Changes in This Release for Oracle Database Advanced Queuing User's Guide

This preface contains:

- Changes in Oracle Database Advanced Queuing Release 23ai
- Changes in Oracle Database Advanced Queuing Release 21c
- Changes in Oracle Database Advanced Queuing Release 19c, Version 19.2
- Changes in Oracle Database Advanced Queuing 12c Release 2 (12.2)
- Changes in Oracle Database Advanced Queuing 12c Release 1 (12.1.0.2)
- Changes in Oracle Database Advanced Queuing 12c Release 1 (12.1)

Changes in Oracle Database Advanced Queuing Release 23ai

The following are changes in *Oracle Database Advanced Queuing User's Guide* for Oracle Database Release 23ai.

New Features

The following features are new in this release:

- Transactional Event Queue (TxEventQ) Propagation

  Transactional Event Queue (TxEventQ) Propagation enables events at source to be sent to destination queues reliably honoring the ordering semantics, with JMS session level ordering.

  TxEventQ gets created with set of “Event Streams” which is configurable, and it allow the applications to parallelize the publish activity using multiple subscribers. “Event Streams” are viewed as a storage mechanism for TxEventQ. Applications publish events in TxEventQs under 'Event Streams'

  Facility of Propagation leads to multiple benefits, the ability to process published messages remotely, the ability to backup critical data at remote locations and any others.

  See Propagation Features for more information
- Enhancement of Kafka Implementation for Transactional Event Queues

  Enhancement to the client library now allows the Kafka applications to atomically produce and or consume messages from multiple queues of Transactional Event Queues (TxEventQ). Support for Kafka Rebalancing which facilitates distribution of topic partitions among available Kafka consumers is also now available.

  See Kafka APIs for Oracle Transactional Event Queues for more information
- Oracle Database Advanced Queuing (AQ) to Transactional Event Queue (TxEventQ) online Migration Tool

AQ to TxEventQ online Migration Tool enables online and automated migration of a AQ deployment to next generation messaging product TxEventQ. The migration interface works without blocking queuing operations. It creates an internal Oracle managed TxEventQ for the AQ that is being migrated. Migration framework will internally drain (or dequeue) messages from AQ and enqueue them into the internal TxEventQ. At an appropriate time, the TxEventQ will assume the name of the AQ, and the AQ will be dropped.

The online migration tool first help assess the feature mapping between AQ and TxEventQ to flag any mismatches. Once this phase passes an online migration is attempted, and a choice is given after some time to continue with the migration or cancel it.

See Migrating from AQ to TxEventQ and DBMS_AQMIGTOOL for more information

- Python, Node.js and REST drivers for Transactional Event Queues (TxEventQ)

  Starting with Oracle Database 23ai, Transactional Event Queues (TxEventQ) also have support for Python, Node.js and REST drivers.

  See Polyglot Programming with Transactional Event Queues for more information.

- Prometheus/Grafana for Oracle

  Oracle Database 23ai introduces Prometheus/Grafana for Oracle (PGO), which provides database metrics for developers running in a Kubernetes/Docker (K8S) environment. Database metrics are stored in Prometheus, a time-series database and metrics tailored for developers are displayed using Grafana dashboards.

  The metrics give a picture of database performance colored by various applications and the modules it comprises.

  See Monitoring Transactional Event Queues for more information.

Deprecated Features

The following features are deprecated in this release:

- Oracle Messaging Gateway
  - `DBMS_MGWADM`

See Oracle Messaging Gateway for more information.

Changes in Oracle Database Advanced Queuing Release 21c

The following are changes in *Oracle Database Advanced Queuing User's Guide* for Oracle Database Release 21c.

- New Features
- Deprecated Features

New Features

The following features are new in this release:

- Advanced Queuing: Kafka Java Client for Transactional Event Queues

  Kafka Java Client for Transactional Event Queues (TxEventQ) enables Kafka application compatibility with Oracle Database. This provides easy migration of Kafka applications to TxEventQ.

Starting from Oracle Database 21c, Kafka Java APIs can connect to Oracle Database server and use Transactional Event Queues (TxEventQ) as a messaging platform. Developers can migrate an existing Java application that uses Kafka to the Oracle Database. A client side library allows Kafka applications to connect to Oracle Database instead of Kafka cluster and use TxEventQ messaging platform transparently. Kafka interoperability is supported by configuring Kafka JMS Connectors to move messages between the two messaging systems.

See Kafka APIs for Oracle Transactional Event Queues for more information.

- PL/SQL Enqueue and Dequeue Support for JMS Payload in Transactional Event Queues

  PL/SQL APIs perform enqueue and dequeue operations for Java Message Service (JMS) payload in Transactional Event Queues. Similarly, the PL/SQL Array APIs are exposed to Transactional Event Queues JMS users. Since JMS support of heterogeneous messages, dequeue gets one of the five JMS message types back, but cannot predict what is the type of the next message received. Therefore, it can run into application errors with PL/SQL complaining about type mismatch. Oracle suggests that the application always dequeue from Transactional Event Queues using the generic type AQ$_JMS_MESSAGE. PL/SQL administration is also supported.

  See Transactional Event Queues and Enqueuing / Dequeuing Messages for more information

- PL/SQL Enqueue and Dequeue Support for non-JMS Payload in Transactional Event Queues

  To improve throughput and reduce overhead and latency, enqueues and dequeues are optimized to use the message cache, the rules engine, and background processing when possible.

  See Transactional Event Queues and Enqueuing / Dequeuing Messages for more information

- Transactional Event Queues for Performance and Scalability

  Oracle Database 21c introduces Transactional Event Queues (TxEventQ), which are partitioned message queues that combine the best of messaging, streaming, direct messages, and publish/subscribe. TxEventQs have their Queue tables partitioned into multiple Event Streams, which are distributed across multiple RAC nodes for high throughput messaging and streaming of events.

  See Oracle Transactional Event Queues and Advanced Queuing Performance and Scalability for more information

- Simplified Metadata and Schema in Transactional Event Queues

  Oracle Database 21c introduces Transactional Event Queues (TxEventQ), which are partitioned message queues that combine the best of messaging, streaming, direct messages, and publish/subscribe. TxEventQ operates at scale on the Oracle Database. TxEventQ provides transactional event streaming, and runs in the database in a scale of 10s to 100s of billions of messages per day on 2-node to 8-node Oracle RAC databases, both on-premise and on the cloud. TxEventQ has Kafka client compatibility, which means, Kafka producer and consumer can use TxEventQ in the Oracle Database instead of a Kafka broker.

- Support for Message Retention and Seekable Subscribers in Transactional Event Queues

  A user can specify a time for which the message can be retained, even after the subscribers have consumed the message. The retention time is specified in seconds by user. It can vary from `0` to `INIFINITE`. Without retention, when a message is dequeued by all subscribers in sharded queues, the message is permanently removed from the queuing system.

The typical way a subscriber can consume from the queue is through a dequeue operation, which now supports seeking an offset into the message queue. Many queueing applications require subscriber to consume messages that were enqueued prior to its creation. Using this seek capability for subscribers, applications can reposition dequeue point to messages that were enqueued prior to the subscriber creation. This offers flexibility for applications to make the communication truly asynchronous between the producer and the consumer of the message.

See Transactional Event Queues and Message Retention and Transactional Event Queues and Seekable Subscribers for more information.

- Native JSON data type support in AQ and Transactional Event Queues

  Starting from Oracle Database 21c, Advanced Queuing also supports JSON datatype. Along with RAW/ADT payload type we can also specify JSON payload type during queue table creation of classic queue and during queue creation of sharded queue. Users can also specify embedded element JSON with simple adt. `DBMS_AQ`, `DBMS_AQADM` procedures like `create_queue_table`/`create_sharded_queue`/`enqueue`/`dequeue` and procedures for OGG/DG replication also accepts JSON datatype.

Deprecated Features

The following features are deprecated in this release:

- Sharded Queue APIs
  - `CREATE_SHARDED_QUEUE`
  - `DROP_SHARDED_QUEUE`
  - `ALTER_SHARDED_QUEUE`
  - `ISSHARDEDQUEUE`
  - `VERIFY_SHARDED_QUEUE`
- Sharded Queue Views
  - `ALL_QUEUE_SHARDS`
  - `DBA_QUEUE_SHARDS`
  - `USER_QUEUE_SHARDS`
  - `GV$AQ_CACHED_SUBSHARDS`
  - `GV$AQ_CROSS_INSTANCE_JOBS`
  - `GV$AQ_DEQUEUE_SESSIONS`
  - `GV$AQ_INACTIVE_SUBSHARDS`
  - `GV$AQ_MESSAGE_CACHE`
  - `GV$AQ_MESSAGE_CACHE_ADVICE`
  - `GV$AQ_MESSAGE_CACHE_STAT`
  - `GV$AQ_NONDUR_SUBSCRIBER_LWM`
  - `GV$AQ_REMOTE_DEQUEUE_AFFINITY`
  - `GV$AQ_SHARDED_SUBSCRIBER_STAT`
  - `GV$AQ_SUBSCRIBER_LOAD`
  - `GV$AQ_UNCACHED_SUBSHARDS`

- GV$AQ_NONDUR_SUBSCRIBER
- GV$AQ_PARTITION_STATS
- GV$AQ_MESSAGE_CACHE_STAT

See Deprecation of Sharded Queues for more information.

Changes in Oracle Database Advanced Queuing Release 19c, Version 19.2

The following are changes in *Oracle Database Advanced Queuing User's Guide* for Oracle Database Release 19c, Version 19.2

New Features

The following feature is new in this release:

- Enhanced Key-based Messaging

  AQ sharded queues perform substantially better when compared to AQ classic (non-sharded) queues. This is achieved by sharding the queue, where each shard is owned by a specific instance of the database. By default, sharding is completely transparent to the user when it comes to enqueue and dequeue operations. AQ internally puts the message in the appropriate shard to get maximum performance and session level ordering as required by the JMS specification. Session level ordering ensures that no two messages will be dequeued in the reverse order of their enqueue order if both the messages are enqueued by the same session and have the same priority and delivery mode.

  In some cases, user applications want to control the sharding. The user application can choose the shard where a message is enqueued. The users can decide the way they plan to shard their messages in the sharded queue to support the application logic as needed. The performance and ordering benefits of AQ sharded queues are still maintained even if the sharding is under control of the user. Applications can control the following:

  - The number of shards of the queue
  - Key based enqueues: The enqueue session can choose the shard of the queue where the message will be enqueued by providing a key with the message at the time of enqueue. AQ server ensures that all the messages of a key are enqueued in the same shard. A shard can have messages of different keys.
  - Sticky dequeues: A shard can have only one active dequeue session for a single-consumer queue or JMS Queue. Similarly, a shard can have only one dequeue session per subscriber for a multi-consumer queue or JMS Topic. That dequeue session will stick to that shard of the queue for the session's lifetime. Such functionality is also available for JMS listeners.

> **See Also:**
>
> User Event Streaming

Desupported Features

The following feature is desupported in this release:

- Desupport of Oracle Streams

  Starting in Oracle Database 19c, the Oracle Streams feature is desupported. Use Oracle GoldenGate to replace all replication features of Oracle Streams.

Changes in Oracle Database Advanced Queuing 12*c* Release 2 (12.2.)

The following are changes in *Oracle Database Advanced Queuing User's Guide* for Oracle Database 12*c* Release 2 (12.2).

New Features

The following features are new in this release:

- PL/SQL enqueue and dequeue support for JMS and non-JMS (ADT or RAW) payload in Sharded Queues

  Oracle Database 12*c* Release 2 (12.2) extends and supports PL/SQL APIs to perform enqueue and dequeue operations for JMS, ADT, and RAW payload in sharded queues. The PL/SQL Array APIs also support sharded queues. Many existing non-JMS applications can now use sharded queues with little or no change.

  Starting from Oracle Database 12*c* Release 2 (12.2), JMS customers using sharded queues can make use of PL/SQL notification to register a PL/SQL procedure that gets automatically invoked by AQ Server on successful enqueue. PL/SQL notification can eliminate the need for clients to poll the queue for messages because messages can be automatically dequeued and processed at the server.

  See Managing Sharded Queues for more information.

- Sharded Queue Diagnosability and Manageability

  Starting from 12*c* Release 2 (12.2), AQ sharded queues provides an advisor, views, and automated management for its message cache to optimize `STREAMS_POOL` memory allocation and throughput.

  See Transactional Event Queues Tuning for more information.

- Longer Identifiers for Oracle Database Advanced Queuing

  Starting from 12*c* Release 2 (12.2), the maximum length of AQ queue names has been increased to 122 bytes. The maximum length of subscriber and recipient names is increased to 128 characters. For the AQ Rules Engine, the maximum length of rule names and rule set names is now 128 bytes.

Changes in Oracle Database Advanced Queuing 12*c* Release 1 (12.1.0.2)

The following are changes in *Oracle Database Advanced Queuing User's Guide* for Oracle Database 12*c* Release 1 (12.1.0.2).

New Features

The following feature is new in this release:

- JMS Streaming

In Oracle Database 12*c* Release 1 (12.1.0.2), Advanced Queuing introduces JMS Streaming with enqueue and dequeue for sharded queues through `AQjmsBytesMessage` and `AQjmsStreamMessage`, for the applications interested in sending and receiving large message data or payload.

See "JMS Streaming" for more information.

Changes in Oracle Database Advanced Queuing 12*c* Release 1 (12.1)

The following are changes in *Oracle Database Advanced Queuing User's Guide* for Oracle Database 12*c* Release 1 (12.1).

New Features

The following features are new in this release:

- JMS Sharded Queues

  In Oracle Database 12*c* Release 1 (12.1), Advanced Queuing introduces high performing and scalable sharded JMS Queues. A sharded queue is a single logical queue that is divided into multiple, independent, physical queues through system-maintained partitioning. A sharded queue increases enqueue-dequeue throughput, especially across Oracle RAC instances, because ordering between two messages on different queue shards is best effort. Each shard is ordered based on enqueue time within a session. Sharded queues automatically manage table partitions so that enqueuers and dequeuers do not contend among themselves. In addition, sharded queues use an in-memory message cache to optimize performance and reduce the disk and CPU overhead of AQ-JMS enqueues and dequeues. Sharded queues are the preferred JMS queues for queues used across Oracle RAC instances, for queues with high enqueue or dequeue rates, or for queues with many subscribers.

  In 12.2, Sharded Queues have been enhanced to support more than JMS. See Sharded Queues for more information.

- Result Cache Enhancement

  In Oracle Database 12*c* Release 1 (12.1), the Rules Engine introduces a result cache to improve the performance of many commonly used rules. The result cache will bypass the evaluation phase if an expression with the same attributes has already been evaluated earlier. Not all rule results are cached, such as when rule results are potentially non-deterministic or when not all rules are evaluated or when attributes include non-scalar data types. For Advanced Queues, the cache is most useful when subscriptions and their dequeue sessions are long-lived.

- `LONG VARCHAR` Support

  The `LONG VARCHAR` data type is supported by Oracle Database Advanced Queuing in Oracle Database 12*c* Release 1 (12.1).

- 3-Tier Background Architecture

  Oracle Database 12c Release 1 (12.1) introduces the AQ background process architecture with a new a 3-tier design.

  See "AQ Background Architecture" for more information.

- Support for Data Guard Database Rolling Upgrade

Databases that use Oracle Database Advanced Queuing can now be upgraded to new Oracle database releases and patch sets in rolling fashion using Data Guard database rolling upgrades (transient logical standby database only). Rolling upgrades are supported beginning in Oracle Database 12*c* Release 1 (12.1).

Data Guard database rolling upgrades reduce planned downtime by enabling the upgrade to new database releases or patch sets in rolling fashion. Total database downtime for such an upgrade is limited to the small amount of time required to execute a Data Guard switchover.

The following packages will have support for rolling upgrade using logical standby:

- `DBMS_AQ`
- `DBMS_AQJMS`
- `DBMS_AQADM`, except for the following procedures:
  - `SCHECULE_PROPAGATION`
  - `UNSCHEDULE_PROPAGATION`
  - `ALTER_PROPAGATION_SCHEDULE`
  - `ENABLE_PROPAGATION_SCHEDULE`
  - `DISABLE_PROPAGATION_SCHEDULE`

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for more information on the Oracle Database AQ packages