26
DBMS_AQ

The DBMS_AQ package provides an interface to Oracle Advanced Queuing (AQ).

This chapter contains the following topics:

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

DBMS_AQ Security Model

Initially, only SYS and SYSTEM have execution privilege for the procedures in DBMS_AQ and DBMS_AQADM.

Queue Security

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

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

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

OCI Applications and Queue Access

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

or DEQUEUE ANY QUEUE system privileges. The EXECUTE right of DBMS_AQ is not checked against the session user's rights.

Security Required for Propagation

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:

- DBMS_AQADM
- *Oracle Database Advanced Queuing User's Guide* for more information on security required for propagation

DBMS_AQ Constants

This topic describes the constants used by DBMS_AQ.

The DBMS_AQ package uses the constants shown in the following table.

When using enumerated constants such as BROWSE, LOCKED, or REMOVE, the PL/SQL constants must be specified with the scope of the packages defining it. All types associated with the operational interfaces have to be prepended with DBMS_AQ. For example: DBMS_AQ.BROWSE.

Note:

The sequence_deviation attribute has no effect in releases prior to Oracle Streams AQ 10g Release 1 (10.1) if message_grouping parameter of DBMS_AQADM subprograms is set to TRANSACTIONAL. The sequence deviation feature is deprecated in Oracle Streams AQ 10g Release 2 (10.2).

Table 26-1 Enumerated Constants

Parameter	Options	Type	Description
VISIBILITY	IMMEDIATE	-	-
-	ON_COMMIT	-	-
DEQUEUE_MODE	BROWSE	-	-
-	LOCKED	-	-
-	REMOVE	-	-

Table 26-1 (Cont.) Enumerated Constants

Parameter	Options	Type	Description
-	REMOVE_NODATA	-	-
NAVIGATION	FIRST_MESSAGE	-	-
-	NEXT_MESSAGE	-	-
STATE	WAITING	-	-
-	READY	-	-
-	PROCESSED	-	-
-	EXPIRED	-	-
SEQUENCE_DEVIATION	BEFORE	-	-
-	TOP	-	-
WAIT	FOREVER	BINARY_INTEGER	-
-	NO_WAIT	BINARY_INTEGER	-
DELAY	NO_DELAY	-	-
EXPIRATION	NEVER	-	-
NAMESPACE	NAMESPACE_AQ	-	-
-	NAMESPACE_ANONYMOUS	-	-
NTFN_GROUPING_CLASS	NFTN_GROUPING_CLASS_TIME	NUMBER	-
NTFN_GROUPING_TYPE	NTFN_GROUPING_TYPE_SUMMARY	NUMBER	-
-	NTFN_GROUPING_TYPE_LAST	NUMBER	-
NTFN_GROUPING_REPEAT_COUNT	NTFN_GROUPING_FOREVER	NUMBER	-

DBMS_AQ Data Types

This topic lists and describes DBMS_AQ data types.

Table 26-2 DBMS_AQ Data Structures

Data Structures	Description
OBJECT_NAME	Names database objects
TYPE_NAME	Defines queue types
JSON	Starting from Oracle Database Release 21c, AQ also supports JSON payload type. A dedicated JSON data type permits us to store the JSON data in a post-parse binary format which allows much faster access to nested JSON values.
Oracle Database Advanced Queuing PL/SQL Callback	Specifies the user-defined PL/SQL procedure, defined in the database to be invoked on message notification

OBJECT_NAME

The `object_name` data structure names database objects. It applies to queues, queue tables, agent names, and object types.

Syntax

```
object_name := VARCHAR2;
object_name := [schema_name.]name;
```

Usage Notes

Names for objects are specified by an optional schema name and a name. If the schema name is not specified, the current schema is assumed. The name must follow object name guidelines in *Oracle Database SQL Language Reference* with regard to reserved characters. Schema names, agent names, and object type names can be up to 128 bytes long. Queue names and queue table names can be up to 122 bytes long. Maximum length of agent names and subscriber names can be 128 characters with datablock size greater than 2k. For 2k datablock size, maximum length for subscriber is 128 bytes.

TYPE_NAME

The `type_name` data structure defines queue types.

Syntax

```
type_name := VARCHAR2;
type_name := object_type | "RAW";
```

Attributes

Table 26-3 Type Name Attributes

Attribute	Description
object_type	Maximum number of attributes in the object type is limited to 900.
"RAW"	To store payload of type `RAW`, Oracle Database Advanced Queuing creates a queue table with a `LOB` column as the payload repository. The theoretical maximum size of the message payload is the maximum amount of data that can be stored in a `LOB` column. However, the maximum size of the payload is determined by which programmatic environment you use to access Oracle Database Advanced Queuing. For PL/SQL, Java and precompilers the limit is 32K; for the OCI the limit is 4G. Because the PL/SQL enqueue and dequeue interfaces accept `RAW` buffers as the payload parameters you will be limited to 32K bytes. In OCI, the maximum size of your `RAW` data will be limited to the maximum amount of contiguous memory (as an `OCIRaw` is simply an array of bytes) that the OCI Object Cache can allocate. Typically, this will be at least 32K bytes and much larger in many cases. Because `LOB` columns are used for storing `RAW` payload, the Oracle Database Advanced Queuing administrator can choose the `LOB` tablespace and configure the `LOB` storage by constructing a `LOB` storage string in the `storage_clause` parameter during queue table creation time.

Oracle Database Advanced Queuing PL/SQL Callback

The `plsqlcallback` data structure specifies the user-defined PL/SQL procedure, defined in the database to be invoked on message notification.

Syntax

If a notification message is expected for a `RAW` payload enqueue, then the PL/SQL callback must have the following signature:

```
procedure plsqlcallback(
  context  IN  RAW,
  reginfo  IN  SYS.AQ$_REG_INFO,
  descr    IN  SYS.AQ$_DESCRIPTOR,
  payload  IN  RAW,
  payloadl IN  NUMBER);
```

Attributes

Table 26-4 Oracle Database Advanced Queuing PL/SQL Callback Attributes

Attribute	Description
`context`	Specifies the context for the callback function that was passed by `dbms_aq.register`. See AQ$_REG_INFO Type.
`reginfo`	See AQ$_REG_INFO Type.
`descr`	See AQ$_DESCRIPTOR Type
`payload`	If a notification message is expected for a raw payload enqueue then this contains the raw payload that was enqueued into a non persistent queue. In case of a persistent queue with raw payload this parameter will be null.
`payloadl`	Specifies the length of `payload`. If `payload` is null, `payload1` = 0.

If the notification message is expected for an ADT payload enqueue, the PL/SQL callback must have the following signature:

```
procedure plsqlcallback(
  context  IN RAW,
  reginfo  IN SYS.AQ$_REG_INFO,
  descr    IN SYS.AQ$_DESCRIPTOR,
  payload  IN VARCHAR2,
  payloadl IN NUMBER);
```

DBMS_AQ Operational Notes

This topic lists various DBMS_AQ operational notes.

DBMS_AQ and DBMS_AQADM Java Classes

Java interfaces are available for `DBMS_AQ` and `DBMS_AQADM`. The Java interfaces are provided in the `$ORACLE_HOME/rdbms/jlib/aqapi.jar`. Users are required to have `EXECUTE` privileges on the `DBMS_AQIN` package to use these interfaces.

Summary of DBMS_AQ Subprograms

The `DBMS_AQ` package uses subprograms described in this table.

Table 26-5 DBMS_AQ Package Subprograms

Subprograms	Description
BIND_AGENT Procedure	Creates an entry for an Oracle Database Advanced Queuing agent in the LDAP directory
DEQUEUE Procedure	Dequeues a message from the specified queue
DEQUEUE_ARRAY Function	Dequeues an array of messages from the specified queue
ENQUEUE Procedure	Adds a message to the specified queue
ENQUEUE_ARRAY Function	Adds an array of messages to the specified queue
LISTEN Procedures	Listen to one or more queues on behalf of a list of agents
POST Procedure	Posts to a anonymous subscription which allows all clients who are registered for the subscription to get notifications
REGISTER Procedure	Registers for message notifications
SEEK Procedure	Enables a subscriber to seek any particular message of choice
UNBIND_AGENT Procedure	Removes an entry for an Oracle Database Advanced Queuing agent from the LDAP directory
UNREGISTER Procedure	Unregisters a subscription which turns off notification

Note:

`DBMS_AQ` does not have a purity level defined; therefore, you cannot call any procedure in this package from other procedures that have `RNDS`, `WNDS`, `RNPS` or `WNPS` constraints defined.

BIND_AGENT Procedure

This procedure creates an entry for an Oracle Database Advanced Queuing agent in the LDAP server.

Syntax

```
DBMS_AQ.BIND_AGENT(
   agent        IN SYS.AQ$_AGENT,
   certificate  IN VARCHAR2 default NULL);
```

Parameters

Table 26-6 BIND_AGENT Procedure Parameters

Parameter	Description
`agent`	Agent that is to be registered in LDAP server.
`certificate`	Location (LDAP distinguished name) of the "organizationalperson" entry in LDAP whose digital certificate (attribute `usercertificate`) is to be used for this agent. Example: "cn=OE, cn=ACME, cn=com" is a distinguished name for a OrganizationalPerson OE whose certificate will be used with the specified agent.

Usage Notes

In the LDAP server, digital certificates are stored as an attribute (`usercertificate`) of the `OrganizationalPerson` entity. The distinguished name for this `OrganizationalPerson` must be specified when binding the agent.

DEQUEUE Procedure

This procedure dequeues a message from the specified queue.

Syntax

```
DBMS_AQ.DEQUEUE (
   queue_name           IN      VARCHAR2,
   dequeue_options      IN      dequeue_options_t,
   message_properties   OUT     message_properties_t,
   payload              OUT     "<ADT_1>"
   msgid                OUT     RAW);
```

Parameters

Table 26-7 DEQUEUE Procedure Parameters

Parameter	Description
`queue_name`	Specifies the name of the queue.
`dequeue_options`	See DEQUEUE_OPTIONS_T Type.
`message_properties`	See MESSAGE_PROPERTIES_T Type.
`payload`	Not interpreted by Oracle Database Advanced Queuing. The payload must be specified according to the specification in the associated queue table. For the definition of *`type_name`* refer to `TYPE_NAME` in DBMS_AQ Data Types.
`msgid`	System generated identification of the message.

Usage Notes

The search criteria for messages to be dequeued is determined by the following parameters in `dequeue_options`:

- `consumer_name`
- `msgid`

  `Msgid` uniquely identifies the message to be dequeued. Only messages in the `READY` state are dequeued unless `msgid` is specified.

- `correlation`

  Correlation identifiers are application-defined identifiers that are not interpreted by Oracle Database Advanced Queuing.

- `deq_condition`

  Dequeue condition is an expression based on the message properties, the message data properties and PL/SQL functions. A `deq_condition` 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). Message properties include `priority`, `corrid` and other columns in the queue table.

To specify dequeue conditions 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.

Example: `tab.user_data.orderstatus='EXPRESS'`

The dequeue order is determined by the values specified at the time the queue table is created unless overridden by the `msgid` and correlation ID in `dequeue_options`.

The database-consistent read mechanism is applicable for queue operations. For example, a `BROWSE` call may not see a message that is enqueued after the beginning of the browsing transaction.

The default `NAVIGATION` parameter during dequeue is `NEXT_MESSAGE`. This means that subsequent dequeues will retrieve the messages from the queue based on the snapshot obtained in the first dequeue. In particular, a message that is enqueued after the first dequeue command will be processed only after processing all the remaining messages in the queue. This is usually sufficient when all the messages have already been enqueued into the queue, or when the queue does not have a priority-based ordering. However, applications must use the `FIRST_MESSAGE` navigation option when the first message in the queue needs to be processed by every dequeue command. This usually becomes necessary when a higher priority message arrives in the queue while messages already-enqueued are being processed.

> **Note:**
>
> It may be more efficient to use the `FIRST_MESSAGE` navigation option when messages are concurrently enqueued. If the `FIRST_MESSAGE` option is not specified, Oracle Database Advanced Queuing continually generates the snapshot as of the first dequeue command, leading to poor performance. If the `FIRST_MESSAGE` option is specified, then Oracle Database Advanced Queuing uses a new snapshot for every dequeue command.

Messages enqueued in the same transaction into a queue that has been enabled for message grouping will form a group. If only one message is enqueued in the transaction, then this will effectively form a group of one message. There is no upper limit to the number of messages that can be grouped in a single transaction.

In queues that have not been enabled for message grouping, a dequeue in `LOCKED` or `REMOVE` mode locks only a single message. By contrast, a dequeue operation that seeks to dequeue a message that is part of a group will lock the entire group. This is useful when all the messages in a group need to be processed as an atomic unit.

When all the messages in a group have been dequeued, the dequeue returns an error indicating that all messages in the group have been processed. The application can then use the `NEXT_TRANSACTION` to start dequeuing messages from the next available group. In the event that no groups are available, the dequeue will time out after the specified `WAIT` period.

Using Secure Queues

For secure queues, you must specify `consumer_name` in the `dequeue_options` parameter. See DEQUEUE_OPTIONS_T Type for more information about `consumer_name`.

When you use secure queues, the following are required:

- You must have created a valid Oracle Database Advanced Queuing agent using `DBMS_AQADM.CREATE_AQ_AGENT`. See CREATE_AQ_AGENT Procedure .
- You must map the Oracle Database Advanced Queuing agent to a database user with dequeue privileges on the secure queue. Use `DBMS_AQADM.ENABLE_DB_ACCESS` to do this. See ENABLE_DB_ACCESS Procedure.

DEQUEUE_ARRAY Function

This function dequeues an array of messages and returns them in the form of an array of payloads, an array of message properties and an array of message IDs. This function returns the number of messages successfully dequeued.

Syntax

```
DBMS_AQ.DEQUEUE_ARRAY (
   queue_name                IN   VARCHAR2,
   dequeue_options           IN   dequeue_options_t,
   array_size                IN   pls_integer,
   message_properties_array  OUT  message_properties_array_t,
   payload_array             OUT  "<COLLECTION_1>",
   msgid_array               OUT  msgid_array_t,
   error_array               OUT  error_array_t)
RETURN pls_integer;
```

Parameters

Table 26-8 DEQUEUE_ARRAY Function Parameters

Parameter	Description
`queue_name`	The queue name from which messages are dequeued (same as single-row dequeue).
`dequeue_options`	The set of options which will be applied to all messages in the array (same as single-row dequeue).
`array_size`	The number of elements to dequeue. For buffered messages, `array_size` should be `1`.
`message_properties_array`	A record containing an array corresponding to each message property. Each payload element has a corresponding set of message properties. See MESSAGE_PROPERTIES_ARRAY_T Type.
`payload_array`	An array of dequeued payload data. "`<COLLECTION_1>`" can be an associative array, varray or nested table in its PL/SQL representation. Users can dequeue `JSON`, `RAW` and `ADT` payloads.
`msgid_array`	An array of message IDs of the dequeued messages. See MSGID_ARRAY_T Type.
`error_array`	Currently not implemented

Usage Notes

A nonzero wait time, as specified in `dequeue_options`, is recognized only when there are no messages in the queue. If the queue contains messages that are eligible for dequeue, then the `DEQUEUE_ARRAY` function will dequeue up to `array_size` messages and return immediately.

Dequeue by `message_id` is not supported. See DEQUEUE Procedure for more information on the `navigation` parameter. Existing `NAVIGATION` modes are supported. In addition, two new `NAVIGATION` modes are supported for queues enabled for message grouping:

- FIRST_MESSAGE_MULTI_GROUP
- NEXT_MESSAGE_MULTI_GROUP

> **See Also:**
>
> ENQUEUE_OPTIONS_T Type

For transaction grouped queues and ONE_GROUP navigation, messages are dequeued from a single transaction group only, subject to the array_size limit. In MULTI_GROUP navigation, messages are dequeued across multiple transaction groups, still subject to the array_size limit. ORA-25235 is returned to indicate the end of a transaction group.

DEQUEUE_ARRAY is not supported for buffered messages, but you can still use this procedure on individual buffered messages by setting array_size to one message.

ENQUEUE Procedure

This procedure adds a message to the specified queue.

Syntax

```
DBMS_AQ.ENQUEUE (
   queue_name          IN      VARCHAR2,
   enqueue_options     IN      enqueue_options_t,
   message_properties  IN      message_properties_t,
   payload             IN       "<ADT_1>",
   msgid               OUT     RAW);
```

Parameters

Table 26-9 ENQUEUE Procedure Parameters

Parameter	Description
queue_name	Specifies the name of the queue to which this message should be enqueued. The queue cannot be an exception queue.
enqueue_options	See ENQUEUE_OPTIONS_T Type.
message_properties	See MESSAGE_PROPERTIES_T Type.
payload	Not interpreted by Oracle Database Advanced Queuing. The payload must be specified according to the specification in the associated queue table. NULL is an acceptable parameter. For the definition of *type_name* refer to TYPE_NAME in DBMS_AQ Data Types.
msgid	System generated identification of the message. This is a globally unique identifier that can be used to identify the message at dequeue time.

Usage Notes

The sequence_deviation parameter in enqueue_options can be used to change the order of processing between two messages. The identity of the other message, if any, is specified by the enqueue_options parameter relative_msgid. The relationship is identified by the sequence_deviation parameter.

Specifying `sequence_deviation` for a message introduces some restrictions for the delay and priority values that can be specified for this message. The delay of this message must be less than or equal to the delay of the message before which this message is to be enqueued. The priority of this message must be greater than or equal to the priority of the message before which this message is to be enqueued.

> **Note:**
>
> The `sequence_deviation` attribute has no effect in releases prior to Oracle Streams AQ 10g Release 1 (10.1) if `message_grouping` parameter of `DBMS_AQADM` subprograms is set to `TRANSACTIONAL`. The sequence deviation feature is deprecated in Oracle Streams AQ 10g Release 2 (10.2).

If a message is enqueued to a multiconsumer queue with no recipient, and if the queue has no subscribers (or rule-based subscribers that match this message), then Oracle error `ORA_24033` is raised. This is a warning that the message will be discarded because there are no recipients or subscribers to whom it can be delivered.

Using Secure Queues

For secure queues, you must specify the `sender_id` in the `messages_properties` parameter. See MESSAGE_PROPERTIES_T Type for more information about `sender_id`.

When you use secure queues, the following are required:

- You must have created a valid Oracle Database Advanced Queuing agent using `DBMS_AQADM.CREATE_AQ_AGENT`. See CREATE_AQ_AGENT Procedure .
- You must map `sender_id` to a database user with enqueue privileges on the secure queue. Use `DBMS_AQADM.ENABLE_DB_ACCESS` to do this. See ENABLE_DB_ACCESS Procedure.

ENQUEUE_ARRAY Function

This function enqueues an array of payloads using a corresponding array of message properties. The output will be an array of message IDs of the enqueued messages.

Syntax

```
DBMS_AQ.ENQUEUE_ARRAY (
   queue_name                IN   VARCHAR2,
   enqueue_options           IN   enqueue_options_t,
   array_size                IN   pls_integer,
   message_properties_array  IN   message_properties_array_t,
   payload_array             IN   "<COLLECTION_1>",
   msgid_array               OUT  msgid_array_t,
   error_array               OUT  error_array_t)
RETURN pls_integer;
```

Parameters

Table 26-10 ENQUEUE_ARRAY Function Parameters

Parameter	Description
queue_name	The queue name in which messages are enqueued (same as single-row enqueue).
enqueue_options	See ENQUEUE_OPTIONS_T Type.
array_size	The number of elements to enqueue. For buffered messages, array_size should be 1.
message_properties_array	A record containing an array corresponding to each message property. For each property, the user must allocate array_size elements. See MESSAGE_PROPERTIES_ARRAY_T Type.
payload_array	An array of payload data. "<COLLECTION_1>" can be an associative array, VARRAY, or nested table in its PL/SQL representation. Users can enqueue JSON, RAW, and ADT payloads.
msgid_array	An array of message IDs for the enqueued messages. If an error occurs for a particular message, then its corresponding message ID is null. See MSGID_ARRAY_T Type.
error_array	Currently not implemented

Usage Notes

ENQUEUE_ARRAY is not supported for buffered messages, but you can still use this procedure on individual buffered messages by setting array_size to one message.

LISTEN Procedures

This procedure listens on one or more queues on behalf of a list of agents. The address field of the agent indicates the queue the agent wants to monitor. Only local queues are supported as addresses. Protocol is reserved for future use.

Syntax

```
DBMS_AQ.LISTEN (
   agent_list            IN    AQ$_AGENT_LIST_T,
   wait                  IN    BINARY_INTEGER DEFAULT DBMS_AQ.FOREVER,
   agent                 OUT   SYS.AQ$_AGENT);

DBMS_AQ.LISTEN (
   agent_list             IN   AQ$_AGENT_LIST_T,
   wait                   IN   BINARY_INTEGER DEFAULT FOREVER,
   listen_delivery_mode   IN   PLS_INTEGER DEFAULT DBMS_AQ.PERSISTENT,
   agent                  OUT  SYS.AQ$_AGENT,
   message_delivery_mode  OUT  PLS_INTEGER);

TYPE aq$_agent_list_t IS TABLE of aq$_agent INDEXED BY BINARY_INTEGER;
TYPE aq$_agent_list_t IS TABLE of aq$_agent INDEXED BY BINARY_INTEGER;
```

Parameters

Table 26-11 LISTEN Procedure Parameters

Parameter	Description
agent_list	List of agents to listen for
wait	Time out for the listen call in seconds. By default, the call will block forever.
listen_delivery_mode	The caller specifies whether it is interested in persistent, buffered messages or both types of messages, specifying a delivery mode of DBMS_AQ.PERSISTENT or DBMS_AQ.BUFFERED or DBMS_AQ.PERSISTENT_OR_BUFFERED
agent	Agent with a message available for consumption
message_delivery_mode	Returns the message type along with the queue and consumer for which there is a message

Usage Notes

If agent-address is a multiconsumer queue, then agent-name is mandatory. For single-consumer queues, agent-name must not be specified.

This procedure takes a list of agents as an argument. You specify the queue to be monitored in the address field of each agent listed. You also must specify the name of the agent when monitoring multiconsumer queues. For single-consumer queues, an agent name must not be specified. Only local queues are supported as addresses. Protocol is reserved for future use.

This is a blocking call that returns when there is a message ready for consumption for an agent in the list. If there are messages for more than one agent, only the first agent listed is returned. If there are no messages found when the wait time expires, an error is raised.

A successful return from the LISTEN call is only an indication that there is a message for one of the listed agents in one the specified queues. The interested agent must still dequeue the relevant message.

> **Note:**
>
> You cannot call LISTEN on nonpersistent queues.

POST Procedure

This procedure posts to a list of anonymous subscriptions that allows all clients who are registered for the subscriptions to get notifications.

Syntax

```
DBMS_AQ.POST (
  post_list       IN  SYS.AQ$_POST_INFO_LIST,
  post_count      IN  NUMBER);
```

Parameters

Table 26-12 POST Procedure Parameters

Parameter	Description
post_list	Specifies the list of anonymous subscriptions to which you want to post. It is a list of AQ$_POST_INFO_LIST Type.
post_count	Specifies the number of entries in the post_list.

Usage Notes

This procedure is used to post to anonymous subscriptions which allows all clients who are registered for the subscriptions to get notifications. Several subscriptions can be posted to at one time.

REGISTER Procedure

This procedure registers an e-mail address, user-defined PL/SQL procedure, or HTTP URL for message notification.

Syntax

```
DBMS_AQ.REGISTER (
   reg_list     IN  SYS.AQ$_REG_INFO_LIST,
   count        IN  NUMBER);
```

Parameters

Table 26-13 REGISTER Procedure Parameters

Parameter	Description
reg_list	Specifies the list of subscriptions to which you want to register for message notifications. It is a list of AQ$_REG_INFO Type.
count	Specifies the number of entries in the reg_list.

Usage Notes

- This procedure is used to register for notifications. You can specify an e-mail address to which message notifications are sent, register a procedure to be invoked on a notification, or register an HTTP URL to which the notification is posted. Interest in several subscriptions can be registered at one time.
- The procedure can also be used to register for grouping notifications using five grouping attributes:
  - Class – grouping criterion (currently only TIME criterion is supported)
  - Value – the value of the grouping criterion (currently only time in seconds for criterion TIME)
  - Type – summary or last, also contains count of notifications received in group (for AQ namespace only, not for DBCHANGE namespace)
  - Repeat count – how many times to perform grouping (Default is FOREVER)
  - Start time – when to start grouping (Default is current time)

- If you register for e-mail notifications, you should set the host name and port name for the SMTP server that will be used by the database to send e-mail notifications. If required, you should set the send-from e-mail address, which is set by the database as the `sent from` field. You need a Java-enabled database to use this feature.
- If you register for HTTP notifications, you may want to set the host name and port number for the proxy server and a list of no-proxy domains that will be used by the database to post HTTP notifications.

> **See Also:**
>
> DBMS_AQELM for more information on e-mail and HTTP notifications

SEEK Procedure

This procedure enables a subscriber to seek any particular message of choice. Once a seek operation is successful, the dequeues move from the new seek position onwards. A new call is added to facilitate the seek operation.

Syntax

```
DBMS_AQ.SEEK (
              queue_name         IN     VARCHAR2 not null,
              consumer_name      IN     VARCHAR2 default null,
              seek_input_array   IN     SEEK_INPUT_ARRAY_T,
              skip_option        IN     pls_integer DEFAULT DBMS_AQ.ERROR_IF_SKIPPED,
              seek_output_array OUT     SEEK_OUTPUT_ARRAY_T);
```

Parameters

Table 26-14 SEEK Procedure Parameters

Parameter	Description
`queue_name`	Specifies the name of the queue.
`consumer_name`	Specifies the name of the subscriber performing seek. `consumer_name` can be `NULL` for single-consumer queues.
`seek_input_array`	Array of type `seek_input_t`, where individual element in the array has per shard seek input. See SEEK_INPUT_T Type.
`skip_option`	Specifies the skip options. The following are the possible values: • `DBMS_AQ.ERROR_IF_SKIPPED` (default): Raises an error if seek operation can result in skipping some undequeued messsages when a seek forward is performed. • `DBMS_AQ.NO_DISCARD_SKIPPED`: Does not discard any skipped messages. Allows the seek operation to go through irrespective of the direction of the seek. Such undequeued messages before seek point will be shown as `READY_SKIPPED` in `AQ$<QUEUE_TABLE_NAME>`. • `DBMS_AQ.DISCARD_SKIPPED`: Discards any skipped messages. As soon as all the messages in a subshard are either dequeued or discarded by all its subscribers, the subshard is "retained" which will be truncated after retention time is over. Before retention time is over, the subscribers who have discarded the messages can seek back and dequeue the discarded messages if they wish to do so. The discarded messages will be shown as `READY_DISCARDED` in `AQ$<QUEUE_TABLE_NAME>`.

Table 26-14 (Cont.) SEEK Procedure Parameters

Parameter	Description
seek_output_array	Array of type seek_output_t, where individual element in the array has seeked from and seeked to fields per shard-priority. See SEEK_OUTPUT_T Type.

UNBIND_AGENT Procedure

This procedure removes the entry for an Oracle Database Advanced Queuing agent from the LDAP server.

Syntax

```
DBMS_AQ.UNBIND_AGENT(
   agent     IN SYS.AQ$_AGENT);
```

Parameters

Table 26-15 BIND_AGENT Procedure Parameters

Parameter	Description
agent	Agent that is to be removed from the LDAP server

UNREGISTER Procedure

This procedure unregisters a subscription which turns off notifications.

Syntax

```
DBMS_AQ.UNREGISTER (
   reg_list      IN  SYS.AQ$_REG_INFO_LIST,
   reg_count     IN  NUMBER);
```

Parameters

Table 26-16 UNREGISTER Procedure Parameters

Parameter	Description
reg_list	Specifies the list of subscriptions to which you want to register for message notifications. It is a list of AQ$_REG_INFO Type.
reg_count	Specifies the number of entries in the reg_list.

Usage Notes

This procedure is used to unregister a subscription which turns off notifications. Several subscriptions can be unregistered from at one time.