Oracle® Tuxedo Message Queue (OTMQ)

Product Overview

12c Release 1 (12.1.1)

June 2012

ORACLE

Oracle® Tuxedo Message Queue (OTMQ) Product Overview, 12c Release 1 (12.1.1)

Copyright © 2012 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Oracle Tuxedo Message Queue Product Overview

Understanding Oracle OTMQ 1
Multiple Queue Types 2
Permanent Queue 2
Primary Queue (PQ) 2
Secondary Queue (SQ) 3
Multi-Reader Queue (MRQ) 3
Unlimited Queue (UNLIMITQ) 3
Temporary Queue 3
Reliable Message Delivery 3
Delivery Interest Point 3
Synchronous/Asynchronous Communication 4
Synchronous mode 4
Asynchronous mode 4
No-reply mode 4
Undelivered Message Action 4
Message Filter 5
Publish/Subscribe 5
Publish/broadcast 5
Subscribe 5
Naming 6
Journal 6
WS SAF 6
Large QSPACE Size Support 7
Automatic Multi-Context Support 7
Oracle OTMQ System Components 7
See Also 9

Oracle Tuxedo Message Queue Product Overview

The following sections provide an overview to the Oracle Tuxedo Message Queue product:

- Understanding Oracle OTMQ
- Oracle OTMQ System Components

Understanding Oracle OTMQ

The Oracle Tuxedo Message Queue (OTMQ) component improves and enhances the Oracle Tuxedo queuing services. Besides the existing queuing features of the Oracle Tuxedo /Q component, the OTMQ also provides richer queuing features, such as Reliable Message Delivery, Synchronous/Asynchronous Messaging, Publish/Subscribe, Message Filtering, Dynamic Queue Alias (Naming), Journal, etc. Also since the OTMQ was implemented based on the Oracle Tuxedo infrastructure, it can provide supports on transaction, security, scalability and HA.

The following sections provide an overview of OTMQ features:

- Multiple Queue Types
- Reliable Message Delivery
- Delivery Interest Point
- Synchronous/Asynchronous Communication
- Undelivered Message Action
- Message Filter

- Publish/Subscribe
- Naming
- Journal
- WS SAF
- Large QSPACE Size Support
- Automatic Multi-Context Support

Multiple Queue Types

Oracle OTMQ can have multiple attributes, which define its behaviors. For more information, see `tmqadmin()` in the Oracle Tuxedo Message Queue Command Reference Guide.

According to the life cycle of queues, the queue types can be:

- Permanent Queue
- Temporary Queue

Permanent Queue

The permanent queue must be pre-allocated. The permanent queue has active property defined as permanent or temporary. The permanent active queue can always receive and store messages even there is no application attaching to it, unless storage quota is exceeded. The temporary active queue can only receive and store messages when it is attached by application. The permanent queue is created by `tmqadmin() qcreate` command.

The permanent queue should be specified as one of the following types when being created:

- Primary Queue (PQ)
- Secondary Queue (SQ)
- Multi-Reader Queue (MRQ)
- Unlimited Queue (UNLIMITQ)

Primary Queue (PQ)

The primary queue acts as the main mailbox for a user process to receive messages from other processes. It can be attached by only one queue client. An application can have only one primary queue, although it may associate with queues of other types.

Secondary Queue (SQ)

The secondary queue acts as an alternate mailbox for a user process to receive messages. An application owns the secondary queue by attaching to it. When an application attaching to a primary queue that is the owner of the secondary queue, the application is automatically attached to the secondary queue at the same time. When the application detaching from its primary queue, all secondary queues associated will be detached too.

Multi-Reader Queue (MRQ)

The multi-reader queue is pre-allocated. It can be attached by multiple queue clients at the same time.

Unlimited Queue (UNLIMITQ)

The unlimited queue's behavior is totally same as the traditional Tuxedo /Q queue. An application need not attach the unlimited queue to dequeue from it.

Temporary Queue

The temporary queue is created at runtime when an application requests to attach a temporary queue. The process attaching to the temporary queue is the owner of this queue, and no other processes can attach or receive messages from this queue. Once the application detaches from the temporary queue, all the messages left in this queue will be deleted. The temporary queue count and range are specified by tmqadmin() qspacecreate command.

Reliable Message Delivery

Oracle OTMQ provides reliable message delivery, which can make sure the message be delivered to the receiver successfully.

For more information, see Using Recoverable Messaging in the Oracle Tuxedo Message Queue Programming Guide.

Delivery Interest Point

Delivery Interest Point (DIP) is the checkpoint during the message delivery. When a message passes the specified checkpoint, an ACK notification message is returned to the sender application to indicate the message delivery status.

Oracle OTMQ supports following DIPs:

- SAF: when a recoverable message is stored in the local queue space storage

- DQF: when a recoverable message is stored in the target queue space storage
- MEM: when a non-recoverable message is stored in the target queue
- DEQ: when a non-recoverable message is dequeued from the target queue
- ACK: when the receiver explicitly acknowledge the non-recoverable message by sending back the ACK notification message using `tpqconfirmmsg(3c)`
- CONF: when a recoverable message is successfully delivered from DQF to the target queue, and explicitly acknowledged using `tpqconfirmmsg(3c)`

Synchronous/Asynchronous Communication

Oracle OTMQ applications can enqueue/send messages in the following modes:

Synchronous mode

The application waits for acknowledgement after sending messages to target. Also the application can choose to wait until the message is sent to specific Delivery Interest Points.

Asynchronous mode

The application doesn't wait after sending messages, but should explicitly call `tpdeqplus(3c)` to get ACK notification message to verify if the message has been successfully delivered to target.

No-reply mode

The application does not wait after sending messages, also doesn't expect any acknowledge messages. This mode is for performance sensitive scenarios.

Undelivered Message Action

Undelivered Message Action (UMA) means the action when the message sender receives a failure acknowledgment notification.

If Reliable Message Delivery is enabled, UMA means the action when the message fails to be stored in the SAF/DQF storage.

Otherwise, UMA means the action when the message fails to be delivered.

The valid Oracle OTMQ UMA options are listed in following table.

Table 1 OTMQ UMA Options

UMA	Description
DISC	Discard - the message is deleted.
RTS	Return to sender - the message is delivered to the sender's response queue.
SAF	Store and forward - the message is written to the message recovery journal on the sender system.
DLQ	Dead letter queue - the message is written to the dead letter queue.
DLJ	Dead letter journal - the message is written to the DLJ.

Message Filter

Messages are read from queues in FIFO order unless another order is defined for the queue. Oracle OTMQ also provides message filter that allow user to read message that matching the selection criteria defined by the message filter. For more information, see Using Filter in the Oracle Tuxedo Message Queue Programming Guide.

Publish/Subscribe

The publish-and-subscribe capability sends a message to multiple recipients registered to receive information from a message broadcasting stream. It is the asynchronous routing of events among the message queuing clients and servers.

Publish/broadcast

Oracle OTMQ application can send a broadcast message using the standard `tpqpublish ()` function. The sending application can generate broadcast messages without knowing the location or number of recipient programs.

Subscribe

Oracle OTMQ application can selectively receive a broadcast message by first subscribing to a broadcast stream. To subscribe to a message stream, the receiving application first use `tpqsubscribe()` to subscribe a message stream. Broadcast messages are then enabled for the application and flow into the receiver's queue for processing using the standard `tpdeqplus()` function.

For more information, see Publish/Subscribe in the Oracle Tuxedo Message Queue Programming Guide.

Naming

Naming is a powerful capability that enables applications to refer to queues by alias instead of by their queue names. Naming separates applications from the details of the current environment configuration and enables system managers to make configuration changes without requiring developers to change their applications.

For more information, see Using Naming in the Oracle Tuxedo Message Queue Programming Guide.

Journal

Oracle OTMQ uses message recovery journal queues to store messages that are designated as recoverable.

The message journal queue on the local queue space is called the store-and-forward (SAF). The message journal queue on the remote queue space is called the destination-queue-file (DQF). If a recoverable message cannot be delivered, it is stored in either the SAF or DQF, and automatically re-sent to the target.

Dead letter queue (DLQ) is a memory-based mechanism for storing undeliverable messages that cannot be stored for automatically recovery. Similar as DLQ, dead letter journal (DLJ) is a disk-based mechanism for storing undeliverable messages that cannot be stored for automatically recovery.

Another auxiliary journal queue is the post confirmation journal (PCJ). Recoverable messages that are successfully confirmed by the receiver can be written into the PCJ for audit.

Oracle OTMQ provides the `tpqreadjrn()` function to dump messages from these journal queues.

For more information, see Using Recoverable Messaging in the Oracle Tuxedo Message Queue Programming Guide.

WS SAF

In WS mode, OTMQ messages that are sent using a recoverable delivery mode are written to the local store-and-forward (SAF) journal file (tmqsaf.jrn), when the connection to the server system is not available.

Note: this WS SAF feature is available only when the WS client is in "Single-context" mode. For more information, see `tpinit(3c)` in the Oracle Tuxedo Function Reference.

Large QSPACE Size Support

For OTMQ, QSPACE can exceed 2G size limitation at 64bit unix platform for both persistent and non-persistent messages.

In tmqadmin, sub-command "`crdl`" will create persistent device for qspace. The maximun value of this command is 2147483647, which is a system value, and the unit is system block size. So, the maximun size of persistent QSpace is 2147483647*block (bytes).

The `tmqadmin->qspacecreate` command has a -n optional, which indicates the size of the area that reserved in shared memory for non-persistent messages for all queues in the queue space. The size may be specified in bytes (b) or blocks (B), where the block size is equivalent to the disk block size.For both "block(B)" and "byte(b)" argument of qspacecreate -n command, the max value is 2147483647. If using block(B) argument, the maximun size for non-persistent messages reserved is 2147483647*block.

Notes: For HPUX ia 64-bit platform, in `hpux_ma.h`, `defined _TMSIGNLESHM` as 1, which means one process creates a shared memory segment and attaches it to its address space. So, if need to create over 2G share memory thru tmqadmin command, set Oracle Tuxedo environment variable `TM_ENGINE_TMSHMSEGSZ` to a larger value, which should less than or equal to the `SHMAX` value that the system defines.

There is another environment variable `TM_QM_NAPTIME` that defines the thread sleeping time in nanosecs. Note that using this variable may expand the time window of the racing condition of queue dead lock.

Automatic Multi-Context Support

The Oracle Tuxedo API `tpinit()` has two modes of operation: single-context mode and multicontext mode. For OTMQ, if an application calls `tpinit()` explicitly, OTMQ *does not* go into multi-context mode automatically.

If application does no tcall `tpinit()`, then following calling of `tpqattach()` OTMQ goes into mulitcontext mode automatically. For WS client, the automation of multicontext mode association is not available.

Oracle OTMQ System Components

Oracle OTMQ provides following system-level components for queuing services:

- `TuxMsgQ()`

  The Oracle OTMQ Message Queue Manager Server. It does the actual message enqueue/dequeue operations on behalf of the processes that request its queuing service.

- `TuxMQFWD()`

  The Oracle OTMQ Message Forwarding Server. It is also known as the Oracle OTMQ Offline Trade Driver, which can forward messages from Journal to target queues automatically, when the original queuing operation was chosen to use recoverable delivery mode, or the operation was failed and the message was put into Journal for re-delivery.

To support the queuing service, Oracle OTMQ provides a new Transaction Management Server TMS_TMQM as the X/OPEN XA-compliant resource manager interface to manage the Oracle OTMQ queue space.

- `TMS_TMQM()`, is the TMS server for the Oracle OTMQ resource manager. It must be configured in the server group for `TuxMsgQ()` and `TuxMQFWD()`.

Besides the basic queuing services, Oracle OTMQ also provides supplemental features for customer. To support these features, Oracle OTMQ has following system level component:

- `TMQ_NA()`, is the Oracle OTMQ Naming Server. It provides naming service which allows runtime queue alias binding and lookup.

- `TMQEVT()`, is the Oracle OTMQ Event Broker, which is used to notify subscriber when topics are published using `tpqpublish()`. The subscriber uses `tpqsubscribe(3c)` to subscribe a topic.

- `TMQFORWARDPLUS()`, is the message forwarding server. It forwards messages that have been stored using `tpenqueue(3c)/tpenqplus(3c)` for later processing. The application administrator enables automated message processing for the application servers by specifying this server as an application server in the `*SERVERS` section.

Oracle OTMQ can also interoperate with another traditional message queue product: Oracle Message Queue (OMQ). To achieve the message level interoperability, the Oracle OTMQ provides following two system level components:

- `TuxMsgQLD()`, is the Oracle OTMQ Link Driver Server. It is the bridge for Oracle OTMQ to interoperate with another messaging product, Oracle Message Queue (OMQ). For more information, see Interoperablity in the Oracle Tuxedo Message Queue Release Notes.

- `TuxCls()`, is the Oracle OTMQ Client Library Server. It acts as a remote agent to perform queuing operations base on Oracle OTMQ queue space and queues on behalf of the

existing Oracle Message Queue clients. For more information, see Interoperablity in the Oracle Tuxedo Message Queue Release Notes.

See Also

- Oracle Tuxedo Message Queue Installation Guide
- Oracle Tuxedo Message Queue Administration Guide
- Oracle Tuxedo Message Queue Programming Guide
- Oracle Tuxedo Message Queue Reference