operation.

DROP_QUEUE_TABLE Procedure

This procedure drops an existing queue table.

Syntax

```
DBMS_AQADM.DROP_QUEUE_TABLE (
   queue_table        IN    VARCHAR2,
   force              IN    BOOLEAN DEFAULT FALSE,
   auto_commit        IN    BOOLEAN DEFAULT TRUE);
```

Parameters

Table 27-36 DROP_QUEUE_TABLE Procedure Parameters

Parameter	Description
queue_table	Name of a queue table to be dropped.
force	FALSE means the operation does not succeed if there are any queues in the table. This is the default. TRUE means all queues in the table are stopped and dropped automatically.

Table 27-36 (Cont.) DROP_QUEUE_TABLE Procedure Parameters

Parameter	Description
auto_commit	TRUE causes the current transaction, if any, to commit before the DROP_QUEUE_TABLE operation is carried out. The DROP_QUEUE_TABLE operation becomes persistent when the call returns. This is the default. FALSE means the operation is part of the current transaction and becomes persistent only when the caller enters a commit. Caution: This parameter has been deprecated.

Usage Notes

All the queues in a queue table must be stopped and dropped before the queue table can be dropped. You must do this explicitly unless the force option is used, in which case this is done automatically.

DROP_SHARDED_QUEUE Procedure

This procedure drops an existing sharded queue from the database queuing system.

You must stop the queue before calling DROP_SHARDED_QUEUE. User must stop the queue explicitly if force is set to FALSE before calling DROP_SHARDED_QUEUE. If force is set to TRUE then queue will be stopped internally and then dropped.

Syntax

```
DBMS_AQADM.DROP_SHARDED_QUEUE(
       queue_name IN VARCHAR2,
       force      IN BOOLEAN DEFAULT FALSE )
```

Parameters

Table 27-37 DROP_SHARDED_QUEUE Procedure Parameters

Parameter	Description
queue_name	This required parameter specifies the name of the sharded queue.
force	The sharded queue is dropped even if the queue is not stopped.

DROP_TRANSACTIONAL_EVENT_QUEUE Procedure

This procedure drops an existing TEQ queue from the database queuing system.

You must stop the queue before calling DROP_TRANSACTIONAL_EVENT_QUEUE. User must stop the queue explicitly if force is set to FALSE before calling DROP_TRANSACTIONAL_EVENT_QUEUE. If force is set to TRUE then queue will be stopped internally and then dropped.

Syntax

```
DBMS_AQADM.DROP_TRANSACTIONAL_EVENT_QUEUE(
       queue_name IN VARCHAR2,
       force      IN BOOLEAN DEFAULT FALSE )
```

Parameters

Table 27-38 DROP_TRANSACTIONAL_EVENT_QUEUE Procedure Parameters

Parameter	Description
queue_name	This required parameter specifies the name of the TEQ queue.
force	The TEQ queue is dropped even if the queue is not stopped.

ENABLE_DB_ACCESS Procedure

This procedure grants an Oracle Database Advanced Queuing Internet agent the privileges of a specific database user.

Syntax

```
DBMS_AQADM.ENABLE_DB_ACCESS (
  agent_name                IN VARCHAR2,
  db_username               IN VARCHAR2)
```

Parameters

Table 27-39 ENABLE_DB_ACCESS Procedure Parameters

Parameter	Description
agent_name	Specifies the username of the Oracle Database Advanced Queuing Internet agent.
db_username	Specified the database user whose privileges are to be granted to the Oracle Database Advanced Queuing Internet agent.

Usage Notes

The Oracle Database Advanced Queuing Internet agent should have been previously created using the CREATE_AQ_AGENT Procedure.

For secure queues, the sender and receiver agent of the message must be mapped to the database user performing the enqueue or dequeue operation.

The SYS.AQ$INTERNET_USERS view has a list of all Oracle Database Advanced Queuing Internet agents and the names of the database users whose privileges are granted to them.

ENABLE_JMS_TYPES Procedure

Enqueue JMS types and XML types.

Syntax

```
DBMS_AQADM.ENABLE_JMS_TYPES (
   queue_table    IN    VARCHAR2);
```

Parameters

Table 27-40 ENABLE_JMS_TYPES Procedure Parameters

Parameter	Description
queue_table	Specifies name of the queue table to be enabled for JMS and XML types.

ENABLE_PROPAGATION_SCHEDULE Procedure

This procedure enables a previously disabled propagation schedule.

Syntax

```
DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE (
   queue_name          IN   VARCHAR2,
   destination         IN   VARCHAR2 DEFAULT NULL,
   destination_queue   IN   VARCHAR2  DEFAULT NULL);
```

Parameters

Table 27-41 ENABLE_PROPAGATION_SCHEDULE Procedure Parameters

Parameter	Description
queue_name	Name of the source queue whose messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the user.
destination	Destination database link. Messages in the source queue for recipients at this destination are propagated. If it is NULL, then the destination is the local database and messages are propagated to other queues in the local database. The length of this field is currently limited to 128 bytes, and if the name is not fully qualified, then the default domain name is used.
destination_queue	Name of the target queue to which messages are to be propagated in the form of a dblink

GET_MAX_STREAMS_POOL Procedure

This procedure retrieves the value of Oracle Database Advanced Queuing maximum streams pool memory limit.

Syntax

```
DBMS_AQADM.GET_MAX_STREAMS_POOL (
   value     OUT      NUMBER);
```

Parameters

Table 27-42 GET_MAX_STREAMS_POOL Procedure Parameter

Parameter	Description
value	Value in megabytes.

GET_MIN_STREAMS_POOL Procedure

This procedure retrieves the value of Oracle Database Advanced Queuing minimum streams pool memory limit.

Syntax

```
DBMS_AQADM.GET_MIN_STREAMS_POOL (
   value     OUT      NUMBER);
```

Parameters

Table 27-43 GET_MIN_STREAMS_POOL Procedure Parameter

Parameter	Description
value	Value in megabytes.

GET_QUEUE_PARAMETER Procedure

This procedure allows user to get different parameters for sharded queues at queue or database level.

For database level the queue_name should be NULL. Note that queue overrides database level parameter values.

> **See Also:**
>
> *Oracle® Database Advanced Queuing User's Guide* for information about sharded queues

Syntax

```
PROCEDURE  GET_QUEUE_PARAMETER(
    queue_name          IN VARCHAR2,
    param_name          IN VARCHAR2,
    param_value         OUT NUMBER);
```

Parameters

Table 27-44 GET_QUEUE_PARAMETER Procedure Parameters

Parameter	Description
queue_name	The name of the sharded queue.
param_name	The name of the parameter. Table 27-45 and Table 27-46 describe the valid parameter names.
param_value	The value of the parameter.

Table 27-45 Sharded queue parameters

Parameter Name	Scope	Allowed Values	Description
SHARD_NUM	Queue level	[1, UB4MAXVAL]	Number of shards created per instance for a queue, when KEY_BASED_ENQUEUE is not set. When KEY_BASED_ENQUEUE is set, the value is maximum number of shards created for the queue across all instances. The default value is 5.
KEY_BASED_ENQUEUE	Queue level	[0,1]	When set, the shard to which a message gets enqueued is determined by the key value specified in the message. Refer to key-based sharding (link) for more details. This parameter cannot be unset once set. When this parameter is not set (default), a session is bound to a shard at the time of first enqueue to the queue. All messages enqueued by the session will go to the same shard to which the session is bound.
STICKY_DEQUEUE	Queue level	[0,1]	When set, dequeue session sticks to a shard in the queue. A session is bound to a shard on first dequeue from the queue. All messages dequeued by the session come from the same shard to which it is bound. This parameter cannot be unset once set. When this parameter is not set, messages dequeued by a session can spread across multiple shards of the queue.

Table 27-46 Key-based Parameters

Parameter Name	Scope	Description
AQ$KEY_TO_SHARD_MAP	Queue level	Shard number to which a given key is mapped. When key-based sharding is enabled, this parameter is used to establish mapping between a key and a shard number or retrieve the shard number to which given key is mapped.
AQ$GET_KEY_SHARD_INST	Queue level	Instance number that owns the shard to which a given key is mapped. Applicable only when key-based sharding is enabled. It is a read-only parameter.

Example 27-1 Key to shard mapping

Users can explicitly map a new key to an existing shard or to a new shard.

To map a key value 'RED' to a shard with id 0 for queue named MY_SHQ1, submit the following:

```
execute
        dbms_aqadm.set_queue_paramter('MY_SHQ1',  'AQ$KEY_TO_SHARD_MAP=RED',
0);
```

If a key is being enqueued which has not been explicitly mapped to a shard, then one of the shards is chosen at random and mapped to that key permanently. Once a key is mapped to shard, the mapping cannot be changed.

To get the shard identifier for a key which is mapped already, submit the following:

```
declare
  pval number;
begin
  dbms_aqadm.get_queue_parameter('MY_SHQ1', 'AQ$GET_KEY_SHARD=RED', pval);
  dbms_output.put_line('The key RED is mapped to shard id ' || pval);
END;
/
```

Example 27-2 Key to instance mapping

User sharding performs best when cross instance enqueues are not involved. To know the instance where a key can be enqueued without any cross instance enqueue, submit the following:

```
declare
  pval number;
begin
  dbms_aqadm.get_queue_parameter('MY_SHQ1', 'AQ$GET_KEY_SHARD_INST=RED', pval);
  dbms_output.put_line('The key RED is owned by instance id ' || pval);
END;
/
```

GET_WATERMARK Procedure

This procedure retrieves the value of watermark set by SET_WATERMARK.

Syntax

```
DBMS_AQADM.GET_WATERMARK (
   wmvalue     IN      NUMBER);
```

Parameters

Table 27-47 GET_WATERMARK Procedure Parameter

Parameter	Description
wmvalue	Watermark value in megabytes.

GRANT_QUEUE_PRIVILEGE Procedure

This procedure grants privileges on a queue to users and roles. The privileges are ENQUEUE or DEQUEUE. Initially, only the queue table owner can use this procedure to grant privileges on the queues.

Syntax

```
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE (
   privilege        IN    VARCHAR2,
   queue_name       IN    VARCHAR2,
   grantee          IN    VARCHAR2,
   grant_option     IN    BOOLEAN := FALSE);
```

Parameters

Table 27-48 GRANT_QUEUE_PRIVILEGE Procedure Parameters

Parameter	Description
privilege	The Oracle Database Advanced Queuing queue privilege to grant. The options are ENQUEUE, DEQUEUE, and ALL. ALL means both ENQUEUE and DEQUEUE.
queue_name	Name of the queue.
grantee	Grantee(s). The grantee(s) can be a user, a role, or the PUBLIC role.
grant_option	Specifies if the access privilege is granted with the GRANT option or not. If the privilege is granted with the GRANT option, then the grantee is allowed to use this procedure to grant the access privilege to other users or roles, regardless of the ownership of the queue table. The default is FALSE.

GRANT_SYSTEM_PRIVILEGE Procedure

This procedure grants Oracle Database Advanced Queuing system privileges to users and roles.

The privileges are ENQUEUE_ANY, DEQUEUE_ANY, and MANAGE_ANY. Initially, only SYS and SYSTEM can use this procedure successfully.

> **Note:**
>
> Starting from Oracle Database 12*c* Release 2, `MANAGE_ANY`, `ENQUEUE_ANY`, and `DEQUEUE_ANY` privileges will not allow access to `SYS` owned queues by users other than `SYS`.

Syntax

```
DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE (
   privilege         IN    VARCHAR2,
   grantee           IN    VARCHAR2,
   admin_option      IN    BOOLEAN := FALSE);
```

Parameters

Table 27-49 GRANT_SYSTEM_PRIVILEGE Procedure Parameters

Parameter	Description
`privilege`	The Oracle Database Advanced Queuing system privilege to grant. The options are `ENQUEUE_ANY`, `DEQUEUE_ANY`, and `MANAGE_ANY`. `ENQUEUE_ANY` means users granted this privilege are allowed to enqueue messages to any queues in the database. `DEQUEUE_ANY` means users granted this privilege are allowed to dequeue messages from any queues in the database. `MANAGE_ANY` means users granted this privilege are allowed to run `DBMS_AQADM` calls on any schemas in the database. **Note:** Starting from Oracle Database 12*c* Release 2, `MANAGE_ANY`, `ENQUEUE_ANY`, and `DEQUEUE_ANY` privileges will not allow access to `SYS` owned queues by users other than `SYS`.
`grantee`	Grantee(s). The grantee(s) can be a user, a role, or the `PUBLIC` role.
`admin_option`	Specifies if the system privilege is granted with the `ADMIN` option or not. If the privilege is granted with the `ADMIN` option, then the grantee is allowed to use this procedure to grant the system privilege to other users or roles. The default is `FALSE`.

MIGRATE_QUEUE_TABLE Procedure

This procedure upgrades an 8.0-compatible queue table to an 8.1-compatible or higher queue table, or downgrades an 8.1-compatible or higher queue table to an 8.0-compatible queue table.

Syntax

```
DBMS_AQADM.MIGRATE_QUEUE_TABLE (
   queue_table    IN   VARCHAR2,
   compatible     IN   VARCHAR2);
```

Parameters

Table 27-50 MIGRATE_QUEUE_TABLE Procedure Parameters

Parameter	Description
queue_table	Specifies name of the queue table to be migrated.
compatible	Set this to 8.1 to upgrade an 8.0-compatible queue table, or set this to 8.0 to downgrade an 8.1-compatible queue table.

MOVE_QUEUE_TABLE Procedure

This procedure is used either to move the AQ queue table within the same tablespace (local redefinition) or to move to a different tablespace (non-local redefinition).

> **Note:**
>
> The MOVE_QUEUE_TABLE interface currently supports the movement of Non-SYS queue tables only. An attempt to move the SYS owned queue tables will be failed.

Syntax

```
DBMS_AQADM.MOVE_QUEUE_TABLE (
  queue_table    IN VARCHAR2,
  to_tablespace  IN VARCHAR2 DEFAULT NULL,
  flags          IN BINARY_INTEGER DEFAULT DBMS_AQADM.MOVEQT_OFFLINE);
```

Parameters

Table 27-51 MOVE_QUEUE_TABLE Procedure Parameters

Parameter	Description
queue_table	Name of a queue table to be moved
to_tablespace	Name of the tablespace to which the queue table is to be moved. If you intend to move within the same tablespace, then this parameter can be specified as NULL.
flags	The set of flags to control the move operation. This parameter accepts multiple values which can be bitwise OR'ed together. The following constants available in the DBMS_AQADM package can be specified as flags. • MOVEQT_OFFLINE - to perform movement in offline mode (default) • MOVEQT_ONLINE - to perform movement in online mode • MOVEQT_LOCALREDEF - to perform local movement (movement within same tablespace)

Modes of movement

The MOVE_QUEUE_TABLE interface can be used in the following two modes:

- Offline Mode
  - DBMS_AQAM.MOVEQT_OFFLINE

  - This is a default mode of movement. In this mode, the underlying queues associated with a queue table are stopped for enqueue and dequeue activity before move operation is triggered. The queues are restored to their original state after the movement.

- Online Mode
  - `DBMS_AQAM.MOVEQT_ONLINE`
  - This offers the movement in online mode. That means, you can have enqueue and dequeue activity on the queue while AQ manages its movement.

Types of movement

- Local movement
  - This allows you to move (or redefine) the queue table within its current tablespace. Local movement is useful for various usecases - such as 1. freeing the space consumed by queue table segment, 2. defragmenting the queue table IOT and Index segments, 3. shrinking the tablespace, 4. Rectifying the block corruption within queue table segment etc.
  - The `to_tablespace` parameter becomes optional to specify for this type of movement.
  - If `to_tablespace` is not specified, then it will default to `NULL` and the `MOVE_QUEUE_TABLE` interface will automatically take care of redefining the queue table within its current tablespace.
  - If a non-null `to_tablespace` is specified, then you also need to specify the flag `MOVEQT_LOCALREDEF` to be able to carry out local movement. If you omit specifying `MOVEQT_LOCALREDEF` flag when a non-null `to_tablespace` is specified, then the `MOVE_QUEUE_TABLE` interface will treat it as non-local movement.
  - The local movement can be carried out in either offline or online mode, so you need to specify the appropriate flag by doing a bitwise `OR` with the other applicable flags.
- Non-local movement
  - This allows you to move the queue table physically from its current tablespace to another tablespace.
  - The `to_tablespace` parameter must be non-null for this type of movement.
  - If the `to_tablespace` specified matches with the current tablespace of the queue table, then the `MOVE_QUEUE_TABLE` interface optimizes the movement and returns silently without throwing any error.
  - The non-local movement can be carried out in either offline or online mode, so you need to specify the appropriate flag by doing a bitwise `OR` with the other applicable flags.

User Permissions and Privileges

The `MOVE_QUEUE_TABLE` interface requires the following user permissions and privileges.

- A queue table owner schema can move its own queue table, provided it has the required privileges on the target tablespace for non-local movement.
- A user with `MANAGE ANY` AQ privilege or `AQ_ADMINISTRATOR_ROL` can move any queue table, provided it also has the required privileges on owning tablespace of the queue table for local movement and the target tablespace for non-local movement.
- A `SYS` user can move any queue table.

PURGE_QUEUE_TABLE Procedure

This procedure purges messages from queue tables. You can perform various purge operations on both single-consumer and multiconsumer queue tables for persistent and buffered messages.

Syntax

```
DBMS_AQADM.PURGE_QUEUE_TABLE(
   queue_table        IN   VARCHAR2,
   purge_condition    IN   VARCHAR2,
   purge_options      IN   aq$_purge_options_t);
```

where type `aq$_purge_options_t` is described in Oracle Database Advanced Queuing (AQ) Types.

Parameters

Table 27-52 PURGE_QUEUE_TABLE Procedure Parameters

Parameter	Description
`queue_table`	Specifies the name of the queue table to be purged.
`purge_condition`	Specifies the purge condition to use when purging the queue table. The purge condition must be in the format of a SQL `WHERE` clause, and it is case-sensitive. The condition is based on the columns of `aq$queue_table_name` view. When specifying the `purge_condition`, qualify the column names in `aq$queue_table_name` view with `qtview`. To purge all queues in a queue table, set `purge_condition` to either `NULL` (a bare null word, no quotes) or `''` (two single quotes).
`purge_options`	Type `aq$_purge_options_t` contains a `block` parameter and a `delivery_mode` parameter. • If `block` is `TRUE`, then an exclusive lock on all the queues in the queue table is held while purging the queue table. This will cause concurrent enqueuers and dequeuers to block while the queue table is purged. The purge call always succeeds if `block` is `TRUE`. The default for `block` is `FALSE`. This will not block enqueuers and dequeuers, but it can cause the purge to fail with an error during high concurrency times. • `delivery_mode` is used to specify whether `DBMS_AQADM.PERSISTENT`, `DBMS_AQADM.BUFFERED` or `DBMS_AQADM.PERSISTENT_OR_BUFFERED` types of messages are to be purged. You cannot implement arbitrary purge conditions if buffered messages have to be purged.

Usage Notes

- You an purge selected messages from the queue table by specifying a purge_condition. Table 27-52 describes these parameters. Messages can be enqueued to and dequeued from the queue table while the queue table is being purged.

- A trace file is generated in the `udump` destination when you run this procedure. It details what the procedure is doing.
- This procedure commits batches of messages in autonomous transactions. Several such autonomous transactions may get executed as a part of one `purge_queue_table` call depending on the number of messages in the queue table.

QUEUE_SUBSCRIBERS Function

This function returns the subscribers to an 8.0-compatible multiconsumer queue in the PL/SQL index by table collection type `DBMS_AQADM.AQ$_subscriber_list_t`.

Each element of the collection is of type `sys.aq$_agent`. This functionality is provided for 8.1-compatible queues by the `AQ$_queue_table_name_S` view.

Syntax

```
DBMS_AQADM.QUEUE_SUBSCRIBERS (
   queue_name          IN          VARCHAR2);
RETURN aq$_subscriber_list_t IS
```

Parameters

Table 27-53 QUEUE_SUBSCRIBERS Function Parameters

Parameter	Description
`queue_name`	Specifies the queue whose subscribers are to be printed.

REMOVE_SUBSCRIBER Procedure

This procedure removes a default subscriber from a queue. This operation takes effect immediately, and the containing transaction is committed. All references to the subscriber in existing messages are removed as part of the operation.

Syntax

```
DBMS_AQADM.REMOVE_SUBSCRIBER (
   queue_name          IN          VARCHAR2,
   subscriber          IN          sys.aq$_agent);
```

Parameters

Table 27-54 REMOVE_SUBSCRIBER Procedure Parameters

Parameter	Description
`queue_name`	Name of the queue.
`subscriber`	Agent who is being removed. See AQ$_AGENT Type.

REVOKE_QUEUE_PRIVILEGE Procedure

This procedure revokes privileges on a queue from users and roles. The privileges are `ENQUEUE` or `DEQUEUE`.

Syntax

```
DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE (
   privilege         IN      VARCHAR2,
   queue_name        IN      VARCHAR2,
   grantee           IN      VARCHAR2);
```

Parameters

Table 27-55 REVOKE_QUEUE_PRIVILEGE Procedure Parameters

Parameter	Description
`privilege`	The Oracle Database Advanced Queuing queue privilege to revoke. The options are `ENQUEUE`, `DEQUEUE`, and `ALL`. `ALL` means both `ENQUEUE` and `DEQUEUE`.
`queue_name`	Name of the queue.
`grantee`	Grantee(s). The grantee(s) can be a user, a role, or the `PUBLIC` role. If the privilege has been propagated by the grantee through the `GRANT` option, then the propagated privilege is also revoked.

Usage Notes

To revoke a privilege, the revoker must be the original grantor of the privilege. The privileges propagated through the `GRANT` option are revoked if the grantor's privileges are revoked.

REVOKE_SYSTEM_PRIVILEGE Procedure

This procedure revokes Oracle Database Advanced Queuing system privileges from users and roles. The privileges are `ENQUEUE_ANY`, `DEQUEUE_ANY` and `MANAGE_ANY`. The `ADMIN` option for a system privilege cannot be selectively revoked. Starting from Oracle Database 12*c* Release 2, `MANAGE_ANY`, `ENQUEUE_ANY`, and `DEQUEUE_ANY` privileges will not allow access to `SYS` owned queues by users other than `SYS`.

Syntax

```
DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE (
   privilege         IN   VARCHAR2,
   grantee           IN   VARCHAR2);
```

Parameters

Table 27-56 REVOKE_SYSTEM_PRIVILEGE Procedure Parameters

Parameter	Description
privilege	The Oracle Database Advanced Queuing system privilege to revoke. The options are ENQUEUE_ANY, DEQUEUE_ANY, and MANAGE_ANY. The ADMIN option for a system privilege cannot be selectively revoked. **Note:** Starting from Oracle Database 12*c* Release 2, MANAGE_ANY, ENQUEUE_ANY, and DEQUEUE_ANY privileges will not allow access to SYS owned queues by users other than SYS.
grantee	Grantee(s). The grantee(s) can be a user, a role, or the PUBLIC role.

SCHEDULE_PROPAGATION Procedure

This procedure schedules propagation of messages from a queue to a destination identified by a specific database link.

Syntax

```
DBMS_AQADM.SCHEDULE_PROPAGATION (
   queue_name          IN    VARCHAR2,
   destination         IN    VARCHAR2 DEFAULT NULL,
   start_time          IN    DATE     DEFAULT SYSDATE,
   duration            IN    NUMBER   DEFAULT NULL,
   next_time           IN    VARCHAR2 DEFAULT NULL,
   latency             IN    NUMBER   DEFAULT 60,
   destination_queue   IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-57 SCHEDULE_PROPAGATION Procedure Parameters

Parameter	Description
queue_name	Name of the source queue whose messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the administrative user.
destination	Destination database link. Messages in the source queue for recipients at this destination are propagated. If it is NULL, then the destination is the local database and messages are propagated to other queues in the local database. The length of this field is currently limited to 390 bytes, and if the name is not fully qualified, then the default domain name is used. The pattern *schema*.*queue*@dblink is used.

Table 27-57 (Cont.) SCHEDULE_PROPAGATION Procedure Parameters

Parameter	Description
start_time	Initial start time for the propagation window for messages from the source queue to the destination.
duration	Duration of the propagation window in seconds. A NULL value means the propagation window is forever or until the propagation is unscheduled.
next_time	Date function to compute the start of the next propagation window from the end of the current window. If this value is NULL, then propagation is stopped at the end of the current window. For example, to start the window at the same time every day, next_time should be specified as SYSDATE + 1 - duration/86400.
latency	Maximum wait, in seconds, in the propagation window for a message to be propagated after it is enqueued. For example, if the latency is 60 seconds and there are no messages to be propagated during the propagation window, then messages from that queue for the destination are not propagated for at least 60 more seconds. It is at least 60 seconds before the queue is checked again for messages to be propagated for the specified destination. If the latency is 600, then the queue is not checked for 10 minutes, and if the latency is 0, then a job queue process will be waiting for messages to be enqueued for the destination. As soon as a message is enqueued, it is propagated.
destination_queue	Name of the target queue to which messages are to be propagated in the form of a dblink

Usage Notes

Messages may also be propagated to other queues in the same database by specifying a NULL destination. If a message has multiple recipients at the same destination in either the same or different queues, the message is propagated to all of them at the same time.

Oracle extensions for JMS such as JMS propagation and remote subscribers are not currently supported for sharded queues. Propagation between sharded and non-sharded queues is not supported

Related Topics

- *Oracle Database Advanced Queuing User's Guide*

SET_QUEUE_PARAMETER Procedure

This procedure allows user to set different parameters for sharded queues at queue or database level. For database level the `queue_name` should be `NULL`. Note that queue overrides database level parameter values.

> **See Also:**
>
> *Oracle® Database Advanced Queuing User's Guide* for information about sharded queues

Syntax

```
PROCEDURE  SET_QUEUE_PARAMETER(
  queue_name          IN VARCHAR2,
  param_name          IN VARCHAR2,
  param_value         IN NUMBER);
```

Parameters

Table 27-58 SET_QUEUE_PARAMETER Procedure Parameters

Parameter	Description
`queue_name`	The name of the sharded queue.
`param_name`	The name of the parameter. Table 27-59 and Table 27-60 describe the valid parameter names.
`param_value`	The value of the parameter.

Table 27-59 Sharded queue parameters

Parameter Name	Scope	Allowed Values	Description
`SHARD_NUM`	Queue level	`[1, UB4MAXVAL]`	Maximum number of shards allowed for the queue.
`SUBSHARD_SIZE`	Queue level	`[1, UB2MAXVAL]`	Default value is 20,000. This is the size of each partition for allocation of memory in a shard. Each shard consists of multiple subshards, which are allocated, used, and purged automatically based on message enqueue and dequeue rate for the shard.

Table 27-59 (Cont.) Sharded queue parameters

Parameter Name	Scope	Allowed Values	Description
KEY_BASED_ENQUEUE	Queue level	[0,1]	When set, the shard to which a message gets enqueued is determined by the key value specified in the message. Refer to key-based sharding (link) for more details. This parameter cannot be unset once set. When this parameter is not set (default), a session is bound to a shard at the time of first enqueue to the queue. All messages enqueued by the session will go to the same shard to which the session is bound.
STICKY_DEQUEUE	Queue level	[0,1]	When set, dequeue session sticks to a shard in the queue. A session is bound to a shard on first dequeue from the queue. All messages dequeued by the session come from the same shard to which it is bound. This parameter cannot be unset once set. When this parameter is not set, messages dequeued by a session can spread across multiple shards of the queue.

Table 27-60 Key-based Parameters

Parameter Name	Scope	Description
AQ$KEY_TO_SHARD_MAP	Queue level	Shard number to which a given key is mapped. When key-based sharding is enabled, this parameter is used to establish mapping between a key and a shard number or retrieve the shard number to which given key is mapped.

Table 27-60 (Cont.) Key-based Parameters

Parameter Name	Scope	Description
AQ$GET_KEY_SHARD_INST	Queue level	Instance number that owns the shard to which a given key is mapped. Applicable only when key-based sharding is enabled. It is a read-only parameter.

Example 27-3 Create a sharded queue with key based enqueues

To create a sharded queue with key based enqueues (user sharding), the PL/SQL procedure DBMS_AQADM.SET_QUEUE_PARAMETER is called after CREATE_SHARDED_QUEUE.

```
execute
        sys.dbms_aqadm.create_sharded_queue(queue_name => 'MY_SHQ1');


execute
        dbms_aqadm.set_queue_parameter('MY_SHQ1', 'KEY_BASED_ENQUEUE', 1);
```

Example 27-4 Create a sharded queue with sticky dequeues

To create a sharded queue with key based enqueues (user sharding), the PL/SQL procedure DBMS_AQADM.SET_QUEUE_PARAMETER is called after CREATE_SHARDED_QUEUE.

```
execute
        sys.dbms_aqadm.create_sharded_queue(queue_name => 'MY_SHQ1');


execute
        dbms_aqadm.set_queue_parameter('MY_SHQ1', 'STICKY_DEQUEUE', 1);
```

Example 27-5 Setting or changing the number of shards

Once a queue is created, it can have a maximum of 5 shards by default in non-Oracle RAC databases. In Oracle RAC databases, each queue can have a maximum of 5 shards per database instance by default.

```
execute
        dbms_aqadm.set_queue_parameter('MY_SHQ1', 'SHARD_NUM', 200);
```

> **Note:**
>
> Odd numbered shard identifiers are reserved for internal use. In the above example, the 200 shard identifiers used will be 0, 2, 4, …, 398.

SET_MAX_STREAMS_POOL Procedure

This procedure is used for Oracle Database Advanced Queuing to specify and limit maximum streams pool memory use.

Syntax

```
DBMS_AQADM.SET_MAX_STREAMS_POOL (
   value     IN      NUMBER);
```

Parameters

Table 27-61 SET_MAX_STREAMS_POOL Procedure Parameter

Parameter	Description
value	Value in megabytes.

SET_MIN_STREAMS_POOL Procedure

This procedure is used for Oracle Database AQ to specify and limit minimum streams pool memory use.

Syntax

```
DBMS_AQADM.SET_MIN_STREAMS_POOL (
   value     IN      NUMBER);
```

Parameters

Table 27-62 SET_MIN_STREAMS_POOL Procedure Parameter

Parameter	Description
value	Value in megabytes.

SET_WATERMARK Procedure

This procedure is used for Oracle Database Advanced Queuing notification to specify and limit memory use.

Syntax

```
DBMS_AQADM.SET_WATERMARK (
   wmvalue     IN      NUMBER);
```

Parameters

Table 27-63 SET_WATERMARK Procedure Parameter

Parameter	Description
wmvalue	Watermark value in megabytes.

START_QUEUE Procedure

This procedure enables the specified queue for enqueuing or dequeuing.

Syntax

```
DBMS_AQADM.START_QUEUE (
   queue_name      IN     VARCHAR2,
   enqueue         IN     BOOLEAN DEFAULT TRUE,
   dequeue         IN     BOOLEAN DEFAULT TRUE);
```

Parameters

Table 27-64 START_QUEUE Procedure Parameters

Parameter	Description
queue_name	Name of the queue to be enabled
enqueue	Specifies whether ENQUEUE should be enabled on this queue. TRUE means enable ENQUEUE. This is the default. FALSE means do not alter the current setting.
dequeue	Specifies whether DEQUEUE should be enabled on this queue. TRUE means enable DEQUEUE. This is the default. FALSE means do not alter the current setting.

Usage Notes

After creating a queue, the administrator must use START_QUEUE to enable the queue. The default is to enable it for both ENQUEUE and DEQUEUE. Only dequeue operations are allowed on an exception queue. This operation takes effect when the call completes and does not have any transactional characteristics.

STOP_QUEUE Procedure

This procedure disables enqueuing or dequeuing on the specified queue.

Syntax

```
DBMS_AQADM.STOP_QUEUE (
   queue_name      IN   VARCHAR2,
   enqueue         IN   BOOLEAN DEFAULT TRUE,
   dequeue         IN   BOOLEAN DEFAULT TRUE,
   wait            IN   BOOLEAN DEFAULT TRUE,
   free_memory     IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 27-65 STOP_QUEUE Procedure Parameters

Parameter	Description
queue_name	Name of the queue to be disabled
enqueue	Specifies whether ENQUEUE should be disabled on this queue. TRUE means disable ENQUEUE. This is the default. FALSE means do not alter the current setting.

Table 27-65 (Cont.) STOP_QUEUE Procedure Parameters

Parameter	Description
dequeue	Specifies whether DEQUEUE should be disabled on this queue. TRUE means disable DEQUEUE. This is the default. FALSE means do not alter the current setting.
wait	Specifies whether to wait for the completion of outstanding transactions. TRUE means wait if there are any outstanding transactions. In this state no new transactions are allowed to enqueue to or dequeue from this queue. FALSE means return immediately either with a success or an error.
free_memory	Specifies whether the queue should be stopped.

Usage Notes

By default, this call disables both ENQUEUE and DEQUEUE. A queue cannot be stopped if there are outstanding transactions against the queue. This operation takes effect when the call completes and does not have any transactional characteristics.

UNSCHEDULE_PROPAGATION Procedure

This procedure unschedules previously scheduled propagation of messages from a queue to a destination identified by a specific database link.

Syntax

```
DBMS_AQADM.UNSCHEDULE_PROPAGATION (
   queue_name         IN  VARCHAR2,
   destination        IN  VARCHAR2 DEFAULT NULL
   destination_queue  IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 27-66 UNSCHEDULE_PROPAGATION Procedure Parameters

Parameter	Description
queue_name	Name of the source queue whose messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the administrative user.
destination	Destination database link. Messages in the source queue for recipients at this destination are propagated. If it is NULL, then the destination is the local database and messages are propagated to other queues in the local database. The length of this field is currently limited to 128 bytes, and if the name is not fully qualified, then the default domain name is used.
destination_queue	Name of the target queue to which messages are to be propagated in the form of a dblink

UNSET_QUEUE_PARAMETER Procedure

This procedure allows user to unset different parameters for sharded queues at queue or database level.

For database level the `queue_name` should be `NULL`. Note that queue overrides database level parameter values.

> **See Also:**
>
> *Oracle® Database Advanced Queuing User's Guide* for information about sharded queues

Syntax

```
PROCEDURE  UNSET_QUEUE_PARAMETER(
  queue_name          IN VARCHAR2,
  param_name          IN VARCHAR2);
```

Parameters

Table 27-67 UNSET_QUEUE_PARAMETER Procedure Parameters

Parameter	Description
`queue_name`	The name of the sharded queue.
`param_name`	The name of the parameter. Table 27-68 and Table 27-69 describe the valid parameter names.

Table 27-68 Sharded queue parameters

Parameter Name	Scope	Allowed Values	Description
`SHARD_NUM`	Queue level	`[1, UB4MAXVAL]`	Maximum number of shards allowed for the queue.
`SUBSHARD_SIZE`	Queue level	`[1, UB2MAXVAL]`	Default value is 20,000. This is the size of each partition for allocation of memory in a shard. Each shard consists of multiple subshards, which are allocated, used, and purged automatically based on message enqueue and dequeue rate for the shard.

Table 27-68 (Cont.) Sharded queue parameters

Parameter Name	Scope	Allowed Values	Description
KEY_BASED_ENQUEUE	Queue level	[0,1]	When set, the shard to which a message gets enqueued is determined by the key value specified in the message. Refer to key-based sharding (link) for more details. This parameter cannot be unset once set. When this parameter is not set (default), a session is bound to a shard at the time of first enqueue to the queue. All messages enqueued by the session will go to the same shard to which the session is bound.
STICKY_DEQUEUE	Queue level	[0,1]	When set, dequeue session sticks to a shard in the queue. A session is bound to a shard on first dequeue from the queue. All messages dequeued by the session come from the same shard to which it is bound. This parameter cannot be unset once set. When this parameter is not set, messages dequeued by a session can spread across multiple shards of the queue.

Table 27-69 Key-based Parameters

Parameter Name	Scope	Description
AQ$KEY_TO_SHARD_MAP	Queue level	Shard number to which a given key is mapped. When key-based sharding is enabled, this parameter is used to establish mapping between a key and a shard number or retrieve the shard number to which given key is mapped.

Table 27-69 (Cont.) Key-based Parameters

Parameter Name	Scope	Description
AQ$GET_KEY_SHARD_INST	Queue level	Instance number that owns the shard to which a given key is mapped. Applicable only when key-based sharding is enabled. It is a read-only parameter.

VERIFY_QUEUE_TYPES Procedure

This procedure verifies that the source and destination queues have identical types.

The result of the verification is stored in the table `sys.aq$_message_types`, overwriting all previous output of this command.

Syntax

```
DBMS_AQADM.VERIFY_QUEUE_TYPES (
   src_queue_name    IN    VARCHAR2,
   dest_queue_name   IN    VARCHAR2,
   destination       IN    VARCHAR2 DEFAULT NULL,
   rc                OUT   BINARY_INTEGER);
```

Parameters

Table 27-70 VERIFY_QUEUE_TYPES Procedure Parameters

Parameter	Description
src_queue_name	Name of the source queue whose messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the user.
dest_queue_name	Name of the destination queue where messages are to be propagated, including the schema name. If the schema name is not specified, then it defaults to the schema name of the user.
destination	Destination database link. Messages in the source queue for recipients at this destination are propagated. If it is `NULL`, then the destination is the local database and messages are propagated to other queues in the local database. The length of this field is currently limited to 128 bytes, and if the name is not fully qualified, then the default domain name is used.
rc	Return code for the result of the procedure. If there is no error, and if the source and destination queue types match, then the result is 1. If they do not match, then the result is 0. If an Oracle error is encountered, then it is returned in `rc`.

Note:

- `SYS.AQ$_MESSAGE_TYPES` can have multiple entries for the same source queue, destination queue, and database link, but with different transformations.
- `VERIFY_QUEUE_TYPES` check happens once per AQ propagation schedule and not for every propagated message send.
- In case the payload of the queue is modified then the existing propagation schedule between source and destination queue needs to be dropped and recreated.