联动北方
LANDING

企业级 IT 运维宝典系列

联动北方 出品

目　录

第 8 章 Tuxedo 常用的管理操作

Tuxedo 常用的管理命令其实比较简练，最基本的操作都可基于简洁的字符界面，下面我们从启动和停止 Tuxedo 应用开始逐步展开阐述。

8.1 启停 Tuxedo 应用

在正式操作一个 Tuxedo 应用系统的时候，确认一个完整的操作环境是必须要做的工作，比如类似于 TUXCONFIG 这种关键的配置文件指定，如果在一个主机上有多套 Tuxedo 应用并行，环境变量的失误，也会带来跨域的误操作。

8.1.1 相关应用环境

为了方便起见，一帮把相关的 Tuxedo 应用系统环境变量都放在一个批处理文件中，这样在设置的时候，执行一下这个批处理文件就可以完成。自然，也可以把这些设置放在用户上下文环境中，比如 UNIX 的.profile，或者 Window 的系统面板中环境变量设置。

8.1.1.1 tux.env 环境文件

基本的 Tuxedo 环境变量其实很简单，基本有 Tuxedo 目录，执行目录以及共享库目录这三项基本内容就足够了。但如果涉及到使用 Tuxedo 自带的 WebGui，或者使用 TSAM，所涉及的环境变量就会大大增多，尤其是如果再涉及到 Tuxedo 的 COBOL 语言编程，就会更复杂一些。

如下给出了一个相对完整复杂的示例列表：

```
    TUXDIR=/home/tuxedo/tuxedo11gR1; export TUXDIR
    JAVA_HOME=$TUXDIR/jre; export JAVA_HOME

    PATH=$TUXDIR/bin:$JAVA_HOME/bin:$PATH; export PATH
    JVMLIBS=$JAVA_HOME/lib/i386/server:$JAVA_HOME/jre/bin
    SHLIB_PATH=$TUXDIR/lib:$JVMLIBS:$SHLIB_PATH; export SHLIB_PATH
    LIBPATH=$TUXDIR/lib:$JVMLIBS:$LIBPATH; export LIBPATH
    LD_LIBRARY_PATH=$TUXDIR/lib:$JVMLIBS:$LD_LIBRARY_PATH; export
LD_LIBRARY_PATH
    #SHLIB, LIBPATH, LD_LIBRARY_PATH 针对不同的操作系统，设置一个就可以

    COBCPY=:$TUXDIR/cobinclude; export COBCPY
    COBOPT="-C ANS85 -C ALIGN=8 -C NOIBMCOMP -C TRUNC=ANSI -C OSEXT=cbl";
export COBOPT
    WEBJAVADIR=$TUXDIR/udataobj/webgui/java; export WEBJAVADIR

    LANG=C; export LANG

    APPDIR=/home/landingbj/simpapp; export APPDIR
    TUXCONFIG=$APPDIR/tuxconfig; export TUXCONFIG
```

示例 8-1

安装 Tuxedo 时会自动生成一个环境文件，其中包括 Tuxedo 安装目录、Java 安装目录、PATH、lib 路径(不同的操作系统对应不同的变量名：LD_LIBRARY_PATH 用在 solaris 或 Linux 上，SHLIB_PATH 用在 HP-UX 上，LIBPATH 在 IBM AIX 上使用)；

还有一些是应用配置需要的，如字符集、应用目录、domain 配置文件位置等。

8.1.1.2 UBBCONFIG 配置文件

```
#  (c) 2003 BEA Systems, Inc. All Rights Reserved.
#ident    "@(#) samples/atmi/simpapp/ubbsimpl"

#Skeleton UBBCONFIG file for the Tuxedo Simple Application.
#Replace the <bracketed> items with the appropriate values.

*RESOURCES
IPCKEY          123456

DOMAINID        simpapp
MASTER          landingbj
MAXACCESSERS    11
MAXSERVERS      5
MAXSERVICES     10
MODEL           SHM
LDBAL           N

*MACHINES
DEFAULT:
                APPDIR="/home/landingbj/simpapp"
                TUXCONFIG="/home/landingbj/simpapp/tuxconfig"
                TUXDIR="/home/tuxedo/tuxedo11gR1"
                MAXWSCLIENTS = 10

DEMOSERVER      LMID=landingbj

#Example:
#beatux         LMID=simple

*GROUPS
GROUP1    LMID=landingbj      GRPNO=1      OPENINFO=NONE
GROUP2    LMID=landingbj      GRPNO=2      OPENINFO=NONE

*SERVERS
DEFAULT:
          CLOPT="-A"

simpserv  SRVGRP=GROUP1 SRVID=1

DMADM           SRVGRP=GROUP2 SRVID=1
GWADM           SRVGRP=GROUP2 SRVID=2
GWTDOMAIN SRVGRP=GROUP2 SRVID=3
```

```
WSL SRVGRP=GROUP1 SRVID=2 CLOPT="-A -t -- -n //192.168.0.168:8888
                                -m 2 -M 5 -x 10"

*SERVICES
TOUPPER
```

示例 8-2

需要配置的几个关键点：

- IPCKEY：对应 IPC 资源的键值，每个在同一主机上的配置文件的 IPCKEY 必须保证不同
- DOMAINID：域 ID
- APPDIR：应用目录
- TUXCONFIG：域配置文件路径
- TUXDIR：Tuxedo 安装路径
- LMID：机器逻辑名
- SERVICES：用来定义服务相关属性

8.1.2 启动 Tuxedo 应用

tmboot 常用参数：

- -A 启动所有管理 SERVER
- -l 启动指定逻辑机器的所有 SERVER
- -g 启动某一组的 SERVER
- -i 启动指定 SERVER id 的 SERVER
- -s 启动指定可执行文件名的 SERVER
- -w 快速启动
- -y 启动时不再问是否确定启动
- -e 指定一个命令，如果任何一个 SERVER 启动失败就执行这个命令

8.1.3 停止 Tuxedo 应用

tmshutdown 常用参数：

- -A 关闭所有管理 SERVER
- -l 关闭指定逻辑机器的所有 SERVER
- -g 关闭某一组的 SERVER
- -i 关闭指定 SERVER id 的 SERVER
- -s 关闭指定可执行文件名的 SERVER
- -y 关闭时不再问是否确定关闭

- -e 指定一个命令，如果任何一个 SERVER 关闭失败就执行这个命令
- -w <delay>等待一断时间，然后执行强制关闭 SERVER
- -c 关闭 SERVER 时不管客户端是否有连接

8.2 管理和监控

8.2.1 一般管理监控 tmadmin

8.2.1.1 tmadmin 主要功能

tmadmin 通常的管理监控工作有以下 3 类：

- 查看系统运行状态，一般查看以下信息：
  - 应用
  - 服务
  - 客户端
  - 交易
  - 队列
  - 组
  - 会话
  - 网络
- 动态修改服务或 Tuxedo 系统参数
- 进行启动，关闭，迁移服务进程等管理任务

8.2.1.2 tmadmin 常用操作

- 输出服务进程信息 printserver（简写为 psr）

```
Prog Name      Queue Name   Grp Name      ID RqDone Load Done Current Service
---------      ----------   --------      -- ------ --------- ---------------
BBL            133333       landingbj      0      0         0       (  IDLE )
DMADM          00002.00001  DWGRP          1     12       600       (  IDLE )
simpserv       00001.00001  GROUP1         1      0         0       (  IDLE )
GWADM          00032.00020  GWGRP2        20      0         0       (  IDLE )
GWADM          00031.00020  GWGRP         20      0         0       (  IDLE )
GWTDOMAIN      gw2          GWGRP2        30      0         0       (  IDLE )
GWTDOMAIN      gw1          GWGRP         30      0         0       (  IDLE )
```

示例 8-3

列号及其描述：

1. 服务进程的可执行文件名
2. 服务进程连接的队列名
3. 服务进程所在的组名

4. 服务进程的 id
5. 服务进程已经处理的请求数
6. 服务进程处理的全部请求的 LOAD
7. 服务进程正在处理的服务，若为 IDLE 则服务进程当前是空闲的

- 输出服务信息 printservice（简写为 psc）

```
Service Name Routine Name Prog Name  Grp Name  ID    Machine   # Done Status
------------ ------------ ---------  --------  --    -------   ------ ------
DMADMIN      DMADMIN      DMADM      DWGRP     1     landingbj   0  AVAIL
TOUPPER      TOUPPER      simpserv   GROUP1    1     landingbj   0  AVAIL
TDOM12       GWS          GWADM      GWGRP2    20    landingbj   0  AVAIL
TDOM1        GWS          GWADM      GWGRP     20    landingbj   0  AVAIL
TOLOWER      GWS          GWTDOMAIN  GWGRP2    30    landingbj   0  AVAIL
TMS          GWS          GWTDOMAIN  GWGRP2    30    landingbj   0  AVAIL
TOLOWER      GWS          GWTDOMAIN  GWGRP     30    landingbj   0  AVAIL
TMS          GWS          GWTDOMAIN  GWGRP     30    landingbj   0  AVAIL
```

示例 8-4

列号及其描述：

1. 服务名
2. 服务函数名
3. 提供服务的程序名
4. 服务所在的组名
5. 服务所在的服务进程的 id
6. 提供服务的机器的 LMID
7. 服务已经执行的次数
8. 服务当前的状态

- 输出服务进程的队列信息 printqueue（简写为 pq）

```
Prog Name    Queue Name    # Serve Wk Queued  # Queued  Ave. Len  Machine
---------    ----------    -------  ---------  --------  --------  -------
GWADM        00032.00020         1          -         0         -  landingbj
BBL          133333              1          -         0         -  landingbj
DMADM        00002.00001         1          -         0         -  landingbj
GWADM        00031.00020         1          -         0         -  landingbj
GWTDOMAIN    gw2                 1          -         0         -  landingbj
simpserv     00001.00001         1          -         0         -  landingbj
GWTDOMAIN    gw1                 1          -         0         -  landingbj
```

示例 8-5

列号及其描述：

1. 队列连接的服务进程名
2. 队列名，由 RQADDR 参数指定或 Tuxedo 生成

3. 连接的服务进程数
4. 当前队列的所有请求的 LOAD
5. 当前队列中的请求数
6. 平均队列长度
7. 队列所在机器的 LMID

- 输出客户端信息 printclient（简写为 pclt）

```
LMID        User Name       Client Name    Time    Status  Bgn/Cmmt/Abrt
---------   ---------     -------------  ------  -------  -------------
Landingbj   landingbj       tmadmin       0:17:39  IDLE        0/0/0
```

示例 8-6

列号及其描述：

1. 已经注册的客户端所在机器的 LMID
2. 用户名，由 tpinit()提供的
3. 客户端名，由 tpinit()提供的
4. 客户端连接后经过的时间
5. 客户端状态

- IDLE ：表示客户端目前没有任何未完成的服务请求或会话
- IDLET ：表示客户端启动了一个全局事务，目前空闲
- BUSY ：表示客户端的服务请求或会话正在被处理
- BUSYT ：表示客户端处于全局事务中的服务请求或会话正在被处理

6. 启动/提交/回滚的事务数

- 输出公告板（BB）统计信息 bbstats（简写为 bbs）

```
Current Bulletin Board Status:
Current number of servers: 7
Current number of services: 20
Current number of request queues: 7
Current number of server groups: 5
Current number of interfaces: 0
```

示例 8-7

- 查看节点间通信状态 printnet（简写为 pnw）

```
> pnw SITE12
SITE12  Connected To:  msgs sent    msgs received
        SITE14         61904           62319
        SITE13         61890           62288
        SITE11         15972           13564
```

示例 8-8

- default

设置默认管理方案

例如：

```
default  -m site1
```

示例 8-9

表示接下来管理时只输出逻辑机器 site1 的信息

- quit（简写为 q）

退出管理界面

- help

显示帮助

8.2.2 域管理监控 dmadmin

8.2.2.1 知识回顾

1. 域配置文件

   文本文件：tuxedo.dm， 二进制文件：bdmconfig，环境变量：BDMCONFIG

   $dmloadcf tuxedo.dm //生成二进制文件

   $dmunloadcf >tuxedo.dm.bak //反向生成文本文件

2. 域网关服务器

   GWTDOMAIN 等服务器进程

3. 域管理服务器

   网关管理（启动时在 DMADM 进行注册，并定期向其汇报自己状态）：GWADM

   域管理服务器（提供 GWADM 注册服务）：DMADM

4. 域管理工具

   dmloadcf：生成二进制配置文件。运行时检查$TUXDIR/udataobj/DMTYPE 文件，查看域类型是否正确。

   Dmunloadcf：将当前域使用的配置导出到文本文件。

   dmadmin：域管理监控工具。

8.2.2.2 dmadmin 常用操作

- 查看域连接状态 printdomain（简写为 pd）

```
$dmadmin
dmadmin - Copyright (c) 1996-1999 BEA Systems, Inc.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by BEA Systems, Inc.
Tuxedo is a registered trademark.
> pd -d LDOM0 //显示域连接状态
Local domain :LDOM0
```

```
Connected domains:
Domainid: trade02

Disconnected domains being retried:
Domainid: trade03
```

示例 8-10

- 手工连接一个域连接 connect（简写为 co）

```
> co -d LDOM0 -R RDOM1
Operation completed successfully. Use printdomain(pd) to obtain results.
```

示例 8-11

- 手工断开一个域连接 disconnect（简写为 dco）

```
> dco -d LDOM0 -R RDOM1
Operation completed successfully. Use printdomain(pd) to obtain results.
```

示例 8-12

8.2.3 队列管理监控 qmadmin

8.2.3.1 知识回顾

/Q 是 Tuxedo 系统的一个重要组成部分，它提供了一种可靠队列机制，允许消息按某种排队规则存储到磁盘上或内存中，然后再转发给其它进程。这种存储转发机制可以保证在两个通信实体之间传递的消息不丢失、不重传，从而保证交易的完整性和可靠性。Tuxedo /Q 提供管理工具和编程接口用于对 /Q 进行管理和操作。

1. /Q 的组成

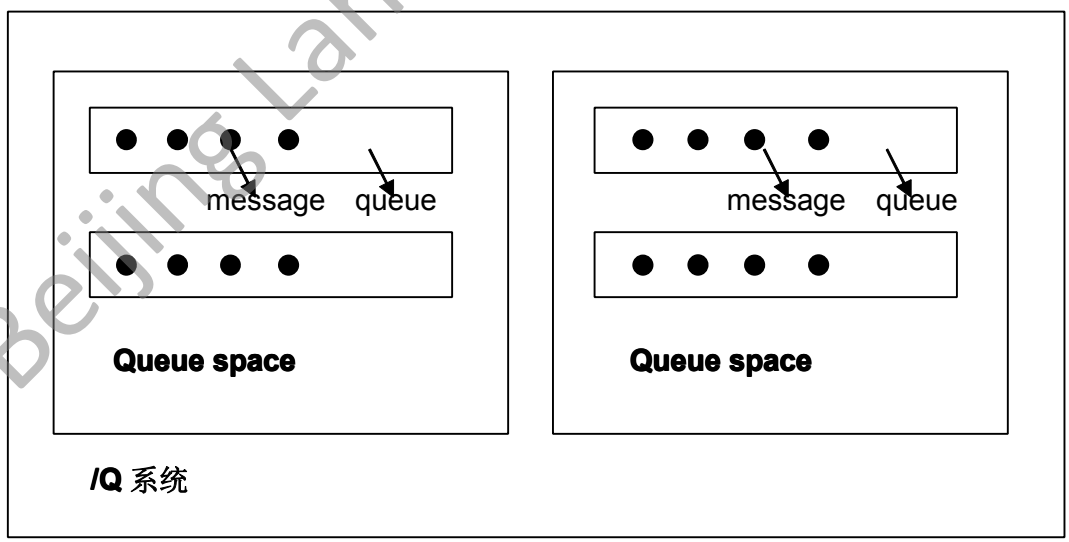

图 8-1

2. /Q 的使用方式

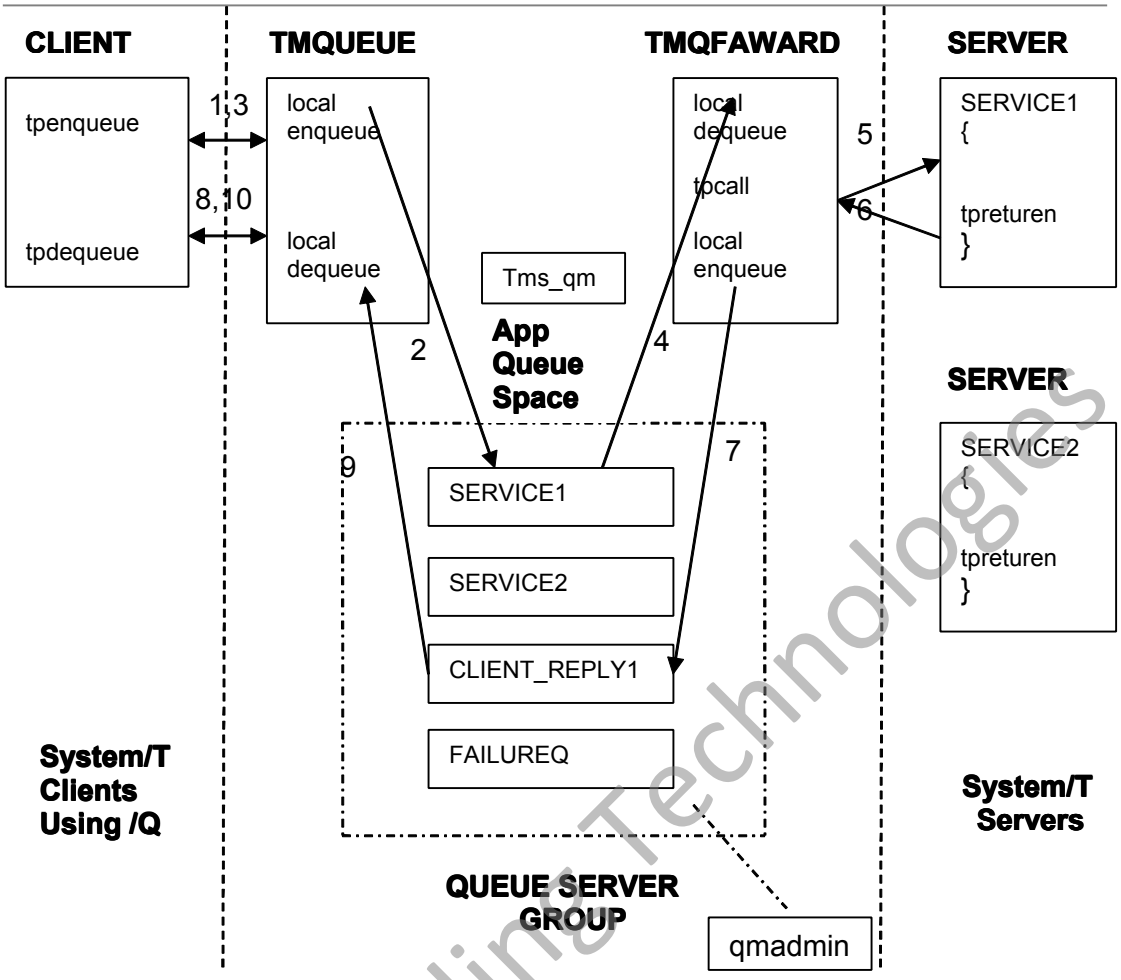

图 8-2

图中左侧为 Tuxedo 客户端，右侧为 Tuxedo 服务进程，中间为 Tuxedo 可靠消息队列系统/Q。右侧服务进程提供了两个服务：SERVICE1 和 SERVICE2。在消息队列系统/Q 中，一个 QUEUE SPACE 对应一个 GROUP，TMS_QM 是/Q 的事务管理进程，在该 GROUP 中要进行定义，这个 QUEUE SPACE 中定义了有四个消息队列，SERVICE1 和 SERVICE2 分别是对应于同名的两个服务（SERVICE）的队列（这是一种命名规则，客户机若请求服务器中的 SERVICE1 服务，就把请求消息放入 SERVICE1 队列中）。SERVICE1 的处理结果放到 CLIENT_REPLY1 中，如果 SERVICE1,SERVICE2 在处理过程中发生错误，把错误信息保存到队列 FAILUREQ 中。

/Q 有两种使用方式，我们称之为基本模式和转发模式

1) 基本模式：

只用到 TMQUEUE，不使用 TMQFORWARD，不调用 SERVICE 对 QUEUE 中的消息进行处理。

流程说明如下(在上面的例子中，不需要进行第 4-7 中的操作,也不需要定义与 SERVICE 同名的 QUEUE)：

- 客户端调用 tpenqueue()把数据发送到 SERVICE1 队列
- TMQUEUE 接收 tpenqueue()发送来的数据,并把他们保存到 SERVICE1 队列中
- 如果以上操作成功,那么 tpenqueue()返回成功.

- 客户端调用 tpdequeue(),请求从 SERVICE1 队列中取数据
- TMQUEUE 收到该请求，它把相应的消息从 SERVICE1 队列中取出，并发送给客户端

2) 转发模式：

用到 TMQUEUE 和 TMQFORWARD，要定义与 SERVICE 同名的 QUEUE，并调用 SERVICE 对 QUEUE 中的消息进行处理。

流程说明如下：

- 客户端调用 tpenqueue()把消息发送到 SERVICE1 队列
- TMQUEUE 接收 tpenqueue()发送来的消息,并把数据保存到 SERVICE1 队列中
- 如果以上操作成功,那么 tpenqueue()返回成功.
- TMQFORWARD 在设定的周期,从 SERVICE1 队列中取消息
- TMQFORWARD 开始一个全局事务，并调用 SERVICE1 服务来处理这个消息
- SERVICE1 服务把处理的结果用 tprerurn()返回给 TPQFORWARD.
- TPQFORWARD 把收到的处理结果发送到 REPLYQ，这里是 CLIENT_REPLY1
- 客户端调用 tpdequeue(),请求从 REPLYQ 中取回响应消息
- TMQUEUE 收到该请求，它把相应的消息从 CLIENT_REPLY1 中取出，并发送给客户端
- 如果以上操作成功,那么 tpdequeue()返回成功

注意：当一条消息从 QUEUE 中被取出后,在该 QUEUE 中它将被删除,其他的进程就看不到这个消息了.如果有多个进程同时要取这个消息,只有最早的那个进程能取到该消息。

3. /Q 的使用场合

首先，/Q 的常见用法是用于实现数据的可靠传送,把数据从一台机器可靠的传送到另一台机器.如:在电信计费业务中,可以用/Q 把采集到的计费数据发送到计费中心进行处理；在银行中,不同的银行间可用/Q 传送结算数据。数据传送可以是在 Tuxedo 客户端与 Tuxedo 服务端之间，或 Tuxedo 服务端与服务端之间。

其次，可以用/Q 实现工作流，如图 8-3,前面的处理流程把处理结果保存到 QUEUE 中,后面的处理流程从相应的 QUEUE 中取要处理的消息，也把处理结果保存到 QUEUE 中，如此下去，直到完成。

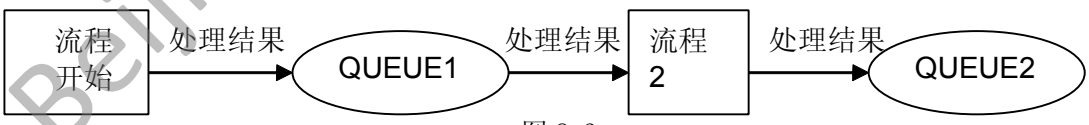

图 8-3

另外/Q 也常用来做批处理：可以把很多消息发送到一个 QUEUE 中,并设置在某个时间这些消息才生效,再把这些消息取出,进行批处理。

8.2.3.2 /Q 的配置和管理

/Q 的管理工作包括：QMCONFIG 环境变量的设置，用 qmadmin 或图形化管理工具进行 QUEUE SPACE，QUEUE 的创建及管理，因为/Q 也是一种资源管理器，所以要象数据库那样在 UBBCONFIG 的 GROUP 中进行配置。在 UBBCONFIG 中还要配置 TMQUEUE，TMQFORWARD 这两个 SERVER。下面分别进行说明：

- QMCONFIG 环境变量的设置

QMCONFIG 环境变量指定存储 QUEUE SPACE 的设备(文件)名,以便 qmamdmin 对它进行管理.它可在环境变量中设置,也可在执行 qmadmin 时在命令行中指定。

在 UNIX 下

```
QMCONFIG=/usr/tuxedo/qsample/QUE; export QMCONFIG
qmadmin /usr/tuxedo/qsample/QUE
```

示例 8-13

在 NT 下:

```
SET QMCONFIG=d:\landingbj\qsample\QUE
qmadmin d:\landingbj\qsample\QUE
```

示例 8-14

QMCONFIG 中指定的设备(文件)要先用 crdl 在 Tuxedo 文件系统中创建, 如:

```
D:\>qmadmin
>crdl d:\landingbj\qsample\QUE 0 5000
>q
```

示例 8-15

该设备(文件)中可以有多个 QUEUE SPACE。 QUEUE 中的消息等数据就保存在该设备(文件)中.

- qmadmin 的使用方法

Tuxedo 提供一个命令行管理工具 qmadmin,用于对/Q 进行管理,它类似 tmadmin, 常用的命令介绍如下:

- qspacecreate : 创建一个新的 QUEUE SPACE
- qcreate : 在某个 QUEUE SPACE 上创建 QUEUE
- qinfo : 查看某个 QUEUE 中的信息
- qlist : 显示一个 QUEUE SPACE 所包含的 QUEUE,及其中的当前消息个数
- qopen : 打开一个 QUEUE SPACE
- qclose : 关闭一个 QUEUE SPACE

说明:在 qmadmin 中输入 help 可以列出所有的命令,用 help 命令名也可以得到其子命令的帮助。

例如:

```
> help qinfo

qinfo [queue_name]
------------------
List information about the specified queue or for all queues.
This command lists the number of messages on the specified
queue or all queues if no argument is given, and the amount
of free space in the queue space.  In verbose mode, this
command also lists the queue creation parameters for each queue.
```

示例 8-16

1. QUEUE SPACE 的创建

在 qmadmin 中执行 qspacecreate,按提示进行操作,说明如下:

```
> qspacecreate
Queue space name: myqueuespace
IPC Key for queue space: 230458
Size of queue space in disk pages: 200
Number of queues in queue space: 3
Number of concurrent transactions in queue space: 3
Number of concurrent processes in queue space: 3
Number of messages in queue space: 12
Error queue name: errq
Initialize extents (y, n [default=n]):
Blocking factor [default=16]: 16
```

示例 8-17

参数说明:

1) --IPC Key for queue space:

该 QUEUE SPACE 的 IPC KEY，范围为 32,768 到 262,143.记住不要和系统的其他 IPC 资源的 ID 号冲突。

2) --Size of queue space in disk pages:

该 QUEUE SPACE 的大小，以页为单位

3) --Number of queues in queue space:

该 QUEUE SPACE 中最多可以有多少个 QUEUE 在里面

4) --Number of concurrent transactions in queue space:

在该 QUEUE SPACE 中最多可以有多少个事务同时存在,计算方法如下:

- 该 GROUP 中的每个 TMS_QM 会用到一个事务
- 该 GROUP 中的 TMQUEUE,TMQFORWARD 也会各自用到一个事务
- qmadmin 会用到一个事务
- 客户端在调用 tpenqueue（）,tpdequeue()之前开始的事务也要计算上

要估计最多可能有多少个这样的客户端同时使用该 QUEUE SPACE。

5) --Number of concurrent processes in queue space:

最多可以有多少个进程同时存取该 QUEUE SPACE，要包括 TMQUEUE，TMFORWARD 这两个进程。

6) --Number of Messages in queue space:

该 QUEUE SPACE 中最多可以有多少个 MESSAGE 在里面

7) --Error queue name:

当一个消息被回滚达到指定次数,Tuxedo 把该消息发送到 ERROR QUEUE 中,如果输入了 ERROR QUEUE NAME,则必须用 qcreate 创建该 QUEUE,如果没有创建 ERROR QUEUE,那么本来应该发送到 ERROR QUEUE 中的消息将被丢弃.

在创建 ERROR QUEUE 时,以下几个参数不能设置.

```
Queue order (priority, time, fifo, lifo):
Out-of-ordering enqueuing (top, msgid, [default=none]):
Retries [default=0]:
Retry delay in seconds
```

示例 8-18

8) --Initialize extents (y, n [default=n]):

是否初始化该 QUEUE SPACE 的存储空间

注意：因为在创建 QUEUE SPACE 过程中会用到 IPC 资源,所以如果在创建 QUEUE SPACE 时失败，在重新创建之前，最好把这些 IPC 资源释放掉，可在 qmadmin 中用 ipcrm 命令释放某个 QUEUE SPACE 所占用的 IPC 资源。

2. QUEUE 的创建

```
> qcreate
Queue name: service1
Queue order (priority, time, fifo, lifo): fifo
Out-of-ordering enqueuing (top, msgid, [default=none]): none
Retries [default=0]: 2
Retry delay in seconds [default=0]: 30
High limit for queue capacity warning (b for bytes used, B for blocks used,
% for percent used, m for messages [default=100%]): 80%
Reset (low) limit for queue capacity warning [default=0%]: 0%
Queue capacity command:
No default queue capacity command
Queue 'service1' created
```

示例 8-19

参数说明:

1) --Queue name:

要创建的 QUEUE 的名称

2) --Queue order (priority, time, fifo, lifo):

发送到该 QUEUE 的消息的存取顺序：

- priority:按优先级(在 tpenqueue()的 TPQCTL 参数中指定)
- time:按时间(在 tpenqueue()的 TPQCTL 参数中指定)
- fifo:先进先出
- lifo:先进后出

3) --Out-of-ordering enqueuing (top, msgid, [default=none]):

指定一个消息可以放在该 QUEUE 的最前面,或在某个 MSGID 之前

4) --Retries [default=0]: 2

默认情况下,当从一个 QUEUE 中取出某个 MESSAGE 的事务回滚时,该 MESSAGE 会被重新放回到该 QUEUE 中,当该消息又处于该 QUEUE 的最前面时,TMQUEUE 将再次试图取出该消息,这里指定重试的次数.默认值为 0,也就是不进行重试.当达到重试的次数时,如果该 QUEUE SPACE 设置了 ERROR QUEUE,那么这个消息将被移到该 ERROR QUEUE 中,如果该 QUEUE SPACE 没有设置 ERROR QUEUE,那么这个消息将被丢弃。

5) --Retry delay in seconds [default=0]: 30

指定重试的时间间隔

6) --High limit for queue capacity warning (b for bytes used, B for blocks used, % for percent used, m for messages [default=100%]): 80%

7) --Reset (low) limit for queue capacity warning [default=0%]: 10%

8) --Queue capacity command: /usr/app/bin/mailme myqueuespace service1

以上 3 个设置,当该 QUEUE 中的消息对空间的使用或消息数达到设定的阀值时,Tuxedo 系统系统自动执行一个命令,以达到对该 QUEUE 中的消息进行自动处理的目的.

在以上的设置中,当该 QUEUE 中 80%的空间被使用时,将执行/usr/app/bin/mailme, myqueuespace service1 是 mailme 的参数.

当第一次到达 80%之后, /usr/app/bin/mailme 被执行,只有当降为 10%之后,又到达 80%时, /usr/app/bin/mailme 才再次被执行.

9) --Reply Queue 和 Failure Queue

当采用转发方式时,TMQFORWARD 把用 tpcall()调用的与该 QUEUE 同名的 SERVICE 的处理结果放到 REPLY QUEUE 中,该 REPLY QUEUE 的名字在 tpenqueue()中指定,如果该 SERVICE 处理失败,TPRETURN(TPFAIL….),那么 TMQFORWARD 将把错误信息写到 FAILUER QUEUE 中,如果没有创建 Reply QUEUE 或 Failure Queue,TMQFORWARD 将把该 SERVICE 的返回丢弃,调用 tpdequeue()的客户端将收不到任何信息,如果创建了 REPLY QUEUE,那么即使该 SERVICE 没有返回信息,TMQFORWARD 也会往该 REPLY QUEUE 中写入一条长度为 0 的消息.客户端可以收到该消息。

3. UBBCONFIG 中要做的配置

- GROUP 中的配置

在 GROUP 中的配置与数据库通过 XA 协议与 Tuxedo 连接的配置差不多, 因为 QUEUE SPACE 是资源管理器, 而一个组只能有一个资源管理器。所以 QUEUE SPACE 与 QUEUE SERVER GROUP 之间是一对一的关系, 在 GROUP 中的配置有:

1) TMS(TRANSACTION MANAGEMENT SERVER): TMS_QM

2) OPENINFO, 它的设置格式如下:

```
OPENINFO="Tuxedo/QM:<device_name>:<queue_space_name>"
```

示例 8-20

Tuxedo/QM : 为/Q 所对应资源管理器的名称, 在$TUXDIR/udataobj/RM 指定;

device_name : 指定存储该 QUEUE SPACE 的设备(文件)名;

queue_space_name: 为该 QUEUE SPACE 的名称。

在 UNIX 下示例:

```
*GROUPS
QUE1
LMID = SITE1 GRPNO=2
TMSNAME = TMS_QM TMSCOUNT = 2
OPENINFO = “Tuxedo/QM:/home/QUE:QSPACE”
```

示例 8-21

在 NT 下示例:

```
*GROUPS
QUE1
LMID = SITE1 GRPNO=2
TMSNAME = TMS_QM TMSCOUNT = 2
OPENINFO = “Tuxedo/QM:d:\landingbj\qsample\QUE;QSPACE”
```

示例 8-21

- 在 SERVER 中的配置

在 SERVER 这一节中要配置 TMQUEUE(必须),TMQFORWARD(可选)这两个 SERVER。

TMQUEUE 的设置格式如下:

```
TMQUEUE CLOPT=“-s QSPACENAME:TMQUEUE --[-t timeout]”
```

示例 8-22

“-s“：指定该 SERVER 要发布的 SERVICE 的名称,采用别名方式:用 QUEUE SPACE 的名字加上 TMQUEUE;

“-t”：非事务中的/Q 操作的超时时间。

可参考下面的例子:

```
*SERVERS
TMQUEUE SRVGRP = QUE1 SRVID = 1
CLOPT = “-s QSPACE:TMQUEUE -- -t 60”
```

示例 8-23

TMQFORWARD 的的设置格式如下：

```
TMQFORWARD CLOPT=“-- -q qname[,qname...] [-t trantime]
[-i idletime] [-e] [-d] [-n][-f delay]”
```

示例 8-24

“-q”：“-q qname[,qname...] ”是用“,”隔开的 QUEUE 的名称,TMQFORWARD 将从这些 QUEUE 中取数据,并用 tpcall()调用与该 QUEUE 同名的 SERVICE 进行后续处理;

“-t”：TMQFORWARD 调用 tpbegin()时指定的事务超时时间.默认值为 60 秒;

“-i”：指定当该 QUEUE 中的消息都已被取出后,隔多长时间,TMQFORWARD 再次读该 QUEUE 看是否有新的消息到来;

“-e”：如果再这些 QUEUE 中都没有消息,那么 TMQFORWARD 将退出;

“-d”：删除导致所调用的服务失败，但返回了响应的原始消息.使它不用再被重试;

“-n”：当 TMQFORWARD 调用 tpcall()时,所用的 FLAG 为:TPNOTRAN;

“-f”：“-f delay”指定 TMQFORWARD 发送异步服务请求的时间间隔。

可参考下面的例子:

```
*SERVERS
TMQFORWARD SRVGRP=QUE1 SRVID = 5
CLOPT=” -- -i 2 -q STRING”
```

示例 8-25

8.3 动态配置 tmconfig

8.3.1 概述

tmconfig 是一个交互式工具可以用来动态修改 Tuxedo 运行环境配置。

8.3.2 配置 tmconfig 运行环境

```
TUXDIR=/home/oracle/tuxedo11g
TUXCONFIG=/home/landingbj/test/tuxconfig
EDITOR=vi
```

示例 8-26

当输入 tmconfig 出现以下输出时表示环境设置正确。

```
$ tmconfig
   Section: 1) RESOURCES, 2) MACHINES, 3) GROUPS 4) SERVERS
    5)SERVICES  6) NETWORK 7) ROUTING q) QUIT 9) WSL
    10) NETGROUPS 11) NETMAPS 12) INTERFACES [1]:
```

示例 8-27

8.3.3 tmconfig 常用操作

- 用 tmconfig 增加新主机

  1、 进入 tmconfig 后，选择 2）MACHINES 项

  2、 然后可以先选择 3）RETRIEVE 看看当前的配置，缺省是第一个 MACHINE 的配置

  3、 通过选择 2）NEXT 可以一直向后搜索 MACHINE 配置，直到空记录为止。

  4、 选择 4）ADD

```
Enter editor to add/modify fields [n]? y
```

示例 8-28

  5、 进入 vi 编辑状态，可以按照一定格式增加配置。

  格式为：MIB 域名[tab]值

  6、 增加 MACHINE 必须加入以下的域：

  - oTA_TUXCONFIG

- oTA_TUXDIR
- oTA_APPDIR
- oTA_TLOGDEVICE
- oTA_TLOGSIZE
- oTA_PMID
- oTA_LMID
- oTA_TYPE

例如：

```
TA_TUXCONFIG /home/landingbj/test/tuxconfig
TA_TUXDIR /home/oracle/tuxedo11g
TA_APPDIR /home/landingbj/test
TA_TLOGDEVICE /home/landingbj/test/TLOG
TA_ULOGPFX /home/landingbj/test/ULOG
TA_ENVFILE /home/landingbj/test/ENVFILE
TA_TLOGSIZE 150
TA_PMID SERVER109
TA_LMID SITE1
TA_TYPE Sun
```

示例 8-29

7、 存盘退出 vi，执行操作即可；

8、 激活新增的 MACHINE： 重新选择 2）MACHINES -> 5)UPDATE；

9、 进入 vi 后查找 TA_STATE,将其值从 NEW 改为 ACTIVE；

10、 存盘退出 vi 并执行操作。

- 用 tmconfig 增加新 Server 进程

1、 进入 tmconfig 后，选择 2）SERVER 项

11、 可以先选择 3）RETRIEVE 看看当前的配置，缺省是第一个 SERVER 的配置

12、 通过选择 2）NEXT 可以一直向后搜索 SERVER 配置，直到空记录为止。

13、 选择 4）ADD

```
Enter editor to add/modify fields [n]? y
```

示例 8-30

14、 进入 vi 编辑状态，可以按照一定格式增加配置。

格式为：MIB 域名[tab]值

15、 增加 MACHINE 必须加入以下的域：

- oTA_SERVERNAME
- oTA_SRVGRP
- oTA_SRVID

例如：

```
TA_SERVERNAME /home/landingbj/test/teller_server
TA_SRVGRP GROUP1
TA_SRVID 15
```

示例 8-31

16、 存盘退出 vi，执行操作即可。