

搭建 STAF+STAX 环境，完成基本的文件传输、调度等功能

1.1 STAF 的安装与配置

STAF 的安装文件可以从 [STAF 的网站](#) 下载。对于不同的平台和 JVM 环境有不同的安装文件，请选择合适的文件下载。如果下载的是 jar 文件，要确保需要安装 STAF 的机器上已经安装有相应的 JRE，然后运行如下命令安装 STAF：java -jar STAF 安装文件.jar。如果下载的是可执行文件，则直接运行即可。

STAF 的安装比较简单，只需要按照向导提示进行操作即可。安装完毕后，可以通过 STAFProc 命令启动 STAF。关闭 STAF 可以用如下的命令：staf local shutdown shutdown。从这条命令我们可以看出上面提到的 STAF 的命令格式。local 表示 STAF 的本地系统，shutdown 表示服务，此服务提供了 STAF 的关闭操作。第二个 shutdown 表示传递给服务的参数，指示 STAF 把本地的 STAF 服务关闭。

STAX 的安装文件也可以从 [STAF 的网站](#) 下载。STAX 本身不需要安装，只需要更改 STAF 的配置文件以便 STAF 在启动的时候能够加载 STAX 服务。从这个角度来说，STAX 是 STAF 的一种外部服务，可以根据需要来决定是否加载它。

下载完 STAX 后，将其解压到 \$STAF_Install_Directory/services/stax 目录中，然后更改 STAF 的配置文件 STAF.cfg。此文件在 \$STAF_Install_Directory/bin 目录下。在 STAF.cfg 文件末尾加上如下的代码，然后重启 STAF。

代码 1: STAX 配置

```
SERVICE STAX LIBRARY JSTAF EXECUTE /
```

```
{STAF/Config/STAFRoot}/services/stax/STAX.jar OPTION J2=-Xmx384m
```

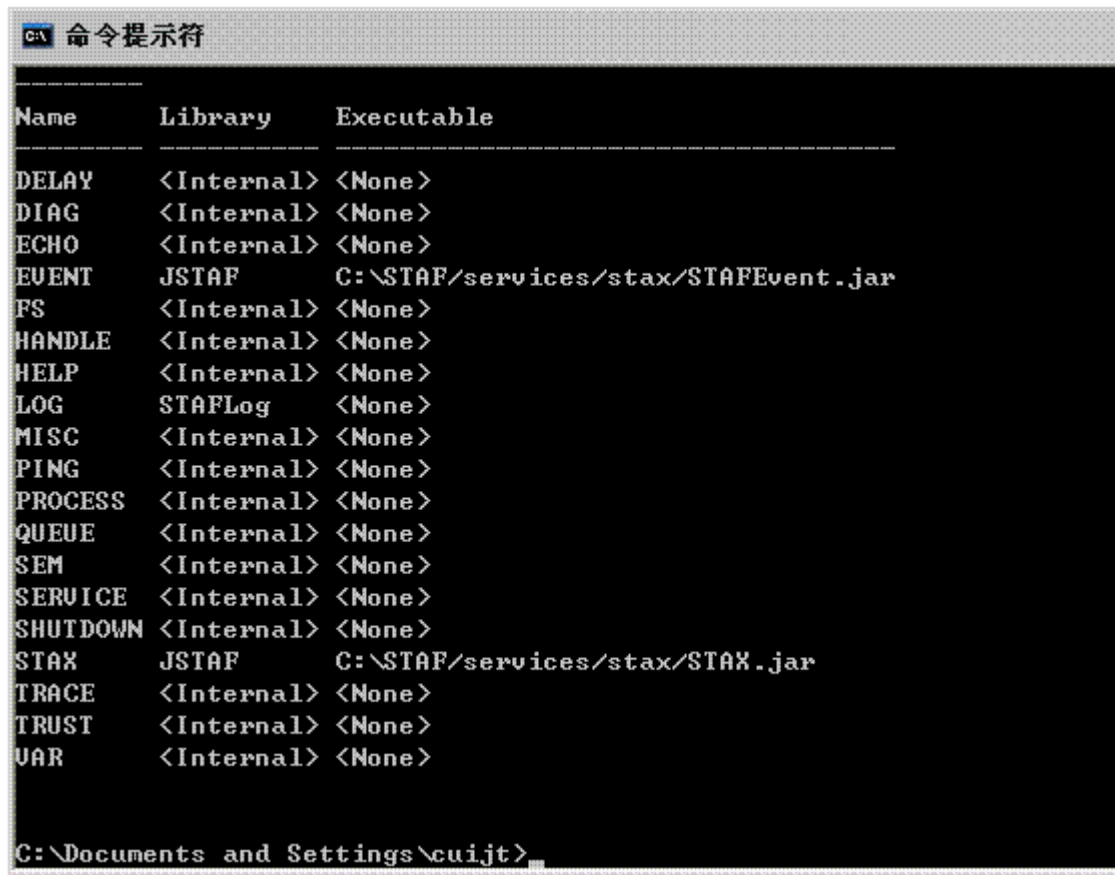
```
SERVICE EVENT LIBRARY JSTAF EXECUTE /
```

```
{STAF/Config/STAFRoot}/services/stax/STAFEvent.jar
```

```
SET MAXQUEUE SIZE 10000
```

STAF 重启之后，运行命令 `staf local service list`，查看输出结果，如果显示有 STAX 和 EVENT，如图 1 所示，则说明 STAX 已经成功加载。

图 1. STAF 服务列表



Name	Library	Executable
DELAY	<Internal>	<None>
DIAG	<Internal>	<None>
ECHO	<Internal>	<None>
EVENT	JSTAF	C:\STAF\services\stax\STAFEvent.jar
FS	<Internal>	<None>
HANDLE	<Internal>	<None>
HELP	<Internal>	<None>
LOG	STAFLog	<None>
MISC	<Internal>	<None>
PING	<Internal>	<None>
PROCESS	<Internal>	<None>
QUEUE	<Internal>	<None>
SEM	<Internal>	<None>
SERVICE	<Internal>	<None>
SHUTDOWN	<Internal>	<None>
STAX	JSTAF	C:\STAF\services\stax\STAX.jar
TRACE	<Internal>	<None>
TRUST	<Internal>	<None>
UAR	<Internal>	<None>

SERVICE STAX LIBRARY JSTAF EXECUTE

{STAF/Config/STAFRoot}/services/stax/STAX.jar 通知 STAF 在启动时以名字 STAX（这样在 STAF 服务列表中，我们看到的 STAX 的服务名字就叫做 STAX）来加载 STAX.jar，也就是 STAX 服务。传递的参数 `J2=-Xmx384m` 表示更改 JVM 的堆栈大小。如果 STAX 会出现 `OutOfMemory` 错误，则需要调整这个参数，增加 JVM 的堆栈大小。建议在加载 STAX 时总是指定这个参数，并且根据系统环境来调整参数大小。

SERVICE EVENT LIBRARY JSTAF EXECUTE

{STAF/Config/STAFRoot}/services/stax/STAFEvent.jar 通知 STAF 在启动时以名字 EVENT 来加载 STAFEvent.jar。

如果需要在运行 STAX 的机器上运行 STAX Monitor （STAX 任务的监控工具），则需要设置 MAXQUEUE SIZE，以保证 STAXMonitor 能够正确运行。

2.1 STAF Java 代码示例

代码 2 所示的是 STAF Java 代码示例。

代码 2: STAF Java 代码示例

```
STAFHandle handle = null;

try {

    handle = new STAFHandle("Java_Sample_Test");

} catch (STAFException e) {

    System.exit(1);

}

STAFResult result = handle.submit2("Linux1", "process",

    "start command ls parms -l wait stdout /root/lsjava.log");

if (result.Ok != result.rc) {

    System.out.println("Error starting the process ls, RC: " +
        result.rc);
}
```

```
}

result = handle.submit2("Linux1", "fs", "copy FILE /root/lsjava.log

    TODIRECTORY C:/STAF TOMACHINE windows' % machineName");

if (result.Ok != result.rc) {

    System.out.println("Error coping file, RC: " + result.rc);

}
```

在调用 **STAF** 服务之前，首先需要注册 **STAFHandle**，所有的 **STAF** 服务调用都要通过这个句柄来进行，因此一般把这个句柄设置成静态的。通过 **handle.submit2()** 函数可以向 **STAF** 服务发送请求并且接收处理结果。

2.2 STAX 脚本示例

STAX 为我们简化了调用 **STAF** 服务的过程，因此我们通过 **STAX** 脚本来调用 **STAF** 服务。本节将根据一个简单的示例来简要介绍 **STAX** 脚本的语法。

代码 3: STAX 脚本 SampleScript.xml 示例

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>

2 <!DOCTYPE stax SYSTEM "stax.dtd">
```

3 <!-- sample1.xml - Sample of a job definition file for STAX

4 Job Description:

5 This job executes some STAF commands and sends messages to the
STAX Job Monitor.

6 -->

7 <stax>

8 <script> LinuxMachine = ['Linux1', 'Linux2'] </script>

9 <defaultcall function="ListDirectory">

10 </defaultcall>

11 <function name="ListDirectory">

12 <paralleliterate var = "machineName" in="LinuxMachine">

13 <testcase name = "'listDirectory' ">

14 <sequence>

15 <stafcmd>

```
16          <location>' %s' % machineName</location>

17          <service>'process' </service>

18          <request>'start command "ls" parms "-l" wait stdout
/root/ls.log' </request>

19      </stafcmd>

20      <if expr="RC == 0">

21          <sequence>

22              <tcstatus result="" pass' "/>

23              <log message="1">'List directory successfully
on %s' % machineName</log>

24          </sequence>

25          <else>

26              <sequence>

27                  <tcstatus result="" fail' "/>

28                  <log message="1">'Error in listing directory
on %s' % machineName</log>

29              </sequence>

30          </else>
```

```
31         </if>

32         <stafcmd>

33             <location>' %s' % machineName</location>

34             <service>' fs' </service>

35             <request>' copy FILE /root/ls.log TOFILE ls%s.log
TODIRECTORY C:/STAF

                    TOMACHINE windows' % machineName</request>

36         </stafcmd>

37     </sequence>

38 </testcase>

39 </paralleliterate>

40 </function>

41 </stax>
```

这个示例调用两台 Linux 机器上的 **ls** 命令，将结果输出到文件，根据命令返回的结果判断调用是否成功，然后复制文件到另外的 **STAF** 机器中。为了方便描述，为脚本加上行号。

STAX 采用现在流行的 XML 语言作为其脚本语言。第一行是 XML 语言的标准格式，第二行表示此 XML 文件使用 `stax.dtd` 样式表进行验证。所有的 STAX 脚本文件都应该保留这两行。

3-6 行是 XML 的注释，用来描述这个脚本的功能。

第 7 行是 STAX 脚本命令的开始符，所有 STAX 脚本内容都要用它起始。第 8 行中 `script` 类似于编程语言中的定义变量的语句，在这里定义一个长度为 2 的数组 `LinuxMachine`，其值为 `Linux1` 和 `Linux2`。

第 9-10 行指定 STAX 脚本运行时调用的函数。第 11-40 行是函数的定义体。11 行指定函数名为 `ListDirectory`。

第 12-39 行定义一个循环，类似于 Java 中的 `for`，但是这个循环是并行的。
`var="machineName" in="LinuxMachine"` 表示此循环从 `LinuxMachine` 数组中获得输入，并且赋给 `machineName` 变量。

13 行定义测试用例，在 STAX 脚本的运行中，可以根据运行结果来决定测试用例的结果，方便用户查看。

第 14-37 行表示其中的 STAX 脚本是顺序执行的。15-19 行执行具体的 STAF 命令，其中 `location` 指定需要运行 STAF 命令的机器，可以由变量来动态指定，比如 `'%s' % machineName`。`service` 表示需要调用的服务，在这里为 `process` 进程服务。`request` 为需要传递给服务的参数。进程服务的参数分为几部分，首先需要调用的命令 `"ls"`，`parms` 指定需要传递给 `"ls"` 的参数 `"-l"`。`wait` 表示需要等待这个命令结束才能返回。`stdout` 表示将命令运行的结果输出到文件中去。

20-31 行判断上个命令的返回结果，并根据返回结果的值设定测试用例的状态，并且记录日志以及将消息发送到 `STAFMonitor`。`expr="RC==0"` 判断返回结果是否为 0。`RC` 表示上个命令的返回结果，0 表示命令执行成功。`<tcstatus result="" pass'"/>` 设置测试用例状态为通过，`fail` 则表示测试用例失败。`<log message="1">` 表示不仅将消息记录到 STAX 的日志中，而且将其发送到 `STAFMonitor`（如果 `STAFMonitor` 处于运行状态）。

32-36 行是 STAF 的文件拷贝命令。`fs` 表示文件系统服务，`copy FILE` 指定复制文件操作，`TOFILE` 指定目标文件的名称，STAX 会用命令后面的参数 `% machineName` 替换 `%s`，因此目标文件的名称为 `lsLinux1.log` 和 `lsLinux2.log`。`TODIRECTORY` 指定目标文件夹，`TOMACHINE` 指定目标机器。

上述 STAX 脚本可以用 `staf local stax execute file SampleScript.xml wait` 执行，或者通过 `java -jar STAXMon.jar` 启动 STAXMonitor 来调用。

[↑ 回页首](#)

3. 自动部署更新包

图 2. STAF(STAX)环境拓扑

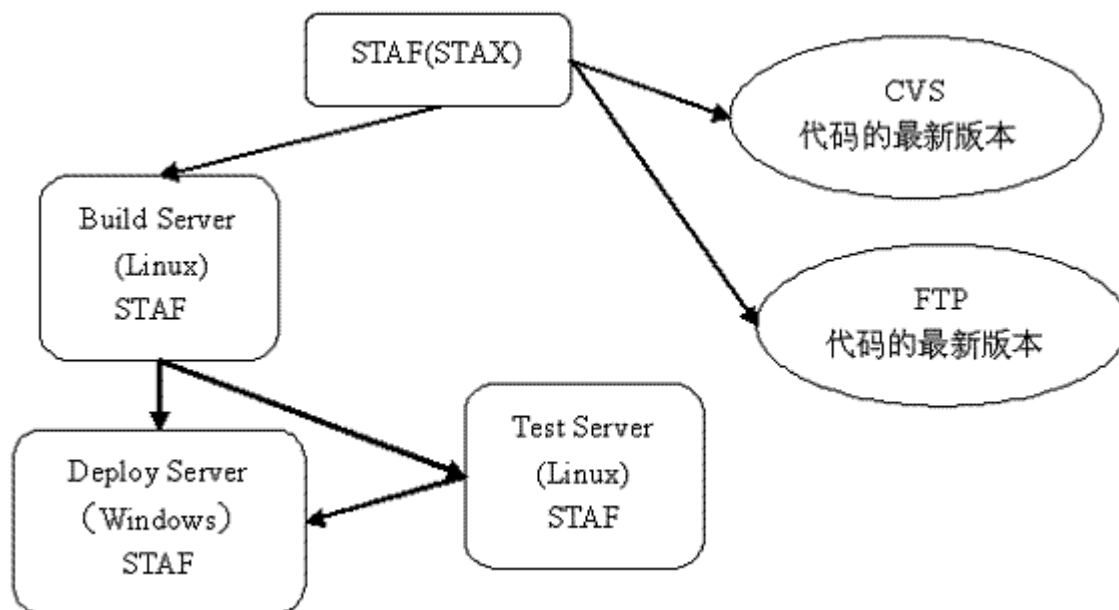


图 2 是本节将要介绍的简化的场景图。软件开发分为两部分，一部分是在 CVS 上的最新的软件源码，另外一部分是在 FTP 服务器上的执行脚本。在 STAF(STAX)自动部署更新包的过程中，STAX 需要同时从 CVS 和 FTP 上下载最新的代码和安装脚本。测试环境中，测试机器上都装有 STAF，并且在从 CVS 和 FTP 下载代码的机器上安装有 STAX。

STAX 下载完代码后，将代码拷贝到用于编译的服务器上。因为代码的编译需要特殊的环境，比如需要 WAS (WebSphere Application Server) 的环境，因此我们把 STAX 服务器

和编译服务器分开。编译服务器编译好源码之后，将其分发到部署和测试服务器上。部署服务器负责向应用程序服务器部署程序，而测试服务器则用来进行自动化测试。

本节根据这个场景介绍如何通过 STAF(STAX)来实现部署和测试的自动化。

3.1 FTP 脚本

STAF(STAX)实现了自动化测试的框架，但并没有实现具体的常用功能，比如 FTP，CVS。因此我们需要借助 FTP 命令来完成 FTP 源码的下载。自动化下载一般通过命令行实现，因此我们使用 Windows 自带的 FTP 命令来完成。

FTP 命令提供了一个参数-s，可以指定一个 FTP 脚本文件来存放将要执行的 FTP 命令。因此我们把需要执行的 FTP 命令存放到某个文件，然后通过 STAX 调用 FTP 命令实现 FTP 上源码的自动下载。

代码 4：FTP 脚本(ftpSample.conf)示例

```
open ftp.ibm.com

username

password

binary

prompt

cd /code/latest/unix

lcd C:/latest/unix

mget *
```

bye

这个 FTP 脚本表示以用户名 `username`, 密码 `password` 访问 `ftp.ibm.com`, 设置传输方式为 `binary`, 然后下载 `/code/latest/unix` 下的文件到本地目录 `C:/latest/unix`。可以通过 `ftp -s:ftpSample.conf` 来运行此脚本。

调用 `ftp` 命令的 STAX 脚本如下所示:

代码 5: 调用 FTP 命令的 STAX 脚本

```
<process>

  <location>'local'</location>

  <command>'ftp'</command>

  <parms>'-s:ftpSample.conf'</parms>

  <workdir>'C:/STAF'</workdir>

</process>
```

`process` 标签表示调用 STAF 的进程服务(`process`), `location` 表示请求被发送的目标机器, `command` 表示需要执行的进程, 而 `parms` 表示传递给进程的参数, `workdir` 表示进程运行的工作目录。

通过 FTP 脚本和 STAX 脚本, 我们可以控制 STAX 来自动下载 FTP 上的源代码。

3.2 CVS 下载

和 FTP 类似，CVS 源码下载也使用命令行的方式，但由于 CVS 服务器使用的协议不同，对 CVS 客户端的要求也不同，因此我们在这里不再介绍如何使 CVS 客户端工作的内容。

假定我们能够使用如下的命令更新 CVS 代码: cvs -

```
d :ext:username@cvs.ibm.com:/cvsroot/ checkout -d directory  
modulename.
```

根据这个 CVS 命令，调用此命令更新 CVS 代码的 STAX 脚本如下：

代码 6：调用 FTP 命令的 STAX 脚本

```
<process>
```

```
  <location>'local'</location>
```

```
  <command>'cvs'</command>
```

```
  <parms>' -d :ext:username@cvs.ibm.com:/cvsroot/ checkout -d  
directory modulename'</parms>
```

```
  <workdir>'C:/CVS'</workdir>
```

```
  <stdout>'C:/CVS/cvsupdate.log'</stdout>
```

```
</process>
```

与代码 5 不同的是，代码 6 使用了 stdout 标签，此标签表示将进程 cvs 的输出重定向到 cvsupdate.log 中，以便于我们查看 cvs 命令执行的状态和结果。

3.3 拷贝编译源码

从 CVS 和 FTP 上下载源码之后，需要将源码拷贝到编译服务器上。本节介绍如何使用 STAF 的文件传输命令以及 STAX 的循环指令。

代码 7：传输文件的 STAX 脚本

```
<script> directoryList = ['CVSDirectory', 'FTPDirectory'] </script>

<iterate var = "directory" in="directoryList">

    <testcase name = "" sourceCopy="">

        <sequence>

            <stafcmd>

                <location>'local'</location>

                <service>'fs'</service>

                <request>'copy DIRECTORY C:/Source/%s TODIRECTORY
/root/build/%s TOMACHINE

                buildserver RECURSE KEEPEMPTYDIRECTORIES' % directory %
directory</request>

            </stafcmd>

        </sequence>

    </testcase>
```

</iterate>

代码 7 拷贝 CVS 和 FTP 源码到编译服务器中。script 标签定义了一个数组 `directoryList`，这个数组有两个值，分别表示 CVS 源码目录和 FTP 目录。iterate 定义了一个顺序循环，分别从 CVS 目录和 FTP 目录拷贝文件到编译服务器中。stafcmd 标签调用 STAF 命令，此处我们调用的是 FS（文件系统）服务。copy DIRECTORY 表示我们需要拷贝整个目录到编译服务器中。如果编译服务器已经有原来的代码，为了正确起见，可以在拷贝之前使用 `fs delete entry` 命令来删除原有的文件。

拷贝文件后，需要通知编译服务器对更新后的源码进行编译。假定在编译服务器上存在用于编译源码的脚本文件 `/root/build/build.sh`，则调用此脚本文件编译源码的 STAX 脚本如代码 8 所示。

代码 8: 编译源码的 STAX 脚本

<stafcmd>

 <location>'buildserver'</location>

 <service>'process'</service>

 <request>'start command "/root/build/build.sh" username "root"
password "password"

 workdir "/root/build" wait stdout
 /root/build/build.log'</request>

</stafcmd>

代码 8 指定以用户 `root` 的身份来运行编译脚本 `build.sh`，并且将输出重定向到文件 `build.log` 中，以便分析编译运行的过程和结果。另外如果编译脚本 `build.sh` 用到某些和路径相关的命令，比如相对路径，则需要指定工作目录。`workdir` 指定工作目录为 `build.sh` 所在的目录，这相当于在 `/root/build` 目录中运行 `build.sh` 命令。

3.4 部署测试

更新包编译完成后，需要将编译之后的更新包分发到部署服务器和测试服务器，然后部署服务器部署程序，测试服务器调用测试程序来测试更新包。将更新包分发到部署和测试服务器的 STAX 脚本如代码 9 所示。

代码 9：更新包分发

```
<script> serverList = ['deployServer', 'testServer'] </script>

<iterate var = "server" in="serverList">

    <testcase name = "'buildCopy'">

        <if expr="server != 'deployServer'">

            <stafcmd>

                <location>'buildserver'</location>

                <service>'fs'</service>

                <request>'copy DIRECTORY /root/build/result TODIRECTORY
/root/build/result
```

```
        TOMACHINE %s RECURSE KEEPEMPTYDIRECTORIES' % server
</request>

    </stafcmd>

<else>

    <stafcmd>

        <location>'buildserver'</location>

        <service>'fs'</service>

        <request>'copy DIRECTORY /root/build/result TODIRECTORY
C:/build/result

        TOMACHINE %s RECURSE KEEPEMPTYDIRECTORIES' % server
</request>

    </stafcmd>

</else>

</if>

</testcase>

</iterate>
```

代码 9 使用了判断语句来判断目标机器的平台，根据目标机器的平台选择不同的文件路径。当只有两台机器时，使用 **if-else** 的好处并不明显，甚至还不如分别向 **windows** 和

linux 机器上单独拷贝方便。但考虑如下的情况，环境中大量的部署服务器和测试服务器，这时一台一台的拷贝显然很难维护，而使用 **if-else** 加上循环的方式则要方便的多。

部署测试的 **STAX** 脚本如代码 10 所示。

代码 10: 部署测试

```
<sequence>
```

```
    <stafcmd>
```

```
        <location>'deployServer'</location>
```

```
        <service>'process'</service>
```

```
            <request>'start command "C:/build/deploy.bat > deploy.log"
username "Administrator"
```

```
password "password" workdir "C:/build" wait ' </request>
```

```
    </stafcmd>
```

```
    <stafcmd>
```

```
        <location>'testServer'</location>
```

```
        <service>'process'</service>
```

```
            <request>'start command "/root/build/runtest.sh" username "root"
password "password"
```

```
workdir "/root/build" wait stdout  
/root/build/runtest.log' </request>
```

```
</stafcmd>
```

```
</sequence>
```

代码 10 中在 Windows 和 Linux 平台运行命令的方式有细微的区别，在 Windows 中我们使用 `> deploy.log` 来重定向输出，而在 Linux 中我们使用 `stdout` 来重定向输出。具体的原因将在经验教训中说明。

至此，我们已经完成了更新包下载、分发、编译、部署和测试的整个过程，根据本节提供的示例代码，读者应该能够根据自己的环境编写出适合环境的 STAX 脚本。另外，读者也可以自定义一些附加的操作，比如在更新代码之前，先把原有的代码删除；在测试完毕后，把分散于各个服务器上的日志汇总到一台集中的机器上；甚至和 CruiseControl 结合实现定时或者基于 CVS 上的代码更新来运行，以及将测试的日志发布到某台服务器上。

[↑ 回页首](#)

4.经验教训

虽然现在 STAF(STAX) 已经比较完善，但在实际使用的过程中，我们还是发现了一些问题。本节介绍这些问题以及解决或者避免这些问题的方法，使读者在碰到这些问题时能够及时的解决它们。

4.1 使用 STAF CMD 的 process 服务，不要使用 STAX 的 process 标签

为了编写 STAX 脚本方便，STAX 定义了 process 标签用来调用 STAF 中的进程 (process) 服务。但在使用过程中，发现 STAX 的 process 标签在某些情况下存在着一定的问题，其所调用的进程不能返回。代码 11 的 STAX 脚本就是这样一个例子。

代码 11: process 标签不能返回

```
<process>

  <location>'linuxServer' </location>

  <command>'ls' </command>

  <parms>'-l' </parms>

</process>
```

代码 11 调用 Linux 机器上的 ls 命令，并且传给 ls 命令 -l 参数。使用 STAXMonitor 执行此脚本，任务始终无法返回。因此推荐使用 stafcmd 标签直接调用 STAF 服务，如代码 12 所示。

代码 12: 修改后的任务

```
<stafcmd>
```

```
<location>'linuxServer'</location>
```

```
<service>'process'</service>
```

```
<request>'start command "ls" parms "-l" wait '</request>
```

```
</stafcmd>
```

4.2 在 Windows 平台上不要使用 STDOUT 重定向输出

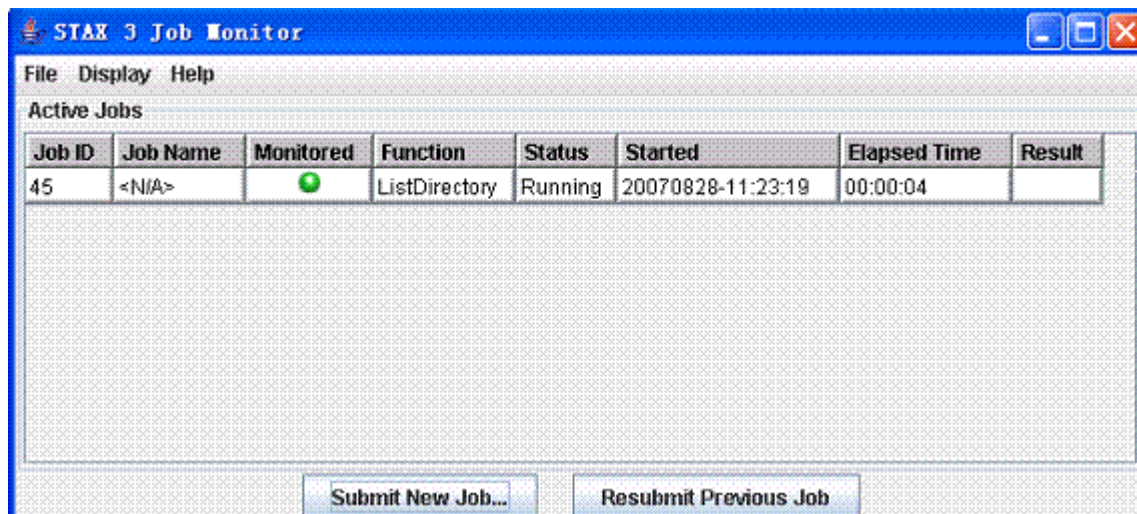
STAF 使用 STDOUT 来为启动的进程重定向输出，类似于>参数，比如 `ls -l > ls.log`。但在 Windows 平台使用中，我们发现使用 STDOUT 会带来一些问题。如果调用的进程为批处理文件，并且此批处理文件中包含某些特定的功能，比如 `xcopy`，则 `xcopy` 将不会工作。另外一些检查目录和文件的命令也不能与 STDOUT 共存。在 Linux 环境中并不存在这样的问题。因此，如果需要在 Windows 平台中使用重定向输出的功能时，建议使用>来重定向输出。

4.3 使用 STAXMonitor 监控任务的执行情况

对于 STAF 和 STAX 新手来说，尽可能使用 STAXMonitor 来监控 STAX 任务的执行情况。STAXMonitor 为我们提供了足够详细的信息，比如测试用例的执行结果，任务执行的消息，当前执行的命令。使用 STAXMonitor 有助于我们对正在进行的任务进行分析并且监控其执行情况和结果。

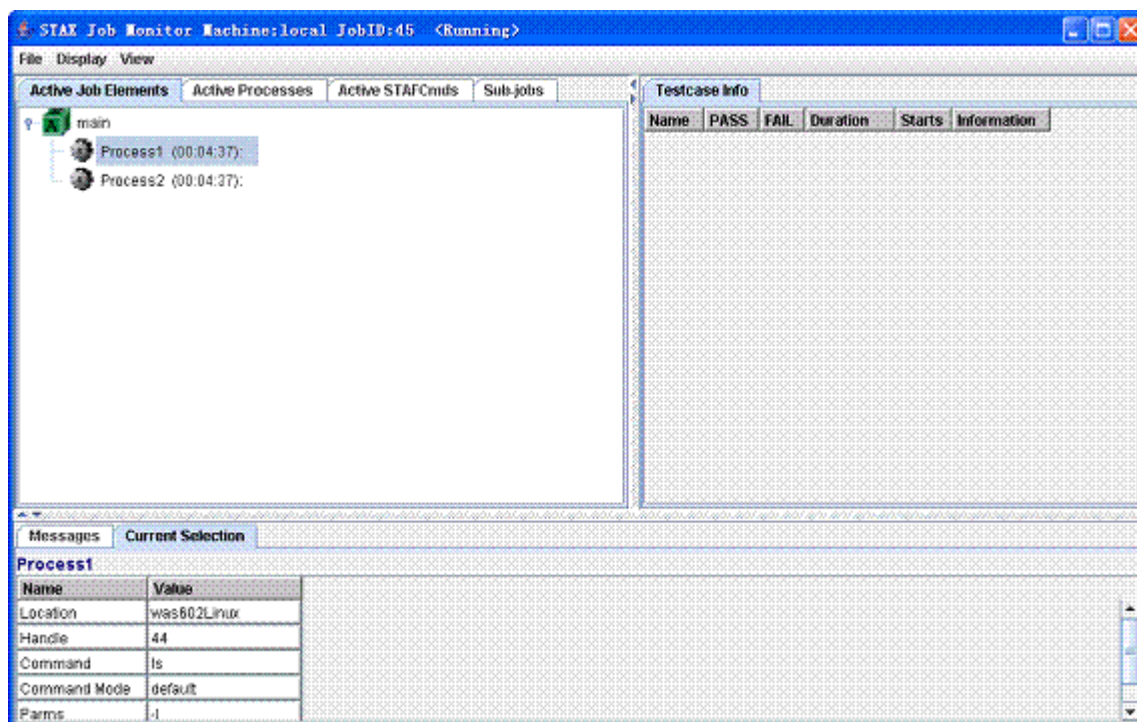
STAXMonitor 在 STAX 安装文件中，可以用 `java -jar STAXMon.jar` 来启动 STAXMonitor。STAXMonitor 的界面如图 3 所示。

图 3. STAXMonitor 运行界面



STAXMonitor 会显示当前正在运行的 STAX 任务，任务号，任务名字，功能，开始时间，执行时间以及结果。Monitored 表示是否正在使用 STAXMonitor 来监控任务。右键单击任务，然后选择 Start monitoring，将出现如图 4 所示的监控界面。

图 4. STAXMonitor 监控界面



监控界面会显示正在运行的进程或者 STAF 命令，命令的详细信息，比如开始时间、进程或者命令的参数，状态等。另外还显示测试用例的状态。通过 STAXMonitor，我们可以很好的监控 STAX 任务的执行情况。

4.4 将 STAF 注册为 Windows 平台上的服务

STAF 并没有提供开机自动启动的功能，在 Windows 平台上，只有当某个用户登录后，才会启动 STAF。这对于自动化测试的环境来说不是一个好消息。因此我们需要自动启动 STAF 的功能，这在 Linux 上比较简单，只要在 `/etc/rc.d/rc.local`（如果是 SuSE Linux，就是 `/etc/rc.d/boot.local`）中加入 STAF 的启动命令 `/usr/local/staf/bin/STAFProc &` 就可以了。Windows 平台上就没有那么方便，因此本小节介绍如何将 STAF 注册为 Windows 的服务，以便能开机自动重启。

1. 首先使用 `instsrv` 命令注册一个基本的服务 STAF：`instsrv STAF c:/winnt/system32/srvany.exe`。
2. 打开注册表编辑器(`regedit`)，找到键值 `My Computer/HKEY_LOCAL_MACHINE/SYSTEM/ControlSet001/Services/STAF`。在 STAF 下创建一个键，名字为 `Parameters`。
3. 在 `Parameters` 键下面，创建一个字符串值（String Value），名字为 `Application`，值为 STAFProc 的完整路径，比如 `C:/STAF/bin/STAFProc.exe`。
4. 使用命令 `services.msc` 启动 Windows 服务窗口，找到 STAF，右键选择属性，然后定位到登录窗口，选择“允许服务与桌面交互”。
5. 使用命令 `net start staf` 或者重启机器来启动 STAF 服务。
6. 使用命令 `staf local service list` 来验证 STAF 是否已经成功启动。