**PRIMETON TECHNOLOGIES, LTD.**

**上海普元信息技术有限责任公司**

**CEP Express**

**总体设计说明书**

No part of this document may be reproduced, stored in any electronic retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording, otherwise, without the written permission of the copyright owner.



**COPYRIGHT 2003 by Primeton Technologies, Ltd. ALL RIGHTS RESERVED.**

文档OARP

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 分类 | 姓名 | 范围要求 |
| 1 | Owner |  |  |
| 2 | 作者 |  |  |
| 3 | 审核人 |  |  |
| 4 | 审核人 |  |  |
| 5 | 审核人 |  |  |
| 6 | 批准人 |  |  |

文档修订记录

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 版本号 | 修订日期 | 修订概述 | 修订人 | 审核人 | 批准人 | 备注 |
| 1 | 1 | 2013/7/12 | 细化流程，增加部署包结构，规则库结构 | 罗南钦 |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**目 录**

[1 引言 5](#_Toc361387264)

[1.1 目的 5](#_Toc361387265)

[1.2 范围 5](#_Toc361387266)

[1.3 文档约定 5](#_Toc361387267)

[1.4 阅读对象 5](#_Toc361387268)

[1.5 参考资料 5](#_Toc361387269)

[1.6 术语和缩略语定义 5](#_Toc361387270)

[2 总体设计约束 6](#_Toc361387271)

[2.1 系统运行环境 6](#_Toc361387272)

[2.2 第三方组件依赖 6](#_Toc361387273)

[2.3 系统技术规范 6](#_Toc361387274)

[3 总体设计图 7](#_Toc361387275)

[3.1 开发期 7](#_Toc361387276)

[3.2 部署期 8](#_Toc361387277)

[3.2.1 规则模板注册 8](#_Toc361387278)

[3.2.2 生成规则实例 8](#_Toc361387279)

[3.2.3 加载规则实例 9](#_Toc361387280)

[3.3 运行期 9](#_Toc361387281)

[4 开发期设计 10](#_Toc361387282)

[4.1 事件源定义 10](#_Toc361387283)

[4.1.1 开发界面 10](#_Toc361387289)

[4.1.2 格式定义 10](#_Toc361387290)

[4.1.3 持久化 11](#_Toc361387291)

[4.2 规则定义 11](#_Toc361387292)

[4.2.1 开发界面 11](#_Toc361387294)

[4.2.2 格式定义 12](#_Toc361387295)

[4.2.3 持久化 13](#_Toc361387296)

[4.3 Action定义 14](#_Toc361387297)

[4.3.1 开发界面 14](#_Toc361387299)

[4.3.2 格式定义 14](#_Toc361387300)

[4.4 EPS Listener开发 15](#_Toc361387301)

[4.4.1 开发界面 15](#_Toc361387302)

[4.4.2 接口实现 15](#_Toc361387303)

[4.5 自定义扩展开发 16](#_Toc361387304)

[4.5.1 开发界面 16](#_Toc361387305)

[4.5.2 接口实现 16](#_Toc361387306)

[4.6 规则模板持久化 16](#_Toc361387307)

[5 部署期设计 17](#_Toc361387308)

[5.1 新增业务规则实例 17](#_Toc361387309)

[5.2 修改业务规则实例 17](#_Toc361387310)

[5.3 事件源部署 18](#_Toc361387311)

[5.3.1 事件源反序列化 18](#_Toc361387312)

[5.3.2 事件源加载 18](#_Toc361387313)

[5.4 规则部署 19](#_Toc361387314)

[5.4.1 规则反序列化 19](#_Toc361387315)

[5.4.2 规则加载 19](#_Toc361387316)

[5.5 Action部署 19](#_Toc361387317)

[5.6 EPS Listener部署 19](#_Toc361387318)

[5.7 自定义扩展部署 19](#_Toc361387319)

[5.8 规则实例持久化 20](#_Toc361387320)

[6 系统进程视图 20](#_Toc361387321)

[7 系统物理视图 21](#_Toc361387322)

[7.1 web控制台结构 21](#_Toc361387323)

[7.2 分析引擎集群结构 21](#_Toc361387324)

[7.3 Zookeeper结构 22](#_Toc361387325)

[7.4 部署包结构 22](#_Toc361387326)

[7.5 规则库结构 23](#_Toc361387327)

[8 系统开发视图 23](#_Toc361387328)

[8.1 接口定义 23](#_Toc361387329)

[8.2 开发工具 23](#_Toc361387330)

[8.3 开发方式 23](#_Toc361387331)

[8.4 设计规范 24](#_Toc361387332)

[8.5 开发规范 24](#_Toc361387333)

[9 其他非功能约束 24](#_Toc361387334)

[*9.1* *日志非功能约束* 24](#_Toc361387335)

[*9.2* *异常非功能约束* 24](#_Toc361387336)

[*9.3* *兼容性非功能约束* 24](#_Toc361387337)

[*9.4* *国际化非功能约束* 24](#_Toc361387338)

[10 FAQ 25](#_Toc361387339)

[10.1 FAQ1 25](#_Toc361387340)

[11 TBD 25](#_Toc361387341)

[11.1 TBD 25](#_Toc361387342)

# 引言

## 目的

*[提出了对软件总体设计的纵览，对此文档的编写目的，这有助于读者理解文档如何编写并且如何阅读和解释。]*

## 范围

*[提供了对指定的软件总体设计要阐述的内容，和其他文档的关系描述。]*

## 文档约定

*[描述编写文档时所采用的标准或排版约定，包括正文风格、提示区或重要符号。]*

## 阅读对象

*[列举了该文档的不同读者，例如开发人员、项目经理、营销人员、用户、测试人员或文档的编写人员。描述了文档中剩余部分的内容及其组织结构。提出了最适合于每一类型读者阅读文档的建议。]*

## 参考资料

*[列举了编写软件需求规格说明时所参考的资料或其它资源。这可能包括用户界面风格指导、合同、标准、系统需求规格说明、使用实例文档，或相关产品的软件需求规格说明。*

*提示：列出本文档的所有参考文献（可以是非正式出版物），格式如下：*

*资料名称，出版单位（或归属单位）或作者，版本号]*

## 术语和缩略语定义

*[定义所有必要的术语，以便读者可以正确地解释软件总体设计，包括词头和缩写。你可能希望为整个公司创建一张跨越多项项目的词汇表，并且只包括特定于单一项目的软件需求规格说明中的术语。]*

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 描述 |
|  |  |  |
|  |  |  |

# 总体设计约束

## 系统运行环境

*[描述最终系统支持的操作系统，从服务器端和客户端两个角度，明确说明操作系统的版本，语言，SP版本号，浏览器版本等*

*描述系统支持的数据库版本*

*描述系统支持的应用服务器版本*

*描述系统支持的JDK版本等]*

## 第三方组件依赖

*[描述如何使用第三资源的策略，比如职能使用License为GPL类型的开源组件，只能使用公司购买License的文档制作软件等]*

*EOS 6.3 主要依赖以下第三方组件或产品，对于其他依赖，每个子系统可以自行定义，但是必须遵循以下原则：*

* 1. *必须具有可商用的 License，尽量使用Apache的License；*
  2. *必须使用成熟的商业或开源产品*

*在EOS6.3中确定要引用的第三方组件是：*

1. *Spring -2.0-rc3*

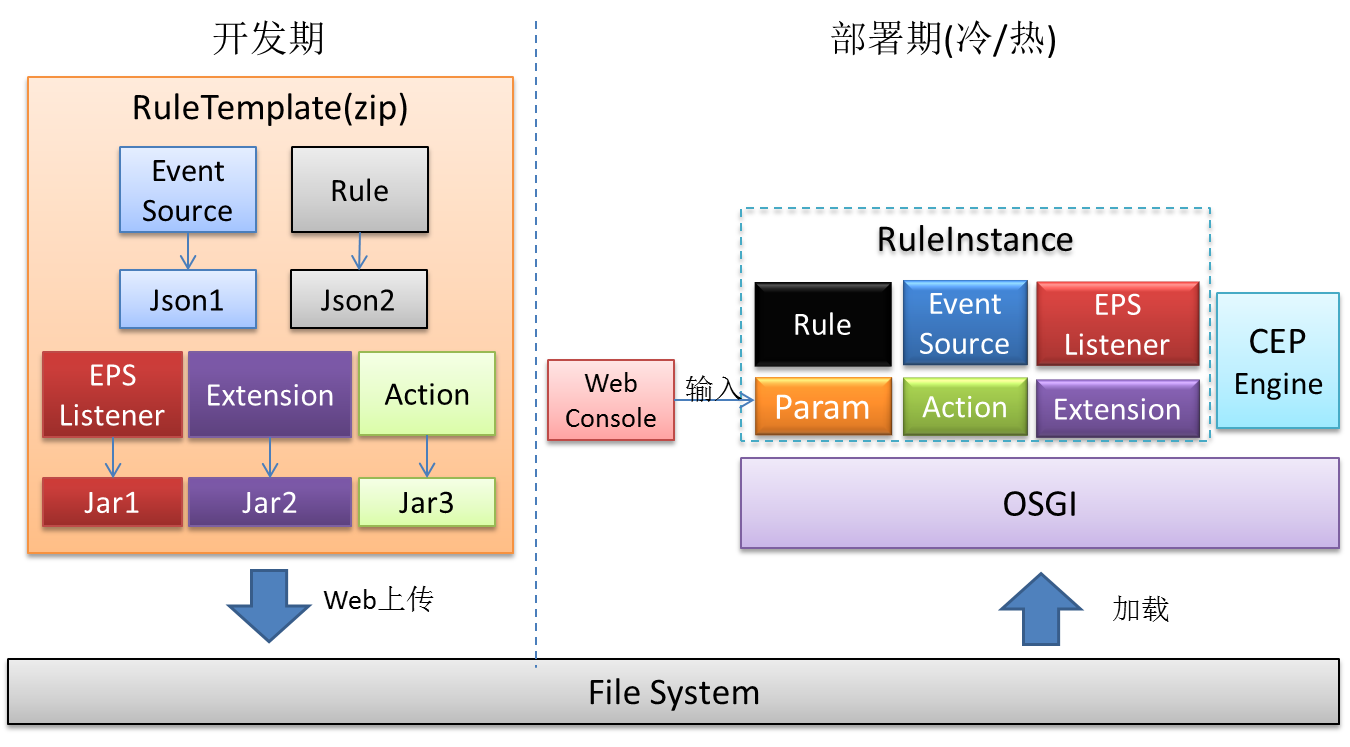
*因Spring 2.5 版本与EOS6.1中使用的Spring -2.0的接口不兼容，因此在EOS6.3中仍然使用Spring -2.0-rc。*

## 系统技术规范

*[描述系统的实现机制，如果是传统的应用软件，要求软件各个层次的交互关系，比如数据访问层、业务逻辑层、展现层的特性，并且要描述层与层之间的交互关系*

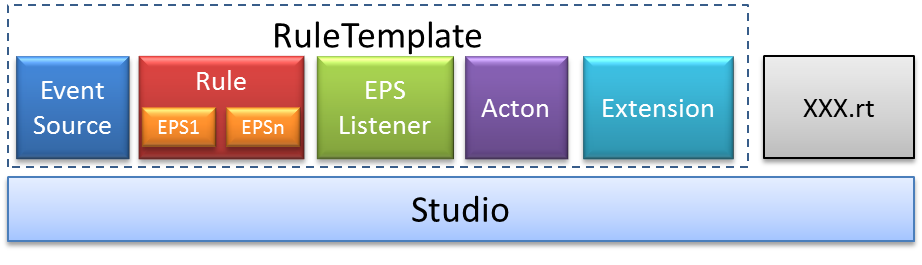
*对于平台型软件，要说明系统实现了那些规范，是部分实现还是完整实现等。]*

# 总体设计图



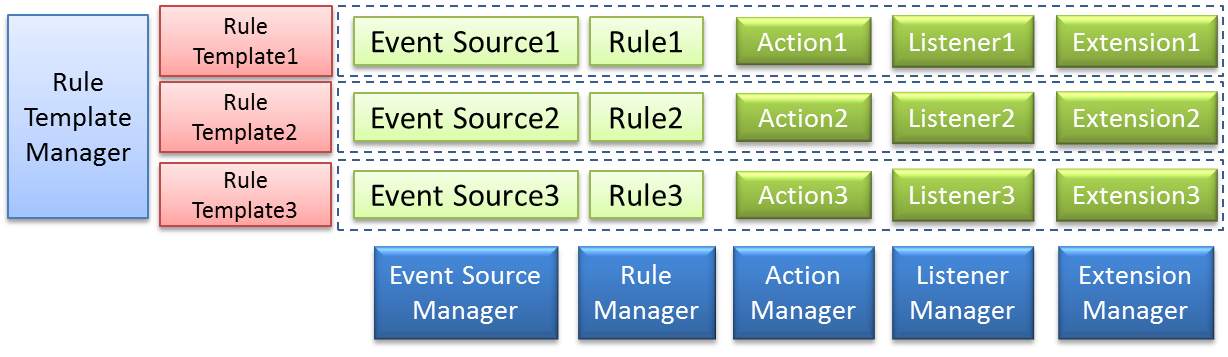
此图面向开发人员，包括事件源、Rule、EPS Listener、Action、自定义扩展的开发及部署。

## 开发期



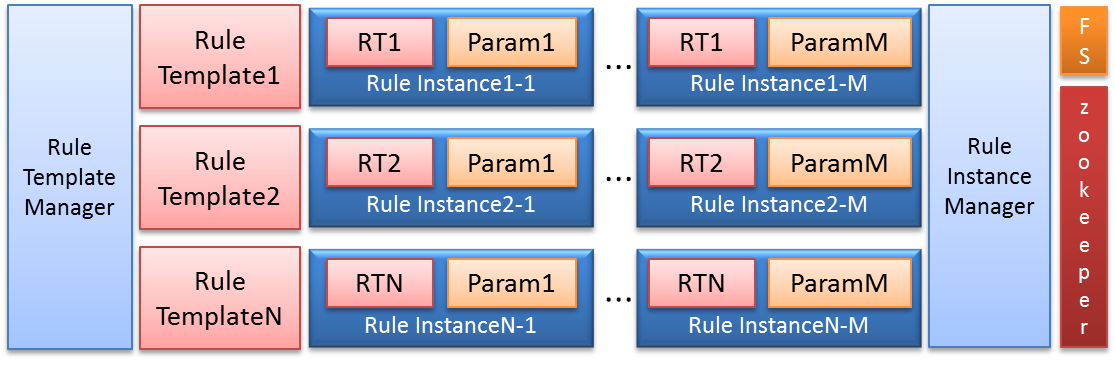
## 部署期

### 规则模板注册



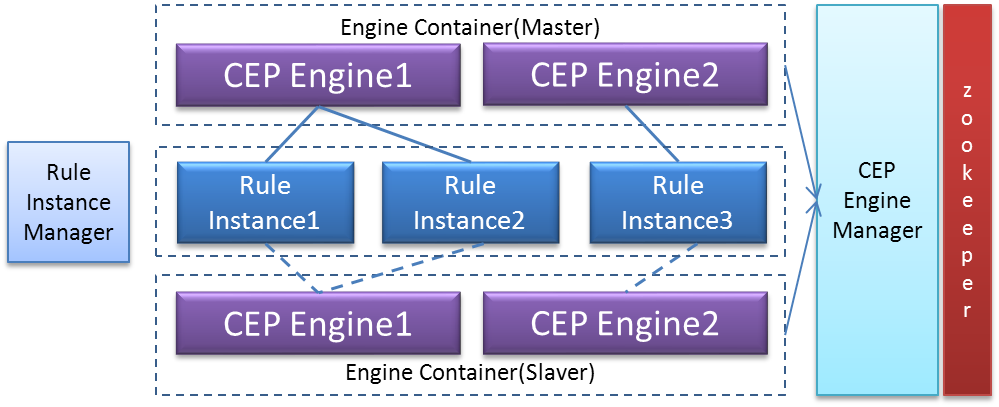
* 事件源，EPS从文件系统中反序列化后注册到各自的Manager中。Action，EPS Listener和自定义扩展Jar会作为插件部署到OSGI平台中成为bundle，由OSGI提供Action，EPS Listener和自定义扩展服务，并注册到各自的Manager中。
* RuleTemplate由事件源，Rule，Action，Listener，Extension组成，多个RuleTemplate由RuleTemplateManager管理

### 生成规则实例



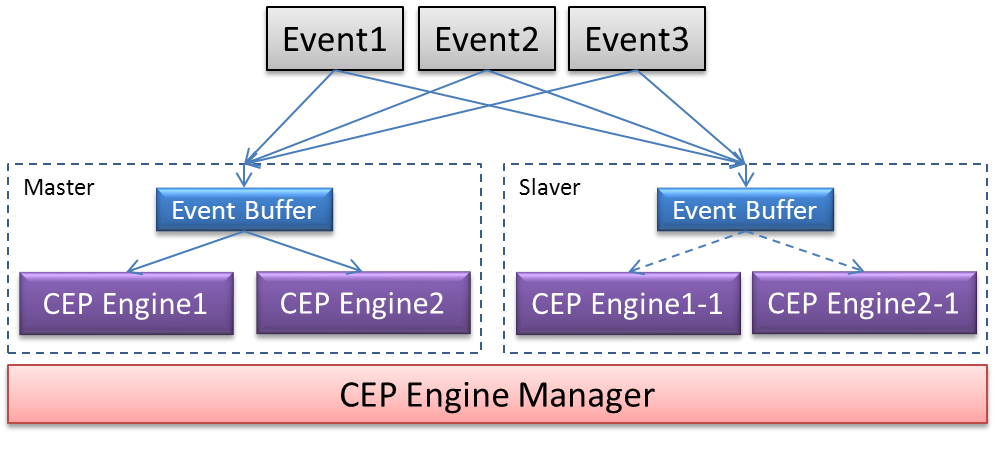
* 每个RuleTemplate加上不同的参数，可组成不同的Rule Instance，多个Rule Instance由RuleInstanceManager管理。运行时的Rule Instance都会注册到zk上，并且持久化到文件系统中。

### 加载规则实例



* 不同的Rule Instance部署到多个CEP Engine中。CEP Engine Manager管理多个CEP Engine容器。CEP Engine容器中含两个Engine。由Zookeeper监控每个Engine的运行情况。

## 运行期



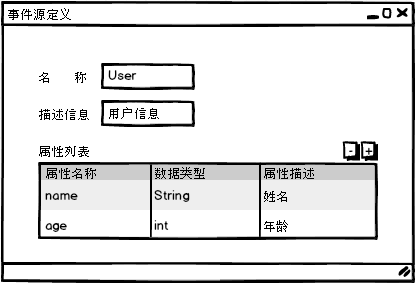
* Master和Slaver都接收事件的输入，首先进入EventBuffer，其大小可配置。
* Master 中的Engine从buffer中获取事件并进行处理，这里并没有事件的路由，所以所有事件都进入两个Engine。
* Slaver的EventBuffer同样接收输入的事件，但是Engine并不处理buffer中的事件，所以当然buffer装满后，抛弃掉最先进入的事件来空出空间接收新的事件。
* 当Master停止并运行Slaver时，Engine会从buffer中取出旧事件进行处理，所以理论上会重复处理Master处理过的事件。

# 开发期设计

## 事件源定义



### 开发界面

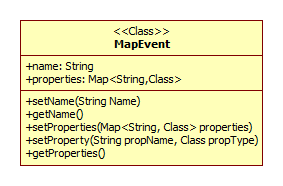


开发人员可添加多个属性，并选择数据类型。除了Java自带的数据类型，还可以选择已经定义好的其他数据源

### 格式定义

根据需求，规定CEP Engine接收事件源为Map格式。其Map的key为事件属性名称，value为属性的数据类型。如Map：<name, String.class>，<age, int.class>等

为了将事件源定义序列化，定义了类来保存事件源的定义。类设计如下：



开发人员可新建一个MapEvent对象，并设置事件的名称和属性名/类型对，即可完成事件源的定义。

### 持久化

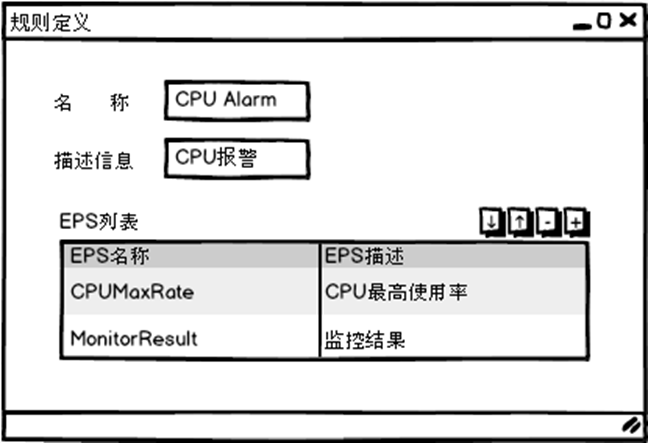


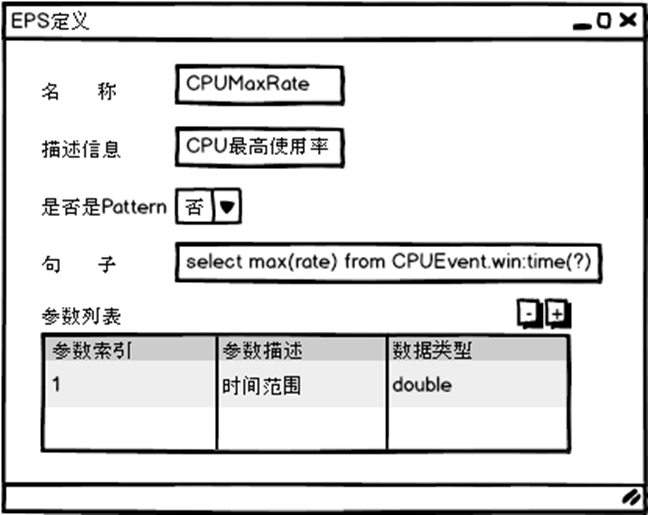
MapEvent对象通过JSON序列化工具序列化成JSON串，然后以事件源的名称作为文件名，event作为后缀保存该JSON串。即完成事件源的序列化及持久化。

## 规则定义



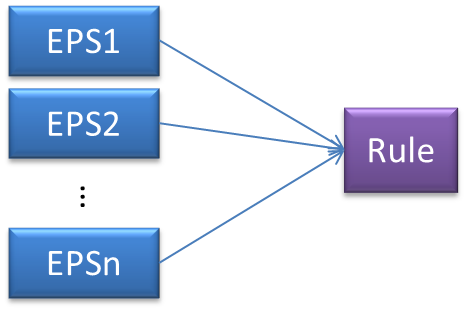
### 开发界面





规则开发包含多个EPS的开发。如上所示，添加新的EPS时需要填写相关信息

### 格式定义



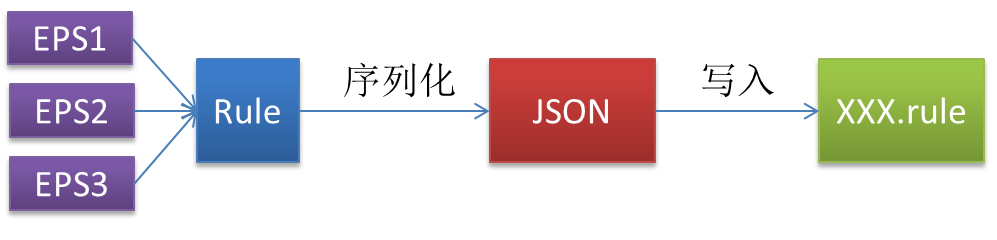
如上图所示，一个Rule由多个EPS组成

为了序列化规则，定义了EPS类和Rule类进行规则的保存。类设计如下：



* EPS实例需要保存名称和eps句子，paramMap保存了EPS中的参数索引和对应的参数数据类型，如果该eps无参数，则不需要往paramMap里写入。如果EPS是Pattern，则需要将pattern值设置为true。
* Rule实例要保存规则的名称及规则对应的所有EPS实例，其中epsMap的key为EPS实例的序列，因为当CEP Engine加载EPS的时候必须根据按照规定的顺序进行，否则会加载失败。

### 持久化

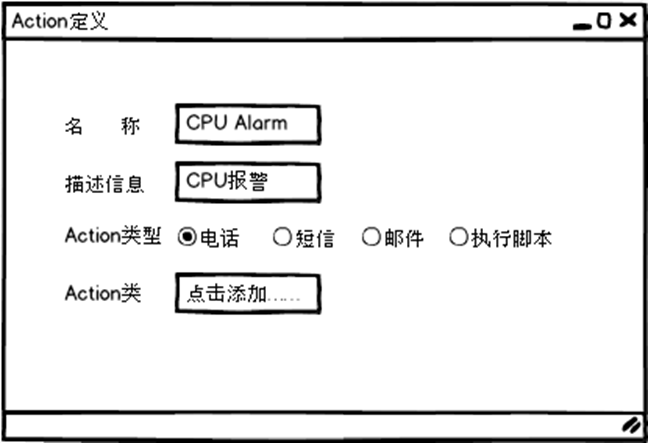


Rule对象保存了多个EPS对象，并通过JSON序列化工具序列化成JSON串，然后以规则名称作为文件名，rule作为后缀保存该JSON串。即完成规则的序列化及持久化。

## Action定义



### 开发界面



开发人员要实现Action接口，并且在添加Action类时要与Action类型一致

### 格式定义

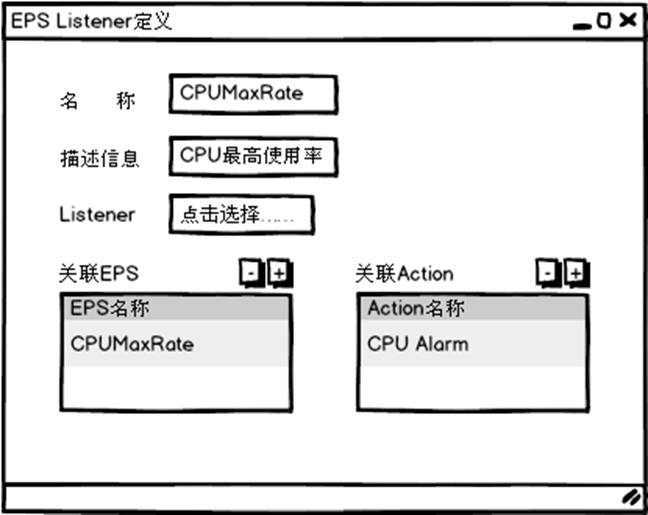
Action包含电话，短信，邮件，执行脚本四类。开发人员需要实现Action接口完成具体的Action操作。接口设计如下：



PHONE\_ACTION、EMAIL\_ACTION、MESSAGE\_ACTION、SCRIPT\_ACTION为四类Action的标识，getType方法必须返回这四类中的某一类。doAction方法内的参数为EPS Listener被触发时产生的newEvents和oldEvents。开发人员可根据此内容决定具体的操作。

## EPS Listener开发

### 开发界面



开发人员可以选择Listener关联的EPS以及Action

### 接口实现

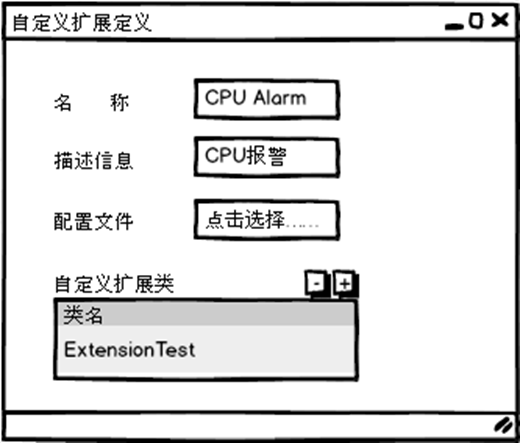
EPS Listener用于监听EPS的运行情况，开发人员需要实现特定的Listener接口。接口定义如下：



newEvents为新事件，oldEvents为过期事件。当EPS满足条件后，该Listener被触发，即可获得newEvents和oldEvents。然后开发人员可根据两类事件完成特定的操作。

## 自定义扩展开发

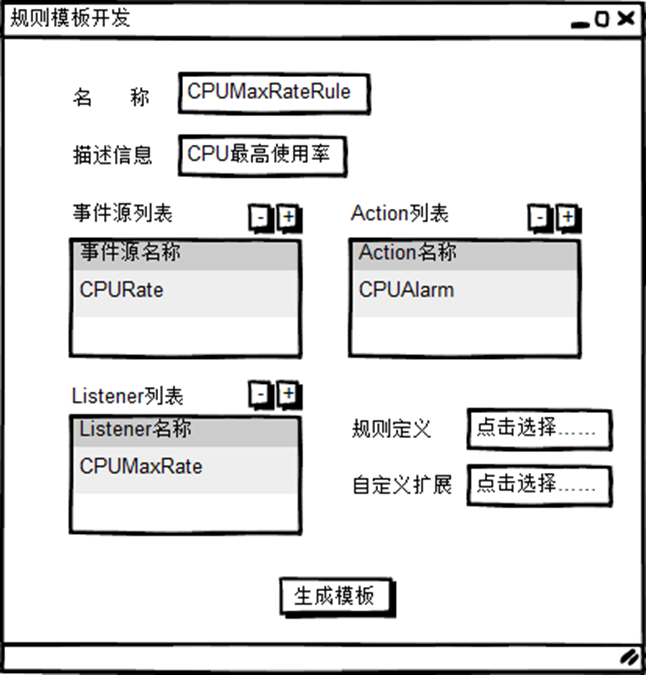
### 开发界面



### 接口实现

开发人员可根据需要自定义实现Esper中的view，具体的接口可参看Esper文档。

## 规则模板开发



## 规则模板持久化

规则模板保存了规则定义，事件源定义，Action，Listener，自定义扩展的名称。类图如下所示（省略了set/get方法）：



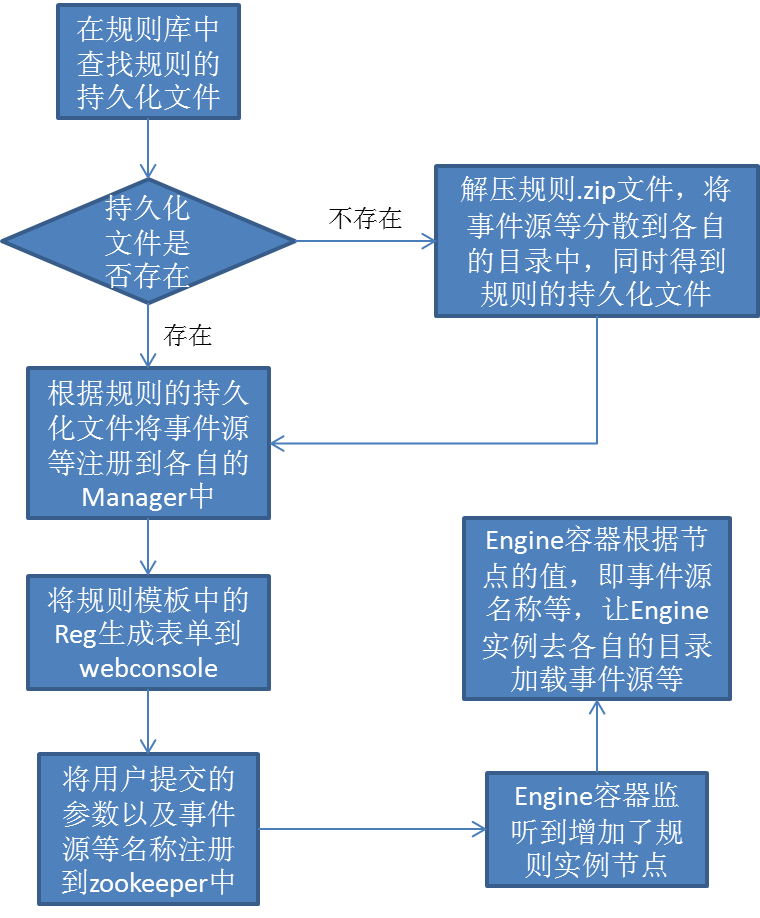
规则模板对象先由JSON工具序列化后，再持久化为xxx.rule文件。

# 部署期设计

## 新增业务规则实例

场景：用户选择规则模板🡪填写表单(规则参数)🡪提交

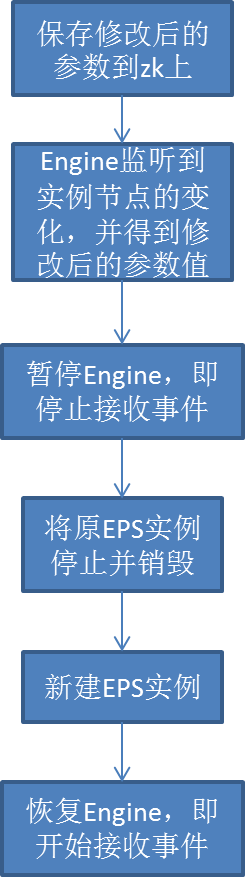
后台处理流程：



## 修改业务规则实例

场景：用户选择某个规则实例🡪修改规则参数🡪提交

后台流程：



## 事件源部署

### 事件源反序列化



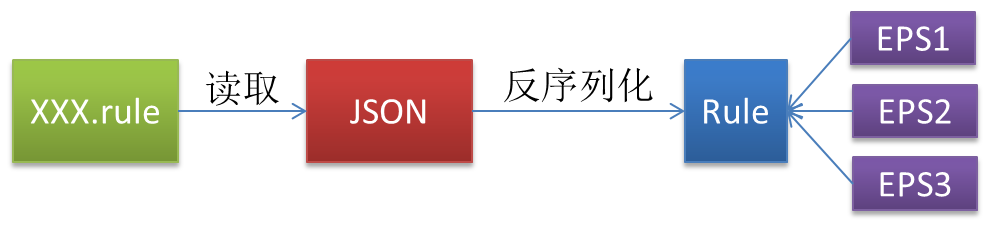
Manager从文件系统中读取JSON串，然后通过JSON工具反序列化为MapEvent对象。

### 事件源加载

Manager提供事件源加载接口，将反序列化得到的MapEvent对象加载到Manager中，供多个业务规则实例使用。

## 规则部署

### 规则反序列化



Manager从文件系统中读取JSON串，然后通过JSON工具反序列化为Rule对象，其中包含了n个EPS对象实例。

### 规则加载

Manager提供规则加载接口，将反序列化得到的Rule对象中的EPS对象加载到Manager中，供多个业务规则实例使用。

## Action部署

由于Action以Jar包形式进行了持久化，所以为了管理Jar，Manager内部将使用OSGI进行Jar包管理，对外提供管理接口，包括Jar的冷部署，热部署，为业务规则提供服务等。

## EPS Listener部署

由于Listener以Jar包形式进行了持久化，所以为了管理Jar，Manager内部将使用OSGI进行Jar包管理，对外提供管理接口，包括Jar的冷部署，热部署，为业务规则提供服务等。

## 自定义扩展部署

和EPS Listener一样，自定义扩展时以Jar包形式进行了持久化，所以为了管理Jar，Manager内部将使用OSGI进行Jar包管理，对外提供管理接口，包括Jar的冷部署，热部署，为业务规则提供服务等。

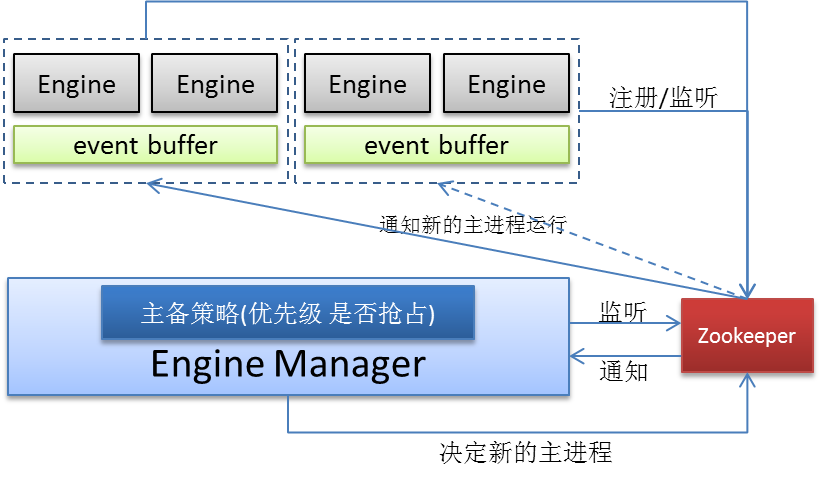
## 规则实例持久化

规则实例中保存了规则实例的名称，规则模板的名称，以及规则内包含的EPS所对应的业务参数列表。类图如下所示（省略了set/get方法）：



规则实例先由JSON工具序列化后，再持久化为xxx.ri文件。

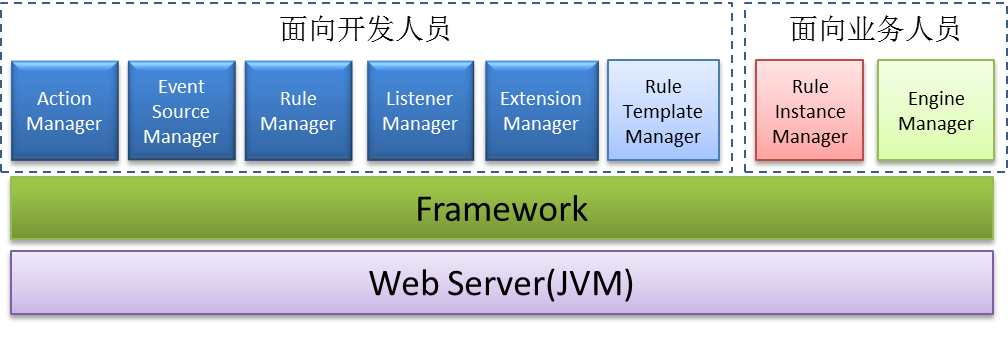
# 系统进程视图



* 如上图所示。每个Engine Manager作为一个独立进程，管理自己的Engine，并且将其注册到Zookeeper中。
* Engine Framework作为一个独立进程通过监听Zookeeper管理正在运行的Engine Manager。并且提供主备策略（进程优先级 或者 进程是否可抢占），当收到Zookeeper上节点变化的通知，决定如何切换Engine Manager进程
* zookeeper为一个单独的进程。

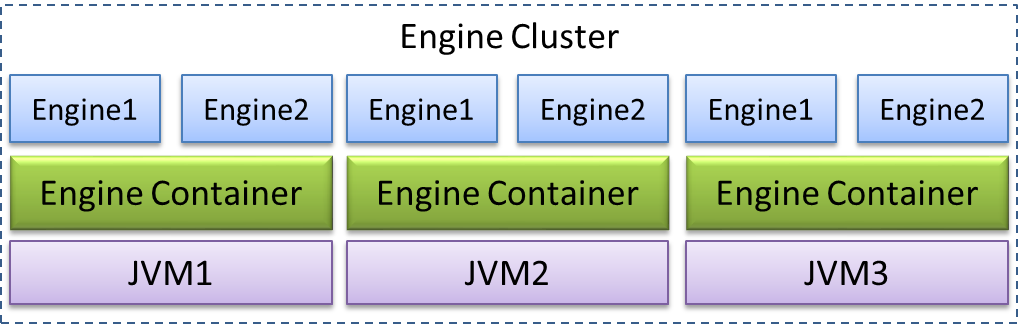
# 系统物理视图

## web控制台结构



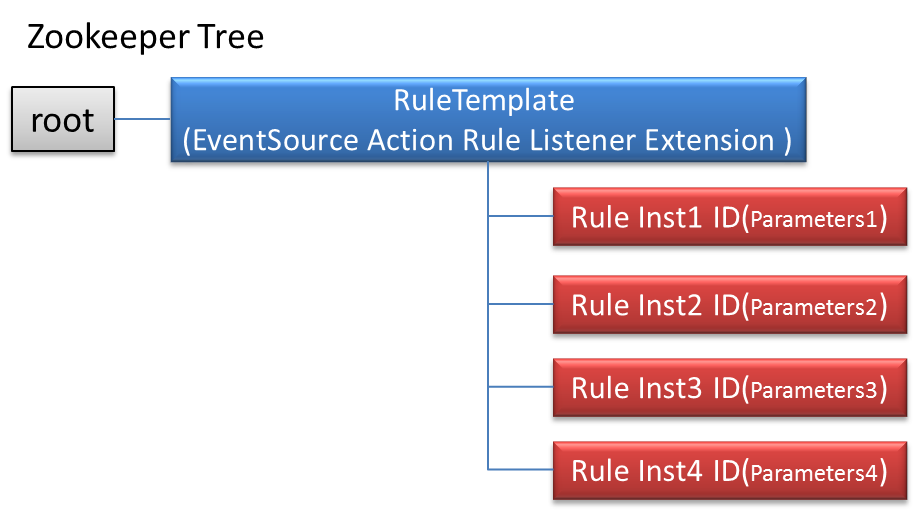
所有的管理类都由Framework进行管理，并作为web应用运行在Web Server中

## 分析引擎集群结构



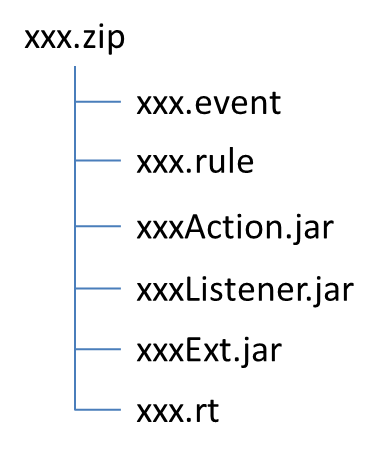
Engine Cluster由多一个JVM进程组成，每个JVM运行一个Engine Manager，每个Engine Manager运行多个Engine

## Zookeeper结构

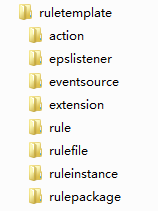


Zookeeper上会注册业务规则实例，一级节点为Rule标识，二级节点以业务规则实例命名，三级节点存放EventSource、Action、Rule、Listener、Extension以及业务规则参数

## 部署包结构

xxx.rt为规则模板的持久化文件

## 规则库结构

rulefile存放xxx.rt，即规则模板的持久化文件，rulepackage存放开发人员上传的部署包，ruleinstance存放规则实例持久化文件

# 系统开发视图

## 接口定义

*[描述在系统开发过程中对于公共接口定义的]*

1. *EAR WAR打包接口*

*由Server端提供统一的打包接口，安装包在新建EAR/WAR、Studio导出EAR/WAR时进行调用，具体接口定义参见《Server详细设计》文档。*

1. *EXF扩展资源接口*

*对扩展资源处理时，提供扩展框架由扩展接口决定如何对扩展的资源进行处理，如bps,btf的扩展资源，具体接口定义参见《Server详细设计》文档。*

## 开发工具

*[描述在系统开发过程中使用的开发工具，包括编程工具、文档编写工具、版本管理工具、分析设计工具、画图工具。需要明确版本]*

## 开发方式

*[描述整个系统的开发方式，工作目录，如何日构建、如何使单元测试自动化。]*

## 设计规范

*[描述设计期建模约定，比如定义出系统的版型、模型元素的颜色含义说明、模型元素的分包规则等]*

## 开发规范

*[此项目为实现层次的项目，比如Java项目，Eclipse Plug-in项目，EOS项目*

*描述子系统项目的组织和依赖，描述子系统间如何进行依赖，比如是项目依赖，还是类库依赖等]*

# 其他非功能约束

## *日志非功能约束*

*各个模块的对外接口(在com.eos包下暴露的API，值对象除外)，以及核心API，都需要在入口和出口记录Trace日志，入口处要记录接口的输入参数，出口处要记录接口的返回参数。如果API抛出异常且被catch而不再继续抛出异常，需要在抛出异常前将异常信息记录到日志，日志的信息必须包含异常栈。*

## *异常非功能约束*

*Spring新增模块分配的异常代码范围为：*

*普通异常：19003\*\*\* 运行期异常：19103\*\*\**

*异常资源文件ExceptionResource.properties，对应包为com.primeton.spring*

*编号常量文件ExceptionConstant.java，对应包为com.primeton.spring*

## *兼容性非功能约束*

*由于jar安装位置的改变，EOS 6.3安装的ear应用与EOS 6.1及以前版本不能安装在同一应用服务器上，如果安装在一起，将有jar冲突。*

## *国际化非功能约束*

*异常支持国际化配置，规则同EOS 6.1。*

# FAQ

*[FAQ，是指文档中可能存在一些大家不容易理解的问题，或者不理解文档中为什么存在某些内容，特此提供一个FAQ章节，用来帮助相关人员了解在阅读本文档时常见的问题]*

## FAQ1

# TBD

*[TBD，是指在编写文档时，可能因为某些需求或者技术，及不可预知的原因，对于某些问题无法准确地加以描述，或者遗漏了一些文档本应该包含的内容，又或者是针对某个问题而给出了某种方案，而放弃了其它方案，以上种种，均可记录下来，帮助相关人员了解以上问题出现的原因。TBD章节中所描述的内容通常为上面所说的内容，但不限于以上内容，如果大家暂时无法为某项内容找到合适的位置，均可放入该章节]*

## TBD