**AAS-V9.0 测试用例**

# 1. 基础功能测试用例

## 1.1 安装运行卸载测试

### 1.1.1介质完整性测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **介质完整性测试** | **测试用例标识** | | **GHtest01** | |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0安装完成后介质的完整性，包括文档，各个实例等介质 | | | | |
| **前置条件** | | 已经安装了Apusic9.0应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 打开Apusic9.0安装目录，查看%APUSIC\_HOME%\docs\目录下是否存在chm,html,pdf版本的文档，并查看文档内容是否存在错误。 | | | hm,html,pdf版本的文档存在并且不存在错误 | |  |
| 2 | 打开Apusic9.0安装目录，查看%APUSIC\_HOME%\domains\下mydomain以及sample目录的完整性 | | | 文件完整 | |  |
| Apusic实例对应的目录结构如下：   1. config目录：Apusic9.0配置文件目录 2. bin目录：启动脚本目录 3. lib目录：实例jar包存储目录 4. upload目录：上传应用目录 5. applications目录：本地部署应用目录 6. logs目录：日志存储目录 7. store目录：存储ejb,session等的目录 8. deploy目录：应用部署后解压目录   sample在applications目录比mydomain多了jpa.ear，jsp.ear，mdb.ear，servlet.ear，stateful.ear，stateless.ear应用。  确认目录下文件正确并且内容不存在错误。 | | | | | |
| 3 | 确认xml格式文件使用notepad打开所有文件并阅读文件不存在错误内容或格式； | | | 没有乱码内容和错误的格式 | |  |
| 4 | 参测人员签字 | | |  | | |

### 1.1.2 安装运行

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Windows下应用服务器安装** | | **测试用例标识** | **GHtest02** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器在Windows下安装的正确性 | | | | |
| **前置条件** | | 准备zip格式的应用服务器安装文件  设置好系统的JDK环境 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 把zip格式的安装文件拷贝到硬盘的对应目录下，如D:\Apusic9目录下 | | 拷贝成功 | | |  |
| 2 | 解压zip格式的文件 | | 执行成功，正确解压到硬盘中 | | |  |
| 3 | 到解压目录下的domains\ mydomain\bin目录，运行startapusic.cmd命令，启动应用服务器 | | 后台出现应用服务就绪字样  应用服务器启动成功 | | |  |
| 4 | 通过访问<http://ip:6888> | | 出现应用服务器首页，应用服务器安装完成 | | |  |
| 5 | 参测人员签字 | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Linux应用服务器安装** | | **测试用例标识** | **GHtest03** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器在Linux下安装的正确性 | | | | |
| **前置条件** | | 准备zip格式的应用服务器安装文件  设置好系统的JDK环境 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 把zip格式的安装文件拷贝到硬盘的对应目录下，如/home目录下 | | 拷贝成功 | | |  |
| 2 | 解压zip格式的文件 | | 执行成功，正确解压到硬盘中 | | |  |
| 3 | 到解压目录下的domains/ mydomain/bin目录，运行startapusic命令，启动应用服务器 | | 后台出现应用服务就绪字样  应用服务器启动成功 | | |  |
| 4 | 通过访问<http://ip:6888> | | 出现应用服务器首页，应用服务器安装完成 | | |  |
| 5 | 参测人员签字 | |  | | | |

### 1.1.3 卸载及再安装

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Windows系统下应用服务器卸载及再安装** | | **测试用例标识** | **GHtest04** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器在Windows下的进行卸载的正确性，卸载后是否可以再次安装 | | | | |
| **前置条件** | |  | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 使用ctrl+c，或在进程管理器中停掉应用服务器进程； | | 进程能够正常停止 | | |  |
| 2 | 直接删除对应的应用服务器安装目录 | | 能够成功删除安装目录，应用服务器卸载成功 | | |  |
| 3 | 重新解压安装包安装 | | 解压成功 | | |  |
| 4 | 到解压目录下的/domains/ mydomain/bin目录，运行startapusic.cmd命令，启动应用服务器 | | 启动成功 | | |  |
| 5 | 参测人员签字 | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Linux系统下应用服务器卸载及再安装** | | **测试用例标识** | **GHtest05** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器在Linux下的进行卸载的正确性，卸载后是否可以再次安装 | | | | |
| **前置条件** | |  | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 使用ctrl+c，kill -9 +pid停掉应用服务器进程； | | 进程能够正常停止 | | |  |
| 2 | 直接删除对应的应用服务器安装目录 | | 能够成功删除安装目录，应用服务器卸载成功 | | |  |
| 3 | 重新解压安装包安装 | | 解压成功 | | |  |
| 4 | 到解压目录下的/domains/ mydomain/bin目录，运行startapusic命令，启动应用服务器 | | 启动成功 | | |  |
| 5 | 参测人员签字 | |  | | | |

### 1.1.4 启动和关闭

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Windows系统下应用服务器启动和关闭** | | **测试用例标识** | **GHtest06** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器在Windows下的启动和停止 | | | | |
| **前置条件** | | 应用服务器部署成功 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 到%APUSIC\_HOME%/domains/ mydomain/bin目录，运行startapusic.cmd命令，启动应用服务器 | | 进程能够正常启动 | | |  |
| 2 | 使用ctrl+c关闭应用服务器 | | 关闭成功 | | |  |
| 3 | 在进程管理器中找到该进程进行结束进程操作 | | 关闭成功 | | |  |
| 4 | 使用脚本关闭进程：  在%APUSIC\_HOME%/domains/ mydomain/bin目录下，执行  Stopapusic.cmd admin password iiop://ip:port | | 关闭成功 | | |  |
| 5 | 参测人员签字 | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Linux系统下应用服务器启动和关闭** | | **测试用例标识** | **GHtest07** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器在Linux下的启动和停止 | | | | |
| **前置条件** | | 应用服务器部署成功 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 进入  ${APUSIC\_HOME}/domains/ mydomain/bin目录，运行startapusic命令，启动应用服务器 | | 进程能够正常启动 | | |  |
| 2 | 使用ctrl+c关闭应用服务器 | | 关闭成功 | | |  |
| 3 | 执行kill -9 pid(应用服务器进程号)命令 | | 关闭成功 | | |  |
| 4 | 使用脚本关闭进程：  在${APUSIC\_HOME}/domains/ mydomain/bin目录下，执行  ./Stopapusic admin password iiop://ip:port  使用ps-ef查看进程是否关闭成功 | | 关闭成功 | | |  |
| 5 | 参测人员签字 | |  | | | |

## 1.2 管理域

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Windows下管理域** | | **测试用例标识** | **GHtest08** | |
| **测试用例说明（方法、目的）** | | 应用服务器域的管理包括创建域，删除域，创建域模板三部分  应用服务器默认自带了两个域，sample和mydomain | | | | |
| **前置条件** | | Windows环境下应用服务器安装成功 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 创建域：  a.在%APUSIC\_HOME%/bin下执行config.cmd  b.选择1-创建一个新的Apusic域  c.安装提示填写域的相关信息，直到完成创建 | | 域创建成功，创建成功后  再%APUSIC\_HOME%/domains下查看到 | | |  |
| 2 | 删除域：  a.在%APUSIC\_HOME%/bin下执行config.cmd  b.选择2-删除一个现有的Apusic域  c.选择需要删除的域完成删除 | | 删除域成功，在%APUSIC\_HOME%/domains下查看到删除成功 | | |  |
| 3 | 构建域模板：  a.在%APUSIC\_HOME%/bin下执行config.cmd  b.选择3-构建一个新的Apusic域模板  c.根据提示填写域模板的信息完成域模板的创建 | | 构建域模板成功，可以在  %APUSIC\_HOME%/templates目录下查看到新建的域模板，格式为jar包 | | |  |
| 5 | 参测人员签字 | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Linux下管理域** | | **测试用例标识** | **GHtest09** | |
| **测试用例说明（方法、目的）** | | 应用服务器域的管理包括创建域，删除域，创建域模板三部分 | | | | |
| **前置条件** | | Linux环境下应用服务器安装成功 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 创建域：  a.在${APUSIC\_HOME}/bin下执行config  b.选择1-创建一个新的Apusic域  c.安装提示填写域的相关信息，直到完成创建 | | 域创建成功，创建成果功  再${APUSIC\_HOME}/domains下查看到 | | |  |
| 2 | 删除域：  a.在${APUSIC\_HOME}/bin下执行config  b.选择2-删除一个现有的Apusic域  c.选择需要删除的域完成删除 | | 删除域成功，在${APUSIC\_HOME}/domains下查看到删除成功 | | |  |
| 3 | 构建域模板：  a.在${APUSIC\_HOME}/bin下执行config  b.选择3-构建一个新的Apusic域模板  c.根据提示填写域模板的信息完成域模板的创建 | | 构建域模板成功，可以在  ${APUSIC\_HOME}/templates目录下查看到新建的域模板，格式为jar包 | | |  |
| 5 | 参测人员签字 | |  | | | |

## 1.3 系统备份

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **系统备份与还原** | | **测试用例标识** | **GHtest10** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器可以进行配置文件的备份与还原 | | | | |
| **前置条件** | | 应用服务器安装成功，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 登录管控，切换到系统备份界面 | |  | | |  |
| 2 | 在系统备份界面点击【备份】按钮进行备份 | | 生成一个zip文件，里面备份了当前域config目录下所有的而配置文件，并且存储在${APUSIC\_HOME}/domains/${domain\_name}/store/config目录下 | | |  |
| 3 | 在系统备份界面相应的备份文件栏内点击【还原】操作 | | 将备份文件里面所有的文件还原到当前域下面的/config目录下 | | |  |
| 5 | 参测人员签字 | |  | | | |

## 1.4 管理监控测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **管理监控功能测试** | **测试用例标识** | | **GHtest13** | |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0管理控制台对部署应用以及JVM资源等信息是否具有监控功能 | | | | |
| **前置条件** | | 已经安装Apusic9.0应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 打开Apusic9.0安装目录，打开%APUSIC\_HOME%\mydomain\bin目录下，运行startapusic，设置初始化密码 | | | 应用服务器启动成功 | |  |
| 2 | 打开浏览器，访问<http://ip:port/admin>打开管理控制台首页 | | | 成功进入到应用服务器管理控制首页 | |  |
| 3 | 输入默认用户名密码admin/密码（步骤一设置的密码），登陆管理控制台 | | | 成功登陆 | |  |
| 4 | 查看管理监控台对应用等资源的监控项是否存在，可监控包括运行中间件中的各种服务、虚拟机运行状况、硬件资源以及用户所感兴趣的各种资源 | | | 管理控制台包含所需要的监控项，包括：应用监控，数据源监控，会话监控，事务监控，JVM监控，http链接监控等。 | |  |
| 1. 可以对应用进行监控配置 2. 可以对数据源进行监控 3. 可以对session会话进行监控 4. 可以对事务进行监控 5. 可以对JVM进行监控 6. 可以对http链接进行监控 7. 除此之外，还可以对Apusic9.0的配置项等进行配置等操作 | | | | | |
| 5 | 参测人员签字 | | |  | | |

## 1.5 数据库连接服务

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **数据库连接服务** | | **测试用例标识** | **GHtest14** | |
| **测试用例说明（方法、目的）** | | 以daytrader应用来测试数据库连接服务，包括moracle, mysql，MSSQLServer,DB2,人大金仓等数据库等数据库 | | | | |
| **前置条件** | |  | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 进入安全管理控制台，点击“数据源”，添加数据源， 名称为：TradeDataSource；JNDI名称：jdbc/TradeDataSource  ，数据库类型可以选择为oracle,mysql, DB2等 | | 1.可以创建数据源，创建的数据源配置保存在datasources.xml，  2.应用可以连接数据源，可以通过应用对数据库进行插入、查询、修改、删除等操作，  3.可以修改数据库连接池的参数，  4.可以对数据源进行启动、停止、删除等操作 | | |  |
| 2 | 看%APUSIC\_HOME%\domains\mydomain\config下数据源配置文件datasources.xml文件，其中是否包含正确的已经新建的数据源的配置信息；在数据源界面点击配置数据源test,查看连接池信息是否正确。 | |
| 3 | 部署daytrader应用 | |
| 4 | 登录daytrader应用，进行插入、查询、修改、删除等操作。 | |
| 5 | 修改数据源TradeDataSource的参数配置，修改连接池参数，如最大连接数设置为100，重启数据源，300并发用户压力测试，检查最大连接数是否会超过100。 | |
| 6 | 选择数据源  jdbc/TradeDataSource，其状态为启动；点击停止并确定停止数据源 | |  |
| 7 | 选择数据源  jdbc/TradeDataSource，其状态为停止；点击启动并确定停止数据源 | |  |
| 8 | 点击删除数据源jdbc/TradeDataSource，界面删除后查看datasources.xml文件是否已经移除数据源test的配置文件 | |  | | |  |
| 9 | 参测人员签字 | |  | | | |

## 1.6 指定应用启动顺序

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **指定应用启动顺序** | **测试用例标识** | | **GHtest14** | |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic应用服务器可以指定应用启动的先后顺序 | | | | |
| **前置条件** | | 已经安装了apusic应用服务器9.0 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管理控制台，在应用管理界面部署两个应用A，B | | | 在启动日志中可以查看到A应用启动后才启动B应用 | |  |
| 2 | 设置A,B应用加载顺序分别为0,1  也可以直接修改${domain\_home}/config/server.xml中应用的loadon值。  如：  <application name="EJB-client.ear" base="applications/EJB-client.ear" start="auto" loadon="0"/> | | |  | |  |
| 3 | 重启应用服务器 | | | | |  |
| 4 | 参测人员签字 | | |  | | |

## 1.7 Apusic工具测试

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Apusic9.0工具功能测试** | | **测试用例标识** | **GHtest26** | | | | |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0安装完成后bin目录下工具功能是否正确，包含apclient,appctl,，adminmain，jspc等工具 | | | | | | | |
| **前置条件** | |  | | | | | | | |
| **测试执行步骤** | | | | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** | | | |
| 1 | 部署测试应用EJB-client.ear测试应用，执行$APUSIC\_HOME$\bin目录下apclient命令，使用用户名密码admin/password(第一次启动时设置的密码)查看能否登陆应用服务器 | | 能够登陆apusic应用服务器 | | |  | | | |
| (1)将appclient.jar拷贝到AAS的根目录；  (2)执行AAS根目录下bin目录下的apclient.cmd命令，参数为：  apclient -url iiop://localhost:6888 -jar appclient.jar  （3）填写用户名密码admin/admin登陆 | | | | | | | | |
| 2 | 执行$APUSIC\_HOME$\bin目录下adminmain命令，可以查询服务或者可配置项 | | 可以使用adminmain命令查询服务或者可配置项 | | | |  | | |
| 用法：adminmain [选项] [命令]  选项包括:  -s <服务器主机名> (默认为 localhost:6888)  -p <管理员密码> (默认为 admin)  命令包括:  list [服务名] [配置项名]  <服务名><配置项名> [<参数>...]  范例:  1. 查询 localhost:6888 可用服务列表  adminmain -s localhost:6888 -p admin list  2. 查询 localhost:6888 http服务可用配置项列表  adminmain -s localhost:6888 -p admin list http  3. 查询 localhost:6888 http服务可用配置项 KeepAlive 的说明  adminmain -s localhost:6888 -p admin list http KeepAlive  4. 查询 localhost:6888 http服务可用配置项 KeepAlive 的值  adminmain -s localhost:6888 -p admin http KeepAlive  5. 设置 localhost:6888 http服务可用配置项 KeepAlive 的值为 false  adminmain -s localhost:6888 -p admin http KeepAlive false | | | | | | | | |
| 3 | 执行$APUSIC\_HOME$\bin目录下jspc命令，可以对JSP进行预编译 | | 可以对指定应用的JSP进行预编译操作 | | | | |  | |
| 1. 编辑$APUSIC\_HOME/bin/jspc.xml。需要修改的几个属性说明如下：   <!-- 需预编译的web模块路径 -->  <propertyname="src"value="${apusic.home}/domains/mydomain/applications/default/public\_html"/>  将value属性修改为需预编译的web模块的路径，默认是mydomain下的default应用  <!-- 预编译的class文件的存放路径 -->  <propertyname="dest"value="${apusic.home}/domains/mydomain/applications/default/public\_html/WEB-INF/classes"/>   1. 在命令行提示符下进入$APUSIC\_HOME/bin目录，执行jspc即可开始预编译 | | | | | | | | |
| 4 | 执行$APUSIC\_HOME$\bin目录下appctl 命令，可以对应用进行控制 | | 能够通过appctl查看应用状态以及安装，卸载应用等 | | | | | |  |
| 用法: appctl [选项] <命令>  选项包括:  -s <服务器主机名> (默认为 localhost:6888)  -p <管理员口令> (如未提供则要求从控制台输入)  命令包括:  install <应用名><应用档案文件(.ear)> [-noverify] [-remote]  [-config <配置文件名>] [-vhost <虚拟主机名>] [-context <虚拟目录名  >]  uninstall <应用名>  list  start <应用名>  stop <应用名>  restart <应用名>  list-module <应用名>  list-role <应用名>  assign-role <应用名><角色名> [[-u] <用户>...] [-g <组>...]  unassign-role <应用名><角色名> | | | | | | | | |
|  | 参测人员签字 | |  | | | | | | |

## 1.8 日志功能测试

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **日志功能测试** | | **测试用例标识** | | **GHtest30** | |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0安装完成后日志功能的正确性 | | | | | |
| **前置条件** | | 已经安装了Apusic应用服务器9.0 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** |
| 1 | 打开%APUSIC\_HOME%\domains\mydomain\config\目录，修改logging.xml文件中默认logger:apusic的配置，修改为DEBUG;重新启动apusic应用服务器，查看启动日志是否包含启动时的DEBUG信息 | | | | 修改后重新启动日志会打印DEBUG日志信息 | |  |
|  | | | | | | |
| 2 | 打开%APUSIC\_HOME%\domains\mydomain\config\目录，修改logging.xml文件，删除fileHandler的配置文件；重新启动apusic应用服务器，查看domains\mydomain\logs下是否产生日志文件 | | | | 不再产生日志文件 | |  |
| 删除的FileHandler配置如下：  <handler class="java.util.logging.FileHandler">  <property name="level" value="ALL"/>  <property name="limit" value="1000000"/>  <property name="count" value="10"/>  <property name="pattern" value="%h/logs/apusic.log"/>  <property name="append" value="true"/>  <formatter class="com.apusic.logging.SimpleFormatter"/>  </handler> | | | | | | |
| 3 | 参测人员签字 | |  | | | | |

## 1.9 JNDI 命名服务

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Naming JNDI Context功能测试** | **测试用例标识** | | **GHtest33** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器对命名服务的支持 | | | | |
| **前置条件** | | 已经安装并可以正常访问Apusic应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 建立数据源JNDI名称：jdbc/TradeDataSource，选择为oracle或mysql数据库类型，指向daytrader数据库中。 | | | 数据库建立连接成功，并且能够连接上 | |  |
| 2 | 部署daytrader应用 | | | 应用部署成功 | |  |
| 3 | 浏览器访问daytrader应用  <http://ip:port/daytrader/>，切换到Configuration标签>，点击标准的“(Re)-create DayTrader Database Tables and Indexes”， 查看daytrader数据库中表是否新建成功 | | | daytrader数据库中新建了accountejb，accountprofileejb，holdingejb，keygenejb，orderejb，quoteejb | |  |
| 4 | 点击“(Re)-populate DayTrader Database”，查看表中是否插入了相应的数据 | | | daytrader数据库的几个表中插入了相应的数据 | |  |
| 5 | 参测人员签字 | | |  | | |

## 1.10 类库管理

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **类库管理** | **测试用例标识** | | **GHtest34** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器对应用的类库管理的支持，可以对对应用进行类的新增，删除，查看等操作 | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 打开Apusic应用服务器监控管理平台“应用管理”操作界面，选择某个应用后点击“类库管理”按钮对该应用的类库进行操作 | | | Apusic应用服务器支持对应用的类库进行新增、删除等管理 | |  |
| 2 | 进入类库管理界面，可以对类库进行增加删除等操作 | | |
| 3 | 点击“新增”按钮，进入到新增类库窗口，选择好需要的文件，点击“选择“按钮，新增成功后在类库管理列表中显示 | | |
| 4 | 删除单个类库：选中某个类库，点击后面的删除图标按钮，删除该类库 | | |
| 5 | 批量删除类库：勾选需要删除的类库，点击左上方的删除按钮图标进行删除 | | |
| 6 | 参测人员签字 | | |  | | |

# 2. 标准符合性测试

## 2.1 多路复用

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **多路复用** | **测试用例标识** | | **GHtest11** | |
| **测试用例说明（方法、目的）** | | 测试Apusic应用服务器支持多路复用技术 | | | | |
| **前置条件** | | Apusic应用服务器已安装好 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 多路复用的配置段在配置文件*apusic.conf*中，此文件在域主目录下的config目录下。  缺省的在*apusic.conf*配置文件中的多路复用配置段如，多路服务端口默认端口号为6888  <SERVICE  CLASS="com.apusic.net.Muxer"  >  <ATTRIBUTE NAME="Port" VALUE="6888"/>  <ATTRIBUTE NAME="Backlog" VALUE="50"/>  <ATTRIBUTE NAME="Timeout" VALUE="300"/>  <ATTRIBUTE NAME="MaxWaitingClients" VALUE="500"/>  <ATTRIBUTE NAME="WaitingClientTimeout" VALUE="5"/>  <ATTRIBUTE NAME="SSLEnabled" VALUE="True"/>  <ATTRIBUTE NAME="SecurePort" VALUE="6889"/>  <ATTRIBUTE NAME="KeyStore" VALUE="config/sslserver"/>  <ATTRIBUTE NAME="KeyPassword" VALUE="keypass"/>  </SERVICE> | | | 客户应用只需向多路复用端口发出请求，服务器通过分析客户请求，将其转发给相关的服务或是拒绝此请求 | |  |
| 2 | 部署应用，使用多路复用端口端口访问应用，确认客户端是否可以只通过多路复用端口域服务端通讯 | | |
| 3 | Apusic应用服务器的Web服务、JMS服务、JNDI服务，在一个端口上进行复用；另外，可配置一个安全端口，此端口与上一端口功能一样，只是使用进行SSL通讯。 | | |  | |
| 4 | 参测人员签字 | | |  | | |

## 2.2 snmp协议的监控

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | snmp协议的监控 | | **测试用例标识** | **GHtest12** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器支持通过SNMP协议进行监控 | | | | |
| **前置条件** | | Apusic应用服务器已安装好 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 2 | 具体操作请参照以下文档 | |  | | |  |
| 5 | 参测人员签字 | |  | | | |

## 2.3 web应用

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试Web应用-支持jsp，**  **servlet测试** | | **测试用例标识** | | **GHtest16** | |
| **测试用例说明（方法、目的）** | | 测试JSP内置对象、EL、Include及Servlet基础功能 | | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** |
| 1 | 1. 把testweb.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/testweb/index.jsp> 4. 分别点击页面显示后的测试各个功能的连接，包括：   [jsp内置对象exception测试](http://localhost:6888/testweb/testException.jsp)  [jsp内置其他对象测试](http://localhost:6888/testweb/testObject1.jsp?requestTest=request%20test%20pass)  [jsp include指令测试](http://localhost:6888/testweb/testInclude.jsp)  [jsp el支持测试](http://localhost:6888/testweb/testel1.jsp)  [servlet 基本方法使用测试](http://localhost:6888/testweb/TestServlet1?requestTest=request%20test%20pass) | | | | 能正确返回结果 | |  |
| 2 | 参测人员签字 | |  | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试Web应用-支持jsp** | | **测试用例标识** | | | **GHtest15** |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0安装完成后是否具有以下功能：支持Scripting Elements；Jsp页面支持基本指令，如page指令，include指令；支持EL表达式；JSP页面热部署；支持自定义标签以及JSTL（Taglib）。 | | | | | |
| **前置条件** | | 已经安装了apusic应用服务器9.0 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** | |
| 1 | 启动Apusic9.0应用服务器，保证Apusic9.0能够正常访问。 | | 执行成功 | | | 通过 | |
| 2 | 部署应用test-webapp.war测试对Scripting Elements的支持性，对Jsp页面进行编译，并正确显示页面内容。页面内容包括Declarations、Scriptlets、EL Expressions | | 每一步执行结果正确 | | | 通过 | |
| 1. 部署应用test-webapp.war,并访问http://localhost:6888/test-webapp/tld.jsp页面   （2）查看页面JSP代码与页面，验证Declarations、Scriptlets、EL Expressions。对应JSP页面上的：  <div>  **<%!**String str = "This is a string!";**%>  <%** out.println("<br/>");  for (int i = 1; i < 10; i++) {  for (int j = 1; j < 10; j++) {  out.println(j);  }  out.println("<br/>");  }**%>** <center>  100-50=**${**100-50**}** <br>  8\*9=**${**8\*9**}** <br>  9/3=**${**9/3**}**<br>  **${**"123"+9+3**}** </center> </div> <div>  ===========================out标签===================<br/>  <**c:out** value="**${**name**}**" default="<h1>out测试<h1>" escapeXml="false"></**c:out**>   =======================choose标签=====================<br/>  <**c:choose**>  <**c:when** test="**${**10>5**}**">  10>5<br />  </**c:when**>  <**c:when** test="**${**6>2**}**">  6>2<br />  </**c:when**>  <**c:otherwise**>  条件都不成立  </**c:otherwise**>  </**c:choose**>  ========================自定义标签======================<br/>  <**p:test** hidden="false">  <span>hidden=false显示内容</span></br>  </**p:test**>  ========================引用开源标签=====================<br/>  **${**es:encodeForHTML("<script>alert(\"hello\")</script>")**}**<br/>  =========================forEach=======================<br/>  **<%** //List数据准备  List<Student> list = new ArrayList<Student>();  list.add(new Student("name001", 18));  list.add(new Student("name002", 19));  list.add(new Student("name003", 20));  pageContext.setAttribute("list", list);//放入域中   //Map数据准备  Map<String, Student> map = new HashMap<String, Student>();  map.put("001", new Student("name001", 18));  map.put("002", new Student("name002", 19));  map.put("003", new Student("name003", 20));  pageContext.setAttribute("map", map);//放入域中   //array数据准备  Student[] array = new Student[]{  new Student("name001", 18),  new Student("name002", 19),  new Student("name003", 20)};  pageContext.setAttribute("array", array);//放入域中  **%>** <**c:forEach** var="student" begin="0" end="2" items="**${**list**}**" step="1" varStatus="varSta">  name:**${**student.name**}**---age:**${**student.age**}**<br/>  </**c:forEach**>  <**c:forEach** var="student" begin="0" end="2" items="**${**map**}**" step="1" varStatus="varSta">  key:**${**student.key**}**---name:**${**student.value.name**}**---age:**${**student.value.age**}**<br/>  </**c:forEach**>  <**c:forEach** var="student" begin="0" end="2" items="**${**array**}**" step="1" varStatus="varSta">  name:**${**student.name**}**---age:**${**student.age**}**<br/>  </**c:forEach**>  <**c:forEach** var="student" items="**${**array**}**">  name:**${**student.name**}**---age:**${**student.age**}**<br/>  </**c:forEach**>  =======================forTokens====================<br/>  **<%** //数据准备  String str = "a-b-c=d=e";  pageContext.setAttribute("str", str);  **%>** <**c:forTokens** var="item" items="**${**str**}**" delims="-=">  **${**item**}**<br/>  </**c:forTokens**> </div>  （3）验证Declarations、Scriptlets、Expressions分别显示正常  （4）查看目录 deploy/test-webapp/tmpfiles/jsp/\_jspx下是否存在\_tld\_\_jsp.class文件 | | | | | | |
| 3 | 部署应用test1.war来测试对Jsp页面支持基本指令的支持 | | 执行成功 | | | 通过 | |
| 1. 访问<http://localhost:6888/test1/testInclude.jsp>，查看页面显示   页面显示为使用<include file="include html"> 使用<jsp:include page="include jsp">   1. 查看testInclude.jsp源代码，其中用page指令定义了使用的是java语言   <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ include file="include.html" %> | | | | | | |
| 4 | 部署并访问应用wow.ear来验证Apusic9.0对JSP页面热部署功能的支持 | | 每一步执行结果均正确 | | 通过 | | |
| 1. 访问http://192.168.10.33:6888/wow/login.jsp查看页面显示   页面显示为：“hello123”   1. 将wow.ear包中的web.war包下的login.jsp中的“hello123”修改为“hellotest”后保存应用，并重新部署wow.ear，这里不重启应用服务器 2. 刷新页面，页面显示为英文：“hellotest” | | | | | | |
| 5 | 参测人员签字 | |  | | | | |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试Web应用-支持servlet** | | **测试用例标识** | | **GHtest16** | | |
| **测试用例说明（方法、目的）** | | 测试应用服务器对servlet功能的支持，包括Servlet生命周期管理，请求与Servlet的映射，请求分发功能。 | | | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器 | | | | | | |
| **测试执行步骤** | | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | | **实际结果** |
| 1 | 部署应用test-webapp.war并访问，来测试servlet的生命周期管理，Servlet生命周期管理着Servlet的装载，实例化，初始化，响应客户请求和消亡。当容器启动或者请求调用Servlet时，容器使用一般类装载器将Servlet装载进来，并且对其实例化。在响应客户请求之前，必须对Servlet进行初始化（读取配置文件、初始化资源等）。 | | | |  | | | 通过 |
| 1. 访问http://localhost:6888/test-webapp/test 2. 查看日志输出：   2012-03-15 10:24:01 信息 [con.out] servlet is inited...  2012-03-15 10:24:01 信息 [con.out] do filter..  2012-03-15 10:24:01 信息 [con.out] servlet is processing request...  2012-03-15 10:24:04 信息 [con.out] do filter..  2012-03-15 10:24:04 信息 [con.out] servlet is processing request...  2012-03-15 10:25:23 信息 [con.out] do filter..  2012-03-15 10:25:23 信息 [con.out] servlet is processing request...  2012-03-15 10:28:15 信息 [con.out] sevlet is desttoryed  2012-03-15 10:28:15 信息 [con.out] filter is destoryed.. | | | | | | | |
| 2 | 通过部署并访问wow.ear来测试分发请求功能，即在应用中，需要一个Web组件对请求（request）作预处理，然后利用请求分发功能将请求转移到另一个Web组件，并产生响应（response）。 | | 每一步执行结果均正确 | | | | 通过 | |
| 1. 访问<http://192.168.10.33:6888/wow/index1.jsp>,查看页面内容   页面内容显示为：this is the hello.jsp conetent!   1. 查看index1.jsp和hello.jsp的源代码，显示的内容为hello.jsp中的，在index1.jsp中定义转发，代码如下：   request.getRequestDispatcher("hello.jsp").forward(request, response); | | | | | | | |
| 3 | 参测人员签字 | |  | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试Web应用部署、反部暑、停止、重启** | | **测试用例标识** | **GHtest17** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器支持应用的部署、卸载、停止、启动等功能 | | | | |
| **前置条件** | | 应用服务器已安装并正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 进入管理控制台，点击“应用管理”，点击“部暑应用”，根据提示完成部暑；  或者直接将web应用拷贝到对应的域的applications目录下如${APUSIC\_HOME}/domain/mydomain/applications/ | | 可部署、反部暑、停止、重启web应用 | | |  |
| 2 | 已部暑的应用，点击其对应的“卸载”，进行反部暑； | |
| 3 | 已部暑的应用，点击其对应的“停止应用”，则停止； | |  |
| 4 | 已部暑的应用，点击其对应的“停止应用”，然后再点击“启动应用”，则重启。 | |  |
|  | 参测人员签字 | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试支持\*.war,\*.ear及以文件夹形式部署** | | **测试用例标识** | **GHtest18** | |
| **测试用例说明（方法、目的）** | | **Apusic应用服务器支持各种形式的web应用产品包格式部署包括**\*.war,\*.ear及以文件夹型式 | | | | |
| **前置条件** | | 应用服务器已安装并正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 测试部署daytrader.war; | | 都可部署成功。 | | |  |
| 2 | 测试部署EJB-client.ear； | |
| 3 | 测试部署default文件夹。 | |  |
|  | 参测人员签字 | |  | | | |

## 2.4 EJB应用

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试EJB3.1Singleton模式以及异步调用** | | **测试用例标识** | | **GHtest16** | |
| **测试用例说明（方法、目的）** | | 测试ejb3.1的Singleton模式以及异步调用等 | | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** |
| 1 | 1. 把ejb310.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问[http://ip:port/ejbtest310/index.jsp](http://ip:port/ejb310/index.jsp) 4. 分别点击页面显示后的测试各个功能的连接   观察页面返回以及启动服务器的终端 | | | | 定时器在终端打印信息，其他功能在页面能正确返回结果 | |  |
| 2 | 参测人员签字 | |  | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **EJB实体bean生命周期功能测试** | | **测试用例标识** | |  | |
| **测试用例说明（方法、目的）** | | 测试ejb实体bean生命周期测试 | | | | | |
| **前置条件** | | 1.AAS已经安装好  2.创建jndi名称为jdbc/test的数据源，并创建表table1，语句为：create table table1(id int,name varchar(20)); | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** |
| 1 | 1. 把ejb311.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/ejbtest311/testEntity.jsp>   观察启动服务器的终端打印信息和页面返回的信息 | | | | 页面返回信息字符串为test，在终端打印信息：  the entity bean is created!!  the entity bean is destoried!!  the entity bean is created!!  the entity bean is destoried!! | |  |
| 2 | 参测人员签字 | |  | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **EJB 有状态会话bean功能使用测试** | | **测试用例标识** | |  | |
| **测试用例说明（方法、目的）** | | 测试有状态会话bean生命周期功能测试 | | | | | |
| **前置条件** | | AAS已经安装好 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** |
| 1 | 1. 把ejb311.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/ejbtest311/statefulTest.jsp>   观察页面返回的的信息 | | | | 在页面打印信息类似：  stateful ejb invoked ...infor:0 stateful ejb invoked ...infor:1  （多次类似信息）  在终端打印：  bean is created.....(多次信息)  the number 0 bean is working.....  the number 0bean is destoryed....:num::0  the number 1 bean is working.....  the number 1bean is destoryed....:num::1  （多次类似信息） | |  |
| 2 | 参测人员签字 | |  | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **EJB无状态会话bean生命周期功能测试** | | **测试用例标识** | |  | |
| **测试用例说明（方法、目的）** | | 测试无状态会话bean生命周期测试 | | | | | |
| **前置条件** | | AAS已经安装好 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** |
| 1 | 1. 把ejb311.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/ejbtest311/statelessTest.jsp>   观察启动服务器的终端打印信息和页面返回信息 | | | | 页面返回信息：stateless mesthod Invoke successfully!stateless mesthod Invoke…. 等信息  在终端打印信息：  stateless bean is created.....  stateless bean will be destoryed..  stateless bean is created.....  stateless bean will be destoryed..  stateless bean is created.....  stateless bean will be destoryed..  stateless bean is created.....  （多次类似信息） | |  |
| 2 | 参测人员签字 | |  | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **EJB标准功能测试** | | **测试用例标识** | | | **GHtest19** |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0安装完成后是否具有以下功能：无状态Session Bean的部署及功能调用，有状态Session Bean的生命周期：创建，事务上下文绑定，钝化和激活，销毁，对Session Bean（EJB3），MDB（EJB3），Entity Bean（EJB3）以及Time service的支持 | | | | | |
| **前置条件** | | 已经安装了apusic应用服务器9.0 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | **实际结果** | | |
| 1 | 启动Apusic9.0应用服务器，部署应用stateless.ear、stateful.ear测试对有、无状态sessionbean的部署以及功能调用 | | 执行成功 | |  | | |
| 部署应用后，访问http://localhost:6888/samples/stateful/bookshelf.jsp,查看页面内容 | | | | | | |
| 2 | 部署应用stateless30.ear，stateful30.ear来测试金蝶Apusic9.0对EJB3的支持 | | 执行成功 | |  | | |
| 分别访问两个应用：  <http://localhost:6888/samples/stateful/bookshelf.jsp>  <http://localhost:6888/samples/stateless/converter.jsp>  应用能够正确访问无错误，则测试通过 | | | | | | |
| 3 | 部署应用mdb.ear来测试Apusic9.0对MDB（ejb3）的支持 | | 执行每一步结果正确 | | |  | |
| 访问http://192.168.10.33:6888/samples/mdb/,查看页面内容并进行一次消息发送与接收  查看是否能够发送并接收到消息 | | | | | | |
| 4 | 参测人员签字 | |  | | | | |

## 2.5 JPA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | JPA2.0功能测试-1 | **测试用例标识** | | **GHtest14** | |
| **测试用例说明（方法、目的）** | | 测试JPA2.0功能 | | | | |
| **前置条件** | | 1.AAS已经安装好  2.创建jndi名称为jdbc/ejbTester的数据源 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 把ejbtest313.ear应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/ejbtest/EJBTester?testcase=jpa>   观察启动服务器的终端打印信息和页面返回信息 | | | 页面返回信息：ok,record size:1  （每访问一次创建一条记录，size数值会递增） | |  |
| 4 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | JPA2.0功能测试-2 | **测试用例标识** | | **GHtest14** | |
| **测试用例说明（方法、目的）** | | 测试JPA2.0功能 | | | | |
| **前置条件** | | 1. AAS已经安装好 2. 创建jndi名称为jdbc/ejbjpa的数据源，创建表并插入数据：   **create** **table** id\_table(  id\_key **varchar**(100) **not** **null** **primary** **key**,  id\_value **int** **not** **null**  );  **create** **table** simple\_entity\_table(  simple\_id **int** **not** **null** **primary** **key**,  simple\_content **varchar**(100),  simple\_date **timestamp** **default** **now**()  );  **create** **table** article\_entity\_table(  article\_id **int** **not** **null** **primary** **key**,  article\_title **varchar**(50) **not** **null**,  article\_content **varchar**(1000)  );  **create** **table** type\_of\_article(  article\_id **int** **not** **null**,  article\_type **int** **not** **null**,  **constraint** unique\_entry **unique**(article\_id,article\_type)  );  **create** **table** comment\_of\_article(  article\_id **int** **not** **null**,  author\_name **varchar**(30) **default** '',  comment\_message **varchar**(100),  comment\_date **timestamp** **default** **now**()  );  **create** **table** versioned\_entity\_table(  entity\_id **int** **not** **null** **primary** **key**,  entity\_msg **varchar**(50),  entity\_version **int** **not** **null**  );  **create** **table** constrainted\_entity\_table(  constrainted\_id **int** **not** **null** **primary** **key**,  constrainted\_content **varchar**(100) **not** **null**  );  **create** **table** onetoone\_target\_entity\_table(  target\_id **int** **primary** **key**,  target\_content **varchar**(100) **not** **null**  );  **create** **table** onetoone\_owner\_entity\_table(  owner\_id **int** **primary** **key**,  owner\_content **varchar**(100) **not** **null**,  owned\_target\_id **int**,  **constraint** owned\_target **foreign** **key** (owned\_target\_id) **references** onetoone\_target\_entity\_table(target\_id)  );  **insert** **into** id\_table **values**('SIMPLE\_ENTITY\_ID',0);  **insert** **into** id\_table **values**('VERSIONED\_ENTITY\_ID',0);  **insert** **into** id\_table **values**('ARTICLE\_ENTITY\_ID',0);  **insert** **into** id\_table **values**('CONSTRAINTED\_ENTITY\_ID',0);  **insert** **into** id\_table **values**('DELIMITED\_ENTITY\_ID',0);  **insert** **into** id\_table **values**('ONETOONE\_RELATIONSHIP\_OWNER\_ENTITY\_ID',0);  **insert** **into** id\_table **values**('ONETOONE\_RELATIONSHIP\_TARGET\_ENTITY\_ID',0); | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 把jpa20.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/jpa20/list> 4. 然后分别执行案例TestElementCollectionCreation、TestElementCollectionModification、TestOptimisticLockingCollision、TestPessimisticLockingUnderNormalScope、TestJpaBeanValidation   观察页面返回的信息 | | | 页面返回信息案例测试为PASSED | |  |
| 4 | 参测人员签字 | | |  | | |

## 2.6 WebService

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **WebService功能测试** | **测试用例标识** | | **GHtest14** | |
| **测试用例说明（方法、目的）** | | 测试WebService功能支持 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好 2. 增加/修改vm.options参数如下：   apusic.connect.anytime=true  apusic.webservice.enabled=true  3.将应用案例webservice\service  中的文件拷贝到apusic-ws.jar\META-INF\services | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 部署webservice服务器端testws.war，把应用拷贝到域下的applications目录，域端口为6888 2. 部署webservice客户端wsclient.war，把应用拷贝到域下的applications目录 3. 启动AASv9.0 4. 访问http://localhost:7888/wsclient/   在页面中输入信息test，点击“say hello”按钮 | | | 页面返回信息：  hello-request: test  hello-response: hello,test / [injection:injected-string] | |  |
| 4 | 参测人员签字 | | |  | | |

## 2.7 WebScoket

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **WebScoket功能测试** | **测试用例标识** | | **GHtest14** | |
| **测试用例说明（方法、目的）** | | 测试WebScoket功能支持 | | | | |
| **前置条件** | | AAS已经安装好 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 部署websocket-sample.war，把应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问http://ip:port /websocket-sample/   4.在页面的Message框中输入信息，点击send,可以在界面查看到发送信息和接收信息，并且每隔10s会收到一条当前时间的信息。 | | | | | |
| 4 | 参测人员签字 | | |  | | |

## 2.8 JMS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 消息智能路由测试 | **测试用例标识** | | **GHtest20** | |
| **测试用例说明（方法、目的）** | | 测试消息智能路由功能 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好 2. 分别配置3个节点node1->node2->node3,配置信息参考jms\_route文件中的node说明。   3. 场景说明：往node1和队列testQueue@routerC发送一个消息，能够自动转发到node3中,node3的程序能够接收到node1发送的信息。 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 把jms\_route中的testjms\_send.jsp文件部署到节点1的默认应用中 2. 把jms\_route中的testjms\_rec.jsp文件部署到节点3的默认应用中 3. 分别启动AASv9.0 4. 访问[http://ip:6888/testjms\_send.jsp](http://ip:6888/daytrader/，然后切换到Configuration标签)发送默认消息   访问<http://ip:8888testjms_rec.jsp>接收默认消息 | | | 接收页正确显示默认消息 | |  |
| 5 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 消息负载均衡测试 | **测试用例标识** | | **GHtest20** | |
| **测试用例说明（方法、目的）** | | 测试消息负载均衡功能 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好 2. 分别配置3个节点负载均衡节点和两个后台节点,配置信息参考jms\_route文件夹中的node说明。   3. 场景说明：往负载均衡节点和队列testQueue发送消息，后台节点的程序能够接收到发送的信息，负载均衡使用轮询的发送策略。 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 把jms\_route中的testjms\_lb\_send.jsp文件部署到负载均衡的默认应用中 2. 把jms\_route中的testjms\_rec.jsp文件部署到后台2个节点中(node2和node3)的默认应用中 3. 分别启动AASv9.0 4. 访问<http://ip:6888/testjms_lib_send.jsp发送2>个消息 5. 访问<http://ip:7888testjms_rec.jsp>接收消息   访问<http://ip:8888testjms_rec.jsp>接收消息 | | | 发送的2个信息已轮询的策略分发到后台节点，在一个节点中都接收到一条消息 | |  |
| 5 | 参测人员签字 | | |  | | |

## 2.9 LDAP集成测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 与LDAP集成测试 | **测试用例标识** | | **GHtest20** | |
| **测试用例说明（方法、目的）** | | 测试cas、openldap与AAS集成进行单独登陆 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好   2. 根据LDAP集成.zip中的LDAP集成使用与演示.docx文档进行环境配置。 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 根据LDAP集成.zip中的《LDAP集成使用与演示.docx》文档进行测试 | | | 能够正确完成测试 | |  |
| 5 | 参测人员签字 | | |  | | |

## 2.10 单点登录测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 单点登录测试 | **测试用例标识** | | **GHtest20** | |
| **测试用例说明（方法、目的）** | | 测试与openldap集成进行安全登陆验证 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好   2. 根据cas\_ldap\_apusic.zip中的LDAP\_cas\_apusic.docx文档进行环境配置。 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 根据Lcas\_ldap\_apusic.zip中的《LDAP\_cas\_apusic.docx》文档进行测试 | | | 能够正确完成测试 | |  |
| 5 | 参测人员签字 | | |  | | |

## 2.11 应用集群集群功能

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **集群功能测试** | | **测试用例标识** | | **GHtest21** | |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0与 Apache httpd2.2.5建立集群的正确性，并检查集群会话粘滞、失败转移以及请求分发功能特性，分发策略包括轮询、随机及按权重策略。  集群安装可以参考“Windows/Linux下集群安装”用例说明 | | | | | |
| **前置条件** | | 已经安装Apusic9.0应用服务器以及Apache httpd2.2.5 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | | **实际结果** |
| 1 | 安装两个Apusic应用服务器实例，使用mydomain进行集群配置，分别部署测试应用test1.war | | 配置apusic后置节点成功 | | | |  |
| 1. 修改两个mydomain实例的服务端口分别为6888，7888避免冲突 2. 分别在每一个实例目录下的config\apusic.conf配置文件中添加cluster服务配置，如下：   <SERVICE CLASS="com.apusic.cluster.ClusterService">  <ATTRIBUTE NAME="ClusterName" VALUE="test"/>//test为所建集群名称  <ATTRIBUTE NAME="LoadWeight" VALUE="100"/>  <ATTRIBUTE NAME="ServerName" VALUE="$DOMAIN\_NAME"/>//用实例名称作为服务器名称  <ATTRIBUTE NAME="AutoReConnect" VALUE="true"/>  </SERVICE>   1. 分别在\config\vm.option中添加配置添加com.apusic.jvm.route=proxyserver1到文件末尾，如下所示，其中route= proxyserver1表示apache转发请求时的路由名称，当存在多个Apusic后置节点时，保证每一个route的名称不同 | | | | | | |
| 2 | 修改apache2.2安装目录下配置文件，添加集群配置文件。 | | 成功添加并重启 | | | |  |
| 1. 进入到%Apache2.2\_HOME\conf目录下，修改httpd.conf配置文件 2. 添加集群配置文件   ProxyPass / balancer://proxy/  ProxyPassReverse / balancer://proxy  <Proxy balancer://proxy>  BalancerMember http://localhost:6888/test route=proxyserver1  BalancerMember http://localhost:7888/test route=proxyserver2  </Proxy>   1. 添加完毕后，重启Apache。 | | | | | | |
| 3 | 重启两个Apusic应用服务器mydomain实例，访问http://localhost:80/test1 | | 成功访问，建立集群成功 | | | |  |
| 4 | 修改Apache2.2配置文件，加入会话粘滞和失效转移的配置，重启Apache；修改Apusic应用服务器配置apusic.conf下session服务，重启Apusic实例。 | | 配置完成后重启Apache和Apusic服务器才成功 | | | |  |
| （1）Apache2.2配置如下：  ProxyPass / balancer://proxy/ **stickysession=JSESSIONIDnofailover=off**  ProxyPassReverse / balancer://proxy  <Proxy balancer://proxy>  BalancerMember http://192.168.11.28:6888/test1 route=proxyserver1  BalancerMember http://192.168.11.28:7888/test1 route=proxyserver2  </Proxy>  （2）Apusic9.0配置如下  SERVICE CLASS="com.apusic.servlet.http.session.SessionService">  <ATTRIBUTE NAME="DefaultSessionTimeout" VALUE="3600"/>  <ATTRIBUTE NAME="MaxSessionsInCache" VALUE="1024"/>  <ATTRIBUTE NAME="SessionInvalidateCheckInterval" VALUE="60"/>  <ATTRIBUTE NAME="Distributable" VALUE="True"/>  <ATTRIBUTE NAME="Replicable" VALUE="True"/>  <ATTRIBUTE NAME="SessionStick" VALUE="True"/>  </SERVICE> | | | | | | |
| 5 | 测试session粘滞成功功能，访问http://localhost:80/test1，观察是否只在一个Apusic实例后台打印日志 | | | | 日志只在一个Apusic实例后台打印日志， | |  |
| 6 | 测试失效转移功能，停止打印日志的Apusic实例，继续访问<http://locahost:80/test>1，观察是否在另一台Apusic实例下打印日志， | | | | 开始在另一台apusic服务器实例启动界面下打印日志 | |  |
| 7 | 测试权重分发策略，修改Apache2.2配置文件，添加loadfacter权重配置后重启apache；  访问<http://locahost:80/test>1，观察日志在两台Apusic服务器实例的打印信息比例 | | | | 根据权重比例，在后台打印测试信息 | |  |
| 1. 轮询策略Apache配置：   ProxyPass / balancer://proxy/  ProxyPassReverse / balancer://proxy  <Proxy balancer://proxy>  BalancerMember http://192.168.11.28:6888/test1 route=proxyserver1  BalancerMember http://192.168.11.28:7888/test1 route=proxyserver2   1. 按权重分配均衡策略配置：   ProxyPass / balancer://proxy/  ProxyPassReverse / balancer://proxy  <Proxy balancer://proxy>  BalancerMember http://192.168.11.28:6888/test1 route=proxyserver1 loadfactor=3  BalancerMember http://192.168.11.28:7888/test1 route=proxyserver2 loadfactor=1   1. 权重请求响应负载均衡策略的配置   ProxyPass / balancer://proxy/lbmethod=bytraffic  ProxyPassReverse / balancer://proxy  <Proxy balancer://proxy>  BalancerMember http://192.168.11.28:6888/test1 route=proxyserver1 loadfactor=3  BalancerMember http://192.168.11.28:7888/test1 route=proxyserver2 loadfactor=1 | | | | | | |
| 8 | 参测人员签字 | |  | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Apusic集群功能测试** | | | **测试用例标识** | | **GHtest22** | | |
| **测试用例说明（方法、目的）** | | 测试使用Apusic9.0作为负载均衡器建立集群，并测试集群会话粘滞、失败转移以及请求分发功能，请求分发策略包括轮询、随机以及权重策略 | | | | | | | |
| **前置条件** | | 已经安装Apusic9.0应用服务器 | | | | | | | |
| **测试执行步骤** | | | | | | | | | |
| **序号** | **测试输入** | | | | | **预期结果** | | **实际结果** | |
| 1 | 安装三个Apusic应用服务器实例，其中2个使用mydomain进行集群配置，分别部署测试应用test1.war；另一台Apusic实例作为负载均衡器 | | | | | 配置apusic后置节点成功 | |  | |
| 1. 修改两个mydomain实例的服务端口分别为6888，7888避免冲突 2. 配置负载均衡器：   进入到$APUSIC\_LB\_HOME$\domains\mydomain\config目录下，删除apusic.conf文件并重命名loadbalancer.conf为apusic.conf文件   1. 修改apusic.conf中配置文件LoadBalancer配置如下：   SERVICE  CLASS="com.apusic.web.loadbalancer.LoadBalancer"  >  <ATTRIBUTE NAME="ServerPort" VALUE="80"/>  <ATTRIBUTE NAME="BackendServers" VALUE="ip:port,ip:port"/>  <ATTRIBUTE NAME="MaxWaitingClients" VALUE="500"/>  <ATTRIBUTE NAME="WaitingClientTimeout" VALUE="5"/>  <ATTRIBUTE NAME="KeepAlive" VALUE="True"/>  <ATTRIBUTE NAME="MaxKeepAliveRequests" VALUE="100"/>  <ATTRIBUTE NAME="KeepAliveTimeout" VALUE="15"/>  <ATTRIBUTE NAME="MaxKeepAliveConnections" VALUE="300"/>  <ATTRIBUTE NAME="EnableLog" VALUE="False"/>  <ATTRIBUTE NAME="LogFileName" VALUE="logs/access.log"/>  <ATTRIBUTE NAME="LogFileLimit" VALUE="1000000"/>  <ATTRIBUTE NAME="LogFileCount" VALUE="10"/>  </SERVICE> | | | | | | | | |
| 2 | Apusic负载均衡器默认情况下是会话粘滞（session-stick）的，同时采用轮循的策略选择可用节点。 | | | 一直在一台后置节点打印日志信息，会话粘滞成功 | | | | |  |
| 访问http://localhost：80/test,一直在一台后置节点打印日志信息 | | | | | | | | |
| 3 | 关闭一直打印日志信息的服务器，继续访问http://localhost：80/test，观察另一台服务器是否开始打印日志信息 | | 另一台服务器开始打印日志信息 | | | | | |  |
| 4 | 修改apusic负载均衡器，关闭会话粘滞配置项，重新启动负载均衡器 | | 轮询策略成功，轮询打印日志信息 | | | | | |  |
| 1. 修改配置项，添加配置：   <ATTRIBUTE NAME="SessionStick" VALUE="false"/>   1. 重启apusic负载均衡器 2. 访问测试应用<http://localhost:80/test>,查看是否是轮询打印访问日志信息 | | | | | | | | |
| 5 | 配置策略为随即访问策略，访问测试应用来测试策略是否生效 | | 查看两个apusic访问日志打印，是随即性的 | | | | | |  |
| 1. 修改apusic负载均衡器配置文件，添加配置：   <ATTRIBUTE NAME="BalancePolicy" VALUE="Random"/>  ( 2) 访问测试应用<http://localhost:80/test>,查看是否是随即打印访问日志信息 | | | | | | | | |
| 6 | 修改BalancePolicy的值为LoadWeight后，添加负载比例配置项后访问测试应用来测试权重策略 | | 访问日志大致比例为3:7 | | | | | |  |
| 1. 修改BalancePolicy的值为LoadWeight 2. 添加权重配置项：   <ATTRIBUTE NAME="LoadWeight" VALUE="30,70"/>   1. 访问测试应用多次后，统计两台负载apusic应用服务器访问比例是否接近3:7 | | | | | | | | |
| 7 | 参测人员签字 | |  | | | | | | |

## 2.12 EJB集群

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **EJB集群** | | **测试用例标识** | | **GHtest23** | | |
| **测试用例说明（方法、目的）** | | 测试应用服务器支持EJB集群 | | | | | | |
| **前置条件** | | 已经安装Apusic9.0应用服务器 | | | | | | |
| **测试执行步骤** | | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** | |
| 1 | 使用同一个AAS服务器中的两个不同的域，如mydomain和sample,端口需要不一致，不然启动不了两个实例。  在apusic.conf文件的**最后**加上如下服务配置：  <SERVICE CLASS="com.apusic.cluster.ClusterService">  <ATTRIBUTE NAME="ClusterName" VALUE="ClusteredEJB\_example"/>  <ATTRIBUTE NAME="LoadWeight" VALUE="100"/>  <ATTRIBUTE NAME="ServerName" VALUE="$DOMAIN\_NAME"/>  <!-- <ATTRIBUTE NAME="AutoReConnect" VALUE="true"/>-->  </SERVICE>  <SERVICE CLASS="com.apusic.naming.cluster.ClusterNameService"/>  <SERVICE CLASS="com.apusic.net.DiscoveryService">  <ATTRIBUTE NAME="GroupAddress" VALUE="230.0.0.2"/>---多播地址和端口  <ATTRIBUTE NAME="GroupPort" VALUE="1500"/>  </SERVICE>  配置文件中的属性：  <ATTRIBUTE NAME="ClusterName" VALUE="ClusteredEJB\_example"/> | | | | | | | |
| 2 | 将hello.ear应用拷贝到上面集群中的2个域的applications目录，完成部署  需要修改应用的jsp1.jsp代码如下：  aas实例的用户名和密码；集群的名称 | | | | | | |  |
| 3 | 启动2个集群实例 | | 其中一个终端显示类似如下信息：  User0， do in this!  User1， do in this!  User6， do in this!  另一个终端显示类似如下信息：  User2， do in this!  User3， do in this!  User4， do in this!  User5， do in this!  User7， do in this!  User8， do in this!  User9， do in this! | | | | |  |
| 4 | 访问http://localhost:6888/cluster/jsp1.jsp，页面显示类似如下格式的信息：  注：6888为集群里面任意一个实例的端口，cluster为应用的基础上下文，代码里面控制的。 | |  |
| 5 | 访问多次，而观察2个AAS启动后台日志 | | 可以在日志的打印信息中查看到访问的集群信息 | | | | |  |
| 6 | 参测人员签字 | |  | | | | | |

## 2.13 JNDI集群

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **JNDI集群** | | **测试用例标识** | | **GHtest24** | | |
| **测试用例说明（方法、目的）** | | 测试应用服务器支持JNDI集群 | | | | | | |
| **前置条件** | | 已经安装Apusic9.0应用服务器 | | | | | | |
| **测试执行步骤** | | | | | | | | |
| **序号** | **测试输入** | | | | **预期结果** | | **实际结果** | |
| 1 | 使用同一个AAS服务器中的两个不同的域，如mydomain和sample,端口分别为6888,7888，不然启动不了两个实例。  各个域的配置文件增加如下的配置：  <SERVICE  CLASS="com.apusic.cluster.ClusterService">  <ATTRIBUTE NAME="ClusterName" VALUE="JNDI\_SAMPLE"/>  <ATTRIBUTE NAME="ServerName" VALUE="$DOMAIN\_NAME"/>  </SERVICE>  <SERVICE CLASS="com.apusic.naming.cluster.ClusterNameService"/>  <SERVICE CLASS="com.apusic.net.DiscoveryService">  <ATTRIBUTE NAME="GroupAddress" VALUE="230.0.0.2"/>  <ATTRIBUTE NAME="GroupPort" VALUE="1500"/>  </SERVICE>  其中：属性ClusterName为集群的名称，多个节点如果想加入到同一个集群中，ClusterName必须相同。 ServerName为当前节点的名称，必须唯一，如果存在相同名称的节点，则后加入的节点会加不到集群中。默认的ServerName为当前Domain的名称。 | | | | | | | |
| 2 | jndi\_test\_provider.war是一个简单的web应用,绑定jndi名称和值。  jndi\_test\_client.war是一个简单的web应用,获取jndi集群中指定名称的值。  只需要把应用分别拷贝到上面集群中的2个域的applications目录，完成部署。 | | | | | | |  |
| 3 | 启动2个集群实例 | | 页面显示类似如下格式的信息：  bind: test/jndiTest--->Mon Nov 02 15:23:13 CST 2015  第二次访问：  rebind: test/jndiTest--->Mon Nov 02 15:23:14 CST 2015 | | | | |  |
| 4 | 访问  http://localhost:6888/jndi\_test\_provider/index.jsp | |  |
| 5 | 访问  http://localhost:7888/jndi\_test\_client/index.jsp | | 页面显示类似如下格式的信息：  find jndi: name=test/jndiTest, value=Mon Nov 02 15:23:14 CST 2015 | | | | |  |
| 6 | 参测人员签字 | |  | | | | | |

## 2.14 session存储测试

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Session存储功能测试** | | | **测试用例标识** | | **GHtest27** | | | |
| **测试用例说明（方法、目的）** | | 测试应用服务器对Session不同的存储策略的正确性，包括本地文件存储,berkeleyDB存储、数据库存储、分布式缓存存储 | | | | | | | | |
| **前置条件** | | 已经安装了Apusic9.0应用服务器 | | | | | | | | |
| **测试执行步骤** | | | | | | | | | | |
| **序号** | **测试输入** | | | | | **预期结果** | | **实际结果** | | |
| 1 | 打开Apusic9.0安装目录，查看%APUSIC\_HOME%\domains\mydomain\config\apusic.conf文件，查看session存储方式为file。store/http\_sessions目录下可以看到文件存储文件。 | | | | | 启动apusic应用服务器成功，在目录下看到文本存储文件.db | | 通过 | | |
| 本地文件存储配置：  <SERVICE  CLASS="com.apusic.web.session.FileSessionStoreService"  >  <ATTRIBUTE NAME="Directory" VALUE="store/http\_sessions"/>  <ATTRIBUTE NAME="RebuildStoreOnShutDown" VALUE="true"/>  <ATTRIBUTE NAME="RebuildStoreOnMaxFileSize" VALUE="1024"/>  <ATTRIBUTE NAME="RebuildStoreOnSchedule" VALUE="0 0 18 1 \* \* \*"/>  </SERVICE>  其中：  Directory：为Sesssion存储目录。  RebuildStoreOnShutDown：是否在AAS关闭的时候进行Rebuild。  RebuildStoreOnMaxFileSize：确定在session文件达到多大的时候，进行Rebuild。单位为MB  RebuildStoreOnSchedule：用于定时Rebuild，值是一个时间的表达式。 | | | | | | | | | |
| 2 | 打开Apusic9.0安装目录，查看%APUSIC\_HOME%\domains\mydomain\config\apusic.conf文件，修改session存储方式为DB存储；将je.jar拷贝到apusic应用服务器lib目录下，重新启动apusic应用服务器 | | | 执行成功  Session存储方式修改为DB，服务器成功启动，在store/http\_sessions目录下可以看到.jdb文件 | | | | | | 通过 |
| BerkeleyDB存储配置：  <SERVICE  CLASS="com.apusic.web.session.DBSessionStoreService"  NAME="apusic:name=HttpSessionStore,j2eeType=Service"  >  <ATTRIBUTE NAME="RebuildStoreOnMaxFileSize" VALUE="1024"/>  <ATTRIBUTE NAME="ConfigPath" VALUE="config/je\_session.properties"/>  <ATTRIBUTE NAME="ServicePriority" VALUE="L"/>  <ATTRIBUTE NAME="RebuildStoreOnShutDown" VALUE="False"/>  <ATTRIBUTE NAME="Directory" VALUE="store/http\_sessions"/>  <ATTRIBUTE NAME="RebuildStoreOnSchedule" VALUE=""/>  </SERVICE> | | | | | | | | | |
| 3 | 新建数据源jdbc/sample,保证数据源可用，修改session存储服务为SQL存储，重启apusic应用服务器 | | Apusic应用服务器启动成功，session存储服务启动成功 | | | | | | 通过 | |
| Session数据库存储配置：  <SERVICE  CLASS="com.apusic.web.session.SQLSessionStoreService"  NAME="apusic:name=HttpSessionStore,j2eeType=Service"  >  <ATTRIBUTE NAME="RebuildStoreOnMaxFileSize" VALUE="-1"/>  <ATTRIBUTE NAME="DataSourceName" VALUE="**jdbc/sample**"/>  <ATTRIBUTE NAME="CreateTableSQL" VALUE="CREATE TABLE t\_httpsession (f\_id VARCHAR(255) NOT NULL, f\_timestamp VARCHAR(255) NOT NULL, f\_session LONG VARCHAR NOT NULL, CONSTRAINT PK\_httpsession PRIMARY KEY(f\_id))"/>  <ATTRIBUTE NAME="ServicePriority" VALUE="L"/>  <ATTRIBUTE NAME="RebuildStoreOnShutDown" VALUE="False"/>  <ATTRIBUTE NAME="TableName" VALUE="t\_httpsession"/>  <ATTRIBUTE NAME="RebuildStoreOnSchedule" VALUE=""/>  </SERVICE> | | | | | | | | | |
|  | session membase存储：  修改$APUSIC\_HOME/domains/mydomain/conf下apusic.conf  <SERVICE  CLASS="com.apusic.web.session.CacheSessionStoreService"  >  <ATTRIBUTE NAME="ServerAddress" VALUE="192.168.6.66:11211,192.168.6.67:11211"/>  <ATTRIBUTE NAME="BackupServerAddress" VALUE="192.168.6.76:11211,192.168.6.77:11211"/>  <ATTRIBUTE NAME="Weights" VALUE="1,2"/>  <ATTRIBUTE NAME="Failover" VALUE="True"/>  <ATTRIBUTE NAME="ConnectionPoolSize" VALUE="1"/>  <ATTRIBUTE NAME="BinaryProtocol" VALUE="False"/>  <ATTRIBUTE NAME="TokyoTyrant" VALUE="False"/>  <ATTRIBUTE NAME="ConsistentHash" VALUE="False"/>  </SERVICE>  具体属性值如下：  ServerAddress:分布式Session存储数据库节点列表；形如“节点1:port，节点2:port“的字符串，节点间用逗号隔开。  BackupServerAddress : 备份节点列表，格式同上，备份节点顺序和主节点顺序一一对应。  Weights : 节点权重，大于零。形如：“1,2”的字符串，权重间用逗号隔开。  Failover：是否为失效转移模式。当为失效转移模式时，当一个节点挂掉后，它的key值将被分配到其它节点。若为非失效转移时，当某个节点挂掉的时候，不会从节点列表移除，请求也不会转移到下一个有效节点，而是直接将请求置为失败。  ConnectionPoolSize ：NIO连接池大小，默认为1。  BinaryProtocol : 是否使用二进制协议。默认为False。  TokyoTyrant ：是否使用TokyoTyrant服务器，默认为False。  ConsistentHash : 是否使用一致性哈希，默认为False。 | | | | | | | | | |
| 4 | 参测人员签字 | |  | | | | | | | |

## 2.15 HTTPS安全传输

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **数据传输安全测试** | **测试用例标识** | | **GHtest28** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器支持单双向认证，https数据加密传输，支持密码算法加密 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器, Openssl已安装好 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | **单向认证：**  a.修改$APUSIC\_HOME$\domains\mydoamin\config目录下apusic.conf文件  配置文件成功  修改muxer服务配置：  <ATTRIBUTE NAME="SSLEnabled" VALUE="True"/>  b. 重启应用服务器，访问https://localhost:6889查看是否能够打开web管理控制台 | | | | | |
| 2. | **双向认证：**  参考文档 步骤1. 建立自己的认证授权机构CA：  1. 建立工作目录   mkdir ca  cd ca   1. 生成ca私钥   openssl dsaparam -out dsaparam 1024  openssl gendsa -out dsakey dsaparam   1. 生成ca待签名证书   openssl req -new -config openssl.conf -out ca-req.csr -key dsakey  eg：根据提示输入根证书信息，openssl.conf根据实际的路径指定   1. 用CA私钥进行自签名，得到自签名的CA根证书：   openssl x509 -req -in ca-req.csr -out ca-cert.cer -signkey dsakey -days 3650 步骤2. 生成服务器端证书 服务器端证书用来向客户端证明服务器的身份，也就是说在SSL协议握手的时候，服务器发给客户端的证书。生成服务器证书时用到了JDK的密钥管理工具Keytool，本文采用的jdk是sun jdk1.5.0。可以在Sun公司的网站http://java.sun.com下载jdk1.5.0。   1. 生成服务器私钥对及自签名证书，并且保存在密钥库mykeystore中   keytool -genkey -alias myserver -keyalg RSA -keysize 1024 -keypass keypass -storepass keypass -dname "cn=localhost, ou=dev, o=apusic,l=Shenzhen, st=guangdong, c=CN" -keystore mykeystore   1. 生成服务器待签名证书   keytool -certreq -alias myserver -sigalg SHA1withRSA -file server.csr -keypass keypass -storepass keypass -keystore mykeystore   1. 请求CA签名服务器待签名证书，得到经CA签名的服务器证书   openssl x509 -req -in server.csr -out server-cert.cer -CA ca-cert.cer -CAkey dsakey -days 3650 -set\_serial 02   1. 把CA根证书导入密钥库 mykeystore   keytool -import -alias caroot -file ca-cert.cer -noprompt -keypass keypass -storepass keypass -keystore mykeystore   1. 把经过CA签名的服务器证书导入密钥库mykeystore   keytool -import -alias myserver -file server-cert.cer -noprompt -keypass keypass -storepass keypass -keystore mykeystore   1. 把CA根证书导入信任库truststore   keytool -import -alias caroot -file ca-cert.cer -noprompt -keypass keypass -storepass keypass -keystore truststore 步骤3. 颁发个人证书  1. 生成客户端私钥：   openssl genrsa -out clientkey 1024   1. 生成客户端待签名证书：   openssl req -new -config openssl.conf -out client.csr -key clientkey  eg：根据提示输入根证书信息，openssl.conf根据实际的路径指定   1. 请求CA签名客户端待签名证书，得到经CA签名的客户端证书：   openssl x509 -req -in client.csr -out client.cer -CA ca-cert.cer -CAkey dsakey -days 3650 -set\_serial 03   1. 生成客户端的个人证书client.p12：   openssl pkcs12 -export -clcerts -in client.cer -inkey clientkey -out client.p12 步骤4.浏览器导入证书：  1. CA根证书导入客户端   在这里CA的根证书用来在SSL握手时验证服务器发给客户端浏览器的证书。如果没有此证书，浏览器将无法自动验证服务器证书，因此浏览器将弹出确认信息，让用户来确认是否信任服务器证书。 在客户端的IE中使用"工具 -> Internet选项 -> 内容 -> 证书 (受信任的根证书颁发机构)-> 导入"把我们生成的CA根证书**ca-cert.cer**导入，使其成为用户信任的CA。     1. 客户端个人证书导入客户端   在客户端的IE中使用"工具 -> Internet选项 -> 内容 -> 证书(个人) -> 导入"把我们生成的个人证书client.p12导入，使其成为用户的证书。   步骤5. 配置Apusic允许双向认证 Apusic应用服务器默认配置下不支持双向认证，要支持SSL双向认证，  **采用密钥库文件的方式**   1. 在对应的域(如mydomain)目录创建目录，如ca 2. 把相关的文件（mykeystore，truststore）拷贝到ca目录下 3. 修改Muxer服务，如果采用密钥库文件的方式，示例配置如下：   <SERVICE CLASS="com.apusic.net.Muxer">  ……  <ATTRIBUTE NAME="SSLEnabled" VALUE="True"/>  <ATTRIBUTE NAME="SecurePort" VALUE="6889"/>  <ATTRIBUTE NAME="MutualAuthPort" VALUE="6887"/>  <ATTRIBUTE NAME="NeedClientAuth" VALUE="True"/>  <ATTRIBUTE NAME="KeyStore" VALUE="ca/mykeystore "/>  <ATTRIBUTE NAME="KeyPassword" VALUE="keypass"/>  <ATTRIBUTE NAME="TrustStore" VALUE="ca/truststore"/>  <ATTRIBUTE NAME="TrustStorePassword" VALUE="keypass"/>  <ATTRIBUTE NAME="TrustStoreType" VALUE="JKS"/>  ……  </SERVICE>  **步骤6.测试双向认证**  访问AAS默认首页https://localhost:6887，如果前面的操作都正确的话，应该可以看到Apusic的欢迎页面。同时状态栏上的小锁处于闭合状态，且证书为有效状态，表示您已经成功地与服务器建立了要求客户端验证的SSL安全连接。    **Eg:必须使用localhost访问，因为我们的服务器证书写的是localhost,否则会报证书错误。** | | | | | |
| 3 | 参测人员签字 | | |  | | |

## 2.16 JTS分布式事物

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **JTS分布式事务功能测试** | | **测试用例标识** | | **GHtest29** |
| **测试用例说明（方法、目的）** | | 测试金蝶Apusic9.0安装完成后能够实现事务分布式管理 | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | **实际结果** | |
| 1 | 部署应用test-jta-webapp.war并访问，来测试事务分布式管理 | |  | |  | |
| 1. 建立测试数据源   a.建立数据源jdbc/test1 与jdbc/test2,数据源选择oracle 驱动需配置XA的驱动  b. 需要确认应用applicationContext-hibernate.xml 的hibernate.dialect为对应的数据库的hibernate的dialog    c.据源jndi/test1连接的数据库中建立一个表为t\_test\_jta\_person， 建表语句:  create table t\_test\_jta\_person(id varchar(64) not null primary key , name varchar(1024));  在数据源jdbc/test2连接的数据库中建立一个表为t\_test\_jta\_person2， 建表语句: create table t\_test\_jta\_person2(id varchar(64) not null primary key , name varchar(1024));  （2）部署应用test-jta-webapp.war后访问应用，访问：  <http://localhost:6888/test-jta-webapp/index/list.do>  页面显示为：    点击添加按钮，跳转添加页面，并输入添加的内容，添加成功后页面会有刚刚新增的数据。  分别查看数据库表t\_test\_jta\_person 和 t\_test\_jta\_person2看看对应的数据是否存在。  然后点击删除，看看数据库是否把数据都删除了。  最后点击删除(模拟回滚)，看看控制台会抛出一个by zero的异常，模拟数据库回滚，再看看数据是否还在。 | | | | | |
| 2 | 参测人员签字 | |  | | | |

## 2.17 虚拟主机

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **虚拟主机功能测试** | **测试用例标识** | | **GHtest31** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器对虚拟主机功能的支持 | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 修改linux服务器etc/hosts文件  127.0.0.1 localhost.localdomain localhost www.apusic.com | | | 成功修改 | |  |
| 2 | 部署应用default成功后修改domains/mydomain/config下的server.xml文件  <application name="default" base="applications/default" start="auto" loadon="0" virtual-host="www.apusic.com"/>，将应用配置到虚拟主机 | | | 重启apusic应用服务器后应用启动成功 | |  |
| 3 | 访问www.apusic.com:6888/,查看是否可以访问 | | | 可以访问应用 | |  |
| 4 | 参测人员签字 | | |  | |  |

## 2.18 第三方web服务器连接

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试与第三方web服务器（Apache）的连接** | | | **测试用例标识** | **GHtest32** | |
| **测试用例说明（方法、目的）** | | 测试应用服务器是否支持与第三方web服务器Apache的连接 | | | | | |
| **前置条件** | | 已经安装Apusic9.0应用服务器以及Apache httpd2.2.5 | | | | | |
| **测试执行步骤** | | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | | **实际结果** |
| 1 | 安装两个Apusic应用服务器实例，使用mydomain进行集群配置，分别部署测试应用test1.war | | 配置apusic后置节点成功 | | | |  |
| 1. 修改两个mydomain实例的服务端口分别为6888，7888避免冲突 2. 分别在每一个实例目录下的config\apusic.conf配置文件中添加cluster服务配置，如下：   <SERVICE CLASS="com.apusic.cluster.ClusterService">  <ATTRIBUTE NAME="ClusterName" VALUE="test"/>//test为所建集群名称  <ATTRIBUTE NAME="LoadWeight" VALUE="100"/>  <ATTRIBUTE NAME="ServerName" VALUE="$DOMAIN\_NAME"/>//用实例名称作为服务器名称  <ATTRIBUTE NAME="AutoReConnect" VALUE="true"/>  </SERVICE>   1. 分别在\config\vm.option中添加配置添加com.apusic.jvm.route=proxyserver1到文件末尾，如下所示，其中route= proxyserver1表示apache转发请求时的路由名称，当存在多个Apusic后置节点时，保证每一个route的名称不同 | | | | | | |
| 2 | 修改apache2.2安装目录下配置文件，添加集群配置文件。 | | 成功添加并重启 | | | |  |
| 1. 进入到%Apache2.2\_HOME\conf目录下，修改httpd.conf配置文件 2. 添加集群配置文件   ProxyPass / balancer://proxy/  ProxyPassReverse / balancer://proxy  <Proxy balancer://proxy>  BalancerMember http://localhost:6888/test route=proxyserver1  BalancerMember http://localhost:7888/test route=proxyserver2  </Proxy>   1. 添加完毕后，重启Apache。 | | | | | | |
| 3 | 重启两个Apusic应用服务器mydomain实例，访问http://localhost:80/test1 | | 成功访问，建立集群成功 | | | |  |
| 5 | 调整与Apache连接处理Web请求的工作线程数量 | | |  | | |  |
| 6 | 调整与Apache连接处理Web请求的等待队列大小 | | |  | | |  |
| 7 | 调整与Apache连接处理Web请求的超时时间 | | |  | | |  |
| 8 | 参测人员签字 | |  | | | | |

## 2.19 SQL语句监控机制测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | Sql语句监控机制测试 | **测试用例标识** | |  | |
| **测试用例说明（方法、目的）** | | 演示产品对SQL语句监控机制的支持 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好 2. 部署jndi名称为jdbc/testsqltrace的数据源，并使用语句创建表:create table account(id integer,name varchar(50)) 3. 在apusic.conf中增加服务JDBCTraceService,并修改属性TraceAllow值为true，如下：   <SERVICE CLASS="com.apusic.jdbc.trace.JDBCTracerService">  <ATTRIBUTE NAME="TraceAllow" VALUE="True"/>  <ATTRIBUTE NAME="StackTraceAllow" VALUE="False"/>  </SERVICE> | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 把testsqltrace.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/testsqltrace/teststart.jsp启动sql>语句监控 4. 访问<http://ip:port/testsqltrace/testsql.jsp执行测试的sql>语句   访问<http://ip:port/testsqltrace/testlog.jsp显示sql>执行的相关信息 | | | 能够在页面中显示执行的sql语句、开始时间、结束时间、执行时间等信息 | |  |
| 5 | 参测人员签字 | | |  | | |

## 2.20 Daytrader应用 el jsp html测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **Daytrader应用 el jsp html测试** | **测试用例标识** | |  | |
| **测试用例说明（方法、目的）** | | 1. AASv9.0已经安装好   2.部署jndi名称为jdbc/TradeDataSource的数据源，不需要手动创建表 | | | | |
| **前置条件** | | 1. AASv9.0已经安装好 2. 部署jndi名称为jdbc/testsqltrace的数据源，并使用语句创建表:create table account(id integer,name varchar(50)) 3. 在apusic.conf中增加服务JDBCTraceService,并修改属性TraceAllow值为true，如下：   <SERVICE CLASS="com.apusic.jdbc.trace.JDBCTracerService">  <ATTRIBUTE NAME="TraceAllow" VALUE="True"/>  <ATTRIBUTE NAME="StackTraceAllow" VALUE="False"/>  </SERVICE> | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 1. 把daytrader.war应用拷贝到域下的applications目录 2. 启动AASv9.0 3. 访问<http://ip:port/daytrader/，然后切换到Configuration标签>，点击标准的“(Re)-create DayTrader Database Tables and Indexes”，然后再点击“(Re)-populate DayTrader Database”，完成系统的初始化   然后切换到Primitives标签，点击[PingHtml](http://localhost:6888/daytrader/PingHtml.html)、[PingServlet2Servlet](http://localhost:6888/daytrader/servlet/PingServlet2Servlet)、[PingJSP](http://localhost:6888/daytrader/PingJsp.jsp)、[PingJSPEL](http://localhost:6888/daytrader/PingJspEL.jsp)[PingHTTPSession2](http://localhost:6888/daytrader/servlet/PingSession2)链接，完成html、servlet、Jsp、EL、Session功能的测试 | | | 点击的在页面中能正确显示内容，没有发生错误 | |  |
| 5 | 参测人员签字 | | |  | | |

## 2.21 国密SSL VPN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **国密算法功能测试** | **测试用例标识** | | **GHtest35** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器支持国密SSL VPN单、双向认证 | | | | |
| **前置条件** | | 已经安装并可以正常启动访问Apusic9.0应用服务器安全版本 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | JDK默认的加密算法长度比国密算法长度要短，并且进行了限制，我们需要修改JDK相关的jar包打破这个限制。解压附件jce\_policy-6.zip，然后把相关的包替换（JRE目录）/lib/security/下相关文件。 | | |  | |  |
| 2 | 修改该域配置文件apusic.conf中的Muxer服务，让其支持SSL VPN认证，如下所示：  <SERVICE CLASS="com.apusic.net.Muxer">  <ATTRIBUTE NAME="SSLEnabled" VALUE="true"/>  <ATTRIBUTE NAME="GMEnabled" VALUE="true"/>  <ATTRIBUTE NAME="SecurePort" VALUE="6889"/>  <ATTRIBUTE NAME="MutualAuthPort" VALUE="6887"/>  <ATTRIBUTE NAME="KeyStore" VALUE="config/server.jks"/>  <ATTRIBUTE NAME="KeyPassword" VALUE="123456"/>  <ATTRIBUTE NAME="TrustStore" VALUE="config/server.jks"/>  <ATTRIBUTE NAME="TrustStorePassword" VALUE="123456"/>  <ATTRIBUTE NAME="SSLProtocol" VALUE="SM1.1"/>  <ATTRIBUTE NAME="NeedClientAuth" VALUE="true"/>  </SERVICE>  SSLEnabled：是否启用SSL协议  GMEnabled： 是否开启国密SSL VPN  SSLProtocol: SSL协议版本，要使用国密SSL VPN，设置值为SM1.1  SecurePort: 为单向认证端口  MutualAuthPort: 为双向认证端口。  KeyStore,TrustStore: 分别为秘钥库，信任库。拷贝附件中的server.jks到域的config目录下，里面包含了服务证书和私钥、信任的客户端证书内容。 | | | | |
| 3 | 安装并配置360360国密专版浏览器 | | |  | |
| 4 | Usb key的安装配置 | | |
| 5 | 启动AAS。通过命令startapusic –gm启动AAS | | |
| 6 | 单向认证  访问<https://ip>:port/6889 | | | 访问成功，表示单向认证成功 | |
| 7 | 双认证  访问<https://ip:port/6887>，输入usbkey密码 | | | 访问成功，表示双认证成功 | |
| 8 | 参测人员签字 | | |  | | |

# 3. 性能测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **静态文本页面并发数** | **测试用例标识** | | **GHtest36** | |
| **测试用例说明（方法、目的）** | | 使用loadrunner测试AAS能够支持多大的并发用户同时访问静态文本页面 | | | | |
| **前置条件** | | 1.已经安装并可以正常访问Apusic应用服务器  2.将中办测试应用（AppTest.war）修改，一个修改为静态10K纯文本页面，另一个修改为带8个每个149k的图片的页面用于效率测试。 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 部署应用（AppTest.war）到aas; | | | 能支持300用户并发 | |  |
| 2 | loadrunner设置300用户并发，运行5分钟，每5秒增加20用户测试静态10K纯文本页面； | | |
| 3 | 监控测试时的内存占用率、CPU使用率、平均响应时间。 | | |
| 4 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **静态附带资源页面并发数** | **测试用例标识** | | **GHtest37** | |
| **测试用例说明（方法、目的）** | | 使用loadrunner测试AAS能够支持多大的并发用户同时访问静态附带图片资源的页面 | | | | |
| 前置条件 | | 1.已经安装并可以正常访问Apusic应用服务器  2. 将中办测试应用修改，一个修改为静态10K纯文本页面，另一个修改为带8个每个149k的图片的页面用于效率测试。 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 部署应用（AppTest.war）到aas; | | | 能支持300用户并发 | |  |
| 2 | loadrunner设置300用户并发，运行5分钟，每5秒增加20用户测试静态带8个每个149k的图片的文本页面； | | |
| 3 | 监控测试时的内存占用率、CPU使用率、平均响应时间。 | | |
| 4 | 参测人员签字 | | |  | | |

# 4. 易用性测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **图形化安装** | **测试用例标识** | | **GHtest38** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器在Windows下可以基于图形化界面进行安装 | | | | |
| 前置条件 | | 获取到Apusic应用服务器exe格式的安装文件 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 双击AAS-V9.0.exe | | | 金蝶Apusic应用服务器图形化安装能够安装成功并且能够启动成功 | |  |
| 2 | 安装图形上的提示进行每一步的操作，选择功能为“金蝶Apusic应用服务器” | | |
| 3 | 设置端口，用户名和密码，根据向导完成每一步的操作 | | |
| 4 | 完成安装后进入安装目录下/domains/mydomian/bin下运行startapusic.cmd启动应用服务器 | | |  | |
| 5 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **图形化界面管理** | | **测试用例标识** | **GHtest39** | |
| **测试用例说明（方法、目的）** | | 管理控制台提供应用部署、数据源配置、服务配置、系统参数配置、日志配置等图形化管理功能 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | **预期结果** | | | **实际结果** |
| 1 | 进入管理控制台，点击菜单“应用管理” | | 在应用管理界面可以对应用进行部署、启动、停止、卸载、监控等操作 | | |  |
| 2 | 进入管理控制台，点击菜单“数据源” | | 在 数据源界面，可以新增、修改参数，启动、停止和卸载数据源 | | |
| 3 | 进入管理控制台，点击菜单“服务面板” | | 服务面板界面，可以对HTTP服务、系统服务、高级服务、线程池服务进行相关参数进行设置 | | |
| 4 | 进入管理控制台，点击“系统配置”菜单下子菜单“系统参数” | | 系统参数界面，可以对JVM参数、Apusic进行设置 | | |
| 5 | 进入管理控制台，点击“系统配置”菜单下子菜单“日志配置” | | 日志配置参数界面，可以对日志级别进行设置 | | |  |
| 6 | 参测人员签字 | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **图形化监控** | **测试用例标识** | | **GHtest40** | |
| **测试用例说明（方法、目的）** | | 安全管理控制台提供了http线程池、数据库连接池、JVM虚拟机等资源的图形监控界面 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 进入管理控制台，点击监控菜单下的子菜单“监控概览” | | | 可以看到活动线程数、数据源、应用会话统计、JVM类加载监控、HTTP连接监控、JVM堆内存等实时动态图 | |  |
| 2 | 进入管理控制台，点击监控菜单下的子菜单“HTTP连接” | | | 可以看到http线程池实时动态图形 | |  |
| 3 | 进入管理控制台，点击监控菜单下的子菜单“数据源”， | | | 可以看到数据库连接池实时动态图形 | |
| 4 | 进入管理控制台，点击监控菜单下的子菜单“JVM内存” | | |  | |
| 可以看到JVM虚拟机实时动态图形 | |  |
| 5 | 进入管理控制台，点击监控菜单下的子菜单“JavaEE组件” | | | 可以看到当前域下面的JSP组件、Servlet组件、EJB组件情况 | |  |
| 6 | 进入管理控制台，点击监控菜单下的子菜单“应用会话” | | | 可以看到应用会话实时动态图形 | |  |
| 7 | 进入管理控制台，点击监控菜单下的子菜单“JTA事务” | | | 可以看到JTAS事务统计实时动态图形 | |  |
| 8 | 进入管理控制台，点击监控菜单下的子菜单“线程” | | | 可以看到总的活动线程数，各线程池活动线程数监控实时动态图形 | |  |
| 9 | 进入管理控制台，点击监控菜单下的子菜单“类加载” | | | 可以看到JVM类加载监控实时动态图形 | |  |
| 10 | 进入管理控制台，点击 “日志监控”菜单 | | | 可以查看到实时生产的日志信息 | |  |
| 11 | 参测人员签字 | | |  | | |

# 5. 安全性测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **安全构架测试** | **测试用例标识** | | **GHtest41** | |
| **测试用例说明（方法、目的）** | | 测试Apusic应用服务器是否采用的JAAS安全认证框架 | | | | |
| 前置条件 | |  | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 查看产品的设计文档； | | | 系统是采用基于JAAS认证框架、授权框架的安全构架 | |  |
| 2 | 确认系统是否采用基于JAAS认证框架、授权框架的安全构架。 | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **数据传输安全测试** | **测试用例标识** | | **GHtest28** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器支持单双向认证，https数据加密传输，支持密码算法加密 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器, Openssl已安装好 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | **单向认证：**  a.修改$APUSIC\_HOME$\domains\mydoamin\config目录下apusic.conf文件  配置文件成功  修改muxer服务配置：  <ATTRIBUTE NAME="SSLEnabled" VALUE="True"/>  b. 重启应用服务器，访问https://localhost:6889查看是否能够打开web管理控制台 | | | | | |
| 2. | **双向认证：**  参考文档 步骤1. 建立自己的认证授权机构CA：  1. 建立工作目录   mkdir ca  cd ca   1. 生成ca私钥   openssl dsaparam -out dsaparam 1024  openssl gendsa -out dsakey dsaparam   1. 生成ca待签名证书   openssl req -new -config openssl.conf -out ca-req.csr -key dsakey  eg：根据提示输入根证书信息，openssl.conf根据实际的路径指定   1. 用CA私钥进行自签名，得到自签名的CA根证书：   openssl x509 -req -in ca-req.csr -out ca-cert.cer -signkey dsakey -days 3650 步骤2. 生成服务器端证书 服务器端证书用来向客户端证明服务器的身份，也就是说在SSL协议握手的时候，服务器发给客户端的证书。生成服务器证书时用到了JDK的密钥管理工具Keytool，本文采用的jdk是sun jdk1.5.0。可以在Sun公司的网站http://java.sun.com下载jdk1.5.0。   1. 生成服务器私钥对及自签名证书，并且保存在密钥库mykeystore中   keytool -genkey -alias myserver -keyalg RSA -keysize 1024 -keypass keypass -storepass keypass -dname "cn=localhost, ou=dev, o=apusic,l=Shenzhen, st=guangdong, c=CN" -keystore mykeystore   1. 生成服务器待签名证书   keytool -certreq -alias myserver -sigalg SHA1withRSA -file server.csr -keypass keypass -storepass keypass -keystore mykeystore   1. 请求CA签名服务器待签名证书，得到经CA签名的服务器证书   openssl x509 -req -in server.csr -out server-cert.cer -CA ca-cert.cer -CAkey dsakey -days 3650 -set\_serial 02   1. 把CA根证书导入密钥库 mykeystore   keytool -import -alias caroot -file ca-cert.cer -noprompt -keypass keypass -storepass keypass -keystore mykeystore   1. 把经过CA签名的服务器证书导入密钥库mykeystore   keytool -import -alias myserver -file server-cert.cer -noprompt -keypass keypass -storepass keypass -keystore mykeystore   1. 把CA根证书导入信任库truststore   keytool -import -alias caroot -file ca-cert.cer -noprompt -keypass keypass -storepass keypass -keystore truststore 步骤3. 颁发个人证书  1. 生成客户端私钥：   openssl genrsa -out clientkey 1024   1. 生成客户端待签名证书：   openssl req -new -config openssl.conf -out client.csr -key clientkey  eg：根据提示输入根证书信息，openssl.conf根据实际的路径指定   1. 请求CA签名客户端待签名证书，得到经CA签名的客户端证书：   openssl x509 -req -in client.csr -out client.cer -CA ca-cert.cer -CAkey dsakey -days 3650 -set\_serial 03   1. 生成客户端的个人证书client.p12：   openssl pkcs12 -export -clcerts -in client.cer -inkey clientkey -out client.p12 步骤4.浏览器导入证书：  1. CA根证书导入客户端   在这里CA的根证书用来在SSL握手时验证服务器发给客户端浏览器的证书。如果没有此证书，浏览器将无法自动验证服务器证书，因此浏览器将弹出确认信息，让用户来确认是否信任服务器证书。 在客户端的IE中使用"工具 -> Internet选项 -> 内容 -> 证书 (受信任的根证书颁发机构)-> 导入"把我们生成的CA根证书**ca-cert.cer**导入，使其成为用户信任的CA。     1. 客户端个人证书导入客户端   在客户端的IE中使用"工具 -> Internet选项 -> 内容 -> 证书(个人) -> 导入"把我们生成的个人证书client.p12导入，使其成为用户的证书。   步骤5. 配置Apusic允许双向认证 Apusic应用服务器默认配置下不支持双向认证，要支持SSL双向认证，  **采用密钥库文件的方式**   1. 在对应的域(如mydomain)目录创建目录，如ca 2. 把相关的文件（mykeystore，truststore）拷贝到ca目录下 3. 修改Muxer服务，如果采用密钥库文件的方式，示例配置如下：   <SERVICE CLASS="com.apusic.net.Muxer">  ……  <ATTRIBUTE NAME="SSLEnabled" VALUE="True"/>  <ATTRIBUTE NAME="SecurePort" VALUE="6889"/>  <ATTRIBUTE NAME="MutualAuthPort" VALUE="6887"/>  <ATTRIBUTE NAME="NeedClientAuth" VALUE="True"/>  <ATTRIBUTE NAME="KeyStore" VALUE="ca/mykeystore "/>  <ATTRIBUTE NAME="KeyPassword" VALUE="keypass"/>  <ATTRIBUTE NAME="TrustStore" VALUE="ca/truststore"/>  <ATTRIBUTE NAME="TrustStorePassword" VALUE="keypass"/>  <ATTRIBUTE NAME="TrustStoreType" VALUE="JKS"/>  ……  </SERVICE>  **步骤6.测试双向认证**  访问AAS默认首页https://localhost:6887，如果前面的操作都正确的话，应该可以看到Apusic的欢迎页面。同时状态栏上的小锁处于闭合状态，且证书为有效状态，表示您已经成功地与服务器建立了要求客户端验证的SSL安全连接。    **Eg:必须使用localhost访问，因为我们的服务器证书写的是localhost,否则会报证书错误。** | | | | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试已有用户名重新注册** | **测试用例标识** | | **GHtest43** | |
| **测试用例说明（方法、目的）** | | 测试Apusic应用服务器需支持户身份标识唯一性检查功能 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器管理控制台 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 在用户管理界面新增一个用户testuser001 | | | 不能注册相同用户 | |  |
| 2 | 在用户管理界面，再次新增用户testuser001。 | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **测试弱口令或空口令** | **测试用例标识** | | **GHtest44** | |
| **测试用例说明（方法、目的）** | | 使用常用扫描工具对中间件进行暴力破解，检测是否存在弱口令或空口令等 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 使用常用扫描工具（比如：appscan,Acunetix WVS等工具）对中间件进行暴力破解，检测是否存在弱口令或空口令等。 | | | 系统不存在弱口令或空口令 | |  |
| 2 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **日志审计** | **测试用例标识** | | **GHtest35** | |
| **测试用例说明（方法、目的）** | | 安全审计管理员可以看到所有用户的异常登陆，修改等操作，比如连续重复登录失败 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器安全管理控制台 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 计员帐户auditor(默认密码1qazXSW@)登陆安全管理控制台； | | | 在审计日志界面可以查看到登录失败的信息，具备审计日志功能 | |  |
| 2 | 点击“审计日志”，可查看到审计日志 | | |
| 3 | 参测人员签字 | | |  | | |

# 6. 维护性测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **配置管理时维护性** | **测试用例标识** | | **GHtest46** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器管控控制平台进行各项管理配置操作时，操作错误给出提示和对应的错误信息 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够访问管理控制台 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管理控制台 | | | 当输入的信息不正确时会给出提示，如不能为空、字符类型不正确等 | |  |
| 2 | 对应用、数据源、服务、日志配置、系统参数等进行设置 | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **运行时维护-日志管理** | **测试用例标识** | | **GHtest47** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器日志实时输出，系统和人为错误会给出提示和对应的错误日志 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 进入管理控制台，在日志配置界面设置日志级别，如INFO | | | 日志监控界面和日志文件中可以查看到实时的日志信息，并且查看到的日志信息的级别与设置的一致 | |  |
| 2 | 对应用系统进行操作，如登录管理，输入正确的用户名和密码，输入错误的密码登录 | | |
| 3 | 切换到日志监控界面查看日志 | | |
| 4 | tail –f /log/apusic.log查看日志文件，确认日志文件中是否会实时记录日志信息 | | |  | |
| 5 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **运行时维护-HTTP线程池监控** | **测试用例标识** | | **GHtest48** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器管理控制台提供运行时HTTP线程池监控的实时监控 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管控控制台 | | | 可以看到http线程池实时动态图形 | |  |
| 2 | 点击监控菜单下的子菜单“HTTP连接 | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **运行时维护-数据库连接池** | **测试用例标识** | | **GHtest49** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器管理控制台提供运行时数据库连接池监控的实时监控 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管控控制台 | | | 可以看到数据库连接池实时动态图形 | |  |
| 2 | 点击监控菜单下的子菜单“数据源” | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **运行时维护- JVM虚拟机监控** | **测试用例标识** | | **GHtest50** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器管理控制台提供运行时JVM内存监控的实时监控 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管控控制台 | | | 可以看到JVM虚拟机实时动态图形，可以查看到堆内存和非堆内存的大小 | |  |
| 2 | 点击监控菜单下的子菜单“JVM内存” | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **JVM堆栈大小配置** | **测试用例标识** | | **GHtest51** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器提供JVM堆栈大小配置 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管控控制台，在系统参数中设置JVM参数，重启应用服务器时 | | | 通过管理或者修改启动脚本中的JVM参数，都生效 | |  |
| 2 | 或者在启动脚本startapusic中修改JVM堆栈大小  MEMORY\_JVMOPTS="-Xms512m -Xmx1024m -XX:MaxPermSize=256m"  重启应用服务器 | | |
| 3 | 检查重启时日志打印的JVM参数是否是修改后的参数 | | |
| 4 | 通过jmap -heap pid(应用服务器进程) 查看JVM参数是否是修改后的值 | | |
| 5 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **垃圾回收方法** | **测试用例标识** | | **GHtest52** | |
| **测试用例说明（方法、目的）** | | Apusic应用服务器支持多种垃圾回收方法设置 | | | | |
| 前置条件 | | Apusic应用服务器已经安装好，并且能够正常运行 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 登录管控控制台，在系统参数中JVM参数设置，设置垃圾回收方法 | | | Apusic应用服务器支持多种垃圾回收方法，如ParallelGC，ConcMarkSweepGC，  ParallelOldGC | |  |
| 2 | 或者在启动脚本startapusic中设置垃圾回收方法，如  set GC\_JVMOPTS=-XX:+UseParallelGC | | |
| 3 | 重启Apusic应用服务器后通过java  -XX:+PrintCommandLineFlags  -version 命令查看设置的垃圾回收方法是否生效 | | |
| 4 | 参测人员签字 | | |  | | |

# 7.可靠性测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **稳定运行测试** | **测试用例标识** | | **GHtest53** | |
| **测试用例说明（方法、目的）** | | 用修改过的中办应用10Ｋ静态页面测试 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic应用服务器 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 部署修改中办应用（AppTest.war） | | | 能稳定运行48小时 | |  |
| 2 | 用loadrunner100用户并发访问10Ｋ静态页面48小时 | | |
| 3 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **网络故障恢复测试－客户端网络故障** | **测试用例标识** | | **GHtest48** | |
| **测试用例说明（方法、目的）** | | 客户端网络故障情况下，网络恢复后，是否在不重启应用服务器的情况下客户端能够重新连接上服务器 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic安全管理控制台 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 启动aas后登陆安全管理控制台； | | | 客户端与服务器之间网络故障后，无须重启应用服务器可恢复。 | |  |
| 2 | 拔除客户端网线制造网络故障； | | |
| 3 | 访问安全管理控制台； | | |
| 4 | 再插上网线，恢复网络； | | |
| 5 | 再次访问安全管理控制台，检查是否可以正常登陆安全管理控制台。 | | |
| 6 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **网络故障恢复测试－服务器端网络故障** | **测试用例标识** | | **GHtest49** | |
| **测试用例说明（方法、目的）** | | 服务器网络故障情况下，网络恢复后，是否在不重启应用服务器的情况下客户端能够重新连接上服务器 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic安全管理控制台 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 启动aas后登陆安全管理控制台； | | | 客户端与服务器之间网络故障后，无须重启应用服务器可恢复。 | |  |
| 2 | 拔除服务器端网线制造网络故障； | | |
| 3 | 访问安全管理控制台； | | |
| 4 | 再插上网线，恢复网络； | | |
| 5 | 再次访问安全管理控制台，检查是否可以正常登陆安全管理控制台。 | | |
| 6 | 参测人员签字 | | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | **易恢复性测试** | **测试用例标识** | | **GHtest50** | |
| **测试用例说明（方法、目的）** | | 数据库网络故障恢复后，可以直接恢复应用与数据库的连接 | | | | |
| 前置条件 | | 已经安装并可以正常访问Apusic安全管理控制台 | | | | |
| **测试执行步骤** | | | | | | |
| **序号** | **测试输入** | | | **预期结果** | | **实际结果** |
| 1 | 在安全管理控制台创建daytrader应用的数据源（jdbc/TradeDataSource  ）并部署daytrader应用; | | | 在第3步可正常登陆、退出daytrader应用 | |  |
| 2 | 关闭数据库端的网络并访问daytrader应用并进行登陆、退出等与数据库相关操作； | | |
| 3 | 恢复数据库端的网络并访问daytrader应用并进行登陆、退出等与数据库相关操作。 | | |
| 4 | 参测人员签字 | | |  | | |