

优云运维平台

采控模块开发规范

（v1.0）

目 录

[第一章 2](#_Toc11181)

[1.1 编写目的 2](#_Toc22236)

[第二章 采控架构 2](#_Toc21977)

[2.1 采控总体框架 2](#_Toc23864)

[2.2 采控代理框架 3](#_Toc26695)

[2.3 托管模块调度 4](#_Toc6399)

[第三章 模块结构 5](#_Toc12508)

[3.1 模块描述表 5](#_Toc19497)

[3.2 模块包制作 5](#_Toc28354)

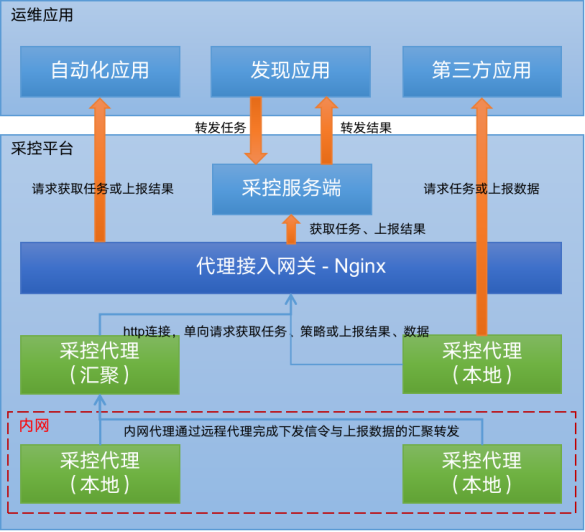
# 

## 编写目的

本文挡用于描述优云采控平台模块设计和开发规范，以便指导开发人员如何开发一个采控模块，或将现有的代理封装为采控平台可管理的模块。

# 采控架构

## 采控总体框架

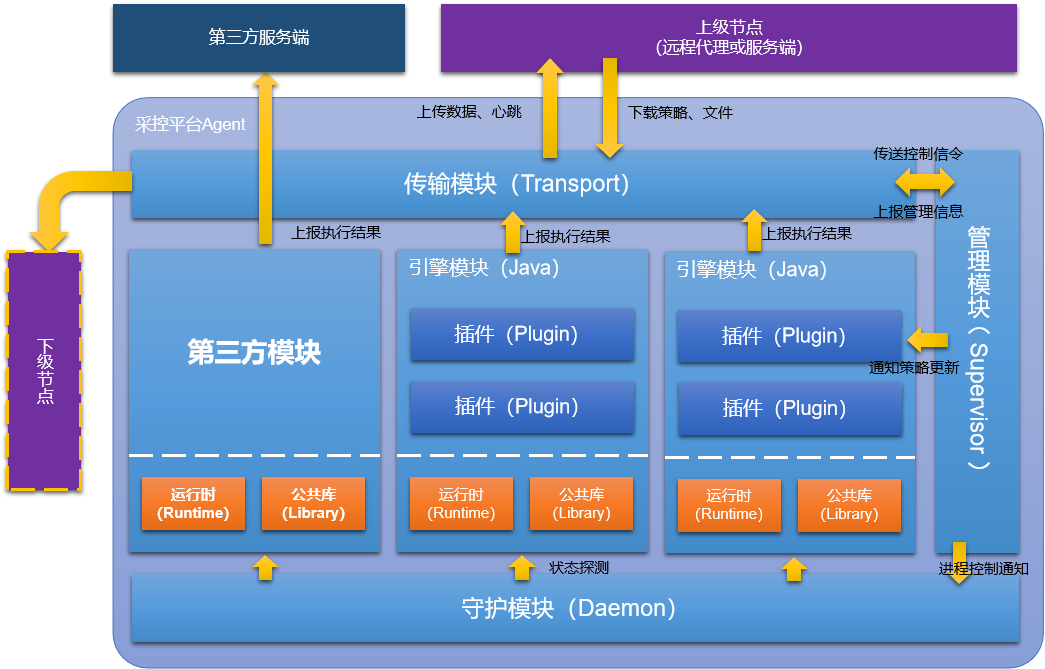


由上至下介绍采控平台各组件：

* **采控服务端：**分布式采控体系中顶层的管理中枢，负责对外提供服务化接口，接收任务并下发至采控代理以进行作业。并且实时收集来自下方采控代理作业过程中输出的各类数据，移交上层应用进行消费处理。
* **代理接入网关：**在实现上是一个Nginx负载均衡器，通过配置能够将下方采控代理发起的连接反向代理至上方的采控服务端、自动化应用、发现应用等。
* **采控代理（汇聚）：**一般部署在特殊分配的服务器上，该服务器仅作运维管理使用，以实现远程作业如远程连接主机执行操作、网络设备巡检、主机设备探活等。或部署在跳板机上，以实现下级代理上报数据的汇聚转发。远程代理提供的远程采控技术，通常用于被管主机设备不方便部署（如安全问题）或不能部署（如网络设备）本地代理程序的情形。远程代理包含所有本地代理的能力，所以也可以执行本地采控。
* **采控代理（本地）：**一般直接部署在生产服务器上，对且仅对当前主机上执行采集、控制，可进行常规运维作业如配置发现、配置变更、系统指标采集、主机安全巡检等。本地代理由于其工作环境的重要性，实现上必须是一个微小、稳定的程序。

**采控模块**运行在采控代理上，由采控代理负责模块的安装、更新、启停及卸载等管理。

## 采控代理框架



采控代理框架层和模块层各组件在运行时表现为一个个的进程，插件/脚本层的代码在运行时视情况作为一个进程执行或在一个线程池中调度执行。

采控框架层的守护模块和管理模块会在采控代理部署时注册为系统服务，受操作系统保护。管理模块负责对模块进行管理，包括模块的安装、启停、升级和卸载等。守护模块对采控模块的运行状态进行探测，当发现异常时可以重启模块。

## 托管模块调度

第三方模块（也称托管模块）会随着采控代理一起启动，随后常驻在内存中，它们的生命周期都由采控代理的管理模块进行管理。托管模块运行过程中发生意外退出会由守护模块进行重启恢复，但托管模块的通信均通过其自身和其服务端建立的通道通信，而非采控平台的通信框架，因此，采控代理不负责托管模块的采控策略或任务调度。

# 模块结构

## 模块描述表

该模块描述表用于进行模块信息的初始化：

|  |  |
| --- | --- |
| **信息** | **说明** |
| **模块代码** | 模块唯一标识，建议是 XXX-XXXX格式，例如：local-discovery |
| **模块名称** | 模块的名称，包括中文和英文名，例如：  中文：本地发现  英文：Local discovery |
| **模块描述** | 模块的描述，包括中文和英文描述，例如：  中文：本地自动化发现插件，可进行本地资源发现。  英文：Local automated discovery plugin for local resource discovery. |
| **模块分组类别** | 可选Local、Remote，Local指本地，Remote指远程模块 |
| **模块所支持的操作** | 例如：安装、卸载、更新、启动、停止 |
| **模块安装参数定义（可选）** | 模块安装需要的参数，在执行安装时界面会弹出界面让用户输入，同时在调用post\_install的时候会通过环境变量方式注入。例如定义：  SERVER\_URL：模块对应的SERVER地址  那么，在post\_install对应脚本里可以通过SERVER\_URL变量获取。 |

## **模块包制作**

* **模块要求**

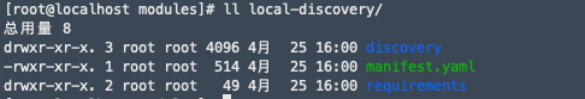
采控模块安装包需要解压完成后即可以通过命名行运行，模块所需的运行环境及相应类库需要包含在模块包中，并按相对目录存放好。

* **模块包文件名**

最终模块打包出的名称没有限制，但尽可能体现它是什么模块、支持什么操作系统，比如可以叫做 local-discovery-linux64.tar.gz 等。

* **模块结构**

各模块结构无严格限制，但需要将所有文件请放到和模块名称相同的文件夹中，且必须在根目录包含一个 manifest.yaml 的模块描述文件，内容定义在下节。例如local-discovery模块目录结构：



* **模块描述文件（manifest.yaml）**

|  |
| --- |
| # 以下字段必填  name: local-discovery  version: 2.1.3  description: description about test module  # 以下字段选填  platforms:  AIX: [64, 32]  Windows: []  Linux: [64]  run: python -m entry  priority: **999**  run:  Windows: .\\bin\\run\_agent.bat  Linux: ./bin/run\_agent  scripts:  post\_install:  Linux: ./bin/post\_install  Windows: bin\\post\_install.bat  post\_start:  Linux: ./bin/post\_start  Windows: bin\\post\_start.bat  post\_stop:  Linux: ./bin/post\_stop  Windows: bin\\post\_stop.bat  post\_install: python -m scripts.initialize  pre\_upgrade: python -m scripts.backup  pre\_start: python -m scripts.prepare  pre\_remove: python -m scripts.backup  post\_remove: python -m scripts.clean  backups:  **-** conf  **-** var/pki  dependencies:  SUSE32@>=11: [xxxxxx.tgz]    actions:  action1: python action1.py  action2: python -m parent.action2  action3: [./jre/bin/java, -jar, action3.jar]  env:  JAVA\_HOME: ./jre/bin  SHELL: /usr/bin/sh |

* **模块描述信息解释**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **必填** | **字段描述** |
| name | char(64) | **是** | 模块名称（英文+数字），在命令行安装一个模块时以该字段的值作为标识符，因此不同模块不能取相同的名字 |
| version | char(32) | **是** | 模块版本格式为语义化版本，参考[yarn 版本](https://yarnpkg.com/en/docs/dependency-versions) 和 [语义化版本](http://semver.org/lang/zh-CN/) |
| description | char(256) | **是** | 描述模块的功能 |
| platforms | dict | 否 | 表示基础环境支持的平台，详见下面关于 platforms的描述 |
| run | char(512) 或 array 或 dict | 否 | 表示模块启动的命令，默认的工作目录是模块根目录，可以针对不同平台 |
| priority | int | 否 | 进程优先级 |
| scripts | dict | 否 | 定义一些信号事件的处理器，可以是单个命令，也可以是包含多个命令的数组，这样将逐个执行这些命令，可以针对不同平台 |
| scripts.post\_install | char(512) 或 array或 dict | 否 | 当模块安装完毕后执行本字段指定的命令，该命令属于安装过程的一部分，只有当该命令同步执行完毕后模块才完成安装，可以针对不同平台 |
| scripts.pre\_upgrade | char(512) 或 array或 dict | 否 | 当准备升级模块前执行本字段指定的命令，该命令属于升级过程的一部分，只有当该命令同步执行完毕后才开始真正的升级工作，可以针对不同平台 |
| scripts.post\_start | char(512) 或 array或 dict | 否 | 当模块启动成功后执行本字段指定的命令，该命令不属于启动过程的一部分，只有在模块成功启动后才被异步触发调用，可以针对不同平台 |
| scripts.pre\_remove | char(512) 或 array或 dict | 否 | 当准备移除模块前执行本字段指定的命令，该命令属于移除过程的一部分，只有当该命令同步执行完毕后才开始真正的移除工作，可以针对不同平台 |
| scripts.post\_remove | char(512) 或 array或 dict | 否 | 当模块移除后执行本字段指定的命令，该命令属于移除过程的一部分，只有当该命令同步执行完毕后模块才完成移除，可以针对不同平台 |
| scripts.post\_stop | char(512) 或 array或 dict | 否 | 当模块关闭成功后执行本字段指定的命令，该命令不属于关闭过程的一部分，只有在模块成功关闭后才被异步触发调用，可以针对不同平台 |
| backups | array | 否 | 在模块升级时（script.pre\_upgrade执行前）会备份本字段指定的文件或文件夹，并在新版模块安装后将备份的文件或文件夹放到对应的路径 |
| dependencies | dict | 否 | 表示需要依赖，详见下文关于 dependencies 的描述 |
| actions | dict | 否 | 模块提供的actions：key是action的名称value是action的执行路径，可以是字符串，或者列表。 |
| env | dict | 否 | 模块提供的环境变量，可以填写相对路径，或绝对路径。推荐路径分隔符都是用/而不是\。/在全平台生效，\只在windows平台生效。 |

* **platforms**

表示基础环境支持的平台，如果不加本字段，说明支持所有平台和所有 cpu 架构。字段内容形如：

Windows: [32]

Windows@6: [64]

Windows@>=10.0.14393: [32, 64]

Linux: [64]

CentOS@6: [64]

CentOS@^6.1: [64]

AIX@6.1: [64]

AIX@>=0.0.0: [32, 64] #表示支持AIX的所有版本

**说明：**

platforms 是一个字典，里面的每一个 key 是个字符串，表示支持的系统大类或发行版；每一个 value 是个列表/数组，表示支持的 cpu 架构。

* 关于 key：

如果只有系统大类（如 Windows、Linux、UNIX），表示支持该系统大类中的所有子版本，填 Linux 等同于同时填 CentOS、Debian、Fedora等。

如果是 “发行版” 加\*\*“@”\*\*加 “比较符”（选填）， 加 “版本号” ，表示支持特定版本号要求的特定系统。版本格式为语义化版本，参考 [yarn 版本](https://yarnpkg.com/en/docs/dependency-versions) 和 [语义化版本](http://semver.org/lang/zh-CN/)

注意，对于Windows，这里的版本号指内核版本号，如10.0.14993；对于非Windows，这里的版本号指发行版本号7.2

* 关于 value

不填表示同时支持 64 和 32，也可填写[32, 64],或 [64], 或 [32]

PS： 苹果系统为 Darwin

* **run和scripts**

当不同平台或者具体发行版上执行的命令不同时，可以根据系统的平台或体发行版分类，当所有平台通用时，直接指明要执行的命令即可。

* **dependencies**
* key ：是在什么样的系统上需要依赖，格式为“系统”+“CPU 位数 或 空”+“@语义化发行版本 或 空”，比如：

SUSE32@>=11 表示 SUSE 系统，32 位 CPU，发行版要>=11 Windows@>=10.0.14993 表示 Windows系统，32 和 64 位 CPU，发行版要>=10.0.14993 CentOS64 表示 CentOS 系统，32 和 64 位 CPU，发行版任意。

* value ：是所依赖的安装包的列表。