Oracle® Tuxedo

Using Oracle Tuxedo Advanced Performance Pack

Release 22c

F79942-02

Oracle Tuxedo Using Oracle Tuxedo Advanced Performance Pack, Release 22c

F79942-02

Primary Author: Preeti Gandhe

Contributing Authors: Tulika Das

Contributors: Maggie Li

Contents

List of Figures

List of Tables

Preface

This document describes about how to configure various features of Oracle Tuxedo Advanced Performance Pack.

About this Guide

This document introduces all features in Oracle Tuxedo Advanced Performance Pack. With this document, you can learn about how to configure these features and run it with your existing application.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1

Introduction to Using Oracle Tuxedo Advanced Performance Pack

This chapter contains the following topics:

1.1 Overview

This section contains the following topics.

1.1.1 About Oracle Tuxedo Advanced Performance Pack

Oracle Tuxedo Advanced Performance Pack is a product option introduced in Tuxedo 12c Release 2 (12.1.3). With this pack, Oracle Tuxedo applications can achieve significantly better application performance and improve application availability, especially when running with Oracle Database/RAC. Features in this pack can be run on all Oracle Tuxedo supported platforms:

- IBM AIX (64-bit) on IBM PowerPC
- HP-UX (64-bit) on Itanium
- Oracle Solaris (64-bit) on SPARC
- Linux x86-64
- Microsoft Windows 64-bit

1.1.2 Features in Oracle Tuxedo Advanced Performance Pack

Oracle Tuxedo Advanced Performance Pack provides the following features:

Table 1-1 Features in Oracle Tuxedo Advanced Performance Pack

Features	Can be used with Oracle RAC only?
Self-Tuning Lock Mechanism	No
Shared Memory Interprocess Communication	No
Tightly Coupled Transactions Spanning Domains	No
Concurrent Global Transaction Table Lock	No
Partial One Phase Read-Only Optimization for RAC	Yes
Single Group Multiple Branches (SGMB)	Yes
Common XID	Yes

Table 1-1 (Cont.) Features in Oracle Tuxedo Advanced Performance Pack

Features	Can be used with Oracle RAC only?
XA Transaction Affinity	Yes
Failover/Failback across Database Instances	Yes
Load Balancing across RAC Instances	Yes

- Self-Tuning Lock Mechanism
- Shared Memory Interprocess Communication
- Tightly Coupled Transactions Spanning Domains
- Concurrent Global Transaction Table Lock
- Partial One Phase Read-Only Optimization for RAC
- Single Group Multiple Branches (SGMB)
- Common XID
- XA Transaction Affinity
- Failover/Failback across Database Instances
- Load Balancing across RAC Instances

1.1.2.1 Self-Tuning Lock Mechanism

This feature can adjust the value of `SPINCOUNT` dynamically for the best use of CPU cycle.

The Tuxedo bulletin board (BB) is one or more shared memory segments in which all the application configuration and dynamic processing information are held at run time. For some Tuxedo system operations (such as service name lookups and transactions), the BB must be locked for exclusive access: that is, it must be accessible by only one process. If a process or thread finds that the BB is locked by another process or thread, it retries, or spins on the lock for `SPINCOUNT` number of times (user level method via spin) before giving up and going to sleep on a waiting queue (system level method via system semaphore). Because sleeping is a costly operation, it is efficient to do some amount of spinning before sleeping.

Because the value of the `SPINCOUNT` parameter is application- and system-dependent, the administrator had to tune the `SPINCOUT` to be a proper value manually by observing the application throughput under different values of `SPINCOUNT`.

Self-Tuning Lock Mechanism takes the job of tuning automatically. It is designed to figure out a proper value of `SPINCOUNT` so that most requests to lock BB are completed by spinning instead of sleeping on a waiting queue.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.2 Shared Memory Interprocess Communication

Starting with release 12.1.3, Oracle Tuxedo offers the option to significantly enhance performance of Tuxedo applications with use of shared memory queues instead of IPC

Message Queues for inter process communication on the same Tuxedo node. With the use of shared memory queues, the sender and receiver processes can exchange pre-allocated messages in shared memory, thus eliminating the need to copy messages several times before message reaches its intended target and resulting in much better throughput and lower latency.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.3 Tightly Coupled Transactions Spanning Domains

Because of different global transaction identifiers (GTRIDs) are used in different domains, the transaction crossing domain are loosely coupled even if the branches of the transaction are running on the same database. This feature enables transactions to be performed in remote domains using the same GTRID. The same GTRID allows branches on the remote domain to share locks, enabling tight coupling. The branches will be tightly coupled if they are running on the same database (if the database allows).

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.4 Concurrent Global Transaction Table Lock

Oracle Tuxedo manages global transactions by maintaining a table of active global transactions and their participants in the Oracle Tuxedo bulletin board called the Global Transaction Table or GTT. As this table is accessed by multiple concurrent processes it must be protected with a semaphore. In the normal Oracle Tuxedo case, the bulletin board lock is used to serialize access to this table. However, under heavy transaction load, the contention for this lock can become substantial resulting in an artificial performance bottleneck.

The bottleneck is eliminated by having a separate lock for each GTT entry.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.5 Partial One Phase Read-Only Optimization for RAC

This feature takes advantage of the read-only optimization of RAC for XA. Given two phase commit scenario, Tuxedo reserves one transaction branch and prepares all other branches concurrently. If all other transaction branches are read-only, Tuxedo does one-phase commit on the reserved branch directly without sending the prepare request and writing `TLOG`; otherwise, Tuxedo does two-phase commit on the reserved branch.

Transactions either within or across domains are supported, including global transaction across Tuxedo domain and WLS via WTC (in WLS 12.1.1 - Contact Oracle Support for a patch, or higher release of WLS).

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.6 Single Group Multiple Branches (SGMB)

In previous releases, servers in the same group use the same transaction branch in a global transaction; if these servers connect to different instances on the same RAC, the transaction branch may fail and an XA error, `XAER_AFFINITY`, will be reported, meaning one branch cannot go through different instances. For this reason, Tuxedo groups can only use singleton

RAC services. A DTP service (if the DTP option, `-x` in `srvctl`, is specified) or a service offered by only one instance could be a singleton RAC service.

In this release, this feature eliminates the need to use singleton RAC service when multiple servers in a server group participate in the same global transaction. If servers in the same server group and same global transaction happen to connect to different RAC instances, a different transaction branch is used. Thus, such applications can balance the load across the available RAC instances.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

> **Note:**
>
> The transaction still fails if more than 16 instances are involved in a single group.

1.1.2.7 Common XID

For global transactions, each participating group has its own transaction branch, and a distinguished transaction branch identifier (XID) identifies each branch. If a global transaction involves multiple groups, Tuxedo adopts two-phase commit on each branch, taking the first participating group as the coordinator.

With the common XID (transaction branch identifier) feature in Oracle Tuxedo Advanced Performance Pack, Tuxedo shares the XID of the coordinator group with all other groups within the same global transaction. If a transaction involves multiple RAC databases or resource managers, then separate branches will be utilized to manage the involved resources. This ensures efficient and effective management of the transaction across multiple systems. This is as opposed to each group having its own XID and thus requiring two-phase commit in earlier releases if multiple groups are participating.

Common XID eliminates the need to XA commit operations for groups that connect to the same Oracle RAC instance through the same service by using the coordinator branch directly.

In the cases where all groups in a global transaction use the coordinator branch directly, one-phase commit protocol (instead of two-phase commit protocol) is used, and thus avoids writing `TLOG`.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.8 XA Transaction Affinity

XA Transaction Affinity provides the ability to route all Oracle Database requests within one global transaction to the same Oracle RAC instance when possible; no matter if the requests come from an Oracle Tuxedo application server or Oracle WebLogic Server. This feature does not require the creation of another transaction branch, which results in a reduced amount of processing for transaction commitments.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.9 Failover/Failback across Database Instances

Fast Application Notification (FAN) is a facility provided by Oracle Database to allow database clients to know about changes in the state of the database. These notifications let an application respond proactively to events such as a planned outage of a RAC node or an imbalance in database load. Tuxedo Advanced Performance Pack provides support for FAN notifications by a system server TMFAN that can monitor Oracle RAC instance and notify Tuxedo application server to establish a new Database connection in case of Database instance up or down.

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

1.1.2.10 Load Balancing across RAC Instances

Based on FAN notification, Tuxedo `TMFAN` server can receive load balancing advisories that include the load information of each RAC instance. If the change in advised load exceeds the threshold specified in the `TMFAN` then Tuxedo attempts to balance the connections to the database to match the load information, connecting more servers to the less loaded instance(s).

For more information about configuration, see Introduction to Using Oracle Tuxedo Advanced Performance Pack.

2

Oracle Tuxedo Advanced Performance Pack Configuration

This section describes how to configure various features of Oracle Tuxedo Advanced Performance Pack.

All of the features in this product are enabled if the `OPTIONS` parameter in `RESOURES` in `UBBCONFIG` is set to `XPP`. On Oracle Exalogic and Oracle SPARC SuperCluster platforms, the `OPTIONS` parameter must be set to `EECS`.

Some features require further configuration. Such configuration for each feature is described below. Each of these features can be individually disabled if needed. How to disable features individually is describe in the following sections:

- Self-Tuning Lock Mechanism
- Shared Memory Interprocess Communication
- Tightly Coupled Transactions Spanning Domains
- Concurrent Global Transaction Table Lock
- Partial One Phase Read-Only Optimization for RAC
- Single Group Multiple Branches (SGMB)
- Common XID
- XA Transaction Affinity
- Failover/Failback across Database Instances
- Load Balancing across RAC Instances
- FAN Integration

2.1 Self-Tuning Lock Mechanism

Two other optional attributes are supported in `UBBCONFIG *MACHINES` section.

`SPINTUNING_FACTOR`
The option `SPINTUNING_FACTOR` controls the tuning target. The default value is 100 which is good enough in most scenarios. It can be changed from 1 to 10000 if necessary. A value of 100 indicates that `SPINCOUNT` will stop tuning as long as less than 1 in 100 attempts to lock result in system level method to get the BB lock and there is sufficient idle CPU. If the number of lock attempts resulting in system level method is higher than 1 and there is sufficient idle CPU time, `SPINCOUNT` will be increased.

`SPINTUNING_MINIDLECPU`: Specifies the CPU idle time.
The negative impact of the user level method is the extra cost of the CPU. Too many reties of user level method will cost many CPU time. This option is used to limit the CPU used by the user level method. The Self-Tuning Lock Mechanism will not increase the `SPINCOUNT` when the limitation of `SPINTUNING_MINIDLECPU` is reached even if the tuning target is not met. On the contrary, the `SPINCOUNT` will be decreased when the limitation of `SPINTUNING_MINIDLECPU` is broken no matter the tuning target is met or not. For example, given the value of 20, the Self-Tuning Lock Mechanism will control the idle CPU time not less than %20 during the adjustment. The default value is 20.

> **Note:**
>
> - If not specified, the default values for these attributes are used.
> - The Self-Tuning Lock Mechanism may adjust the `SPINCOUNT` at each scan unit but may need to adjust by several times to achieve the target.

For more information, see UBBCONFIG(5) and UBBCONFIG(5) Additional Information, Example 2 Self-Tuning Lock Mechanism Configuration, in File Formats, Data Descriptions, MIBs, and System Processes Reference.

You can also set the configuration via `TM_MIB`. For more information, see TM_MIB(5) in File Formats, Data Descriptions, MIBs, and System Processes Reference.

The following listing shows a `UBBCONFIG` file example of enabling Self-Tuning Lock Mechanism.

Listing UBBCONFIG File Example of Enabling Self-Tuning Lock Mechanism

```
*RESOURCES

OPTIONS       XPP
...
```

You can disable this feature by specifying the option `NO_SPINTUNING` in the `UBBCONFIG` file.

The following listing shows a UBBCONFIG file example of disabling Self-Tuning Lock Mechanism.

Listing UBBCONFIG File Example of Disabling Self-Tuning Lock Mechanism

```
*RESOURCES
OPTIONS         XPP,NO_SPINTUNING
...
```

2.2 Shared Memory Interprocess Communication

Another optional attribute is provided in *RESOURCES section.

SHMQMAXMEM numeric_value
Specifies the maximum shared memory size (Megabyte) used for message buffers.

For UNIX platforms and Windows platforms, the numeric_value range is from 1 to 2000 inclusive. For all other platforms, the numeric_value range is from 1 to 96000 inclusive. If SHMQMAXMEM is not specified, a recommended minimum value will be used, which is good enough for almost all of the scenarios.

Run tmloadcf -c to get recommended minimum value. For more information, refer to tmloadcf(1) in Section 1 - Oracle Tuxedo Command Reference.

The following listing shows a UBBCONFIG file example of enabling Shared Memory Interprocess Communication.

Listing UBBCONFIG File Example of Enabling Shared Memory Interprocess Communication

```
*RESOURCES
OPTIONS         XPP
...
```

You can disable this feature by specifying the option NO_SHMQ in the UBBCONFIG file.

The following listing shows a UBBCONFIG file example of disabling Shared Memory Interprocess Communication.

Listing UBBCONFIG File Example of Disabling Shared Memory Interprocess Communication

```
*RESOURCES
OPTIONS         XPP,NO_SHMQ
...
```

2.3 Tightly Coupled Transactions Spanning Domains

The Oracle Tuxedo Advanced Performance Pack includes this feature by default and cannot be disabled.

2.4 Concurrent Global Transaction Table Lock

The Oracle Tuxedo Advanced Performance Pack includes this feature by default and cannot be disabled.

Listing Configuration Example of Enabling Concurrent Global Transaction Table Lock by Default

```
*RESOURCES
OPTIONS       XPP
```

2.5 Partial One Phase Read-Only Optimization for RAC

The following listing shows a `UBBCONFIG` file example of enabling Partial One Phase Read-Only Optimization for RAC.

Listing UBBCONFIG File Example of Enabling Partial One Phase Read-Only Optimization for RAC

```
*RESOURCES
OPTIONS         XPP
...
```

You can disable this feature by specifying the option `NO_RDONLY1PC` in the `UBBCONFIG` file.

The following listing shows a `UBBCONFIG` file example of disabling Partial One Phase Read-Only Optimization for RAC.

Listing UBBCONFIG File Example of Disabling Partial One Phase Read-Only Optimization for RAC

```
*RESOURCES
OPTIONS         XPP,NO_RDONLY1PC
...
```

You can also get/change the configuration via `TM_MIB`. For more information, see TM_MIB(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

2.6 Single Group Multiple Branches (SGMB)

The following listing shows a `UBBCONFIG` file example of enabling SGMB.

Listing Configuration Example of Enabling SGMB by Default

```
*RESOURCES
OPTIONS         XPP
```

You can disable this feature by specifying the option SINGLETON of RMOPTIONS in the UBBCONFIG file.

```
RMOPTIONS {[...|SINGLETON],*}
```

> **Note:**
>
> This option indicates all RAC services being used in the domain are singleton.

The following listing shows a UBBCONFIG file example of disabling SGMB.

Listing Configuration Example of Disabling SGMB Explicitly

```
*RESOURCES
OPTIONS         XPP
RMOPTIONS    SINGLETON
```

You can also set this flag when Tuxedo application is inactive through T_DOMAIN class in TM_MIB. For more information, see TM_MIB(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

2.7 Common XID

The following listing shows a UBBCONFIG file example of enabling Common XID.

Listing Configuration Example of Enabling Common XID by Default

```
*RESOURCES
OPTIONS         XPP
```

You can disable this feature by specifying the option NO_COMMONXID of RMOPTIONS in the UBBCONFIG file.

```
RMOPTIONS {[...|NO_COMMONXID],*}
```

The following listing shows a UBBCONFIG file example of disabling Common XID.

Listing Configuration Example of Disabling Common XID Explicitly

```
*RESOURCES
OPTIONS         XPP
RMOPTIONS       NO_COMMONXID
```

You can also set this flag when Tuxedo application is inactive through T_DOMAIN class in TM_MIB. For more information, see TM_MIB(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

2.8 XA Transaction Affinity

The following listing shows a UBBCONFIG file example of enabling XA Transaction Affinity.

Listing Configuration Example of Enabling XA Transaction Affinity by Default

```
*RESOURCES
OPTIONS         XPP
```

You can disable this feature by specifying the option NO_XAAFFINITY of RMOPTIONS in the UBBCONFIG file.

```
RMOPTIONS {[...|NO_XAAFFINITY],*}
```

The following listing shows a UBBCONFIG file example of disabling XA Transaction Affinity.

Listing Configuration Example of Disabling XA Transaction Affinity Explicitly

```
*RESOURCES
OPTIONS         XPP
RMOPTIONS     NO_XAAFFINITY
```

You can also set this flag when Tuxedo application is inactive through T_DOMAIN class in TM_MIB. For more information, see TM_MIB(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

2.9 Failover/Failback across Database Instances

This feature is implemented using FAN technology. See FAN Integration to enable this technology.

2.10 Load Balancing across RAC Instances

This feature is implemented using FAN technology. See FAN Integration to enable this technology.

2.11 FAN Integration

The following listing shows a UBBCONFIG file example of enabling FAN Integration.

Listing Configuration Example of Enabling FAN by Default

```
*RESOURCES
OPTIONS     XPP
```

You can disable this feature by specifying the option NO_FAN of RMOPTIONS in the UBBCONFIG file.

```
RMOPTIONS {[...|NO_FAN],*}
```

The following listing shows a UBBCONFIG file example of disabling FAN Integration.

Listing Configuration Example of Disabling FAN Explicitly

```
*RESOURCES
OPTIONS                 XPP
RMOPTIONS       NO_FAN
```

You can also set this flag when Tuxedo application is inactive through T_DOMAIN class in TM_MIB. For more information, see TM_MIB(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

TMFAN must be configured for Tuxedo to support FAN, specify Tuxedo system server TMFAN in SERVERS section. For more information, see TMFAN(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

To support Oracle TAF (Transparent Application Failover) for Tuxedo XA server, threads=t must be included in OPENINFO in UBBCONFIG *GROUPS section.

3
Best Practices to Optimize Performance

This section contains the following topics:

3.1 Self-Tuning Lock Mechanism

3.1.1 Scenarios Recommended

A proper `SPINCOUNT` indicates a server can hold the BB lock via user level method at most time. It can significantly improve the performance in the scenarios where BB lock conflict is heavy. The typical scenario is the transactional application using Tuxedo XA mechanism. So it is recommended to enable this feature on the Oracle Exalogic by default in a Tuxedo application unless the CPU is not enough.

3.1.2 Setting the Number of Lock Spins

A process or thread locks the bulletin board through user level method or system level method. Because system level method is a costly operation, it is efficient to set a proper number of lock spins so that most lock attempts are achieved through user level method.

A process on a uniprocessor system should not spin. A `SPINCOUNT` value of 1 is appropriate for uniprocessors. On multiprocessors, the value of the `SPINCOUNT` parameter is application- and system-dependent. Self-Tuning Lock Mechanism can figure out the proper `SPINCOUNT` automatically.

> **See Also:**
>
> - Oracle Tuxedo Advanced Performance Pack Configuration

3.2 Shared Memory Interprocess Communication

3.2.1 Scenarios Recommended

SHMQ helps you to gain higher performance in the native Tuxedo application by reducing unnecessary message copy. It can be considered to enable this feature when one or more cases are met in the following list:

- big request/reply message
- simple/little service
  If the service routine itself does not spend too much time, the performance should be improved with this feature by reducing the message copy.
- IPC resource
  Not only more shared memory, but also extra semaphores are necessary if this features enabled. It is recommended to check the minimum IPC resources via `tmloacf -c` before using this feature.

3.2.2 Adjust SHMQMAXMEM

The default value is good enough for almost all scenarios. But, you require adjust the value of `SHMQMAXMEM` in `UBBCONFIG` if the message size is greater than 32 Kbytes, the detail is as follow:

- `SHMQMAXMEM = (Recommend_value * Message_size) / 32`
- `Recommend_value`: The value returned by `tmloadcf -c`
- `Message_size`: The buffer size for one message (Kbytes)

3.2.3 Memory Usage

Given a specific shared memory used by the SHMQ, the Tuxedo would divide it into several parts for different sized buffers. In general, the bigger the buffer size is, the less the total entries for this kind of buffer are. If some sized buffer is too much, the Tuxedo will convert to use local memory although the whole shared memory for SHMQ is not full.

In this release, there are two new MIB fields, `TA_SHMQSTAT` and `TA_MSG_SHMQNUM`, which are used to get the detailed information about shared memory usage. For more details about `TA_SHMQSTAT` and `TA_MSG_SHMQNUM`, see TM_MIB(5) in Section 5 - File Formats, Data Descriptions, MIBs, and System Processes Reference.

3.2.4 Programming with SHMQ

It is a new flag of `TPNOCOPY` in `tpcall()` for using SHMQ message. A typical Tuxedo user case of zero-copy messaging:

1. Client gets request `SHMMSG` buffer by `tpalloc()`
2. Client sends the request to server's request SHMQ by `tpcall()`, and waits for reply
3. Server receives the request from its request SHMQ, processes the request
4. Server use the same `SHMMSG` buffer for reply
5. Server sends the reply to client's reply SHMQ by `tpreturn()`
6. Client receives the reply from its reply SHMQ

Zero-copy messaging is an ideal circumstance, with the pre-condition that sender and receiver cannot access the shared buffer at the same time. In the real world, to guarantee safe memory access, sender needs to do one copy and send the copy instead of original `SHMMSG`. However, to gain advanced performance, new flag `TPNOCOPY` is provided for `tpcall()` to avoid the copy cost. If application chooses to use this flag, it must take the responsibility to make sure no access to the `SHMMSG` buffer after `tpcall()` fails, except for `tpfree()`.

When `TPNOCOPY` is set for `tpcall()` flags and the send buffer is SHMMSG buffer, no safe copy will be done during message sending. After `tpcall()` succeeds, sender application has full access of the send buffer as normal. But if `tpcall()` fails in any circumstance, sender application cannot access the send buffer any more. In this case the recommended action is `tpfree()` the buffer, and this is the only safe operation on the buffer.

`TPNOCOPY` cannot be set for `tpacall()`, or `tpacall()` will fail with `tperrno` set to `TPEINVAL`.

3.2.5 Exceptions

In general, the tuxedo native request/reply messages will be transferred using shared memory queue (SHMQ) if the feature is available. But the IPC queue is used instead in the following cases:

- Encoded Tuxedo message
- Stateful CORBA object
- Tuxedo message associated with digital signature and encryption
  For example, `tpseal()` or `tpsign()` marks the Tuxedo message for encryption or digital signature.
- Tuxedo messages out from a server specified with `BUFTYPECONV`
- Tuxedo FML32 typed message with any field typed pointer
- Tuxedo FML32 typed message with any field typed embedded FML32

> **See Also:**
>
> - Oracle Tuxedo Advanced Performance Pack Configuration

3.3 Partial One Phase Read-Only Optimization for RAC

In general, Tuxedo will only perform a one-phase commit if only one Tuxedo group is participating in a global transaction. If more than one group is involved, Tuxedo will perform a two-phase commit. Two-phase commit indicates the Tuxedo sends a prepare request to each branch of the global transaction followed by a commit request to all non-read-only branches if all prepare requests are successful. When using RAC, if there is more than one RAC instance involved in the transaction, all prepare requests except the last will return read-only. Tuxedo uses this knowledge to try and improve the performance of the transaction by waiting to prepare one branch until all other RAC branches have been prepared. If all other branches have returned read-only, Tuxedo will then send a one-phase commit to the remaining branch. This saves a prepare call and the need to write a TLOG entry. While this may slightly increase the response time of the transaction, it will reduce the load and likely increase the throughput.

> **See Also:**
>
> For more information, Partial One Phase Read-Only Optimization for RAC.

If the Tuxedo application is using a RAC Database, the customer can take advantage of this feature when the application involves multiple groups. In addition, the branches must be tightly coupled for Oracle Database which is a default property of the `OPENINFO`.

The typical scenario is that the participated groups connect to different RAC instances or use different database service. A typical UBB configuration is as below.

Listing UBB Configuration

```
*RESROUCE
MODEL          SHM
OPTIONS        XPP
...

* MACHINES
"mach1"       LMID=L1
...

*GROUPS
GRP1          LMID=L1 GRPNO=10 TMSNAME="TMSORA1"
              OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=P/
scott/tiger +SesTM=120"
GRP2          LMID=L1 GRPNO=20 TMSNAME="TMSORA2"
              OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux2+ACC=P/
scott/tiger +SesTM=120"

*SERVERS
server1        SRVGRP=GRP1 SRVID=10 MIN=2
server2        SRVGRP=GRP2 SRVID=10 MIN=2
...
```

Tuxedo `GRP1` and `GRP2` use different net services to connect to the same Oracle RAC database, with each net service using a different database service offered by a separate RAC instance. Tuxedo server1 and server2 offer different Tuxedo services. A transactional business requires services from `svc 1` and `svc2`. Therefore, it involves servers 1 and 2.

Enabling Read-only Optimization saves requests by ignoring TLOG writing. Perform a one-phase commit to efficiently commit the remaining branch.

Enabling the Common XID feature is recommended if the participating groups are connected to the same Oracle RAC instance and RAC database. This feature ensures that the global transaction is committed in a single phase. Common XID feature improves performance by ignoring prepare requests and TLOG writing, better than Read-only Optimization.

If read-only optimization is not used in the transactional business, do not enable it to avoid unfavorably impacting response time. In a typical scenario, it is common to use multiple resource managers simultaneously.

See Also:

- Oracle Tuxedo Advanced Performance Pack Configuration

3.4 Single Group Multiple Branches (SGMB)

3.4.1 Scenarios Recommended

If a Tuxedo application is running on the Oracle RAC, you may want to take advantage of non-singleton database service, such as load balance, service failover, and so on. Tuxedo groups can use RAC non-singleton service by enabling this feature. Given that the business may involve multiple groups, it is better to also enable Common XID and XA Transaction Affinity to achieve good performance.

A typical UBB configuration is as follows.

Listing UBB Configuration

```
*RESROUCES
MODEL          SHM
OPTIONS        XPP
...

*MACHINES
"mach1"       LMID=L1
...

*GROUPS
GRP1         LMID=L1 GRPNO=10 TMSNAME="TMSORA1"
             OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux3+ACC=
P/scott/tiger +SesTM=120"
```

```
GRP2        LMID=L1 GRPNO=20 TMSNAME="TMSORA2"
            OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux3+ACC=
P/scott/tiger +SesTM=120"

*SERVERS
server1     SRVGRP=GRP1 SRVID=10 MIN=4
server2     SRVGRP=GRP2 SRVID=10 MIN=4
...
```

`GRP1` and `GRP2` use the same net service `orcl.tux3` to connect the resource manager; `orcl.tux3` is configured to database service `tux3`, which both RAC instance1 and instance2 support. `Server1` offers Tuxedo service `svc1` and `server2` offers Tuxedo service `svc2`. The transactional business A calls `svc1` and then `svc2`, and so involves both `server1` and `server2`. Because `orcl.tux3` is non-singleton database service, `server1` copies associate with either instance1 or instance2, so do `server2` copies.

SGMB can ensure business working well and ensure that business A transactions are distributed evenly on instance1 and instanc2.

Given that both Common XID and XA Transaction Affinity are enabled, all business A transactions become one-phase commit.

3.4.2 Limitations

- Groups with multiple resource managers are not supported.
- A transaction fails if more than 16 instances are involved in a single group.
- Partial One Phase Read-Only Optimization for RAC does not work in a transaction if the preferred reserved group is a multi-branch group. If GWTDOMAIN is not the coordinator, the preferred reserved group is the coordinator group; otherwise, the preferred reserved group is the participated group coming next in the coordinator domain.
- Multi-threaded servers do not provide instance information via MIB; however, SGMB still performs well on server-dispatched threads.

> **See Also:**
>
> - Oracle Tuxedo Advanced Performance Pack Configuration

3.5 Common XID

Common XID shares the coordinator's instance information and branch (common XID) to all participated groups. The servers in a participated group will reuse the common XID if they have the same instance information as the coordinator does. This feature brings significant performance improvement when a global transaction involves multiple groups, especially when all participated groups associate with the same database instance through the same database service.

- Scenarios Recommended

- Limitations

3.5.1 Scenarios Recommended

- Scenario A
- Scenario B
- Scenario C
- Scenario D

3.5.1.1 Scenario A

Only one Oracle Database instance is used in a Tuxedo application. A typical UBB configuration is as follows.

Listing 18 UBB Configuration

```
*RESROUCES
MODEL       SHM
OPTIONS     XPP
...

*MACHINES
"mach1"     LMID=L1
...

*GROUPS
GRP1        LMID=L1 GRPNO=10 TMSNAME="TMSORA1"
            OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=
P/scott/tiger +SesTM=120"
GRP2        LMID=L1 GRPNO=20 TMSNAME="TMSORA2"
            OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=
P/scott/tiger +SesTM=120"

*SERVERS
server1     SRVGRP=GRP1 SRVID=10 MIN=2
server2     SRVGRP=GRP2 SRVID=10 MIN=2
...
```

In the above configuration, `GRP1` and `GRP2` use the same net service (`orcl.tux1`, which is configured to an Oracle Database) to connect resource manager. `Server1` offers Tuxedo service `svc1` and `server2` offers Tuxedo service `svc2`. The transactional business A calls `svc1` followed by `svc2` so it will involve `server1` and `server2`. When Common XID is enabled, all transactions of business A become one-phase commit.

3.5.1.2 Scenario B

All participated groups associate with the same database instance via the same database service when Tuxedo application is running on Oracle RAC.

The typical UBB sample is the same as the Listing UBB Configuration (refer to this listing as described in Scenario A), while the net service `orcl.tux1` is configured to Oracle RAC

instance1 through database service `tux1`. When Common XID is enabled, all transactions of business A become one-phase commit.

3.5.1.3 Scenario C

Redundant servers or groups are configured when they are running on different Oracle RAC instances. Given this scenario, the XA Transaction Affinity feature should be enabled too. It helps the business involves the servers/groups that associate same database instance via same database service with the coordinator.

Listing 19 UBB Configuration

```
*RESROUCES
MODEL       SHM
OPTIONS     XPP
...

*MACHINES
"mach1"     LMID=L1
...

*GROUPS
GRP1        LMID=L1 GRPNO=10 TMSNAME="TMSORA1"
            OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=
P/scott/tiger +SesTM=120"
GRP2        LMID=L1 GRPNO=20 TMSNAME="TMSORA2"
            OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=
P/scott/tiger +SesTM=120"
GRP3        LMID=L1 GRPNO=30 TMSNAME="TMSORA3"
            OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux2+ACC=
P/scott/tiger +SesTM=120"

*SERVERS
server1     SRVGRP=GRP1 SRVID=10 MIN=2
server2     SRVGRP=GRP2 SRVID=10 MIN=2
server3     SRVGRP=GRP3 SRVID=10 MIN=2
...
```

`GRP1` and `GRP2` use the same net service `orcl.tux1` to connect the resource manager; `orcl.tux1` is configured to database service `tux1`, which RAC instance1 supports. `GRP3` uses net service `orcl.tux2` to connect the resource manager; `orcl.tux2` is configured to database service `tux2`, which RAC instance2 supports. `Server1` offers Tuxedo service `svc1`; both `server2` and `server3` offer Tuxedo service `svc2`. The transactional business A calls `svc1` and then `svc2`.

In general, business A may involve `server1` and `server2`, or `server1` and `server3`, because of Tuxedo load balance. When Common XID is enabled, the transactions that involve `server1` and `server2` become one-phase commit; when XA Transaction Affinity feature is enabled, business A always involves `server1` and `server2` so that all transactions of the business A would be one-phase commit.

3.5.1.4 Scenario D

Part of participated groups associate with the same instances through the same database service with the coordinator. In this scenario, it is better to enable both common XID and Read-Only Optimization features.

A typical UBB configuration is as follows.

Listing 20 UBB Configuration

```
*RESROUCES
MODEL           SHM
OPTIONS         XPP
...

*MACHINES
"mach1"         LMID=L1
...

*GROUPS
GRP1            LMID=L1 GRPNO=10 TMSNAME="TMSORA1"
                OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=
P/scott/tiger +SesTM=120"

GRP2            LMID=L1 GRPNO=20 TMSNAME="TMSORA2"
                OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux1+ACC=
P/scott/tiger +SesTM=120"

GRP3            LMID=L1 GRPNO=30 TMSNAME="TMSORA3"
                OPENINFO="Oracle_XA:ORACLE_XA+SqlNet=orcl.tux2+ACC=
P/scott/tiger +SesTM=120"

*SERVERS
server1         SRVGRP=GRP1 SRVID=10 MIN=2
server2         SRVGRP=GRP2 SRVID=10 MIN=2
server3         SRVGRP=GRP3 SRVID=10 MIN=2
...
```

`GRP1` and `GRP2` use the same net service `orcl.tux1` to connect the resource manager; `orcl.tux1` is configured to database service `tux1`, which RAC instance1 supports. `GRP3` uses net service `orcl.tux2` to connect the resource manager; `orcl.tux2` is configured to database service `tux2`, which RAC instance2 supports. `Server1` offers Tuxedo service `svc1`; `server2` offers Tuxedo service `svc2`; `server3` offers Tuxedo service `svc3`. The transactional business B calls `svc1`, then `svc2`, and then `svc3`.

The business B involves `server1/GRP1`, `server2/GRP2`, and `server3/GRP3`. When common XID is enabled, the prepare request to `GRP2` is saved. Given that Read-Only Optimization is enabled as well, the prepare request to `GRP1` is saved as well and one-phase commit is done on `GRP1`, avoiding TLOG writing.

3.5.2 Limitations

- Groups with multiple resource managers are not supported.

- Multi-threaded servers do not provide instance information via `MIB`; however, common XID still performs well on server-dispatched threads.
- In two-phase commit scenarios, `GWTDOMAIN` is always involved to do prepare and/or commit

> **See Also:**
>
> - For detailed information about configuration, see Oracle Tuxedo Advanced Performance Pack Configuration.
> - For more information about how to set up this feature with Oracle Database, see Using Tuxedo with Oracle Real Application Clusters (RAC) in *Oracle Tuxedo Setting Up an Oracle Tuxedo Application*.

3.6 XA Transaction Affinity

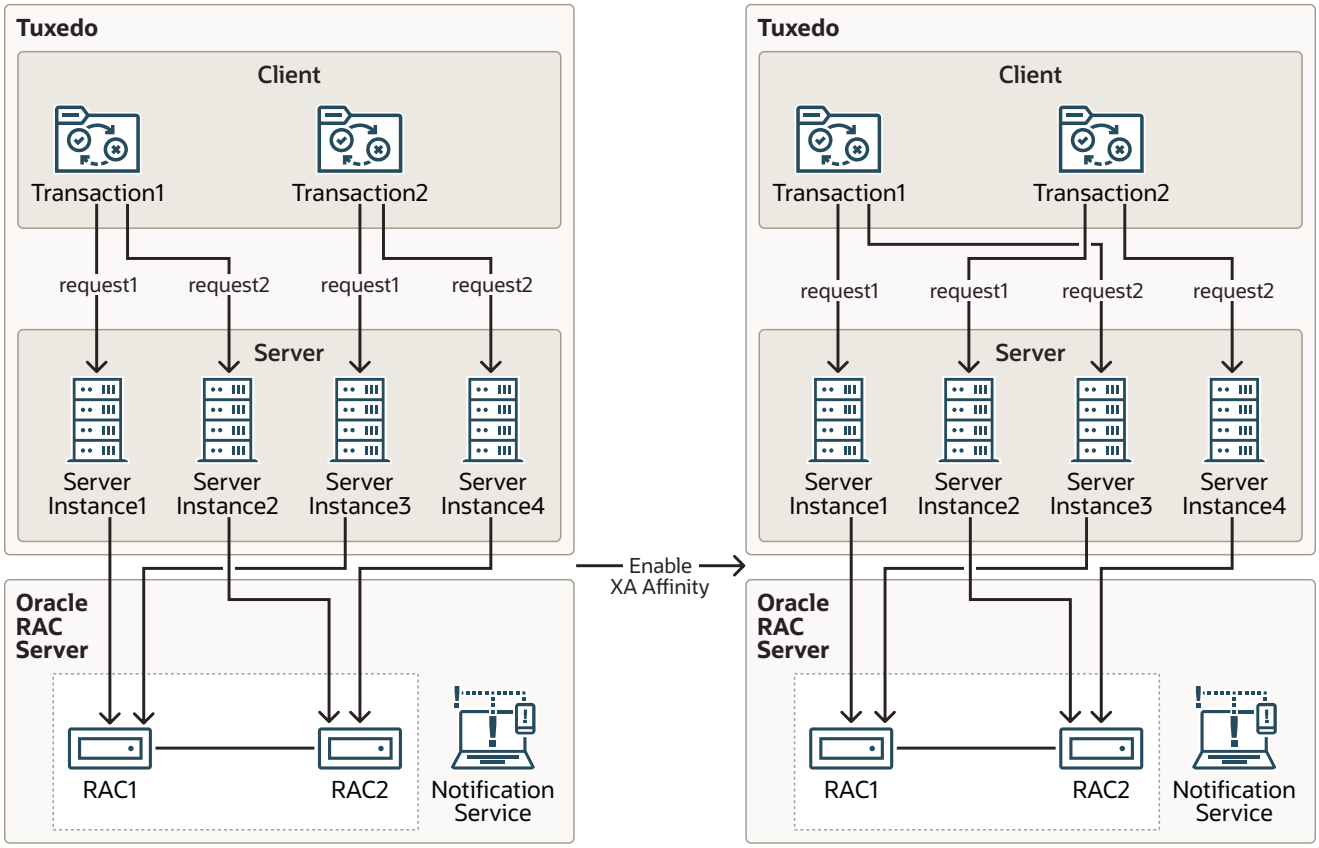
3.6.1 Scenarios Recommended

It is recommended to enable this feature when Tuxedo server has multiple instances running on different Oracle RAC instances via the same Oracle Database service.

In the event that XA Transaction Affinity is enabled, there is no need to utilize the rule of Oracle RAC routing specified by the environment variable `TUXRACGROUPS`, and this rule will be disabled.

The following picture illustrates the changes that will be made when XA Transaction Affinity is enabled.

Figure 3-1 XA Transaction Affinity Enablement



3.6.2 Limitations

- Groups with multiple resource managers are not supported.
- The max number of affinity context (database name+instance name+service name) in one transaction is 16.
- XA Transaction Affinity does not support multi-server single queue, multi-threaded server, or cross-domain services.

> **See Also:**
>
> - Oracle Tuxedo Advanced Performance Pack Configuration

3.7 Failover/Failback across Database Instances

3.7.1 Recommendation for Configuration on Oracle Database

To benefit from Oracle FAN (Fast Application Notification), it is recommended to enable this feature anytime when Tuxedo works with Oracle RAC.

Besides `UBBCONFIG`, set Oracle Database properly for the following configurations:

- ONS (Oracle Notification System)
  This feature depends on ONS (Oracle Notification System) to access FAN events. ONS daemon must be enabled on Oracle Database server side and client side if Tuxedo is taken as a native ONS client. It is recommended that Tuxedo works in remote mode.

  The ONS daemon configuration file is located in `$ORACLE_HOME/opmn/conf/ons.config`. This file tells the ONS daemon how to behave. Configuration information within `ons.config` is defined in simple name and value pairs.
  After configuring ONS, you can start it with `onsctl` command. Make sure that ONS daemon is running all the time.

  > **Note:**
  >
  > On Oracle Database client side, if the Oracle version is lower than 12.1.0.1.0, ONS daemon must be enabled.

- TAF (Transparent Application Failover)
  TAF is recommended when Oracle Tuxedo non-XA server works with Oracle RAC Fast Application Notification (FAN).

  TAF (Transparent Application Failover) is an Oracle Database client-side feature that allows clients to reconnect surviving database instances automatically in the event of database instance failure.

  – If TAF is configured, Oracle client will be responsible for re-establishing the new connection from Oracle Tuxedo to Oracle Database server.

  – If TAF is not configured, Oracle Tuxedo non-XA server will not do the re-establishment and so this feature will not work.

3.7.2 Recommendation for Non-XA Server

To monitor FAN event for the instance associated with the specific non-XA application server, `$TUXDIR/lib/tuxociucb.so.1.0` should be deployed in `$ORACLE_HOME/lib`, and the name of this binary must be specified in `ORA_OCI_UCBPKG` environment variable.

To support TAF, follow the rules as below:

- For OCI application, create OCI environment in `OCI_THREADED` mode.
- For Pro*C application, run pre-compilation with `threads=yes` and use `EXEC SQL ENABLE THREADS` before creating the first executable embedded SQL statement.

`-L` option in `servopts` must be used for a non-XA server to indicate that the server will connect to the Oracle Database. Since the ECID is enabled when `-L` is specified, a

new option `-F` is introduced into `servopts` to close ECID. The usage is `-F noECID`. The example is below.

Listing Example for -L Option

```
*SERVERS
server1
SRVGRP=GRP1 SRVID=1 ClOPT="-L libclntsh.so -F noECID"
```

3.7.3 Limitations

- Groups with multiple resource managers are not supported.
- If the customized server is going to use OCI to connect to Oracle Database, `OCI_NO_UCB` should not be set at OCI initialization time.

> **See Also:**
>
> - For detailed information about configuration, see Oracle Tuxedo Advanced Performance Pack Configuration.
> - For more information about how to set up this feature with Oracle Database, see Using Tuxedo with Oracle Real Application Clusters (RAC) in *Oracle Tuxedo Setting Up an Oracle Tuxedo Application*.

3.8 Load Balancing across RAC Instances

3.8.1 Recommendation for Configuration on Oracle Database

To benefit from Oracle FAN (Fast Application Notification), it is recommended to enable this feature anytime when Tuxedo works with Oracle RAC.

Configure `UBBCONFIG` and Oracle Database the same as Failover/Failback across Database Instances, and set LBA (Load Balance Advisory) as follows.

- LBA (Load Balance Advisory)
  Based on Oracle Database load balance advisory, Tuxedo can distribute service request across Tuxedo application server that is connected to the same Oracle Database service. To enable LBA and publication of FAN load balancing events, the service-level goal for runtime connection load balancing must be specified in Oracle Database service definition. You can use `-B` option to specify the goal via `srvctl` when creating or modifying the service.

3.8.2 Recommendation for Non-XA Server

See Recommendation for Non-XA Server for the recommendation.

3.8.3 Limitations

- Groups with multiple resource managers are not supported.
- If the customized server is going to use OCI to connect to Oracle Database, `OCI_NO_UCB` should not be set at OCI initialization time.
- Load balance based on Oracle RAC LBA does not support multi-server single queue, multi-threaded server, or cross-domain services.

See Also:

- For detailed information about configuration, see Oracle Tuxedo Advanced Performance Pack Configuration.
- For more information about how to set up this feature with Oracle Database, see Configuring Level 1: Basic Application High Availability in *Oracle® Database High Availability Overview and Best Practices*.

4
Software Requirement

Ensure to meet the following software requirements for using features in the Oracle Tuxedo Advanced Performance Pack:

Oracle Tuxedo
Oracle Tuxedo 22*c* Release 1 (22.1.0.0.0) Rolling Patch 001 or later is required.

Oracle Database
For "Failover/Failback across Database Instances" and "Load Balancing across RAC Instances" features, Oracle Database 12.1.0.2 Patch for bug 21462577 or later client is required.

For every other feature, Oracle Database 11.2.0.2.0 or later is required.