Alert动作扩展插件开发手册1.0

> 注：R10版本暂不支持前端页面注册

动作插件可以让开发人员根据特定业务进行定制动作以满足不同客户的不同需求。

扩展说明

注册插件需要实现以下几个接口和类：

```
uyun.eagle.actionplugin.ActionRegister;
uyun.eagle.actionplugin.ActionHandler;
uyun.eagle.actionplugin.ActionParam;
```

ActionRegister

ActionRegister是动作的注册接口，用于提供动作信息，开发人员必须实现该接口的各个方法。

详细说明请阅读该接口的注释：

```java
/**
 * 自定义动作注册接口
 */
public interface ActionRegister {
    /**
     * 获取该动作的唯一标识(不能为空，固定值，每次调用返回同一UUID)
     *
     * @return 唯一标识字符
     */
    String getUUID();
    /**
     * 获取动作显示名称(不能为空)
     *
     * @return 动作显示名称
     */
    String getActionName();
    /**
     * 获取动作版本号
     *
     * @return 动作版本号
     */
    String getActionVersion();
    /**
     * 获取动作自定义参数的页面组件（页面组件按list的顺序显示)
     *
     * @return 页面组件
     */
    List<UIComponent> getUIComponent();
    /**
     * 获取自定义的动作(不能为空)
     *
     * @return 自定义的动作
     */
    ActionHandler getActionHandler();
    /**
     * 获取继承{@link ActionParam ActionParam}的子类类型
     *
     * @return 类类型
     */
    Class getParamClazz();
}
```

ActionHandler

ActionHandler是自定义动作处理接口，开发人员必须实现该接口。具体的业务逻辑就是在ActionHandler的handle方法中实现。

```java
/**
 * 自定义动作处理接口
 */
public interface ActionHandler {
    /**
     * 自定义处理告警
     *
     * @param incidentData 告警数据（对数据本身做处理）
     * @param actionParam  页面填写的动作自定义参数，只有ActionRegister#
     *                     getUIComponent() 不为null，并且页面
     *                     填写值之后，该参数才有值（非必须使用）
     * @param logger       动作记录器，记录自定义备注（非必须使用）
     * @return 动作执行结果，true则保存告警，反之则删除告警
     */
    ActionResult handle(final IncidentData incidentData,
                        final ActionParam actionParam,
                        final ActionLogger logger);
}
```

ActionParam

ActionParam是动作自定义参数的父类。如果扩展动作需要用户在规则页面填值，然后让扩展动作在运行时获取用户填写的值，则需继承ActionParam并且实现ActionRegister#getUIComponent。

注：ActionParam的成员变量只支持简单的数据类型；成员变量的名称必须和UIComponent的key一样，才能被赋值。

```java
/**
 * 动作自定义参数的父类，自定义参数需继承该类
 */
public abstract class ActionParam {
}
```

扩展示例

新建一个maven项目

添加依赖

在pom.xml添加如下依赖

```xml
<dependency>
   <groupId>uyun.eagle</groupId>
   <artifactId>action-plugin</artifactId>
   <version>1.0.0-SNAPSHOT</version>
   <scope>provided</scope>
</dependency>
```

Maven项目结构

```
src
 |-main
    |-java
        |-com
            |-xxx
                |-XxxHandler.java (实现ActionHandler接口)
                |-XxxParam.java (继承ActionParam)
                |-XxxRegister.java (实现ActionRegister接口)
    |-resources
        |-META-INF
            |-services
                |-uyun.eagle.actionplugin.ActionRegister
                (纯文本文件，内容为：com.xxx.XxxRegister)
```

XxxRegister.java

```java
public class XxxRegister implements ActionRegister {

    @Override
    public String getUUID() {
        return "593fa93d2d0e9b3e43ef886d";
    }
    @Override
    public String getActionName() {
        return "XxxHandler";
    }
    @Override
    public String getActionVersion() {
        return "1.0";
    }
    @Override
    public List<UIComponent> getUIComponent() {
        List<UIComponent> list = new ArrayList<>();
        //regex和XxxParam成员变量名一样
        list.add(new UIComponent("regex", "正则表达式",
                UIComponentEnum.SINGLE_LINE_TEXT.getCode()));
        return list;
    }
    @Override
    public ActionHandler getActionHandler() {
        return new XxxHandler();
    }
    @Override
    public Class getParamClazz() {
        return XxxParam.class;
    }
}
```

XxxParam.java

```java
public class XxxParam extends ActionParam {
    private String regex;
    public String getRegex() {
        return regex;
    }
    public void setRegex(String regex) {
        this.regex = regex;
    }
}
```

XxxHandler.java

```java
public class XxxHandler implements ActionHandler {
    @Override
    public ActionResult handle(IncidentData incidentData,
                               ActionParam actionParam,
                               ActionLogger actionLogger) {
        RegexActionParam param = (RegexActionParam) actionParam;
        //注意判空
        if (param == null) {
            return new ActionResult(true);
        }
        String regex = param.getRegex();
        //注意判空
        if (regex == null || incidentData.getEntityAddr() == null) {
            return new ActionResult(true);
        }
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(incidentData
                                          .getEntityAddr());
        StringBuilder newValue = new StringBuilder();
        while (matcher.find()) {
            newValue.append(matcher.group());
        }
        Property property = new Property("newField", "新增字段",
                newValue.toString());
        //新增字段添加到properties中
        if (incidentData.getProperties() == null) {
            List<Property> list = new ArrayList<>();
            list.add(property);
            incidentData.setProperties(list);
        } else {
            incidentData.getProperties().add(property);
        }
        //日志不是必须的，可以根据具体业务自行添加
        actionLogger.log("添加扩展字段：newField");
        return new ActionResult(true);
    }
}
```

META-INF/services/uyun.eagle.actionplugin.ActionRegister

```
com.xxx.XxxRegister
```

该示例展示的是根据自定义的正则表达式从已有字段获取新值，然后添加到properties中。逻辑很简单，只是用于展示功能。**在具体业务逻辑编写时须注意判空**。

Alert加载扩展插件

上述工程完成以后，通过maven打包，打包之后的jar包放到Alert所在主机的目录下

```
/opt/uyun/alert/datacenter/lib/
```

注：Alert集群部署则需要在所有的主机上加上扩展插件。

Alert项目启动以后，可以查看datacenter的日志,日志所在路径为：

```
/opt/uyun/alert/datacenter/logs/
```

看到以下日志：

```
************* LOAD ACTIONS START ***********
load XxxHandler 1.0 success
*************** LOAD ACTIONS END **********
```

提示load XxxHandler 1.0 success，则代表动作扩展插件加载成功。加载失败会提示错误原因，若没有加载信息，请检查插件项目。

Alert加载动作插件以后，需验证插件的功能，如有异常可以通过`perform action plugin error`关键字在eagle-datacenter.log中查找。

其他类说明

IncidentData.java

```
public class IncidentData {
    /**
     * 故障id(不可修改)
     */
    private String id;
    /**
     * 租户id(不可修改)
     */
    private String tenant;
    /**
     * 故障名称
     */
    private String name;
    /**
     * 故障显示名
     */
    private String alias;
    /**
     * 合并故障的父故障id(不可修改)
     */
    private String parentId = null;
    /**
```

```
     * 子故障数量(不可修改)
     */
    private int subNum = 0;
    /**
     * 最后一次发生故障时的严重性等级
     * 恢复 - 0, 提醒 - 1, 警告 - 2, 紧急 - 3
     */
    private int severity;
    /**
     * 来源系统(不可修改)
     */
    private String source;
    /**
     * 故障状态
     * 新告警-0,处理中-150,已解决-190,已关闭-255
     */
    private int status;
    /**
     * 故障发生源名称
     */
    private String entityName;
    /**
     * 故障发生源地址,一般情况下为IP地址
     */
    private String entityAddr;
    /**
     * 故障发生次数(不可修改)
     */
    private int count;
    /**
     * 故障首次发生时间
     */
    private Date firstOccurTime;
    /**
     * 故障最后一次发生时间
     */
    private Date lastOccurTime;
    /**
     * 故障关闭时间
     */
    private Date closeTime;
    /**
     * 故障描述
     */
    private String description;
    /**
     * 故障所关联的工单ID
     */
    private String orderId;
```

```
72      /**
73       * 故障所关联的工单流水号
74       */
75      private String orderFlowNum;
76      /**
77       * 故障解决信息
78       */
79      private String resolveMessage;
80      /**
81       * 故障关闭信息
82       */
83      private String closeMessage;
84      /**
85       * 是否通知
86       */
87      private boolean isNotify;
88      /**
89       * 对应统一资源库的资源ID
90       */
91      private String resObjectId;
92      /**
93       * 对应store的资源类型
94       */
95      private String classCode;
96      /**
97       * 故障扩展属性
98       */
99      private List<Property> properties;
100     /**
101      * 故障关联的标签
102      */
103     private List<Property> tags;
104     /**
105      * 接入应用key(不可修改)
106      */
107     private String appKey;
108     /**
109      * 责任人
110      */
111     private String owner;
112 }
```

IncidentData包含alert数据对外可见的字段，其中注释为不可修改的 id，tenant，parentId，subNum，source，count，appKey字段是不能更改的，即使更改了也不会被保存。

注：如果将某个字段置为null，程序也不会对该字段进行保存，如果要去除这个字段的值，可以将这个字段设置为一个空对象

ActionLogger.java

```java
/**
 * 动作记录器
 */
public interface ActionLogger {
    /**
     * 记录自定义备注（可以在故障详情页的审计日志处查看记录的内容，非错误日志log）
     *
     * @param str 自定义备注
     */
    void log(String str);
}
```

ActionResult.java

```java
/**
 * 执行动作后返回的结果
 */
public class ActionResult {
    /**
     * 是否被保存
     */
    private boolean isSaved;
    // ...
}
```

目前只有isSaved字段表示告警数据是否需要删除，用对象返回方便以后扩展。

Property.java

```java
public class Property {
    /**
     * 键
     */
    private String key;
    /**
     * 键所对应的名称
     */
    private String keyName;
    /**
     * 值
     */
    private String value;
    // ...
}
```

UIComponent.java

```java
public class UIComponent {
    /**
     * 键
     */
    private String key;
    /**
     * 键所对应的名称
     */
    private String keyName;
    /**
     * ui组件类型
     */
    private int type;
    /**
     * 默认值，可以为null
     */
    private String defaultValue;
    /**
     * 下拉框可选参数（多选下拉框返回数组）
     */
    private List<Property> params;
    /**
     * 是否必填
     */
    private boolean isRequired = false;
    // ...
}
```

UIComponentEnum.java

```java
public enum UIComponentEnum {
    SINGLE_LINE_TEXT(1, "单行文本框"),
    MULTI_LINE_TEXT(2, "多行文本框"),
    SINGLE_SELECT(3, "单选下拉框"),
    MULTI_SELECT(4, "多选下拉框");
}
```

目前展示只支持上述几种ui组件。