

文档名称: **自动化框架设计和改进方案**

版本: **0.1**

文档状态: **讨论中**

文档类型: **功能描述**

文档创建人: **徐卫芳**

最后更新日期: **2016/9/6**

**注意:**

此文档为景行锐创软件有限公司所有，其只能在景行锐创内部使用。任何公司，组织和个人未经同意不得拷贝，传播与使用。

|  |
| --- |
| JH Innovation Software Co. 功能描述 |

自动化设计框架及易用性改进方案

|  |  |
| --- | --- |
| **创建人** | 徐卫芳 |
| **复核人** |  |
| **授权** | 内部使用 |
| **项目名称** | 自动化设计框架及易用性改进方案 |
| **项目编号#** |  |
| **项目描述** | 该项目设计了自动化case的框架，增加了自定义功能，扩展了自动化库的灵活性和易用性。 |

文档变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 作者 | 变更描述 |
| 0.1 | 2016/9/8 | 徐卫芳 | 初稿 |
|  |  |  |  |

目录

[1． 概述 4](#_Toc461185499)

[1.1 目的 4](#_Toc461185500)

[1.2 关键词，概念，流行语，专业名词 4](#_Toc461185501)

[2． 详述 4](#_Toc461185502)

[2.1 功能需求 4](#_Toc461185503)

[2.1.1 简要描述 4](#_Toc461185504)

[2.1.2 功能范围 4](#_Toc461185505)

[2.1.3 前提条件与依赖 4](#_Toc461185506)

[2.1.4 功能描述 5](#_Toc461185507)

[2.1.4.1 详细描述 5](#_Toc461185508)

[2.1.4.2 特性交互与影响 6](#_Toc461185509)

[2.1.5 接口需求 6](#_Toc461185510)

[2.1.5.1 用户特性 6](#_Toc461185511)

[2.1.5.2 配置描述 6](#_Toc461185512)

[2.1.5.3 命令行接口描述 6](#_Toc461185513)

[2.1.5.4 图形用户界面描述 7](#_Toc461185514)

[2.1.5.5 API 需求 7](#_Toc461185515)

[2.1.6 安装需求 7](#_Toc461185516)

[2.1.7 许可证需求 7](#_Toc461185517)

[2.1.8 文档需求 7](#_Toc461185518)

[2.2 用户用例 7](#_Toc461185519)

[2.2.1 用例1： 7](#_Toc461185520)

[2.3 非功能需求 10](#_Toc461185521)

[2.3.1 性能与吞吐量 10](#_Toc461185522)

[2.3.2 兼容性 10](#_Toc461185523)

[2.3 参考文献 10](#_Toc461185524)

[2.4 附录 11](#_Toc461185525)

# **概述**

## 目的

1. 针对自动化case，做整体框架设计。通过使用robot framework 的Test Suites、Test Tags等功能，分层次管理 case，提高使用case的灵活性；
2. 修改和添加相关测试库函数，使得判断集群状态、增加自动化case更加方便；
3. 增加自定义功能，用户可以自定义路径、节点、用户名等。

## 关键词，概念，流行语，专业名词

1. Robot Framework：Robot Framework 是一款基于 Python 的功能自动化测试框架。
2. Test suites：Robot Framework中测试用例的集合被称为test suites。每个包含测试用例的文件形成了一个test suites。
3. Setup 和 Teardown：如果需要在每个测试执行之前和之后执行某些操作，可以使用Test Setup和Test Teardown。
4. Tags：Robot Framework允许给测试用例的设置标签。通过Default Tags或Force Tags可以所有的测试用例打上标签（即：test suite 级别的标签）。也可以给单个测试用例打上标签（test case级别的标签）。
5. Robot framework-ride：RIDE是一个图形界面的用于创建、组织、运行测试的软件。

# **详述**

## 功能需求

### 简要描述

1. 将自动化case按照功能分成不同的suite，给每个suite设置force tags。然后将每个suite中的test case按照优先级、属性等设置不同的tag，一个case可以标记多个tags。使用suite和tag方式管理case，使得测试更加灵活。
2. 针对目前自动化测试库限制节点个数、节点名、用户名、安装路径、文件存放路径等问题，提出了通过增加自动化使用脚本和自动化解析库的方式来解决该问题，使得自动化测试环境更加灵活。

### 功能范围

**范围内**

Python2.6.6及以上版本

Master节点Linux节点为

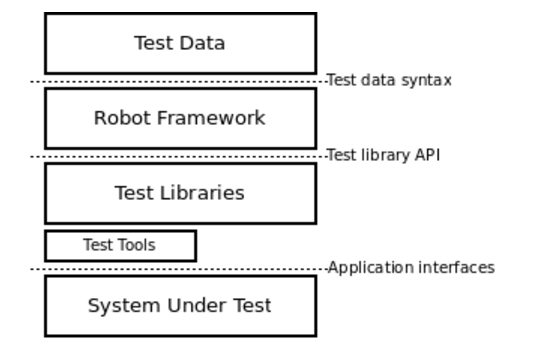
**范围外**

mater节点为windows系统

### 前提条件与依赖

1. 平台依赖（版本号）

自动化测试依赖于robot framework测试框架。该框架介绍如下：Robot Framework 是一款基于 Python 的功能自动化测试框架。它具备良好的可扩展性，支持关键字驱动，可以同时测试多种类型的客户端或者接口，可以进行分布式测试执行。主要用于重复性的回归测试以及验收测试驱动开发（ATDD），同时还可以支持行为驱动开发（BDD）。Robot Framework有着高度模块化得架构如下图所示：



测试数据是一个简单的易于编辑的表格。当Robot Framework启动时，对测试数据进行处理，执行测试用例，并生成日志和报告。测试框架不用知道任何测试应用程序的信息和测试库与应用程序的交互。测试库可直接使用应用程序的接口，或使用低一级的测试工具作为驱动。

1. python2.6.6。
2. Robot Framework 版本号：2.9
3. wxWidets-2.8.12
4. Robot framework-ride 版本号：1.3

### 功能描述

本章通过如下几节，对该项目的功能进行了详细的描述，细节如下。

#### 详细描述

1.将自动化中的case按照功能分成不同的suite，如下图所示：每个命令行、每种调度策略分别是一个suite，每个suite用一个force tags进行标记（该suite中的每个test case都会继承该tags并且tags不能被修改，如果是default tags，则可以被test case修改），所有命令行suite文件组成一个命令行文件夹。

若需要测试所有命令行功能是否正确，只需要执行：pybot jhds（jhds是文件夹名） 即可；若要单独执行某个调度策略，只需要执行：pybot Suite-fcfs.txt(Suite-fcfs.txt是一个Test Suite)即可。

每个Test Suite中的case按照优先级、属性等设置不同的tag。若需要运行每个Test Suite中标记为negative的case，以fcfs.txt为例：执行pybot --include negative --test case\* fcfs.txt即可。

备注：suite的命令格式为：suite\*，即必须以suite开头。Test case的命令格式为：case\*，即必须以case开头。

自动化case整体设计图：autotest4.0



自动化目录结构如下：

autotest4.0

├──testcase

│ └── jhds

│ └── suite-command.txt

│ └── suite-conf.txt

│ └── ……

│ └── scheduling

│ └── suite-fcfs.txt

│ └── suite-fairshare.txt

│ └── ……

├── lib

│ └── hostLib.py

│ └── myUtils.py

│ └── ……

└──spooler

│ └──jhds

│ └──command-case1-input.sh(input文件)

│ └── ……

2.针对现有自动化测试库，将判断节点状态相关函数进行改进，之前行为是：只有当集群中的所有节点状态均为ok，才可以提交作业，否则默认集群环境有问题，不执行该case。修改为判断指定个数的节点是否ok，只要ok的节点个数和指定个数一致，即可以提交作业。

3.增加解析库，eg：params.conf、hosts.conf、users.conf、queues.conf等 解析库。（elim动态收集资源没有考虑在内）

4.设置节点变量：eg：MSTER、LINUXA、WINA、WINB。将case中所有和节点相关的地方对应进行替换，这个只需替换autotestfile.txt文件。

5.设置用户变量：eg：CLUSTER\_ADMIN、USER1-USER10自定义用户名。目前，自动化中使用的集群管理用户为jhadmin并且需要有user1-user10用户，如果不想使用这些用户。那么需要对应替换autotestfile.txt、users.conf、scheduler.conf等文件中的用户名。

6.设置自动化包放置路径：将自动化包放到某个路径下，然后将该路径通过变量AUTOTEST\_TOP设置为环境变量。

##### **实现方案**

1.编写脚本autotest\_prepare.sh，匹配更换节点、用户名、路径等。

2.将函数checkClusterStatus(checkTime=60,user="jhadmin")改为：checkClusterStatus(checkTime=60,num，type，user="jhadmin")，即：执行jhhosts list、jhosts查询一个节点的状态，只有当这两个命令查询出来的结果均为ok时，该节点状态被认为ok。

3.编写 params.conf、hosts.conf、users.conf、queues.conf、share.conf等 解析库（暂时没有考虑elim）。支持增加、修改、删除参数配置等功能。

#### 特性交互与影响

TBD

### 接口需求

#### 用户特性

* **最终用户**
* **管理员**

TBD

#### 配置描述

##### **调度脚本**

TBD

##### **权限**

#### 命令行接口描述

N/A

#### 图形用户界面描述

N/A

#### API 需求

N/A

### 安装需求

TBD

### 许可证需求

N/A

### 文档需求

TBD

## 用户用例

### 用例1：

1. 安装好robotframework测试工具。(安装手册、安装包位置：svn://192.168.0.30/tools/trunk/TestAutomation/JHUnischeduler-4.0/自动化安装包)
2. 安装好unischeduler4.0集群（两个linux节点，两个windows节点，master节点为linux）。
3. 更改autotest\_prepare.sh脚本。脚本内容如下：

#!/bin/sh

MASTER:host1 (将host1修改为自己的mster 节点名，linux节点)

LINUXA:host2 (将host2修改为自己的slave 节点名，linux节点)

WINA:host3 (将host3修改为自己的slave节点名，windows节点)

WINB:host4 (将host4修改为自己的slave节点名，windows节点)

CLUSER\_ADMIN=userxx(修改处，集群管理员用户名)

USER1=user1 (修改处，普通域用户，maste节点和所有slave节点上必须共享该用户)

USER2=user2 (修改处，普通域用户，maste节点和所有slave节点上必须共享该用户)

USER3=user3 (修改处，普通域用户，maste节点和所有slave节点上必须共享该用户)

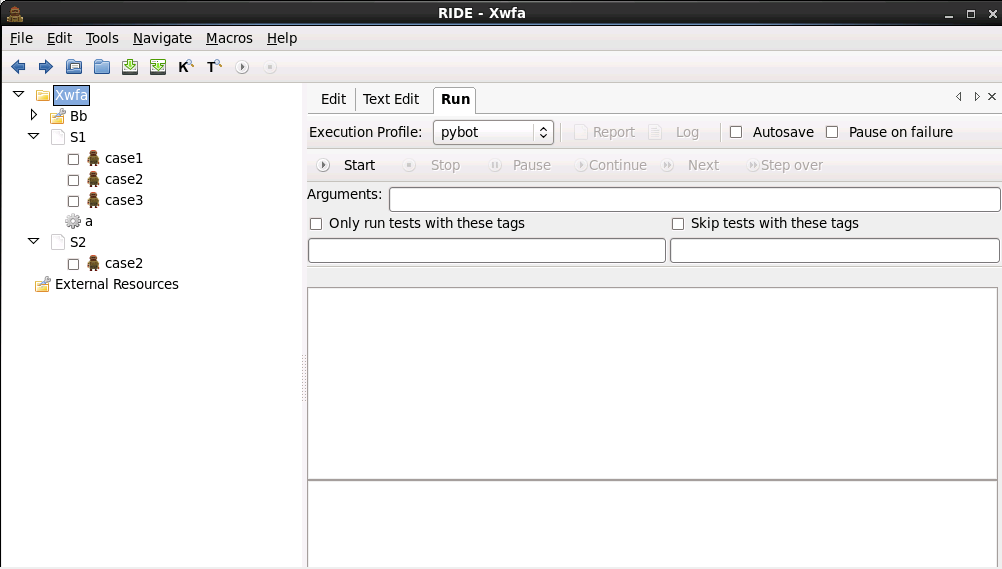
………………………..

USER4=user10 (修改处，普通域用户，maste节点和所有slave节点上必须共享该用户)

export AUTOTEST\_TOP=/apps/（放置自动化测试包的目录，这个需要共享目录, eg：/apps)

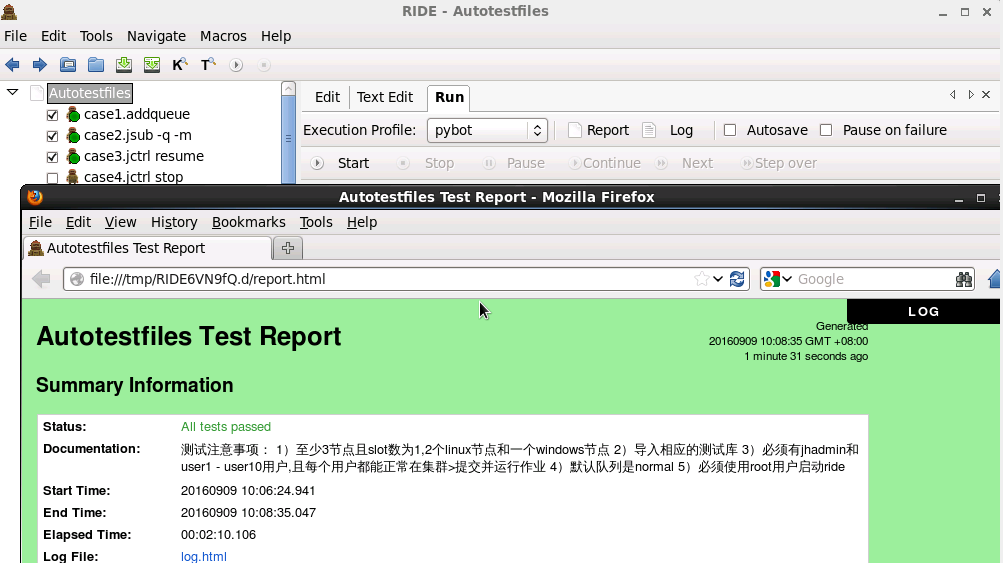
…………………………..

1. 用root用户执行autotest\_prepare.sh脚本。
2. 用图形界面执行case步骤：
3. 用root用户执行ride.py打开测试界面：

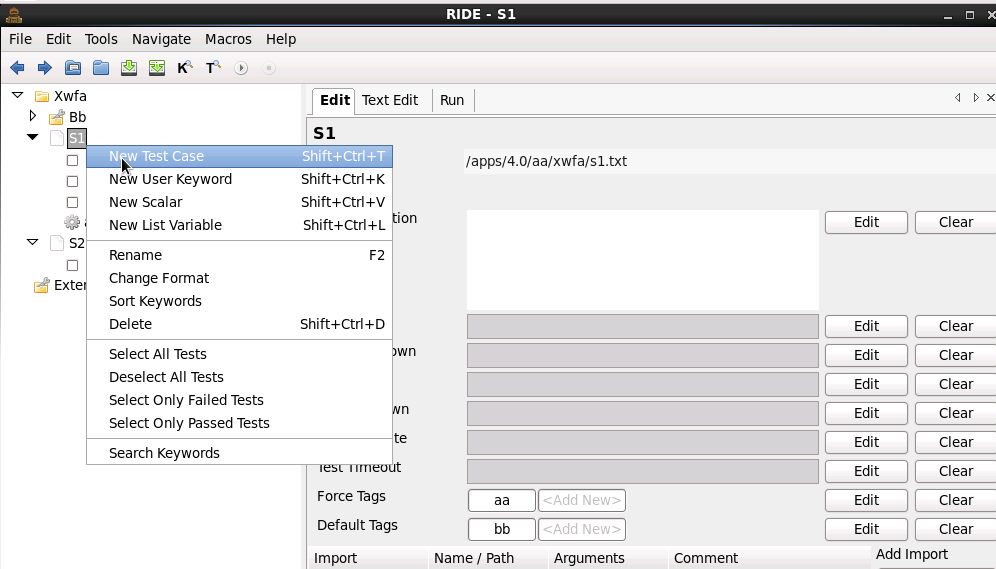


点击右侧的start按钮，即可以run所有test case。也可以勾选左侧的方框，run部分test case。

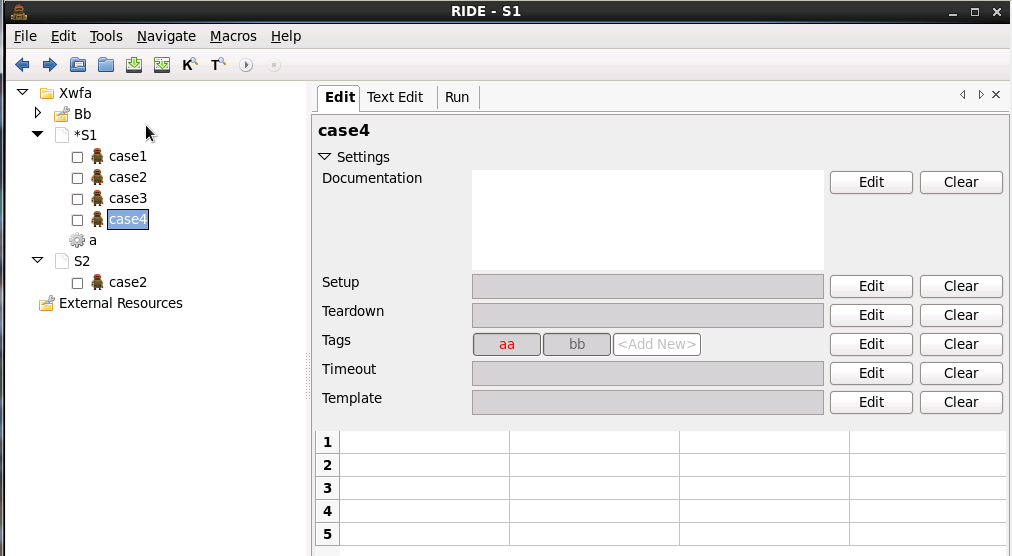
1. 查看测试结果：点击右侧的Report、Log按钮会出现对应的html界面，即可查看test case的总体报告和详细信息。



1. 若要添加case，需要将鼠标点到相应的test suite（eg：S1）上，然后点击右键，即可新建case：



每个case对应的有Setup、Teardown、Tags、Timeout等属性，在对应的方框中填写内容即可。右侧下方以阿拉伯数字开头的方框可以编写test case的详细内容。编写完test case后，点击左上方两个绿色的向下箭头便可以保存case。



1. 修改case：在test case的方框中更改对应方框内容即可。
2. 如果不使用图形界面的形式，使用一下步骤对test case进行操作：

添加case：

首先新建一个文件夹xwfa，然后在xwfa中新建一个s1.txt文件（此时s1，即为一个test suite），然后在s1.txt安装自动化书写格式进行书写（书写格式详见robotframework），便可以添加多个test case：

\*\*\* Settings \*\*\*

Force Tags tag1 tag2

Default Tags tag3

Library /apps/4.0/autotest4.0/lib/OperatingSystem.py

\*\*\* Test Cases \*\*\*

case1.get hostname

[Setup] pretest

[Tags] Tag4 Tag5  
 RUN hostname  
 shouldcontain host1  
 [Teardown] posttest

case2.jsub –J

[Setup] pretest

[Tags] Tag6 Tag7  
 checkClusterStatus 60  
 ${job1} QuerySubmitInfo su jhadmin -c "jsub -J xujob1 sleep 1000"  
 ${jobid1} getJobId ${job1}  
 sleep 8  
 ${job1} queryJobInfo ${jobid1}  
 ${name} getJobName ${job1}  
 containOneKeyword ${name} xujob1  
 [Teardown] postTest

修改case：

直接修改该txt文件对应的内容即可。

执行case：

pybot s1.txt（执行套件 s1中的所有case）

pybot –include tag0 /xwfa(执行xwfa中标签为tag0的suite的所有case)

pybot –include tag1–test case\* s1（执行套件 s1中所有标签为tag1的case）

查看执行结果：

在当前目录下会生成三个文件report.html、log.html、output.xml，这三个文件分别是test case的测试结果摘要、详细的测试结果、portbable XML格式的测试结果（方便与其他工具集成）。

## 非功能需求

TBD

### 性能与吞吐量

TBD

### 2.3.2 兼容性

不兼容unischeduler3.2

## 参考文献

TBD

《robot framework快速入门》

<http://robotframework.org/#documentation>

<http://blog.csdn.net/yydoraemon/article/details/13847781>

<http://www.cnblogs.com/fnng/p/3871712.html>

## 附录