2 Basic Components of Oracle Transactional Event Queues and Advanced Queuing

This topic lists the basic components of Oracle Database Advanced Queuing.

- Object Name
- Type Name
- AQ Agent Type
- AQ Recipient List Type
- AQ Agent List Type
- AQ Subscriber List Type
- AQ Registration Information List Type
- AQ Post Information List Type
- AQ Registration Information Type
- AQ Notification Descriptor Type
- AQ Message Properties Type
- AQ Post Information Type
- AQ$_NTFN_MSGID_ARRAY Type
- Enumerated Constants for AQ Administrative Interface
- Enumerated Constants for AQ Operational Interface
- AQ Background Processes

See Also:

- Oracle Database Advanced Queuing Administrative Interface
- Oracle Database Advanced Queuing Operations Using PL/SQL

Object Name

This component names database objects.

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

This naming convention applies to queues, queue tables, and object types.

Names for objects are specified by an optional schema name and a name. If the schema name is not specified, then the current schema is assumed. The name must follow the reserved character guidelines in *Oracle Database SQL Language ReferenceOracle Database SQL*

Language Reference. The schema name, agent name, and the object type name can each be up to 128 bytes long. However starting from 12c Release 2 (12.2.), queue names and queue table names can be a maximum of 122 bytes.

Type Name

This component defines queue types.

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

The maximum number of attributes in the object type is limited to 900.

To store payloads of type `RAW`, Oracle Database Advanced Queuing creates a queue table with a LOB column as the payload repository. The size of the payload is limited to 32K bytes of data. 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.

Note:

Payloads containing LOBs require users to grant explicit `Select`, `Insert` and `Update` privileges on the queue table for doing enqueues and dequeues.

AQ Agent Type

This component identifies a producer or consumer of a message.

```
TYPE AQ$_AGENT IS OBJECT (
   name             VARCHAR2(30),
   address          VARCHAR2(1024),
   protocol         NUMBER);
```

All consumers that are added as subscribers to a multiconsumer queue must have unique values for the `AQ$_AGENT` parameters. Two subscribers cannot have the same values for the `NAME`, `ADDRESS`, and `PROTOCOL` attributes for the `AQ$_AGENT` type. At least one of the three attributes must be different for two subscribers.

You can add subscribers by repeatedly using the `DBMS_AQADM.ADD_SUBSCRIBER` procedure up to a maximum of 1024 subscribers for a multiconsumer queue.

This type has three attributes:

- `name`

  This attribute specifies the name of a producer or a consumer of a message. It can be the name of an application or a name assigned by an application. A queue can itself be an agent, enqueuing or dequeuing from another queue. The name must follow the reserved character guidelines in *Oracle Database SQL Language Reference*.

- `address`

  This attribute is interpreted in the context of `protocol`. If `protocol` is 0 (default), then `address` is of the form `[schema.]queue[@dblink]`.

- `protocol`

  This attribute specifies the protocol to interpret the address and propagate the message. The default value is 0.

AQ Recipient List Type

This component identifies the list of agents that receive a message.

```
TYPE AQ$_RECIPIENT_LIST_T IS TABLE OF aq$_agent
           INDEX BY BINARY_INTEGER;
```

AQ Agent List Type

This component identifies the list of agents for `DBMS_AQ.LISTEN` to listen for.

```
TYPE AQ$_AGENT_LIST_T IS TABLE OF aq$_agent
           INDEX BY BINARY INTEGER;
```

AQ Subscriber List Type

This component identifies the list of subscribers that subscribe to this queue.

```
TYPE AQ$_SUBSCRIBER_LIST_T IS TABLE OF aq$_agent
          INDEX BY BINARY INTEGER;
```

AQ Registration Information List Type

This component identifies the list of registrations to a queue.

```
TYPE AQ$_REG_INFO_LIST AS VARRAY(1024) OF sys.aq$_reg_info;
```

AQ Post Information List Type

This component identifies the list of anonymous subscriptions to which messages are posted.

```
TYPE AQ$_POST_INFO_LIST AS VARRAY(1024) OF sys.aq$_post_info;
```

AQ Registration Information Type

This component identifies a producer or a consumer of a message.

```
TYPE SYS.AQ$_REG_INFO IS OBJECT (
   name                              VARCHAR2(128),
   namespace                         NUMBER,
   callback                          VARCHAR2(4000),
   context                           RAW(2000)  DEFAULT NULL,
   qosflags                          NUMBER,
   timeout                           NUMBER
   ntfn_grouping_class               NUMBER,
   ntfn_grouping_value               NUMBER    DEFAULT 600,
   ntfn_grouping_type                NUMBER,
   ntfn_grouping_start_time          TIMESTAMP WITH TIME ZONE,
   ntfn_grouping_repeat_count   NUMBER);
```

Its attributes are described in the following list.

Table 2-1 AQ$_REG_INFO Type Attributes

Attribute	Description
`name`	Specifies the name of the subscription. The subscription name is of the form `*schema.queue*` if the registration is for a single consumer queue or `*schema.queue:consumer_name*` if the registration is for a multiconsumer queues.
`namespace`	Specifies the namespace of the subscription. To receive notification from Oracle Database AQ queues, the namespace must be `DBMS_AQ.NAMESPACE_AQ.` To receive notifications from other applications through `DBMS_AQ.POST` or `OCISubscriptionPost(),` the namespace must be `DBMS_AQ.NAMESPACE_ANONYMOUS.`
`callback`	Specifies the action to be performed on message notification. For HTTP notifications, use `http://www.company.com:8080`. For e-mail notifications, use `mailto://xyz@company.com`. For raw message payload for the `PLSQLCALLBACK` procedure, use `plsql://*schema.procedure*?PR=0`. For user-defined type message payload converted to XML for the `PLSQLCALLBACK` procedure, use `plsql://*schema.procedure*?PR=1`
`context`	Specifies the context that is to be passed to the callback function
`qosflags`	Can be set to one or more of the following values to specify the notification quality of service: • `NTFN_QOS_RELIABLE`- This value specifies that reliable notification is required. Reliable notifications persist across instance and database restarts. • `NTFN_QOS_PAYLOAD` - This value specifies that payload delivery is required. It is supported only for client notification and only for `RAW` queues. • `NTFN_QOS_PURGE_ON_NTFN` - This value specifies that the registration is to be purged automatically when the first notification is delivered to this registration location.
`ntfn_grouping_class`	Currently, only the following flag can be set to specify criterion for grouping. The default value will be 0. If `ntfn_grouping_class` is 0, all other notification grouping attributes must be 0. • `NTFN_GROUPING_CLASS_TIME` - Notifications grouped by time, that is, the user specifies a time value and a single notification gets published at the end of that time.
`ntfn_grouping_value`	Time-period of grouping notifications specified in seconds, meaning the time after which grouping notification would be sent periodically until `ntfn_grouping_repeat_count` is exhausted.
`ntfn_grouping_type`	• `NTFN_GROUPING_TYPE_SUMMARY` - Summary of all notifications that occurred in the time interval. (Default) • `NTFN_GROUPING_TYPE_LAST` - Last notification that occurred in the interval.
`ntfn_grouping_start_time`	Notification grouping start time. Notification grouping can start from a user-specified time that should a valid timestamp with time zone. If `ntfn_grouping_start_time` is not specified when using grouping, the default is to current timestamp with time zone

Table 2-1 (Cont.) AQ$_REG_INFO Type Attributes

Attribute	Description
`ntfn_grouping_repeat_count`	Grouping notifications will be sent as many times as specified by the notification grouping repeat count and after that revert to regular notifications. The ntfn_grouping_repeat_count, if not specified, will default to • `NTFN_GROUPING_FOREVER` - Keep sending grouping notifications forever.

AQ Notification Descriptor Type

This component specifies the Oracle Database Advanced Queuing descriptor received by AQ PL/SQL callbacks upon notification.

```
TYPE SYS.AQ$_DESCRIPTOR IS OBJECT (
   queue_name       VARCHAR2(61),
   consumer_name    VARCHAR2(30),
   msg_id           RAW(16),
   msg_prop         MSG_PROP_T,
   gen_desc         AQ$_NTFN_DESCRIPTOR,
   msgid_array      SYS.AQ$_NTFN_MSGID_ARRAY,
   ntfnsRecdInGrp   NUMBER);
```

It has the following attributes:

Table 2-2 AQ$_DESCRIPTOR Attributes

Attribute	Description
`queue_name`	Name of the queue in which the message was enqueued which resulted in the notification
`consumer_name`	Name of the consumer for the multiconsumer queue
`msg_id`	Identification number of the message
`msg_prop`	Message properties specified by the `MSG_PROP_T` type
`gen_desc`	Indicates the timeout specifications
`msgid_array`	Group notification message ID list
`ntfnsRecdInGrp`	Notifications received in group

AQ Message Properties Type

The message properties type `msg_prop_t` has these components.

```
TYPE AQ$_MSG_PROP_T IS OBJECT(
   priority          number,
   delay             number,
   expiration        number,
   correlation       varchar2(128),
   attempts          number,
   recipent_list     aq$_recipient_list_t,
   exception_queue   varchar2(51),
   enqueue_time      date,
   state             number,
   sender_id        aq$_agent,
   original_msgid   raw(16),
   delivery_mode    number);
```

The timeout specifications type `AQ$_NTFN_DESCRIPTOR` has a single component:

```
TYPE AQ$_NTFN_DESCRIPTOR IS OBJECT(
   NTFN_FLAGS    number);
```

`NTFN_FLAGS` is set to `1` if the notifications are already removed after a stipulated timeout; otherwise the value is `0`.

See Also:

"MESSAGE_PROPERTIES_T Type" in *Oracle Database PL/SQL Packages and Types Reference*

AQ Post Information Type

This component specifies anonymous subscriptions to which you want to post messages.

```
TYPE SYS.AQ$_POST_INFO IS OBJECT (
  name       VARCHAR2(128),
  namespace  NUMBER,
  payload    RAW(2000));
```

It has three attributes:

- `name`

  This attribute specifies the name of the anonymous subscription to which you want to post.

- `namespace`

  This attribute specifies the namespace of the anonymous subscription. To receive notifications from other applications using `DBMS_AQ.POST` or `OCISubscriptionPost()`, the namespace must be `DBMS_AQ.NAMESPACE_ANONYMOUS`.

- `payload`

  This attribute specifies the payload to be posted to the anonymous subscription. The default is `NULL`.

AQ$_NTFN_MSGID_ARRAY Type

This component is for storing grouping notification data for AQ namespace, value 2^{30} which is the max varray size.

```
TYPE SYS.AQ$_NTFN_MSGID_ARRAY AS VARRAY(1073741824)OF RAW(16);
```

Enumerated Constants for AQ Administrative Interface

When enumerated constants such as `INFINITE`, `TRANSACTIONAL`, and `NORMAL_QUEUE` are selected as values, 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 2-3 lists the enumerated constants in the Oracle Database Advanced Queuing administrative interface.

Table 2-3 Enumerated Constants in the Oracle Database Advanced Queuing Administrative Interface

Parameter	Options
`retention`	`0,1,2...INFINITE`
`message_grouping`	`TRANSACTIONAL`, `NONE`
`queue_type`	`NORMAL_QUEUE`, `EXCEPTION_QUEUE`,`NON_PERSISTENT_QUEUE`
`delivery_mode`	`BUFFERED`, `PERSISTENT`, `PERSISTENT_OR_BUFFERED`

Note:

Nonpersistent queues are deprecated in Oracle Database Advanced Queuing 10*g* Release 2 (10.2). Oracle recommends that you use buffered messaging instead.

Enumerated Constants for AQ Operational Interface

When using enumerated constants such as `BROWSE`, `LOCKED`, and `REMOVE`, the PL/SQL constants must be specified with the scope of the packages defining them.

All types associated with the operational interfaces must be prepended with `DBMS_AQ`. For example:

```
DBMS_AQ.BROWSE
```

Table 2-4 lists the enumerated constants in the Oracle Database Advanced Queuing operational interface.

Table 2-4 Enumerated Constants in the Oracle Database Advanced Queuing Operational Interface

Parameter	Options
`visibility`	`IMMEDIATE`, `ON_COMMIT`
`dequeue mode`	`BROWSE`, `LOCKED`, `REMOVE`, `REMOVE_NODATA`
`navigation`	`FIRST_MESSAGE`, `NEXT_MESSAGE`, `NEXT_TRANSACTION`
`state`	`WAITING`, `READY`, `PROCESSED`, `EXPIRED`
`wait`	`FOREVER`, `NO_WAIT`
`delay`	`NO_DELAY`
`expiration`	`NEVER`
`namespace`	`NAMESPACE_AQ`, `NAMESPACE_ANONYMOUS`

Table 2-4 (Cont.) Enumerated Constants in the Oracle Database Advanced Queuing Operational Interface

Parameter	Options
`delivery_mode`	`BUFFERED`, `PERSISTENT`, `PERSISTENT_OR_BUFFERED`
`quosflags`	`NTFN_QOS_RELIABLE`, `NTFN_QOS_PAYLOAD`, `NTFN_QOS_PURGE_ON_NTFN`
`ntfn_grouping_class`	`NFTN_GROUPING_CLASS_TIME`
`ntfn_grouping_type`	`NTFN_GROUPING_TYPE_SUMMARY`, `NTFN_GROUPING_TYPE_LAST`
`ntfn_grouping_repeat_count`	`NTFN_GROUPING_FOREVER`

AQ Background Processes

These topics describe the background processes of Oracle Database Advanced Queuing.

- Queue Monitor Processes
- Job Queue Processes
- AQ Background Architecture

Queue Monitor Processes

Oracle recommends leaving the `AQ_TM_PROCESSES` parameter unspecified and let the system autotune.

Many Oracle Database Advanced Queuing tasks are executed in the background. These include converting messages with `DELAY` specified into the `READY` state, expiring messages, moving messages to exception queues, spilling and recovering of buffered messages, and similar operations.

It is no longer necessary to set `AQ_TM_PROCESSES` when Oracle Database AQ is used. If a value is specified, that value is taken into account when starting the `Qxx` processes. However, the number of Qxx processes can be different from what was specified by `AQ_TM_PROCESSES`.

No separate API is needed to disable or enable the background processes. This is controlled by setting `AQ_TM_PROCESSES` to zero or nonzero. Oracle recommends, however, that you leave the `AQ_TM_PROCESSES` parameter unspecified and let the system autotune.

Note:

If you want to disable the Queue Monitor Coordinator, then you must set `AQ_TM_PROCESSES = 0` in your `pfile` or `spfile`. Oracle strongly recommends that you do NOT set `AQ_TM_PROCESSES = 0`.

Job Queue Processes

Propagation and PL/SQL notifications are handled by job queue (Jnnn) processes.

The parameter `JOB_QUEUE_PROCESSES` no longer needs to be specified. The database scheduler automatically starts the job queue processes that are needed for the propagation and notification jobs.

AQ Background Architecture

Oracle Database Advanced Queuing introduces a new AQ background architecture with a 3-tier design.

- Tier1 (AQPC): Asingle background process called the Advanced Queuing Process Coordinator is created at instance startup. It will be responsible for creating and managing various primary processes. The coordinator statistics can be viewed using `GV$AQ_BACKGROUND_COORDINATOR`.
- Tier2 (QM**): There will be many primary processes named Queue Monitors. Each will be responsible for handling a distinct type of job. Jobs could be of type notification(Emon pool), queue monitors (handling TxEventQ time manager etc) , cross process etc.

  > **Note:**
  >
  > The old processes like QMNC and EMNC will be subsumed within one of new primary processes.

  A job can be defined as a type of work which needs own scheduling mechanism across multiple server processes (Q***) to perform its task . The primary process statistics and their jobs can be viewed using `GV$AQ_JOB_COORDINATOR`.
- Tier3(Q***): There will be a single pool of server processes for all above mentioned primary processes. Each process will be associated to a single primary process at a time. But can be rescheduled to another once original primary relinquishes its need to use it. These servers will perform jobs for respective primary processes providing performance and scalability. The server process statistics and its current primary association can be viewed using `GV$AQ_SERVER_POOL`.