联动北方
LANDING

企业级 IT 运维宝典系列

联动北方 出品

目　录

目　录 **2**

第 11 章　Tuxedo 监控 **3**
11.1　监控 Tuxedo 应用的方法 3
11.2　可以监控的系统和应用数据 3
11.3　使用管理控制台监控应用 4
11.4　使用命令行方式监控 4
11.5　使用 EventBroker 监视应用程序 4
11.5.1　相关 API 介绍 4
11.5.2　相关例子参考 5
11.6　使用 MIB 监视应用程序 7
11.7　使用日志文件来监控 9
11.7.1　Tuxedo 日志的分类 9
11.7.2　Tuxedo 事务日志 9
11.7.2.1　什么是 Transaction Log (TLOG) 9
11.7.2.2　Tuxedo 的事务管理 9
11.7.2.3　创建事务管理器 9
11.7.2.4　创建事务日志 10
11.7.2.5　迁移事务日志 10
11.7.2.6　处理事务日志 11
11.7.2.7　查看 TLOG 11
11.7.3　Tuxedo 用户日志 11
11.7.3.1　什么是 User Log（ULOG） 11
11.7.3.2　查看 ULOG 12

第 11 章 Tuxedo 监控

11.1 监控 Tuxedo 应用的方法

作为一个管理员，必须保证 Tuxedo 正常高效的运行，所以我们要监控 Tuxedo 的各个方面。

- 资源，例如：共享内存；
- 活动性，例如：事件；
- 其他一些问题，例如：存在必须校正的安全漏洞。

Tuxedo 系统提供了几种方法来帮助你监控你的 Tuxedo 系统和应用：

1. Tuxedo 管理控制台：基于 Web 的图形用户界面，你可以用它来观察应用程序的行为，并动态地配置其参数。您可以显示和更改配置信息，确定系统各组件的状态，并得到执行的情况，例如关于项目的统计信息和排队的请求。
2. 命令行方式：一组命令（例如，tmboot，tmadmin，tmshutdown）你可以用它来启动，关闭，配置和管理应用程序。
3. 日志文件：一个文件，包含错误和警告消息报告，调试信息，以及对跟踪和解决系统中的问题有所帮助信息。
4. MIB：使用 MIB，可以编写程序，来动态管理监控您的运行时的应用。
5. TSAM（Tuxedo System and Application Monitor）

这些工具帮助您的应用程序快速有效的对应不断变化的业务需求，提高排除故障情况的能力。还可以帮助管理员管理应用程序的性能和安全。

监控工具如下所示：

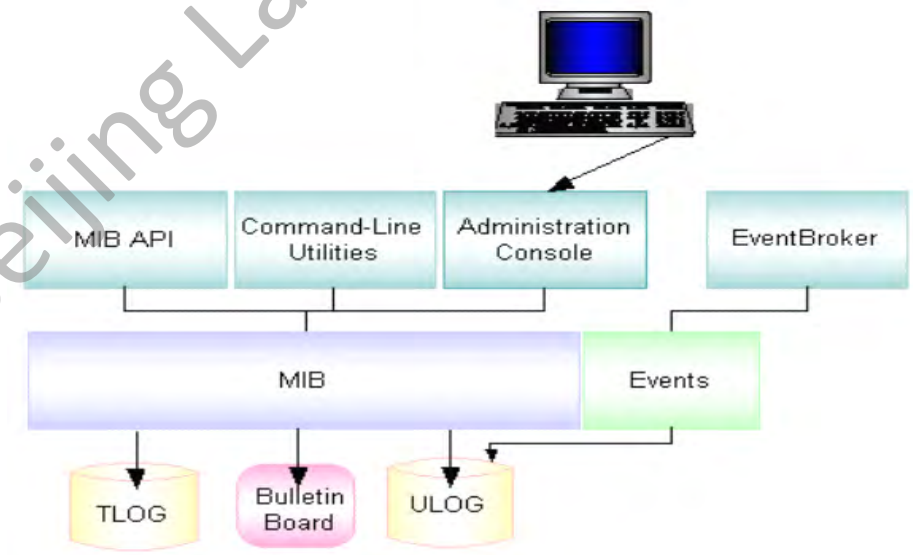

图 11-1

11.2 可以监控的系统和应用数据

Tuxedo 管理员可以用 tmadmin 命令或通过 MIB 来管理监控以下组件：

- Clients
- Conversations
- Groups
- Message queues
- Networks
- Servers
- Services
- CORBA Interfaces
- Transactions

11.3 使用管理控制台监控应用

Tuxedo 的管理控制台是一个图形用户界面，底层通过 MIB 查询修改系统配置、监控应用程序。它是通过通过 Web 浏览器来访问。任何一个管理员通过浏览器可以监控 Tuxedo 的应用。

11.4 使用命令行方式监控

在 Tuxedo 启动的情况下，可以用 tmadmin 命令对 Tuxedo 应用进行管理和监控。常用的子命令请参见 9.2.1。

11.5 使用 EventBroker 监视应用程序

首先我们清晰一下基本概念：事件代理(EventBroker），它其实提供了这样一种通讯方式：一些进程可以发布消息给订阅了这些消息的进程或客户端，例如：服务可以在价格发生变化时发布信息，所有连接系统的客户端都可以收到信息。

事件代理只能执行一些预先定义的动作，这些动作包括：

- 调用服务（tpacall）
- 将请求送入一个可靠队列（tpenqueue）
- 用户通知(tpnotify)
- 记录日志到 ULOG.mmddyy
- 执行系统调用

主要有 2 类事件：

- 系统事件：Tuxedo 系统内部状态变化，如网络故障等
- 用户定义事件：应用中与业务流程有关的事件

11.5.1 相关 API 介绍

其中，发布事件用的 ATMI API 是 tppost()：

```
int tppost(char *eventname, char *data, long len, long flags)
```

示例 11-1

char *eventname : 事件的名字

char *data : 伴随事件发布的数据

long len : 数据长度

flags : 可以是[TPNOTIME|TPSIGRSTRT|TPNOBLOCK]

其中，订阅和取消订阅事件的 ATMI API 是 tpsubscribe()/tpunsubscribe()：

```
long tpsubscribe(char *eventexpr, char *filter, TPEVCTL *ctl, long flags)
```

示例 11-2

char *eventexpr : 表示事件的字符串，可以含有通配符*

char *filter : 一个布尔表达式，决定是否触发相应的动作

TPEVCTL ctl : 描述事件代理将进行的动作

Flags : 可以是[TPNOTIME|TPSIGRSTRT|TPNOBLOCK]

这个函数返回订阅句柄，-1 表示失败

```
int tpunsubscribe(long subscriptn, long flags)
```

示例 11-3

long subscriptn : tpsubscribe()返回的句柄

flags : 可以是[TPNOTIME|TPSIGRSTRT|TPNOBLOCK]

其中，TPEVCTL 结构的定义如下：

```
struct TPEVCTL
 {
     long flags;
     char name1[32];
     char name2[32];
     TPQCTL qctl;
 }
```

示例 11-4

订阅一个事件，指定事件发生时触发一个服务请求的方法如下：

```
ctl->flags=TPEVSERVICE;
strcpy(ctl->name1,” SERVICENAME”);
```

示例 11-5

订阅一个事件，指定事件发生时把消息存入队列的方法如下：

```
ctl->flags=TPEVQUEUE;
strcpy(ctl->name1,” QSPACENAME”);
```

```
strcpy(ctl->name2,” QUEUENAME”);
ctl->qctl.flags=TPQREPLYQ;
strcpy(ctl->qctl.replyqueue,” RPLYQ”);
```

示例 11-6

11.5.2 相关例子参考

下面，我们给出一个使用 tpsubscribe()和 tpunsubscribe()的例子：

```
#include <atmi.h>
static long sub_serv;
int tpsvrinit(int argc, char **argv)
{
   TPEVCTL evctl;
   /* 连接 Tuxedo 系统——tpinit() */
   evctl.flags=TPEVSERVICE;
   strcpy(evctl.name1,” BANK_MANAGER”);
   sub_serv=tpsubscribe(“BANK_TLR_WITHDRAWAL”,
       “AMOUNT>300.00”,&evctl,TPSIGRSTRT);
   if (sub_serv == -1)
   return(-1);
}

int tpsvrdone()
{
   if (tpunsubscribe(sub_serv,TPSIGRSTRT)== -1)
   {
      printf(“Error unsubscribing to service event\n”);
   }
}
```

示例 11-7

这个服务器在初始化时订阅事件，指定事件发生时调用服务 BANK_MANAGER，这个事件是 BANK_TLR_WITHDRAWAL，仅当 AMOUNT 域的值大于 300.00 时触发。

服务器在终止时用 tpunsubscribe()取消订阅。

下面是使用 tppost()的例子：

```
void WITHDRAWAL(TPSVCINFO *transb)
{
   ……
   if (amt > 300.00)
   {
      if ( (Fchg(transf,EVENT_NAME,0,” BANK_TLR_WITHDRAWAL”,0)<0) ||
        (Fchg(transf, EVENT_TIME,0,gettime(),0)<0 )||
        (Fchg(transf, AMOUNT,0,(char *)&amt,0)<0) ))
      {
        sprintf(emsg,” Fchg(event flds) failed :%s”,Fstrerror(Ferror));
      }
      else if( tppost(“BANK_TLR_WITHDRAWAL”,(char *)transf,0L,
```

```
            TPNOTRAN|TPSIGRSTRT)<0)
          {
             if (tperrno!=TPENOENT)
             {
              sprintf(emsg,” tppost() failed :%s”,tpstrerror(tperrno));
             }
        }
        if ( strcmp(emsg,” ”)!=0)
        {
          userlog(“WARN:Event BANK_TLR_WITHDRAWAL not posted:%s”,emsg);
          strcpy(emsg,” ”);
        }
      }
    }
```

示例 11-8

WITHDRAWAL 服务在金额>300.00 时,给 FML 域赋值，并调用 tppost()发布事件。

而相关的事件代理服务器要配置在在 ubbcofig 中

```
*SERVERS
TMSYSEVT  SRVGRP=EVTGRP1       SRVID=100
          RESTART=Y MAXGEN=5 GRACE=3600
          CLOPT=”-A  -- -f tmsysevt.dat”
TMUSREVT  SRVGRP=EVTGRP1       SRVID=150
          RESTART=Y MAXGEN=5 GRACE=3600
          CLOPT=”-A  -- -f tmusrevt.dat”
TMUSREVT  SRVGRP=EVTGRP1       SRVID=200
          RESTART=Y MAXGEN=5 GRACE=3600
          CLOPT=”-A  -- -S -p 120”
```

示例 11-9

TMSYSEVT：系统事件代理服务进程；

TMUSREVT：用户事件代理服务进程；

“-f”指定了包含订阅信息的数据文件。

11.6 使用 MIB 监视应用程序

Tuxedo 提供了一套可编程的管理员 API 接口(Management Information Bases),简称为 MIB。

通过 MIB，可以方便的监控 Tuxedo 运行时的所有系统信息，例如：SERVER 和 SERVICE 的运行状况、SERVER 队列的情况、客户端的使用情况、域间通讯的连接情况、系统的资源配置等，所有的信息都可以通过 MIB 的 API 来获取，或者动态的修改系统配置。

例如，对于某些关键的 SERVER 或者 SERVICE，我们想监控它在一段时间内的被调用情况，服务是否出现异常或者是被挂起，SERVER 对应的消息队列是否出现了堵塞，或者是队列空间紧张；当 SERVER 出现请求反应较慢需要增加进程时，也可以通过 MIB 动态的增加进程个数。

例如，对于消息队列，我们需要监控消息队列当前的请求总数，队列长度是否达到了危险程度需要告警。

例如，对于客户端使用者来说，我们需要监控有那些客户端调用超时或者是调用已经被系统给挂起的情况，某些客户端调用出现异常会导致系统性能下降，我们可以自动先将这些客户端请求给撤销或者杀掉，再根据客户端请求的全过程来找出问题。

例如，如果系统和其它外围系统存在着连接，或者是分布式的 Tuxedo 系统，我们需要监控系统和其它系统的域间通讯连接是否正常，远端的系统是否还在正常运行。

用一句话来概括：Tuxedo 所有的应用都可以通过 MIB 进行实时监控。

MIB 接口规定了系统管理员，系统操作员和其它用户，三者具有不同的操作权限。

MIB 接口将系统资源信息划分为三种类型，分别为 Classes，Attributes 和 States：

- Classes 指资源的分组，例如 SERVER、SERVICE、CLIENT、QUEUE、MSG 等；
- Attributes 指 Classes 对应的属性，例如 SERVER 的属性有 SERVERNAME、SRVGRP、SRVID 和运行时的参数值等属性；
- States 指 Classes 在运行时的状态，以及在运行时可以更改的属性。

MIB 的操作类(OPERATION)分为 GET 和 SET 两种类型，即指查询和设置两种操作。

Classes 的类型有 T_MACHINE、T_GROUP、T_SERVER 、T_SERVICE 、T_SVCGRP、T_QUEUE、T_MSG T_DOMAIN、T_CLIENT 等。

用 MIB 的 API 查询系统资源的编程比较简单，跟客户端调用 SERVICE 方法一样，都是用 tpcall()或者 tpacall()。只是服务名为.TMIB。输入输出的缓冲区类型为 FML32。在调用之前我们需要指定几个查询参数 TA_OPERATION 和 TA_CLASS、TA_FLAGS，以及 Classes 对应的参数，例如我们在查询 T_SERVER 这个 Classes，我们想根据 LMID、SERVERNAME、SRVGRP、SRVID、RQADDR 等参数来查询 SERVER，那么就在输入缓冲区中输入对应的参数值。

下面我们以 T_SERVER 这个 Classes 为例子来做个大致介绍。

T_SERVER 就是我们在 Tuxedo 配置文件的*SERVERS 节中所定义的 SERVER。用 MIB 来查询 SERVER 在运行状态下的数据，可以获取到 SERVER 的配置参数值，以及 SERVER 运行时的其它动态参数值，例如进程 ID(操作系统的 PID)，请求消息队列的 ID，回复消息队列的 ID，队列名称，SERVER 当前被请求的服务，SERVER 被调用的次数、已经完成的调用次数，还没有完成的请求数等。

代码如下：

```
obuf = (FBFR32 *)tpalloc("FML32",NULL,8000);
Finit32(obuf, (FLDLEN32)Fsizeof32(obuf));

Fchg32(obuf, TA_OPERATION, 0, "GET", 0);
Fchg32(obuf, TA_CLASS, 0, "T_SERVER", 0);
flags = MIB_LOCAL;
Fchg32( obuf, TA_FLAGS, 0, (char *)&flags, 0);

//设置想要查询的参数，例如服务名、服务 ID、服务队列等。
if(strlen(t_SrvName)>0) Fchg32(obuf, TA_SERVERNAME, 0,t_SrvName, 0);
if(strlen(t_SrvGrp)>0)  Fchg32(obuf, TA_SRVGRP, 0,t_SrvGrp, 0);
```

```
    ret=tpcall(".TMIB",(char *)obuf,0,(char **)&obuf,&sendlen,TPNOTRAN);

    if(ret == -1)
    {
         userlog("获取系统服务的状态失败:%s",tpstrerror(tperrno));
         tpfree((char *)obuf);
       Tuxedo_Return("-1","查询 Tuxedo 系统状态失败");
    }

    i=Foccur32(obuf,TA_SERVERNAME);
    for(j=0;j<i;j++)
    {
         Fgets32(obuf,TA_SERVERNAME,j,t_ServerName);
         Fgets32(obuf,TA_SRVID,j,t_SrvID);
         Fgets32(obuf,TA_STATE,j,t_STATE);
         Fgets32(obuf,TA_SRVGRP,j,t_SrvGrp);
         Fgets32(obuf,TA_RQADDR,j,t_RQADDR);//SERVER 队列名称
         Fgets32(obuf,TA_TOTWORKL,j,t_TOTWORKL);//工作量。TA_TOTREQC*负载因子
=TA_TOTWORKL

           ..........................
           //根据返回的数据进行其它操作。
           ..............................
    }
```

示例 11-10

从上面代码可以看到，我们可以获取到 SERVER 在运行状态下的所有信息。

例如，假设有一个服务有异常，那么我们就可以根据获取的服务状态，来自动做一些处理。如果 SERVER 包含多个 SERVICE，那么可以监控到某个 SERVICE 是否因为被调用次数过多而且影响到 SERVER 中其它的 SERVICE，如果是这样就要考虑将这个 SERVICE 移到其它 SERVER 或者是单独成为一个 SERVER。通过查询 SERVER 对应的请求消息队列，如果排队过长，则需要自动进行处理或者是告警。

如果我们想在运行状态时动态的修改 SERVER 的配置参数，那么我们可以将 TA_OPERATION 改为 SET，将需要修改的参数输入缓冲区，调用 MIB 的接口即可。

修改某些参数有时需要将 SERVER 的状态改为 INActive，相当于 tmshutdown -s SERVER,然后才能修改。

11.7 使用日志文件来监控

11.7.1 Tuxedo 日志的分类

Tuxedo 系统提供了两种日志文件，分别是 Transaction Log (TLOG)和 User Log (ULOG)，可以帮助管理员快速准确的发现问题。

- Transaction Log (TLOG)—是一个二进制的文件，被事件管理器（TMS）使用。不能用文本编辑器读取。。可以用 tmadmin 管理命令查看全局事务的状态。

- User Log (ULOG)—当应用程序启动的时候同时 Tuxedo 系统产生一个 ULOG 消息日志，记录 Tuxedo 系统输出的信息。用户也可以通过编程输出信息到这个文件。

11.7.2 Tuxedo 事务日志

11.7.2.1 什么是 Transaction Log (TLOG)

TLOG 日志用来跟踪全局事务的两个阶段提交，日志中的记录是用来保证在网络故障或者是机器崩溃的时候，仍能正常完成全局事务。

对于跨域的全局事务，还需要域日志 DMTLOG 来共同保证事务的完整性。

11.7.2.2 Tuxedo 的事务管理

Tuxedo 的事务管理工作包括：创建 TMS、创建 TLOG、运行时事务的迁移和监控。

11.7.2.3 创建事务管理器

系统管理员必须为每一种用到的资源管理器（RM）创建专用的事务管理器（TMS），并且在 UBBCONFIG 文件中，配置使用。Tuxedo 系统提供了 buildtms 命令来创建 TMS，这个命令需要从一个名为 RM 的文件中去读取 RM 信息，包括 RM 名、XA switch 名，以及 XA 支持库。

RM 位于 Tuxedo 的 udataobj 目录下，它保存着系统可能用到的所有 RM 信息，每行格式如下：

```
RM_NAME:XA_SWITCH_NAME:XA_LIBS
```

示例 11-11

其中，UNIX 平台使用“：”分隔，Windows 平台使用“；”分隔。

RM_NAME 是 RM 在 Tuxedo 系统中的标识名，XA_SWITCH_NAME 是 RM 的 xa_switch_t 结构名，XA_LIBS 是 RM 的接口库。XA Switch 名和 XA 接口库由 RM 供应商提供。修改完这些，就可以使用 buildtms 来创建 TMS。Buildtms 有两个选项，“-r”指定 RM 名，用于引用 RM 文件中的一个条目，“-o”用于指定生成的 TMS 名称和存放路径。

建议把 TMS 放在 Tuxedo 的 bin 目录下，以方便重复使用。

11.7.2.4 创建事务日志

TMS 使用 TLOG 来记录事务日志，以便恢复或回滚全局事务。每台 Tuxedo 主机上只需要创建一个 TLOG 文件。它会被这台主机上的所有 TMS 实例共享使用，如果一个全局事务还没有完成，那么它在 TLOG 中将占用一个分页的空间，全局事务完成后，它在 TLOG 中的记录将会被删除。

创建 TLOG 时先进入 tmadmin 界面，用 crdl 命令创建设备文件，命令格式为：

```
crdl  -b blocks  -z config
```

示例 11-12

- -b blocks 为设备文件的大小，以块（block）为单位
- -z config 为设备文件名，应和配置文件中 TLOGDEVICE 相同

然后执行 crlog 命令创建 TLOG，命令格式为：

```
crlog  -m machine
```

示例 11-13

- -m machine 为机器逻辑名

如果用到跨域的全局事务，则必须为每个域网关组创建一个 DMTLOG 日志。该 DMTLOG 文件被定义在 DMCONFIG 文件的*DM_LOCAL 段。

```
DMTLOGDEV=string
```

示例 11-14

其中 string 是日志设备的名称

DMTLOG 还有两个可选参数：

- DMTLOGNAME=identifier 用来指定设备上的 TLOG 名
- DMTLOGSIZE=numeric 用来指定 TLOG 的大小

如果在启动一个 domain 网关组的时候没有创建 DMTLOG，那么网关服务器会自动的创建这个日志。

11.7.2.5 迁移事务日志

当主机出现故障需要更换，或者是管理员需要对运行 Tuxedo 的系统进行升级时，管理员可以把正在执行的应用程序迁移到另一台主机。

事务日志是迁移中的重要部分，迁移步骤如下：

1. 停止所有可能向 TLOG 中写日志的进程；
2. 执行 migrategroup 把所有部署在源主机上的进程迁移到备份机；
3. 执行 dumptlog 命令把事务日志转储在一个 ASCII 文件 logfile 中。

在目标主机上执行的命令如下：

4. 从源主机上复制 logfile，执行 loadtlog 命令从 logfile 中加载事务日志；
5. 执行 logstart，命令强制对事务日志做热恢复；
6. 执行 boot 命令启动所有迁移过来的进程组。

11.7.2.6 处理事务日志

在 Tuxedo 应用程序中，造成事务失败的主要原因有：发起事务的客户机异常终止、TMS 或服务进程异常终止、通信网络中断、Tuxedo 系统出现错误和 RM 响应超时等。事务失败可能发生在提交前、2PC 的第一个阶段（预提交事务阶段）和 2PC 的第二个阶段。在提交前和预提交过程中失败的事务可以由 Tuxedo 自动恢复，而在 2PC 的第二个阶段中失败的事务需要在管理员的协助下进行恢复。

如果一个事务在提交前就失败了，那么在事务超时后，TMS 会把它的状态从 TMGACTIVE 改变为 TMGABORTONLY，在 Tuxedo 下一次进行健康检查的时候，将它从全局事务表（GTT）中清除。如果一个事务在预提交过程中失败了，那么 tpcommit（）将会返回-1，并且把 tperrno 设置为 TPEABORT。在事务超时以后，TMS 会把它的状态从 TMGCOMMCALLED 改变为 TMGABORTED，在 Tuxedo 下一次进行健康检查的时候把它从 GTT 中清除。

如果一个事务在 2PC 的第二阶段失败了，tpcommit（）返回-1，tperrno 设置为 TPEHAZARD 或 TPEHEURISTIC。TPEHAZARD 表示事务已经由 RM 启发式完成，但是状态未知；TPEHEURISTIC 表示事务已经由 RM 启发式完成，但是一部分 RM 提交了事务分支，另一部分回滚了事务分支。

在一个事务提交以前，如果因为 TMS 异常终止而导致事务不能正常结束那么管理员可以通过执行 tmadmin 提供的管理命令来结束它。Tmadmin 提供了 printtrans（pt）、aborttrans（abort）和 committans（commit）来结束没有完成的事务。

11.7.2.7 查看 TLOG

Tuxedo 日志可以帮助管理员，发现和分析系统和应用失败的原因。TLOG 是一个二进制文件，它包含全局事务交易过程的信息。要查看 TLOG 必须先将 TLOG 转换为文本文件，Tuxedo tmadmin 提供了双向转换的命令：

- dumptlog（dl）把二进制 TLOG 导出到一个文本文件
- loadtlog 把 dumptlog 导出的文本加载到 TLOG 二进制文件中

11.7.3 Tuxedo 用户日志

11.7.3.1 什么是 User Log（ULOG）

User Log 是 Tuxedo 系统运行日志文件，包括 error messages，warning messages，information messages 和 debugging messages。应用客户端和服务器通常也会写这个文件。

每天会产生一个新的 ULOG 日志，并且在每台机器上有不同的 ULOG。

当一个共享文件系统被使用的时候，多个机器就可以共享一个 ULOG。

ULOG 提供了一个系统管理记录，包含 Tuxedo 系统和应用失败的原因。可以用文本编辑器查看 ULOG 文件。

11.7.3.2 查看 ULOG

下面来看一个 ULOG 的例子，比如，错误消息是 LIBTUX_CAT:358，问题的原因是没有足够的 UNIX 系统信号量来启动应用程序。

```
    151550. landingbj!BBL.28041.1.0: LIBTUX_CAT:262: std main starting
    151550. landingbj!BBL.28041.1.0: LIBTUX_CAT:358: reached UNIX limit on
semaphore ids
    151550. landingbj!BBL.28041.1.0: LIBTUX_CAT:248: fatal: system init
function ..
    151550. landingbj!BBL.28040.1.0: CMDTUX_CAT:825: Process BBL at SITE1
failed ...
    151550. landingbj!BBL.28040.1.0: WARNING: No BBL available on site SITE1.
Will not attempt to boot server processes on that site
```

示例 11-15

一个 ULOG 消息是由标签和正文组成。

标签由下面几部分组成：

1. 一个六位数字（hhmmss）表示一天中的时分秒；

2. 机器的名字（在 Linux 中 hostname －n 显示的机器名）；

3. 进程的名称和标识（进程 ID、线程 ID、上下文 ID）；

如果当前处于全局事务中，则还包括一个：

4. 事务 ID；

正文由下面几部分组成：

5. Tuxedo 消息目录的名字；

6. Tuxedo 消息编号；

7. Tuxedo 系统消息。

另外，关于 Tuxedo 域连接的管理监控，可以通过 dmadmin 命令或 MIB 完成。dmadmin 常用的子命令请参见第 8 章相关内容。