30

DBMS_AQMIGTOOL

The DBMS_AQMIGTOOL package simplifies migration from Oracle Database Advanced Queuing (AQ) to Transactional Event Queue (TxEventQ) with orchestration automation, source and target compatibility diagnostics and remediation, and a unified user experience. Migration scenarios can be short or long-lived and be performed with or without AQ downtime, eliminating operational disruption.

See Also:

Migrating from AQ to TxEventQ in *Oracle Database Advanced Queuing User's Guide* for detailed information about DBMS_AQMIGTOOL

This chapter contains the following topics:

- Security Model
- DBMS_AQMIGTOOL Constants
- Summary of DBMS_AQMIGTOOL Subprograms

DBMS_AQMIGTOOL Security Model

All DBMS_AQMIGTOOL subprograms require the user to have EXECUTE privilege over the DBMS_AQMIGTOOL package. If the invoker of the package is the owner of the queue, then only EXECUTE privilege on the DBMS_AQMIGTOOL package is sufficient, But if the invoker is non-owner, then it also needs to have MANAGE_ANY queue system privilege, which can be granted through DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE.

DBMS_AQMIGTOOL Constants

The DBMS_AQMIGTOOL package defines several constants that can be used for specifying parameter values.

When using enumerated constants, such as AUTOMATIC, INTERACTIVE, or ENABLE_EVALUATION, 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_AQMIGTOOL. For example, DBMS_AQMIGTOOL.AUTOMATIC.

Table 30-1 DBMS_AQMIGTOOL Constants

Paramater	Options	Value
`mig_mode`	`AUTOMATIC`	1
	`INTERACTIVE`	2
	`OFFLINE`	3
	`ONLY_DEFINITION`	4
`ordering`	`GLOBAL`	1
	`SESSION`	2
`checkmode`	`CURRENT`	1
	`ENABLE_EVALUATION`	2
`cancelmode`	`RESTORE`	1
	`NORESTORE`	2
	`Q_EMPTY`	3
`purge_option`	`ONLY_CQ`	1
	`ONLY_TXEVENTQ`	2
	`BOTH_Q`	3

Summary of DBMS_AQMIGTOOL Subprograms

This section lists and briefly describes the DBMS_AQMIGTOOL subprograms.

Table 30-2 DBMS_AQMIGTOOL Package Subprograms

Subprogram	Description
CANCEL_MIGRATION Procedure	Cancels an ongoing migration by dropping the interim TxEventQ
CHECK_MIGRATED_MESSAGES Procedure	Provides a count of messages in the `READY` state within the AQ and the interim TxEventQ
CHECK_MIGRATION_TO_TXEVENTQ Procedure	Analyzes the AQ's definition and data and report any features unsupported in TxEventQ
CHECK_STATUS Procedure	Returns the current status of the ongoing migration process
CLEAR_UNSUPPORTED_FEATURE_TABLE Procedure	Clears entries from the `USER_TXEVENTQ_MIGRATION_STATUS` view
COMMIT_MIGRATION Procedure	Completes the migration process by dropping AQ, renaming the interim TxEventQ to match AQ's name, and enabling TxEventQ for all operations
DISABLE_MIGRATION_CHECK Procedure	Disables the internal AQ monitoring for detecting unsupported features and stops event insertion in the `USER_TXEVENTQ_MIGRATION_STATUS` view
INIT_MIGRATION Procedure	Analyzes the AQ's definition and data for unsupported features and then starts the migration process by creating an interim TxEventQ copying the AQ's configuration
PURGE_QUEUE_MESSAGES Procedure	Purges messages from the queue

Table 30-2 (Cont.) DBMS_AQMIGTOOL Package Subprograms

Subprogram	Description
RECOVER_MIGRATION Procedure	Helps recovery from any failures during execution of migration procedures such as `DBMS_AQMIGTOOL.INIT_MIGRATION`, `DBMS_AQMIGTOOL.COMMIT_MIGRATION`, or `DBMS_AQMIGTOOL.CANCEL_MIGRATION`
RENAME_QUEUE Procedure	Renames the TxEventQ along with its default exception queue if present

CANCEL_MIGRATION Procedure

This procedure serves the purpose of canceling an ongoing migration. It involves dropping the interim TxEventQ, which was created during the execution of `DBMS_AQMIGTOOL.INIT_MIGRATION`.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.CANCEL_MIGRATION (
    cqschema     IN VARCHAR2,
    cqname       IN VARCHAR2,
    cancelmode   IN NUMBER DEFAULT DBMS_AQMIGTOOL.RESTORE );
```

Parameters

Table 30-3 CANCEL_MIGRATION Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the queue name for which the migration needs to be canceled
`cancelmode`	Specifies the mode in which the user wants to cancel the migration. The following are the possible values: `DBMS_AQMIGTOOL.RESTORE` (default): This option restores the messages from the interim TxEventQ into the AQ, including their message state. New MSGID will get populated for restored messages. **Note:** The priority behavior of restored messages may change as AQ and TxEventQ default values are different. `DBMS_AQMIGTOOL.NORESTORE`: Messages within interim TxEventQ will be discarded. `DBMS_AQMIGTOOL.EMPTY`: If the interim TxEventQ is not empty, selecting this option will trigger an exception, prompting the user to dequeue all messages from the interim TxEventQ. This mode is useful if the user wishes to prevent message migration while transitioning to the AQ.

Usage Notes

A prerequisite for this procedure is that the migration must already be started on the queue, that is, `DBMS_AQMIGTOOL.INIT_MIGRATION` should be invoked before executing this procedure. For `DBMS_AQMIGTOOL.RESTORE` mode, the TxEventQ's exception queue messages are not restored to AQ or its exception queue.

CHECK_MIGRATED_MESSAGES Procedure

This procedure calculates the count of messages in the `READY` state within both the AQ and the interim TxEventQ. This count provides valuable insight before using `DBMS_AQMIGTOOL.COMMIT_MIGRATION` or `DBMS_AQMIGTOOL.CANCEL_MIGRATION`. The calculated count is independent of the number of subscribers.

Syntax

```
PROCEDURE SYS.DBMS_AQMIGTOOL.CHECK_MIGRATED_MESSAGES (
    cqschema                            IN VARCHAR2,
    cqname                              IN VARCHAR2,
    txeventq_migrated_message           IN OUT NUMBER,
    cq_pending_messages                 IN OUT NUMBER);
```

Parameters

Table 30-4 CHECK_MIGRATED_MESSAGES Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the name of the queue on which the migration process has started
`txeventq_migrated_message`	Represents the count of messages in the `READY` state within the interim TxEventQ. The count helps estimate the potential fallback time if the user opts to execute `DBMS_AQMIGTOOL.CANCEL_MIGRATION`.
`cq_pending_messages`	Represents the count of messages in the `READY` state within the AQ. The count helps determine the remaining number of `READY` state messages until the AQ is empty, which is a prerequisite for using the `DBMS_AQMIGTOOL.COMMIT_MIGRATION` procedure.

Usage Notes

A prerequisite for this procedure is that the migration must already be started on the queue, meaning `DBMS_AQMIGTOOL.INIT_MIGRATION` should be invoked before executing this procedure.

CHECK_MIGRATION_TO_TXEVENTQ Procedure

This procedure examines the AQ's definition and data and reports any features that are unsupported in TxEventQ. If no unsupported features are detected, then `migration_report` will be empty.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.CHECK_MIGRATION_TO_TXEVENTQ (
    cqschema            IN VARCHAR2,
    cqname              IN VARCHAR2,
    migration_report    IN OUT TXEVENTQ_MIGREPORT_ARRAY,
    checkmode           IN NUMBER DEFAULT DBMS_AQMIGTOOL.ENABLE_EVALUATION);
```

Parameters

Table 30-5 CHECK_MIGRATION_TO_TXEVENTQ Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the queue name for which unsupported features need to be checked
`migration_report`	A Varray containing details of unsupported events and their corresponding descriptions. It holds the most recent 20 unsupported features.
`checkmode`	Specifies the mode in which the user wants to check. The following are the possible values: `DBMS_AQMIGTOOL.CURRENT`: This mode generates a report using the current definition of the AQ and its data. `DBMS_AQMIGTOOL.ENABLE_EVALUATION` (Default): This mode, along with the `CURRENT` option report, will enable the monitoring of the AQ. It helps the capturing runtime-specific unsupported features. As more workload is applied to the AQ, the unsupported features, if found any, are recorded in an internal table accessible through the `USER_TXEVENTQ_MIGRATION_STATUS` view. Users can disable monitoring using `DBMS_AQMIGTOOL.DISABLE_MIGRATION_CHECK`.

Usage Notes

Several features like relative message identifier, sequence deviation, and transformation are not supported in TxEventQ. If the queue uses any of them, they will be recorded in the `migration_report`.

It is recommended to use the `DBMS_AQMIGTOOL.CHECK_MIGRATION_TO_TXEVENTQ` procedure to detect unsupported features before beginning the migration process.

See Also:

- Limitations and Workarounds in *Oracle Database Transactional Event Queues and Advanced Queuing User's Guide*
- `ENQUEUE_OPTIONS_T` Type in *Oracle Database PL/SQL Packages and Types Reference*

CHECK_STATUS Procedure

This procedure returns the status of the migration process. In case any unsupported features are detected, the procedure will return details about the most recent unsupported feature, including its description. On the other hand, if no unsupported features are detected, it will return a status of 'No Compatibility Error'.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.CHECK_STATUS (
    cqschema                    IN VARCHAR2,
    cqname                      IN VARCHAR2,
    status                      IN OUT VARCHAR2,
    migration_comment           IN OUT VARCHAR2);
```

Parameters

Table 30-6 CHECK_STATUS Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the name of the queue for which the migration process status needs to be checked
`status`	Return the compatibility status. In case of incompatibility, that is, detection of unsupported features; the most recent unsupported event will be returned, and the status return format will be: "Compatibility Error: <feature_name> Unsupported Feature".
`migration_comment`	If the status is incompatible, the description of the unsupported event will be provided

Usage Notes

A prerequisite for this procedure is that the migration must already be started on the queue, meaning `DBMS_AQMIGTOOL.INIT_MIGRATION` should be invoked before executing this procedure.

The following table will store the information regarding events during pre-init or post-init migration. Users can access this information through security views: `DBA_TXEVENTQ_MIGRATION_STATUS`, `USER_TXEVENTQ_MIGRATION_STATUS`, and `ALL_TXEVENTQ_MIGRATION_STATUS`.

```
sys.aq$_migration_status(
        migration_id           RAW(16);
        source_schema          VARCHAR2(128) NOT NULL,
        source_queue           VARCHAR2(128) NOT NULL,
        source_queue_table     VARCHAR2(128),
        target_schema          VARCHAR2(128) NOT NULL,
        target_queue           VARCHAR2(128) NOT NULL,
        status                 NUMBER,
        event                  VARCHAR(128),
        event_timestamp        TIMESTAMP WITH TIME ZONE,
        event_id               NUMBER,
        event_error            VARCHAR2(1024),
        ordering               VARCHAR(30),
        suffix                 VARCHAR2(2),
        mig_mode               NUMBER,
        spare1                 NUMBER,
        spare2                 VARCHAR2(30),
        spare3                 TIMESTAMP WITH TIME ZONE
)
```

A unique `migration_id` will be assigned to each initiated migration.

See Also:

DBA_TXEVENTQ_MIGRATION_STATUS in Oracle Database Reference

USER_TXEVENTQ_MIGRATION_STATUS in Oracle Database Reference

ALL_TXEVENTQ_MIGRATION_STATUS in Oracle Database Reference

CLEAR_UNSUPPORTED_FEATURE_TABLE Procedure

This procedure removes entries from the underlying table of the `USER_TXEVENTQ_MIGRATION_STATUS` view. This view stores records related to unsupported features detected by the `DBMS_AQMIGTOOL.CHECK_MIGRATION_TO_TXEVENTQ` procedure and details of other migration procedure calls (`INIT_MIGRATION`/`COMMIT_MIGRATION`/`CANCEL_MIGRATION`) used for internal purposes.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.CLEAR_UNSUPPORTED_FEATURE_TABLE (
    cqschema             IN    VARCHAR2,
    cqname               IN    VARCHAR2 DEFAULT NULL,
    eraseall            IN    BOOLEAN DEFAULT FALSE);
```

Parameters

Table 30-7 CLEAR_UNSUPPORTED_FEATURE_TABLE Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the name of the queue for which records need to be cleared
`eraseall`	`TRUE` erases all the records for the specified queue. **Note:** The `TRUE` value is intended solely for internal purposes and should not be used without consulting Oracle support. `FALSE` erases records related to unsupported features only.

Usage Notes

Users can use this procedure to erase the records generated by the `DBMS_AQMIGTOOL.CHECK_MIGRATION_TO_TXEVENTQ` procedure. Dropping the queue or the user will also erase records for that queue. However, executing `DBMS_AQMIGTOOL.COMMIT_MIGRATION` or `DBMS_AQMIGTOOL.CANCEL_MIGRATION` will not clear the records for the queue. Therefore, this procedure offers the flexibility to erase records for a queue at any point.

COMMIT_MIGRATION Procedure

This procedure completes the migration process. It drops AQ and renames the interim TxEventQ to the AQ's name, and enables the TxEventQ for all operations. It is important to note that an empty AQ (that is, with no messages in the `READY` state) is required to execute the procedure successfully; otherwise, an exception will be raised.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.COMMIT_MIGRATION (
    cqschema         IN VARCHAR,
    cqname           IN VARCHAR,
    ignore_warning   IN BOOLEAN DEFAULT FALSE));
```

Parameters

Table 30-8 COMMIT_MIGRATION Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the queue name for which the migration needs to be completed
`ignore_warning`	After `DBMS_AQMIGTOOL.INIT_MIGRATION`, all unsupported events are recorded as warnings. `TRUE` means the procedure will ignore the warnings and complete the migration. `FALSE` means the procedure will throw an error if there are any warnings.

Usage Notes

A prerequisite for this procedure is that the migration must already be started on the queue. In other words, the `DBMS_AQMIGTOOL.INIT_MIGRATION` procedure should be invoked before executing this procedure. The messages from the AQ's exception queue will not be copied to the TxEventQ's exception queue.

DISABLE_MIGRATION_CHECK Procedure

This procedure disables the internal monitoring of the AQ aimed at detecting unsupported features. It also stops the recording of events for the `USER_TXEVENTQ_MIGRATION_STATUS` view.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.DISABLE_MIGRATION_CHECK (
    cqschema       IN VARCHAR2)
    cqname         IN VARCHAR2);
```

Parameters

Table 30-9 DISABLE_MIGRATION_CHECK Procedure Parameter

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the name of the queue on which migration monitoring needs to be disabled

Usage Notes

As a pre-migration step, the user can start recording unsupported features by invoking DBMS_AQMIGTOOL.CHECK_MIGRATION_TO_TXEVENTQ with DBMS_AQMIGTOOL.ENABLE_EVALUATION option. This can be followed by running a comprehensive workload on the AQ to detect potential issues and then stop recording unsupported features by calling DBMS_AQMIGTOOL.DISABLE_MIGRATION_CHECK. The user can modify the workload if any migration issues are found before repeating the process. If no migration issues were found, migration of the AQ can be attempted.

INIT_MIGRATION Procedure

This procedure examines the definition and data of the AQ to detect any features that are not supported in TxEventQ. If any unsupported features are detected, an exception is raised. Otherwise, the procedure starts the migration process by creating an interim TxEventQ that copies the configuration of the AQ, which includes payload type, rules, subscribers, privileges, PL/SQL notifications, and more.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.INIT_MIGRATION (
    cqschema            IN VARCHAR2,
    cqname              IN VARCHAR2 DEFAULT NULL,
    txeventqschema      IN VARCHAR2 DEFAULT NULL,
    txeventqname        IN VARCHAR2 DEFAULT NULL,
    mig_mode            IN NUMBER DEFAULT DBMS_AQMIGTOOL.INTERACTIVE,
    ordering            IN NUMBER DEFAULT DBMS_AQMIGTOOL.GLOBAL,
    suffix              IN VARCHAR2 DEFAULT 'M');
```

Parameters

Table 30-10 INIT_MIGRATION Procedure Parameters

Parameter	Description
cqschema	Specifies the schema name where the queue (AQ) exists
cqname	Specifies the name of the queue (AQ) for which the migration process is to be started. If cqname is NULL, then migration initiates for all AQs in cqschema, ignoring queues with unsupported features. Execution displays the count of queues where DBMS_AQMIGTOOL.INIT_MIGRATION succeeded.
txeventqschema	Specifies the schema name where the target TxEventQ is intended to be created. Only provide this value for DBMS_AQMIGTOOL.ONLY_DEFINITION mode. If the invoker is on a different schema than txeventqschema, it needs MANAGE_ANY queue system privilege to execute this procedure successfully.
txeventqname	Specifies the name of the queue for the target TxEventQ to be created. Only provide this value for DBMS_AQMIGTOOL.ONLY_DEFINITION mode.

Table 30-10 (Cont.) INIT_MIGRATION Procedure Parameters

Parameter	Description
`mig_mode`	Specifies the migration mode. The following are the possible values: `DBMS_AQMIGTOOL.AUTOMATIC`: Enqueue and dequeue operations are allowed in this mode, but a background job will attempt to execute `DBMS_AQMIGTOOL.COMMIT_MIGRATION` once no messages are left in the `READY` state in AQ and no unsupported features are detected. `DBMS_AQMIGTOOL.INTERACTIVE` (Default): In this mode, both enqueue and dequeue operations are allowed. `DBMS_AQMIGTOOL.OFFLINE`: Only dequeue operations are allowed in this mode, which helps in reducing the workload by restricting the new enqueue operations. `DBMS_AQMIGTOOL.ONLY_DEFINITION`: This mode creates a separate TxEventQ with the same configuration as the AQ instead of an interim TxEventQ setup. This completes the migration process, with AQ and TxEventQ remaining in the system. The messages present in AQ will not be copied to the newly created TxEventQ. **Note:** For the `DBMS_AQMIGTOOL.ONLY_DEFINITION` mode, there is no need to call `DBMS_AQMIGTOOL.COMMIT_MIGRATION` or `DBMS_AQMIGTOOL.CANCEL_MIGRATION` to complete or cancel the migration; this `DBMS_AQMIGTOOL.INIT_MIGRATION` call is sufficient. However, for other modes, the user must explicitly call other procedures in the `DBMS_AQMIGTOOL` package to proceed further.
`ordering`	Specifies the message level ordering the user wants to follow. The following are the possible values: `DBMS_AQMIGTOOL.GLOBAL` (Default): This option implements global-level message ordering by setting the number of event streams in the TxEventQ to one. `DBMS_AQMIGTOOL.SESSION`: This option imposes a message order only for a session. The number of event streams in the TxEventQ will be set based on the initialization parameter `_aq_init_shards`. Oracle recommends using this option to achieve the full performance benefits of TxEventQ.
`suffix`	Specifies a single character suffix for naming the interim TxEventQ. The interim TxEventQ name will be in the format `<cqname>_<suffix>`. The default value for the suffix is `M`.

Usage Notes

- This procedure will also create the default exception queue on the TxEventQ, following the naming format `AQ$_<TxEventQ_name>_E`.
- The following points are not relevant for `DBMS_AQMIGTOOL.ONLY_DEFINITION` mode but apply to other modes:

  – It restricts AQ from administrative operations to maintain interim TxEventQ configuration integrity until the migration is completed or canceled.
  – The Enqueue - Dequeue operations are allowed on AQ:
    * Enqueue requests for new messages are directed to the interim TxEventQ.
    * Messages are first dequeued from AQ. If no messages are in the `READY` state, then messages are dequeued from the interim TxEventQ.
  – Users are restricted from performing all direct operations on the interim TxEventQ. Enqueue and dequeue operations on interim TxEventQ will always be performed through AQ.
- If the procedure triggers an exception due to the detection of unsupported features, it is recommended to use the `DBMS_AQMIGTOOL.CHECK_MIGRATION_TO_TXEVENTQ` procedure to obtain a detailed list of the detected unsupported features.
- If there is a name conflict, such as when a queue (AQ) being migrated shares the same name as the queue table in which it resides, then only the `DBMS_AQMIGTOOL.ONLY_DEFINITION` mode is supported. For any other modes, attempting to proceed will result in an exception being raised.

PURGE_QUEUE_MESSAGES Procedure

This procedure purges messages from the queue. It can perform message purging from the AQ, the interim TxEventQ, or both, depending on user input.

Specifically, one of the prerequisites for executing `DBMS_AQMIGTOOL.COMMIT_MIGRATION` is to ensure an empty AQ, meaning no messages in the `READY` state. This procedure allows users to efficiently purge all messages from the AQ to fulfill this requirement.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.PURGE_QUEUE_MESSAGES (
    cqschema        IN VARCHAR2,
    cqname          IN VARCHAR2,
    purge_option    IN NUMBER DEFAULT DBMS_AQMIGTOOL.ONLY_CQ);
```

Parameters

Table 30-11 PURGE_QUEUE_MESSAGES Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the name of the queue where the migration process started
`purge_option`	Specifies the option from which queue messages need to be purged. The following options are available: `DBMS_AQMIGTOOL.ONLY_CQ` (Default): Purge messages only from the AQ. `DBMS_AQMIGTOOL.ONLY_TXEVENTQ`: Purge messages only from the interim TxEventQ. `DBMS_AQMIGTOOL.BOTH_Q`: Purge messages from both the AQ and the interim TxEventQ.

Usage Notes

Suppose the count of messages in the `READY` state within the AQ is obtained from the `DBMS_AQMIGTOOL.CHECK_MIGRATED_MESSAGE` procedure is large, and the user wishes to speed up the `DBMS_AQMIGTOOL.COMMIT_MIGRATION` process without waiting for dequeues to consume the messages, the `DBMS_AQMIGTOOL.PURGE_QUEUE_MESSAGES` procedure can be used.

A prerequisite for this procedure is that the migration must already be started on the queue, meaning `DBMS_AQMIGTOOL.INIT_MIGRATION` should be invoked before executing this procedure.

RECOVER_MIGRATION Procedure

This procedure restores the migration state to the nearest feasible and consistent point, either before or after the execution of `DBMS_AQMIGTOOL.CANCEL_MIGRATION`, `DBMS_AQMIGTOOL.COMMIT_MIGRATION`, or `DBMS_AQMIGTOOL.INIT_MIGRATION`. The recovered state is then displayed to the user through the output parameter `recovery_message`, providing guidance for further action. If migration procedures experience unexpected failures, such as instance crashes, then this procedure can be used to recover the migration to the nearest consistent state like before `INIT_MIGRATION`, after `INIT_MIGRATION`, after `COMMIT_MIRGATION`, and after `CANCEL_MIGRATION`.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.RECOVER_MIGRATION (
    cqschema                IN VARCHAR2,
    cqname                  IN VARCHAR2,
    recovery_message        OUT VARCHAR2);
```

Parameters

Table 30-12   RECOVER_MIGRATION Procedure Parameters

Parameter	Description
`cqschema`	Specifies the schema name where the queue exists
`cqname`	Specifies the name of the queue where the migration procedure was attempted
`recovery_message`	Returns a descriptive message indicating the restored migration state's nearest feasible and consistent point

Usage Notes

The following table shows all the possible recovery_message:

Table 30-13   Recommended action table according to recovery_message

Error while executing migration procedure:	**`recovery_message`**	Recommended Action
`DBMS_AQMIGTOOL.INIT_MIGRATION`	`State is restored to before INIT_MIGRATION call execution.`	To start the migration, the user must explicitly call `DBMS_AQMIGTOOL.INIT_MIGRATION` again.

Table 30-13 (Cont.) Recommended action table according to recovery_message

Error while executing migration procedure:	`recovery_message`	Recommended Action
`DBMS_AQMIGTOOL.INIT_MIGRATION`	`State is restored to after INIT_MIGRATION call execution.`	No further action is needed to start the migration, as this procedure has successfully started.
`DBMS_AQMIGTOOL.COMMIT_MIGRATION`	`State is restored to after COMMIT_MIGRATION call execution.`	No further action is needed to complete the migration, as this procedure has successfully completed it.
`DBMS_AQMIGTOOL.CANCEL_MIGRATION`	`State is restored to before CANCEL_MIGRATION call execution.`	To proceed with canceling the migration, the user must explicitly call `DBMS_AQMIGTOOL.CANCEL_MIGRATION` again.
`DBMS_AQMIGTOOL.CANCEL_MIGRATION`	`State is restored to after CANCEL_MIGRATION call execution.`	No further action is needed to cancel the migration, as this procedure has successfully cancelled it.
`No migration procedure`	`No need for recovery call.`	Since no migration procedure execution is detected, restoring it to the nearest feasible and consistent point is not required.

RENAME_QUEUE Procedure

This procedure renames the TxEventQ along with its default exception queue if present.

Syntax

```
PROCEDURE DBMS_AQMIGTOOL.RENAME_QUEUE (
    schema              IN VARCHAR2,
    qname               IN VARCHAR2,
    new_qname           IN VARCHAR2);
```

Parameters

Table 30-14 RENAME_QUEUE Procedure Parameters

Parameter	Description
`schema`	Specifies the schema name where the queue exists
`qname`	Specifies the current name of the queue to be renamed
`new_qname`	Specify the new name to be given to the existing queue. The new name must be unique within a schema and must follow object name guidelines in Oracle Database SQL Language Reference with regard to reserved characters.

Usage Notes

This procedure requires a prerequisite step: the user must perform `DBMS_AQADM.STOP_QUEUE` to ensure there are no concurrent enqueue and dequeue transactions.

If the default exception queue is present, it will be renamed from `<schema>.AQ$_<qname>_E` to `<schema>.AQ$_<new_qname>_E`.