**通用模拟器使用配置手册**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号： | V1.0.0.0 | 文档编号： |  |
| 文档密级： | 保密 | 归属部门/项目： |  |
| 产品名： | MISC | 子系统名： |  |
| 编写人： |  | 编写日期： | 2016-01-07 |

cid:_Foxmail.1@c348d6be-bd89-58f0-4132-cbfa386c34e5

**卓望数码技术（深圳）有限公司 版权所有**

**内部资料 注意保密**

**所有权声明**

本文档的内容将做定期性的变动，且不另行通知。更改的内容将会补充到本手册中。

除特别声明外，此文档所用的公司名称、个人姓名及数据均属为说明的目的而模拟。

本文档的版权属卓望数码技术（深圳）有限公司（本公司）所有，受中华人民共和国法律的保护。

本文档所含的任何构思、设计、工艺及其他技术信息均属本公司所有，受中华人民共和国法律的保护。未经本公司书面同意，任何单位和个人不得擅自摘抄、全部或部分复制本书内容，或者以其他任何方式使第三方知悉。

**修订记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订人** | **修订日期** | **修订描述** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1 简介 5](#_Toc439961775)

[1.1 目的 5](#_Toc439961776)

[1.2 定义、首字母缩写词和缩略语 5](#_Toc439961777)

[2 模拟器组件介绍 5](#_Toc439961778)

[2.1 模拟器构成 5](#_Toc439961779)

[2.1.1 SimLibrary： 6](#_Toc439961780)

[2.1.2 SimServer 6](#_Toc439961781)

[2.2 功能实现原理 7](#_Toc439961782)

[3 模拟器服务配置配置 7](#_Toc439961783)

[3.1 调用查询服务配置 7](#_Toc439961784)

[3.2 模拟服务配置 8](#_Toc439961785)

[3.2.1 HTTPServer 8](#_Toc439961786)

[3.3 模板配置 9](#_Toc439961787)

[3.3.1 请求模板配置 9](#_Toc439961788)

[3.3.2 应答模版配置 10](#_Toc439961789)

[4 模拟器调用过程介绍 10](#_Toc439961790)

[4.1 模拟器调用包 10](#_Toc439961791)

[4.2 模拟器调用关键字 11](#_Toc439961792)

[4.3 “模拟器请求包构造”关键字 11](#_Toc439961793)

[4.4 “模拟器调用请求”关键字 13](#_Toc439961794)

[5 应用举例 13](#_Toc439961795)

[5.1 模板配置 13](#_Toc439961796)

[5.1.1 请求模板配置 13](#_Toc439961797)

[5.1.2 应答模版配置 14](#_Toc439961798)

[5.2 测试过程 15](#_Toc439961799)

[5.2.1 模拟器调用过程 15](#_Toc439961800)

[5.2.2 被测系统向模拟器发送请求 16](#_Toc439961801)

[5.2.3 模拟器向被测系统返回消息 17](#_Toc439961802)

[5.2.4 模拟器调用结果查询 18](#_Toc439961803)

# 简介

## 目的

本文档用户指导测试人员如何使用模拟器组件进行接口自动化测试。介绍模拟器组件原理；服务器配置和模板配置方法，并提供应用示例；

## 定义、首字母缩写词和缩略语

| **缩略语/术语** | **全 称** | | **说 明** |
| --- | --- | --- | --- |
|  |  |  | |
|  |  | |  |
|  |  | |  |

# 模拟器组件介绍

## 模拟器构成

模拟器组件由两个部分组成：模拟器组件的Robot Framework调用库SimLibrary，服务器端引擎SimServer。

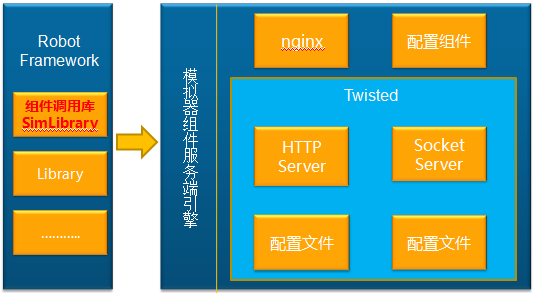


图2-1 模拟器框架

### SimLibrary：

SimLibrary作为Robot Framework的库提供，与RIDE配合使用，通过关键字封装使用。SimLibrary与SimServer交互，告诉SimServer，对被测系统的消息接口校验规则，返回应答消息的规则。

### SimServer

SimServer在基于事件驱动的网络引擎Twisted基础上实现，根据配置启动了一个或多个HTTP Server或TCP Server，这些Server能够模拟第三方系统，接收被测系统的请求，根据SimLibrary设定的校验规则和应答规则，对被测系统请求进行校验，校验通过再根据返回规则发送应答包。

对被测系统的消息接口校验规则支持：IP地址校验，超时控制，消息内容校验；消息内容校验支持：

1. 完全字段匹配：字段值等于期望值；
2. 空字段：期望消息中存在该字段，但为空值；
3. 字段不存在：期望消息中不存在该字段；
4. 字段非空校验：期望消息中存在该字段，且值不为空；
5. 字段格式校验：对字段格式校验，如时间格式，长度；

返回应答消息的规则控制支持：

1. 不返回；
2. 延迟返回；
3. 返回内容可以根据测试案例需要返回成功，返回不同错误码的失败；

## 功能实现原理

一个典型的接口自动化过程由以下步骤组成：

1. 数据环境准备：准备被测系统的测试数据，启动一个模拟器，通知模拟器接收何种消息，如何应答；
2. 发起测试：测试工具向被测系统发起请求，或通过测试工具启动被测系统一个Job；
3. 数据检验：检查被测系统应答，数据库，日志或文件；
4. 环境数据清理：清理脏数据，避免对其他测试案例的运行有影响；

模拟器组件在自动化测试中的工作过程为：

1. Robot Framework通过调用SimLibrary库，通知模拟器服务端SimServer对被测系统请求的校验规则，应答规则；
2. Robot Framework向被测系统发起测试；
3. 被测系统向模拟器服务端SimServer发送请求；
4. SimServer根据SimLibrary设定的校验规则，检验请求消息；
5. 校验通过，SimServer根据SimLibrary设定的应答规则，返回应答；

图2-2 功能原理图

# 模拟器服务配置配置

## 调用查询服务配置

**模拟器调用查询服务**是用来供模拟器的RF库（simLibrary.py）调用，用来设置模拟器的对请求消息的校验规则和返回应答消息的规则；**模拟器调用查询服务**端口库配置在文件conf/system.xml：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <server>  <system>  <port>8000</port>  </system>  …. |

模拟器调用地址为: <http://ip:8000/simSet>

模拟器查询地址为: <http://ip:8000/simQuery>

## 模拟服务配置

### HTTPServer

修改conf/system.xml配置文件，增加httpServer。

|  |
| --- |
| <server>  …  <httpServer version="http1.1" port="9080" msgFormat="json" msgTypeFlag="BusiCode" desc="ismp boss">  <template msgType="OI\_PrepayOrReleasePayOrder" desc="" delayms="0" >  <request>conf/template/ISMP/boss/PrepayOrReleasePayOrder\_req.json</request>  <response>conf/template/ISMP/boss/PrepayOrReleasePayOrder\_rsp.json</response>  </template>  </httpServer>  ...  <server> |

每个httpServer接收一个或多个被测系统请求消息。针对每个请求消息，配置一个消息模板。httpServer包含以下几个属性：

|  |  |  |
| --- | --- | --- |
| 属性名 | 含义 | 说明 |
| Version | 协议版本 | http协议版本 |
| port | 端口 | http端口 |
| msgFormat | 消息格式 | 主要有三种：json、xml、KeyValue |
| msgTypeFlag | 消息类型标识 | 如果是json、xml类型的消息，一般会有某个字段标识消息类型，从而定位到模板，msgTypeFlag取值为该字段的名字；  如果消息格式是KeyValue类型，从消息的内容一般无法识别消息的类型，一般用URL区分，msgTypeFlag取值为“URL” |
| desc | 描述 | 描述httpServer信息 |

模板(template)属性说明

|  |  |  |
| --- | --- | --- |
| 属性名 | 含义 | 说明 |
| msgType | 消息类型 | 模拟器根据msgType定位模板 |
| desc | 描述 | 模板描述 |
| delayms | 延迟时间 | 单位毫秒 |
| encoding | Custom encoding编码 | 缺省为'UTF-8'，可以为'GBK' |

request和response元素指向一个配置文件，分别用于配置请求模板和应答模板。

## 模板配置

### 请求模板配置

根据在system.xml文件中路径添加配置文档，必须根据文档里面所定义的指定其对应模版文件路径和文件名，例如：

<request>conf/template/ISMP/boss/PrepayOrReleasePayOrder\_req.json</request>

请求的配置模版为：conf\template\ISMP\boss\PrepayOrReleasePayOrder\_req.json

|  |
| --- |
| {  "Request": {  "BusiParams": {  "PayMode": "$${PayMode}$$",  "OrderNo": "$${OrderNo}$$",  "MobNum": "$${MobNum}$$",  "UserID": "$${UserID}$$",  "PayNum": "$${PayNum}$$",  "CommodityID": "$${CommodityID}$$",  "Remark": "",  "Unit": "$${Unit}$$",  "PayType": "$${PayType}$$",  "VirtulUserID": "XXXXX13765330673"  },  "BusiCode": "$${BusiCode}$$"  },  "PubInfo": {  "TransactionId": "201",  "TransactionTime": "123456",  "OpId":"OpId",  "OrgId":"OrgId",  "InterfaceId":"6110",  "InterfaceType":"51"  }  } |

### 应答模版配置

<response>conf/template/ISMP/boss/PrepayOrReleasePayOrder\_rsp.json</response>

也需要根据其路径和文件名来找到对应的模版：

|  |
| --- |
| {  "Response": {  "ErrorInfo": {  "Code": "$${Code}$$",  "Message": "$${Message}$$",  "TransactionId": "##{TransactionId}##",  "TransactionTime": "##{TransactionTime}##",  "TransIDH": "##{TransactionId}##",  "TransAccount": "201506",  "BusiCode": "OI\_PrepayOrReleasePayOrder"  },  "RetInfo": {  }  }  } |

# 模拟器调用过程介绍

## 模拟器调用包

模拟器调用通过向模拟器服务端的“模拟器调用”服务发送请求(图2-2 功能原理图 第一步)，设置模拟器对被测系统请求消息的校验规则和应答消息返回规则。

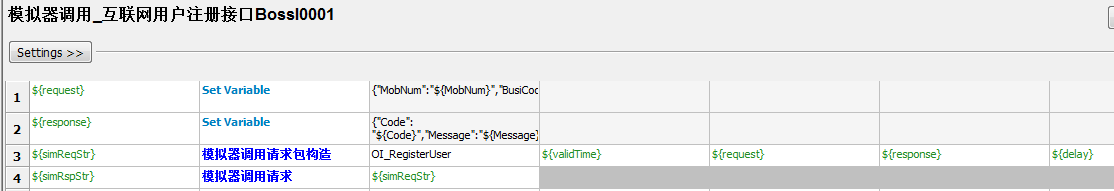
模拟器调用消息如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <message>  <hostIp>10.1.5.79</hostIp>  <validTime>20</validTime>  <msgType>OI\_PrepayOrReleasePayOrder</msgType>  <delay>0</delay>  <request>  <PayMode>01</PayMode>  <BusiCode>OI\_PrepayOrReleasePayOrder</BusiCode>  <CommodityID>20160107175444561426</CommodityID>  <MobNum>15012880663</MobNum>  <PayNum>100</PayNum>  <OrderNo>20160107175444561426</OrderNo>  <UserID>100000000001</UserID>  <PayType>02</PayType>  <Unit>01</Unit>  </request>  <response>  <Message>success</Message>  <Code>0000</Code>  </response>  </message> |

## 模拟器调用关键字

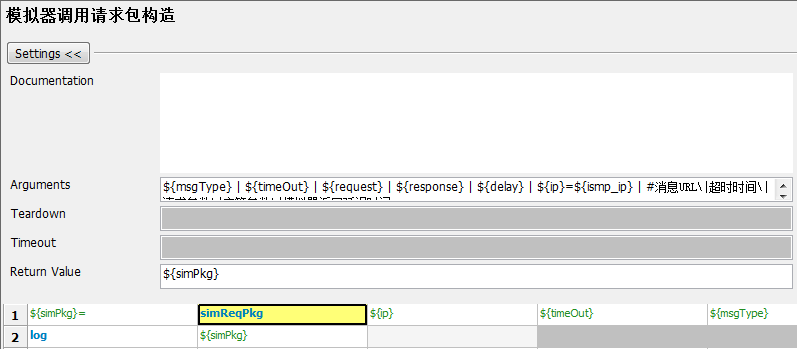
${request}变量保存请求包的变量，dict类型，格式为：{"key1":"value1","key2":"value2"}

${respsone}变量保存应答包的变量，dict类型，格式为：{"key1":"value1","key2":"value2"}



${request}和${respsone}通过“模拟器请求报构造”关键字构造模拟器调用请求包；

## “模拟器请求包构造”关键字



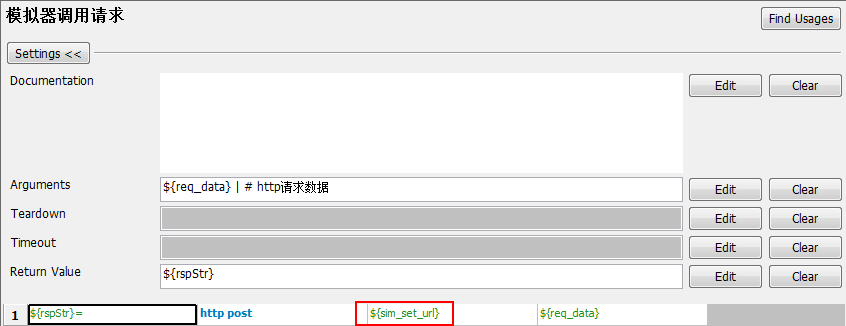
模拟器请求包构造关键字完成请求包构造，关键字的参数有：

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 说明 |
| msgType | 消息类型 | 模拟器根据msgType定位模板 |
| timeOut | 超时时间 | 单位秒，指定时间内没有收到被测系统请求，模拟器会当做超时处理 |
| request | 请求 | 模拟器调用请求参数dict，里面根据"key1":"value1"的方式来填写系统字段 |
| response | 应答 | 模拟器调用响应参数dict，里面根据"key1":"value1"的方式来填写系统字段 |
| delay | 延迟时间 | 单位秒，模拟器在收到被测系统请求后，按照指定时间延迟后回应答 |
| ip | IP地址 | 被测系统IP地址 |
| custom\_header | 自定义头 | 模拟器返回自定义头参数dict，里面根据"key1":"value1"的方式来填写系统字段，value为"REPLY"表示从请求包中获取值后在返回包中回中入，否则以指定的key,value 为头 |

这些参数作为simReqPkg关键字的输入，用于构造模拟器调用请求包，将请求报保存到${simPkg}变量，

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 说明 |
| ${simPkg} | 变量名 | 用于接收simReqPkg它的值 |

## “模拟器调用请求”关键字

****

“模拟器调用请求”负责将模拟器调用请求包，通过http post关键字发送给模拟器服务端模拟器调用服务。${sim\_set\_url}对应模拟器调用服务的URL。http post关键字使用了httpLibrary.py库文件的http\_post()函数。

# 应用举例

## 模板配置

### 请求模板配置

根据在system.xml文件中路径添加配置文档，必须根据文档里面所定义的指定其对应模版文件路径和文件名，例如：

<request>conf/template/ISMP/boss/PrepayOrReleasePayOrder\_req.json</request>

请求的配置模版为：conf\template\ISMP\boss\PrepayOrReleasePayOrder\_req.json

|  |
| --- |
| {  "Request": {  "BusiParams": {  "PayMode": "$${PayMode}$$",  "OrderNo": "$${OrderNo}$$",  "MobNum": "$${MobNum}$$",  "UserID": "$${UserID}$$",  "PayNum": "$${PayNum}$$",  "CommodityID": "$${CommodityID}$$",  "Remark": "",  "Unit": "$${Unit}$$",  "PayType": "$${PayType}$$",  "VirtulUserID": "XXXXX13765330673"  },  "BusiCode": "$${BusiCode}$$"  },  "PubInfo": {  "TransactionId": "201",  "TransactionTime": "123456",  "OpId":"OpId",  "OrgId":"OrgId",  "InterfaceId":"6110",  "InterfaceType":"51"  }  } |

### 应答模版配置

<response>conf/template/ISMP/boss/PrepayOrReleasePayOrder\_rsp.json</response>

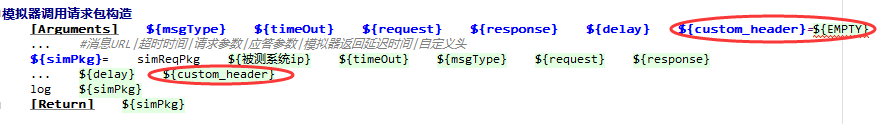
也需要根据其路径和文件名来找到对应的模版：

|  |
| --- |
| {  "Response": {  "ErrorInfo": {  "Code": "$${Code}$$",  "Message": "$${Message}$$",  "TransactionId": "##{TransactionId}##",  "TransactionTime": "##{TransactionTime}##",  "TransIDH": "##{TransactionId}##",  "TransAccount": "201506",  "BusiCode": "OI\_PrepayOrReleasePayOrder"  },  "RetInfo": {  }  }  } |

### 自定义头配置

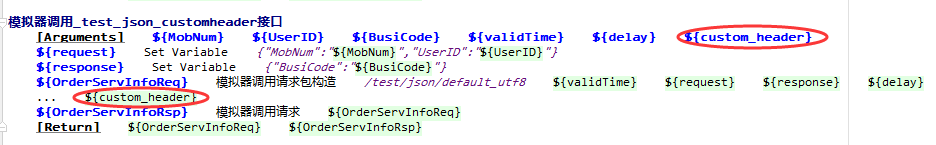
#### 通过SimClient.py进行封装配置

#### "模拟器调用请求包构造"增加新的参数



缺省值为${EMPTY}，使得该功能W向前兼容

#### "模拟器调用\_自定义接口"增加新的参数



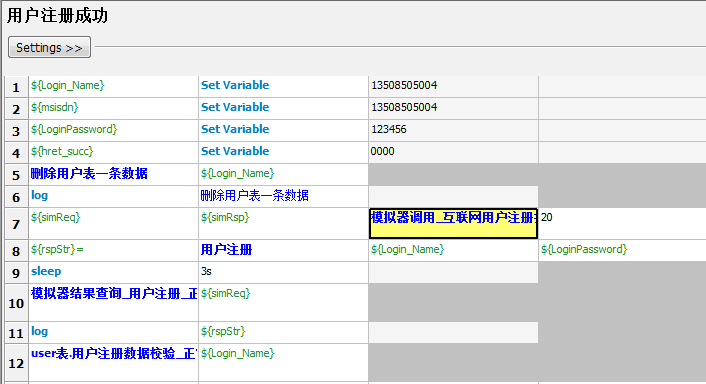
对于每个接口，如果有自定义头的要求，作为参数带入

## 测试过程

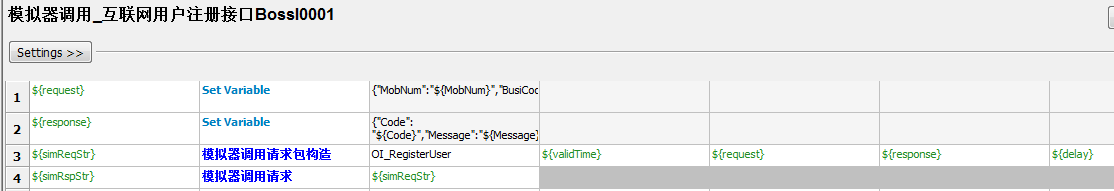
### 模拟器调用过程

#### 发起模拟器调用

测试案例中调用模拟器调用关键字



模拟器调用关键字向模拟器服务端的模拟器调用服务发送调用请求



#### 模拟器调用请求日志

模拟器服务端收到Robot Framework模拟器调用消息的日志如下：



### 被测系统向模拟器发送请求

当Robot Framework向被测系统发起用户注册后，被测系统向模拟器发送请求，日志如下：

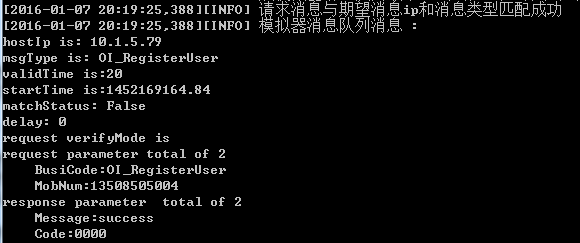


模拟器将收到的消息和期望接收队列中的消息比较，比较过程如图



### 模拟器向被测系统返回消息

比较成功，模拟器向被测系统返回应答，日志如下





### 模拟器调用结果查询

最后，可以通过模拟器查询接口，查询消息匹配结果。模拟器结果查询请求包与模拟器调用包一直。



