Metainfo描述

在实例化过程中，MetaInfo对象由文件metainfo.xml和metainfo.json生成，过程见AgentProviderService#getApplicationMetainfo

appDef <= appConfig.json中定义的路径，一般指定为zip包，解压后在根目录下找到metainfo.json，如果找不到Json，则找metainfo.xml文件

文件解析器为：MetainfoParser，

在有addOnPackage时（极少）为AddOnPackageMetainfoParser

MetainfoParser.fromXmlSteram: 使用org.apache.commons.digester.Digester ,其封装了XMLReader

MetainfoParser.fromJsonStream: 使用com.google.gson.Gson.Gson

#### 将文件转换为Metainfo对象 ####

Metainfo.java

**#schemaVersion[String]** 版本号

**#applicationPackage[ApplicationPackage]**，add on的component

**#application[Application]**

Application，定义一个app的所有信息描述如下：

**#exportConfigs**，对外提供的配置文件，例如storm-site,hbase-site

**#exportGroups**，export Groups信息，每个group可以有多个Export

#name: 组名称

#exports [List<Export>]，export包括name和value

**#osSpecifics**，os类型及安装包的类型

#osType => any，目前仅有any

#packages[OSPackage]，type和name(包的名称)

如type [tarball]，name[files/apache-storm-1.0.tar.gz]

**#commandOrders，**定义app内部组件的启动顺序

<commandOrders>

<commandOrder>

<command>HBASE\_REGIONSERVER-START</command>

<requires>HBASE\_MASTER-STARTED</requires>

</commandOrder>

<commandOrder>

<command>HBASE\_MASTER-START</command>

<requires>HBASE\_REST-INSTALLED</requires>

<requires>HBASE\_THRIFT-INSTALLED</requires>

<requires>HBASE\_THRIFT2-INSTALLED</requires>

</commandOrder>

</commandOrders>

定义了的命令及要求组件所处的状态，其中命令[Command]包括：NOP,INSTALL,

INSTALL\_ADDON,START,STOP,UPGRADER及TERMINATE

命令执行要求的状态[STATE]，INIT,INSTALLING,INSTALLED,STARING,STARTED等。上例中，HBASE\_REGIONSERVER启动命令需要HBASE\_MASTER处于STARTED状态

**#packages，**和OsSpecific作用比较类似，定义应用的安装文件

name，安装包名

type，安装包的类型

**#components**，组件的核心描述

#category，该组件所处的位置，如MASTER,SLAVE,CLIENT

#type，组件的类型，如STANDARD,DOCKER，PYTHON及YARN\_DOCKER

#publisheConfig，该组件是否向外提供config，默认是false。如果是true，则可以 通过getConfig命令获取通过heatbeat传过来的配置文件，从而提供给查询者。该配置 和exportConfig是对应的

#minInstanceCount，最小的启动个数

#maxInstanceCount，最多的启动个数

#autoStartOnFailure，失败后是否自动启动，很重要的特性，默认是false，但是一 般配置为true，从而达到高可用性的目的

#appExports，向外提供的应用的export，格式为[exportGroup]-[exportName]

#compExports，组件的export，需要了解其用处（应用内部？？）

#dockerContainers，如果type为docker，则需要定义docker的属性

#name，docker名称

#image，该docker使用的image名称

#options，docker命令中的可选参数参数，添加到agent中的docker命令中

#mouts[DockerContainerMount]，目录mount，{containerMount:hostMount}

#ports[DockerContainerPort]，container的port映射，{containerPort:hostPort}

#statusCommand，用于检查运行应用的健康状况，返回0时为healthy，非0 为unhealthy。如果不定义，则slider执行docker top ${container\_ID}|grep来获取运 行状况

#commandPath,docker命令的路径

#addtionalParam,为了支持多个port，但是目前还没用

#inputFiles，将本地文件复制到container中,[containerPath,fileLocalPath]

=========以上为slider 0.80的参数，下面是0.90后的参数===========

#runPrivilegedContainer，需要进一步的研究

#configFiles，配置文件，type,fileName,dictionaryName

#network, 网络相关的参数

#useNetworkScript，网络的执行脚本