**Action详解**

CMC的功能就是用来进行监控和处理故障的，当监控的对象出现问题时我们就需要告知维修人员并进行自动修复。那么从监控，到出现监控异常，将异常修复，监控异常报告给维修人员这一系列动作需要CMC的Action来关联完成。

目录

**[1.配置action](#_Toc25588_WPSOffice_Level1)** **[1](#_Toc25588_WPSOffice_Level1)**

[1） 基础信息](#_Toc18689_WPSOffice_Level2) [1](#_Toc18689_WPSOffice_Level2)

[2） 命令操作](#_Toc1563_WPSOffice_Level2) [2](#_Toc1563_WPSOffice_Level2)

[3） 邮件操作](#_Toc3407_WPSOffice_Level2) [3](#_Toc3407_WPSOffice_Level2)

[4） Recover功能](#_Toc22168_WPSOffice_Level2) [3](#_Toc22168_WPSOffice_Level2)

**[2. 实现原理](#_Toc18689_WPSOffice_Level1)** **[4](#_Toc18689_WPSOffice_Level1)**

[1） 处理流程](#_Toc20159_WPSOffice_Level2) [4](#_Toc20159_WPSOffice_Level2)

[2） 脚本创建](#_Toc22578_WPSOffice_Level2) [6](#_Toc22578_WPSOffice_Level2)

[3） 返回值保存](#_Toc23633_WPSOffice_Level2) [8](#_Toc23633_WPSOffice_Level2)

[4） 远程机与主机交互](#_Toc19450_WPSOffice_Level2) [8](#_Toc19450_WPSOffice_Level2)

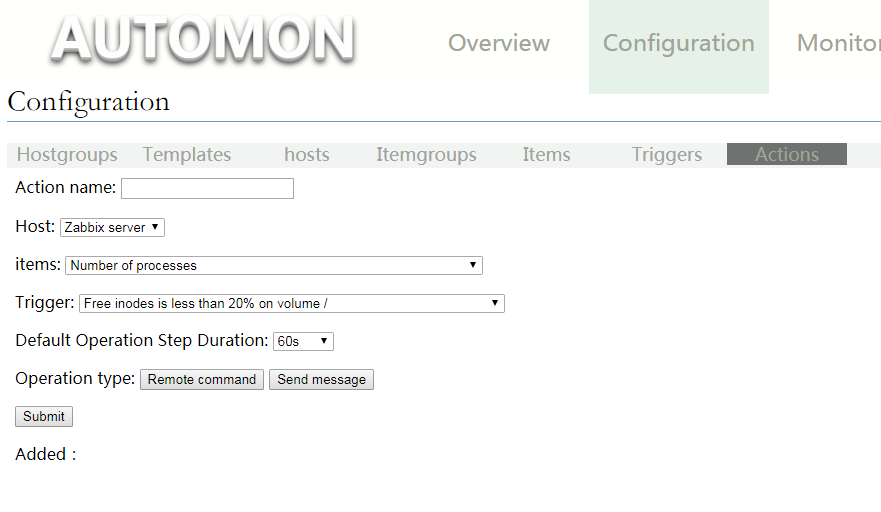
[5） 前端展示](#_Toc12445_WPSOffice_Level2) [8](#_Toc12445_WPSOffice_Level2)

**1.配置action**

点击Configuration-》action  
这里有三部分要填写，基础信息，命令操作，邮件操作

1. 基础信息

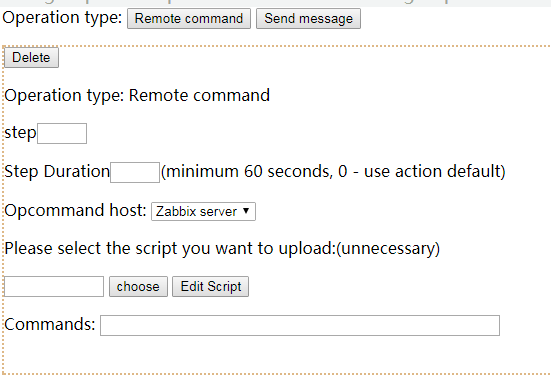
根据提示填写action名字以及对应的host、item、trigger



1. 命令操作

点击Remote command

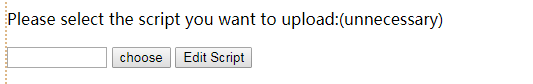
这里的step为action操作的步骤序号，填写1即为第一步，2为第2步，每一步操作间隔时间由Step Duration确定。



Opcommand host：命令执行在哪台计算机。

Commands:需要操作的命令。

当需要操作的命令复杂时需要执行脚本，本功能提供两种方式去选择脚本.第一种直接上传，第二种在线编辑。



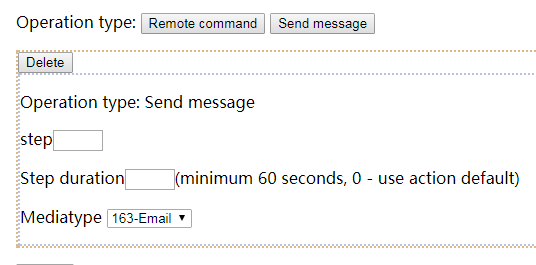
脚本上传或编写完成后Commands将自动填写上操作命令取执行脚本。如下例：名为http的action的第一步操作命令。



需要第2步、第3步。继续添加即可。

1. 邮件操作

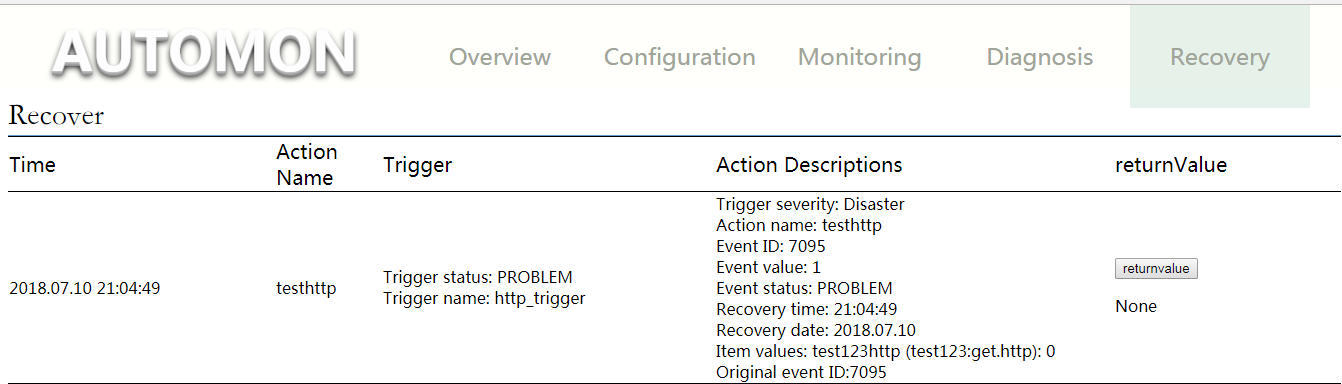
点击Send message。



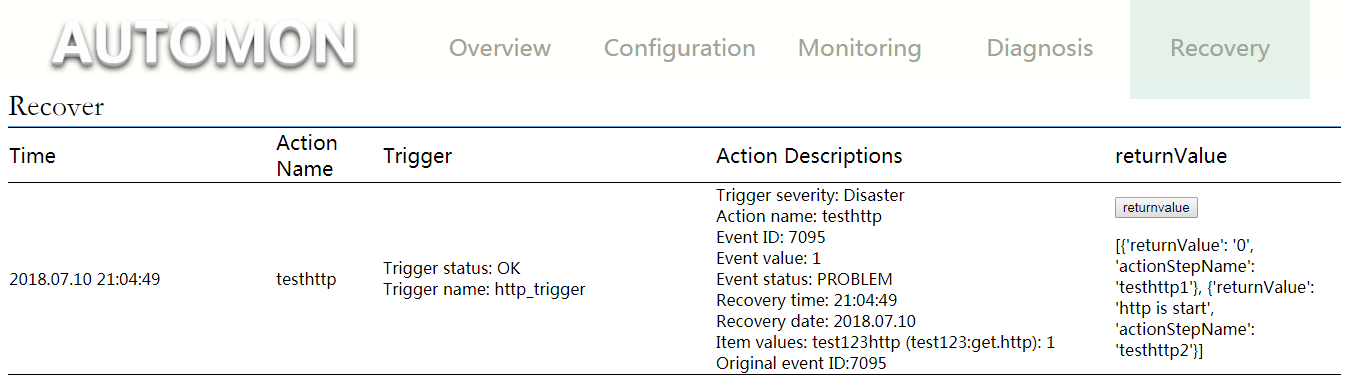
Mediatype：将邮件发送至何处，下拉菜单的为配置好的邮箱。

4）Recover功能

当异常出现，触发action，Recovery会出现如下的action信息，trigger的状态是problem。如下图所示：



处理完成后trigger状态变为ok，此时点击returnvalue将会把返回值信息展示出来。如下图所示：



1. **实现原理**

1）处理流程

当trigger达到阈值触发action，action按step步骤依次进行执行。

下面以执行脚本的action，testhttp为例做介绍：

当添加完2步action后CMC将在本机创建4个脚本



testhttp1.sh,testhttp1.py为第1步脚本，testhttp2.sh,

testhttp2.py为第2步脚本。每一步有两个脚本。

然后，后台通过配置好的免密scp将这4个脚本发送至远程被监控的主机上。

testhttp1.sh为用户上传的脚本，本系统对其进行改名保存至主机。

testhttp1.py为testhttp1.sh的管理脚本，action直接执行的是testhttp1.py。以此脚本去执行testhttp1.sh并获得脚本返回值，将返回值保存至远程机的etc/zabbix/return.yaml。然后将return.yaml以netcat命令通过 8899端口发送回主机，而主机打开了对8899端口的监控，当主机接收后保存至主机。然后CMC的recover将会对其进行处理展示至前端。

处理流程如下图所示：

主机 远程机

执行

执行

Testhttp1.sh

Testhttp1.py

返回

Action的step1

取得返回值存入return.yaml

追加至本机的return.yaml

执行

执行

Testhttp2.sh

Action的step2

Testhttp2.py

读取return.yaml根据第一步的返回值执行第二步sh脚本

返回

追加至本机的return.yaml

取得返回值存入return.yaml

CM读取return处理展示至前端

1. 脚本创建

action的每一步将会有两个脚本，第一个为用户上传或者在线编辑，第二个为CMC在前端生成。前端生成根据模板以及用户填写的数据，以数据去替换模板中的特殊字符，生成该action的脚本。并对脚本的命名规范化。

第一步脚本模板：

A为action步骤名（testhttp1），E为主机ip地址（前端自动获取）

#name: A.py  
#url: /etc/zabbix/script/  
import commands  
import yaml  
import os  
output= commands.getoutput('sh /etc/zabbix/script/A.sh')  
fr = open('/etc/zabbix/script/return.yaml',"a+")  
data ={'A':{'step':'1','value':output}}  
print data  
yaml.dump(data,fr)  
fr.close()  
os.system("nc E 8899 < /etc/zabbix/script/return.yaml")

第二步及以后脚本模板：

A为action步骤名（testhttp2），DC为上一步步骤名（testhttp1），B为步骤号（2）

#name: A.py  
#url: /etc/zabbix/script/  
import yaml  
import commands  
import os  
fr = open('/etc/zabbix/script/return.yaml',"r")  
data= yaml.load(fr)  
data1=data['DC']  
fr.close()  
data\_value=data1['value']  
output=commands.getoutput('sh/etc/zabbix/script/A.sh %s' %(

data\_value))  
data={'A':{'step':'B','value':output}}  
print data  
fr1 = open('/etc/zabbix/script/return.yaml',"a+")  
yaml.dump(data,fr1)  
fr1.close()  
os.system("nc E 8899 < /etc/zabbix/script/return.yaml")

1. 返回值保存

在action脚本中每次都将返回值追加保存至return.yaml。格式如下图：



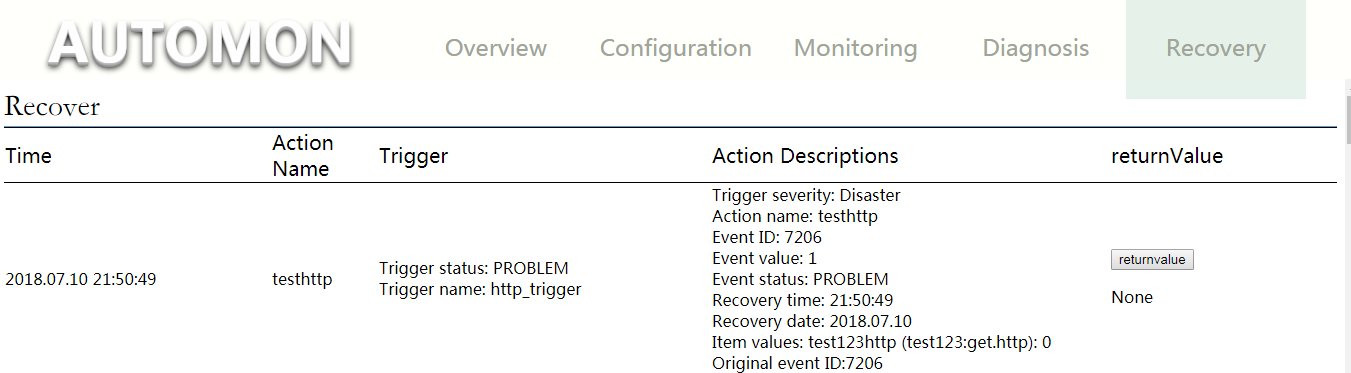
1. 远程机与主机交互

主机通过免密的scp将创建好的脚本发送至远程机。

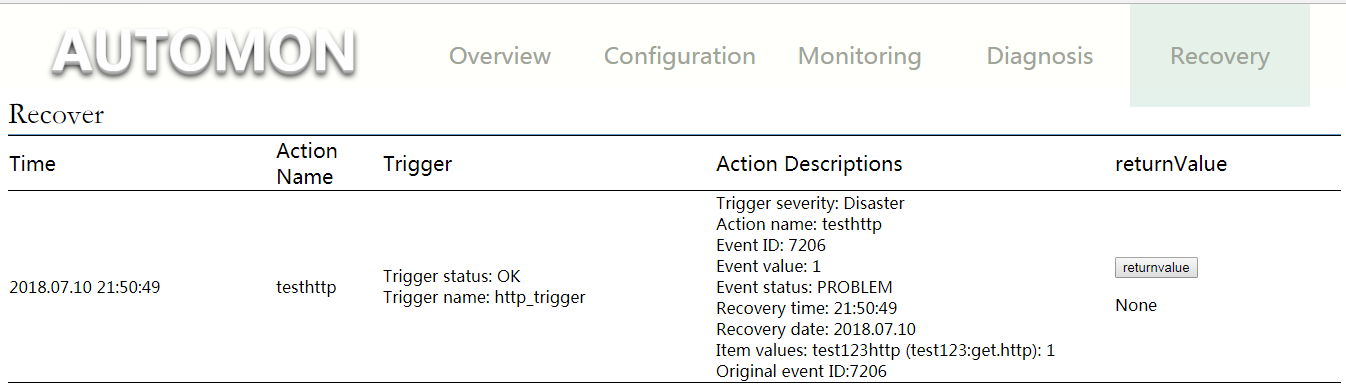
远程机通过netcat命令将返回值（return.yaml）发送回主机。

1. 前端展示

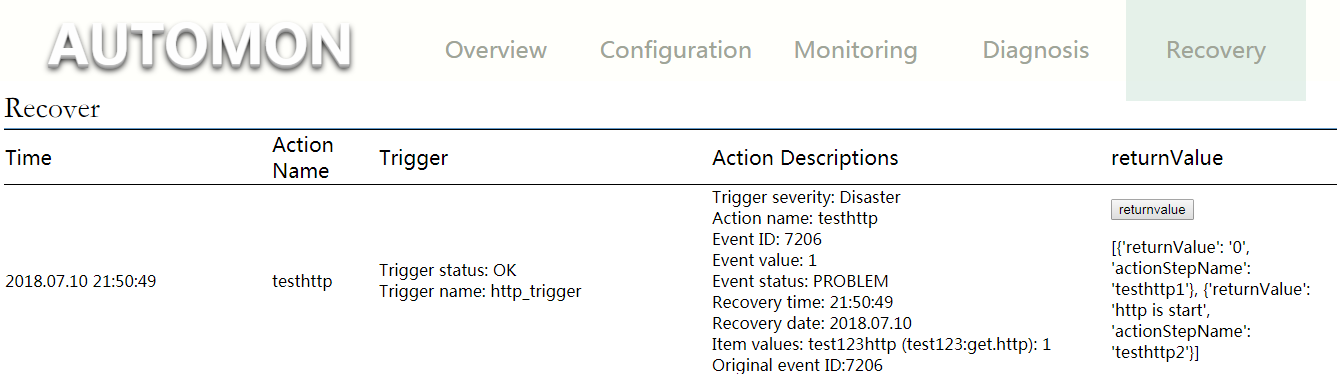
当达到trigger阈值触发action时，将会将此时的action，item，trigger各个相关信息保存至数据库。然后展示在Recover下，如图所示：



当故障修复即trigger状态为ok。



此时点击returnvalue将会匹配上整个处理过程的返回值在returnValue栏中。



匹配原理：在触发action时，第一次存相关数据至数据库，此条记录的returnValue为空。当故障处理CMC会再次发出一条trigger状态为ok的相关数据，此时根据事件的id将第一次保存的数据replace掉。此时点击returnvalue将根据action名搜索数据库最新的记录中哪数据的returnValue为空，然后将返回值保存到此条记录中。然后前端刷新获取。