《听云Java探针功能说明手册》

Version:2.1.1

BuildDate:2016-01-21

《听云Java探针功能说明手册》

Version:2.1.1

BuildDate:2016-01-21

听云Java 探针采集的数据项：

- Application request controller and dispatch activity
- Database Operator
- External web services calls
- View resolve
- Uncaught Exceptions and counts
- Process Memory and CPU usage

通过对特定方法嵌码，采集这些数据：每个方法的响应时间以及响应时间统计分析（最大值、最小值、平均值以及标准差）。在报表系统里，你可以看到这些类和方法的相应时间和调用次数。

除此之外，还可以采集线程堆栈、数据库执行计划、自定义参数、HTTP request parameters等。

数据传输

Java 探针默认启用SSL方式传输数据。可以在Java Agent的配置文件（tingyun.properties）里启用或者禁用使用HTTPS。

Java Agent 通讯使用2个host：

> redirect.networkbench.com
>
> dc*.networkbench.com

*可能为任意的数字。这个数字可能不是一成不变的，启动探针时，您可以通过logs/tingyun_agent.log文件查看。

如果您有防火墙，需要将以下IP和端口添加到许可列表：

IP

> - 123.59.62.110
> - 123.59.62.110
> - 123.59.62.108
> - 123.59.62.118
> - 123.59.62.107
> - 123.59.62.109

端口

> - TCP 80
> - TCP 443

兼容性和要求

安装Java探针之前，请确保您的系统满足如下这些条件。

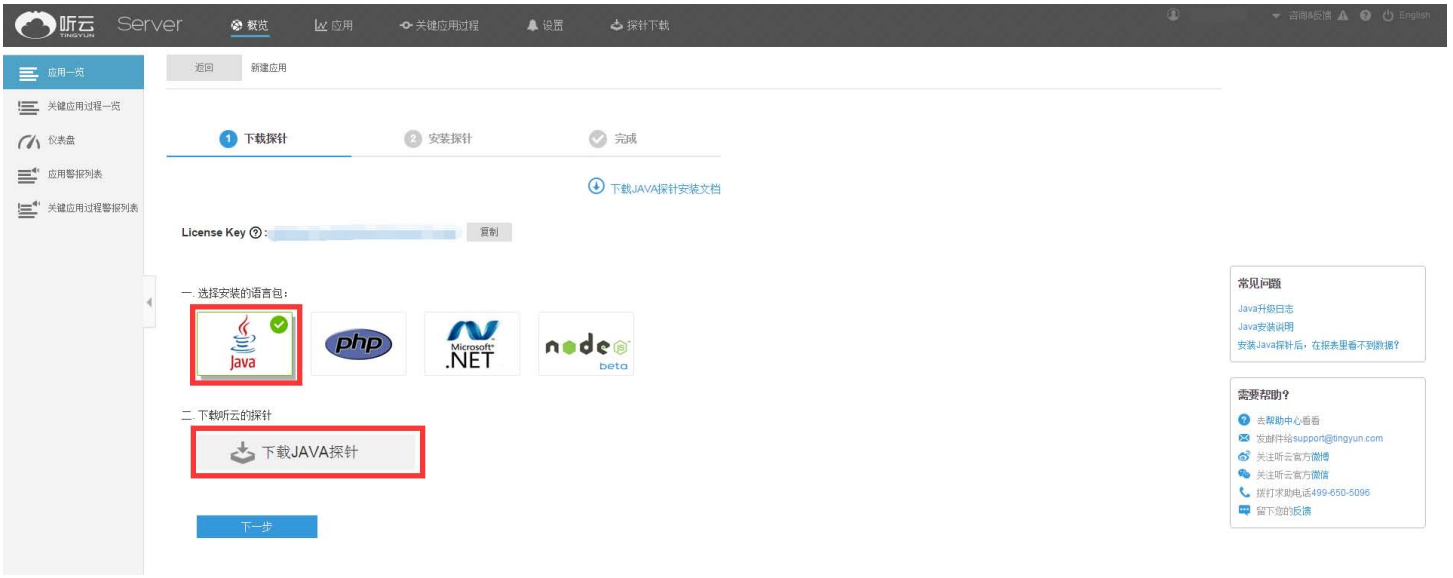
	要求
JVM	• Oracle Hotspot JVM for Linux, Solaris, Windows, OSX 6 - 8 • Apple Hotspot JVM for OSX 6 • IBM JVM for Linux 6 - 7
Application servers	• ColdFusion 10 • Glassfish 3.0 - 4.0 • JBoss 4.0.5 - 7.x • JBoss EAP 6.x • Jetty 6.1 - 9.1 • Netty 3.5-3.9 • Resin 3.1.9 - 4.0 • Tomcat 5.5 - 8 • TomEE 1.5 • WebLogic 10.3.3 - 12.1 • WebSphere 7.0 - 8.5.5 • WildFly 8
Frameworks	• Grails 1.3.7 - 2.3.x • JSF (Java Server Faces) • Play 1.2.2 - 1.2.4 • Play 2.0.3 - 2.3.x • Spring 3.x - 4.x • Struts 2
JDBC drivers	• jTDS open source JDBC 3.0 type 4 driver for Microsoft SQL Server (6.5 up to 2008) and Sybase ASE • MySQL mysql-connector-java-5.1.13-bin • Oracle ojdbc14, ojdbc5, ojdbc6 • Postgres postgressql-8.4-701.jdbc3, 9.0, 9.1 • SQLServer sqljdbc4 • Generic support for other JDBC drivers
NoSQLs	• Memcached • MongoDB 2.12.0 • Jedis 2.2.0
RPC Services	• Dubbo Consumer 2.4.0+ • Thrift Client 0.8.0+ • XmlRpc Client • Apache Axis2 1.5+ • Apache CXF 2.7.0+ • Spring WS 2.0.0+ • Java JAX-WS • Java JAX-RS (REST) • Jersey 2.0+ (REST) • Resteasy 2.2+ (REST)
Others	• HttpClient 3.0.1 - 4.3.0 • java.net (HttpURLConnection) • JMS 1.1 and Spring-JMS

	• JMX • JSP - Java Server Pages 2.0 - 2.2 • Quartz Job Scheduler 1.8.3 - 2.2.x • Scala 2.9 - 2.10 async tracking • Solr 1.4.0 - 4.0

下载Java探针

- 1. 登录到http://report.tingyun.com/
- 2. 选择Server产品，点击“新建应用”

- 3. 选择Java语言，点击“下载Java探针”



在tingyun-agent-java.zip文件中包含一下2个文件，以及其他的文件：

- 1. tingyun-agent-java.jar: 探针类文件
- 2. tingyun.properties: 配置文件，包括你的license key、应用名称等

在Application Server 启动的时候，探针会在tingyun-agent-java.jar的同级目录查找tingyun.properties文件；探针会在tingyun-agent-java.jar所在目录中的子目录logs中记录探针运行日志（请确保改目录有写权限）。

自动安装(支持GlassFish ,Jetty, JBoss和Tomcat)

1.1.Linux or Mac OS

- 1)、解压缩tingyun-agent-java-*${version}*.zip到应用服务器的根目录。GlassFish需要解压到使用的domain目录，如../domains/domain1/:

```
unzip tingyun-agent-java-${version}.zip -d /path/to/appserver/
```

- 2)、在tingyun目录下执行自动安装程序:

```
cd /path/to/appserver/tingyun
java -jar tingyun-agent-java.jar install
```

- 3)、启动或重新启动你的应用服务器。
- 4)、登录到http://report.tingyun.com/查看你的应用性能。

1.2.参数说明

install 命令会查找启动脚本文件，先备份，然后修改脚本文件，资java的启动脚本里增加-javaagent参数。

install 命令参数说明:

参数	描述
-h	显示命令帮助
-l licensekey	指定license key
-s /path/to/appserver	指定应用服务的路径 如果没有将tingyun-agent-java.zip解压到应用服务器的根目录，需要通过该参数指定

install 命令执行结果:

- 1)、执行成功后，会提示您重启应用服务器，访问您的应用，登陆到报表系统查看数据;
- 2)、执行失败后，会提示您失败原因。

手动安装

- 1)、解压缩安装文件包到你的应用服务器的根目录(推荐)或指定目录
- 2)、备份tingyun目录下tingyun.properties文件
- 3)、在应用服务器的启动脚本中增加:
  `-javaagent:/path/to/tingyun/tingyun-agent-java.jar`
- 4)、启动或重新启动你的应用服务器，详细安装文档请查看下章节中不同应用服务器安装说明。
- 5)、登录到http://report.tingyun.com/查看你的应用性能。

不同Application Server安装说明

App Server	Notes
ColdFusion	ColdFusion installation.
Geronimo	使用geronimo run启动时，将tingyun-agent-java.jar添加到 `JAVA_OPTS` 变量里: `export JAVA_OPTS="$JAVA_OPTS -javaagent:/full/path/to/tingyun-agent-java.jar" && geronimo run`
Glassfish	Glassfish installation.
Grails	Grails installation.
JBoss	JBoss installation.
Jetty	在jetty.sh里配置 `JAVA_OPTIONS` : `export JAVA_OPTIONS="${JAVA_OPTIONS} -javaagent:/full/path/to/tingyun-agent-java.jar"`
Play 1.x	Play installation.
Play 2.x	Play 2 installation.
Resin	Resin installation.
Solr	单独运行Solor, 在start.jar 之前添加 `-javaagent` : `java -javaagent:/full/path/to/tingyun-agent-java.jar -jar start.jar` 在应用服务器里运行Solr, 在应用服务器的启动脚本里增加-javaagent ， 并要确保该应用服务器开启JMX
Tomcat	修改catalina.sh/catalina.bat，配置 `JAVA_OPTS` : `export JAVA_OPTS="$JAVA_OPTS -javaagent:/full/path/to/tingyun-agent-java.jar"` 如果tomcat 作为windows的服务, 请见 Tomcat installation. 如果使用 Apache Commons Daemon (jsvc) 启动Tomcat, 详见 The Apache Commons Daemon (jsvc).
WebLogic	WebLogic installation.
WebSphere	WebSphere installation. 对于 WebSphere Community Edition, 启动是增加 `JAVA_OPTS` 配置来启动: `export JAVA_OPTS="$JAVA_OPTS -javaagent:/full/path/to/tingyun-agent-java.jar" && startup`
WildFly	WildFly installation.
其他应用服务器	Java探针可以运行在任何应用服务器上，只要在启动大参数中配置 `-javaagent:/path/to/tingyun-agent-java.jar`

注：所有配置-javaagent的地方请使用完整路径。

ColdFusion installation for Java

在ColdFusion上安装Java探针

注意：探针暂时不支持ColdFusion Version 9 on IIS

- 1、访问ColdFusion Administrator Console；
- 2、从左侧菜单依次选择： SERVER SETTINGS -> Java and JVM;
- 3、在 JVM Arguments 输入框里，配置 `-javaagent` 参数;

```
-javaagent:/path/to/tingyun-agent-java.jar
```

- 5、点击 Submit Changes；
- 6、重启ColdFusion。

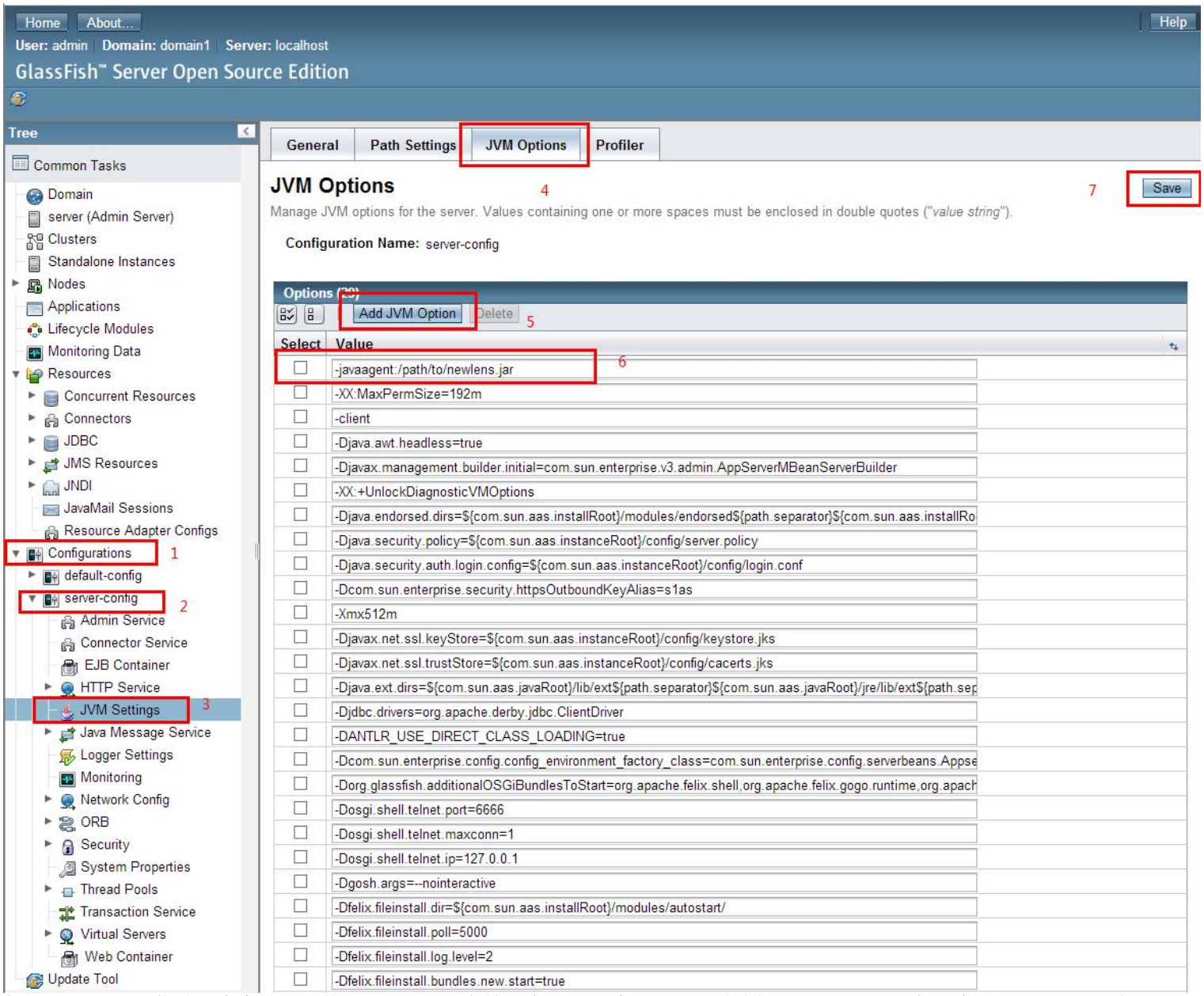
Glassfish installation for Java

在Glassfish上安装Java探针

- 1、访问Glassfish配置界面;
- 2、从左侧菜单依次选择： Configurations -> server-config -> JVM Settings -> JVM Options;
- 3、在 JVM Options 页面，选择 Add JVM Option;
- 4、添加一个条目，配置 `-javaagent` 参数;

```
-javaagent:/path/to/tingyun-agent-java.jar
```

- 5、保存设置
- 6、重启应用服务器



如果Glassfish不能够正常启动，可能是-javaagent参数没有设置正确，可以通过编辑domain.xml中正确配置-javaagent参数。({GLASSFISH_HOME}/glassfish/domains/{domain}/config/domain.xml)

Grails installation for Java

使用run-app启动Grails

当使用 `grails run-app` 启动服务时，需要在run-app之前增加-javaagent参数：

```
grails -noreloading -javaagent:/path/to/tingyun-agent-java.jar run-app
```

Note: `-noreloading` 参数只在 Grails 2.x 版本需要。

使用run-war启动Grails

修改{your-grails-app}/conf/BuildConfig.groovy 文件，在 `grails.tomcat.jvmArgs` 属性里添加-javaagent参数：

```
grails.tomcat.jvmArgs = ["-javaagent:/path/to/tingyun-agent-java.jar"]
```

JBoss installation for Java

在JBoss上安装Java探针 说明：从8.0 以后的版本，JBoss 更名为WildFly。如果是正在使用WildFly，请参见 WildFly installation for Java

目录:

Domain mode

Domain mode 使用在JBoss 6.X EAP 或 7.0.X 及以后的版本。 每个服务器组的JVM 配置信息可以在domain/configuration/domain.xml文件配置。

```
<server-group name="main-server-group" profile="full">
  <jvm name="default">
    <jvm-options>
      <option value="-javaagent:/path/to/tingyun-agent-java.jar"/>
    </jvm-options>
  </jvm>
</server-group>
```

请确认 `-javaagent` 指定路径是tingyun-agent-java.jar的完整路径。

如果你在Windows使用，路径使用的是斜杠'/'。例如： `d:/newlens/tingyun-agent-java.jar`

警告: JBoss 存在一个Bug 详见：JBoss bug in 7.0.2.Final and 7.1.0.Alpha1不允许在domain.xml中配置jvm-options。如果你使用的是存在这个bug的JBoss，请升级JBoss 应用服务器。

Standalone mode

警告:请确认 `-javaagent` 指定路径是tingyun-agent-java.jar的完整路径。

平台	用法
Unix / Mac OS 使用 6.x EAP 或 7.0.x AS 及更高版本	在bin/standalone.conf文件的底部, 添加: `JAVA_OPTS="$JAVA_OPTS -javaagent:/path/to/tingyun-agent-java.jar"`

Windows 使用 6.x EAP 或 7.0.x AS 及更高版本	在bin/standalone.bat文件的该行之前:
	`set JBOSS_ENDORSED_DIRS=%JBOSS_HOME%\lib\endorsed`
	添加:
	`set "JAVA_OPTS=-javaagent:C:/path/to/tingyun-agent-java.jar %JAVA_OPTS%"`
	使用斜线: `C:/newlens/tingyun-agent-java.jar` .

Unix / Mac OS 使用 6.x 及更低版本	在bin/run.conf文件的底部，添加: `JAVA_OPTS="$JAVA_OPTS -javaagent:/full/path/to/tingyun-agent-java.jar"`
Windows 使用 6.x 及更低版本	在bin/run.bat文件中该行之前: `set JBOSS_CLASSPATH=%RUN_CLASSPATH%` 添加: `set "JAVA_OPTS=-javaagent:C:/path/to/tingyun-agent-java.jar %JAVA_OPTS%"` 使用斜线: `C:/newlens/tingyun-agent-java.jar` .

Play installation

- Play 2.3.x
- Play 2.2.x production mode
- Play 2 production mode
- Play 2 development mode
- Play 1

Play 2.3.x

进入项目目录

`./activator -J-javaagent:/path/to/tingyun/tingyun-agent-java.jar run`

Play 2.2+ production mode

注意:2.2+之后`play start`不允许使用 -javaagent参数。

解压包含`start`脚本的zip文件:

```
play clean dist && unzip target/universal/*.zip
```

在启动应用时增加`-J-javaagent`参数:

```
cd UNZIPPEDFOLDER;
./bin/SCRIPTNAME -J-javaagent:/path/to/tingyun-agent-java.jar
```

Play 2 production mode

Note:该版本的Play使用`play start`命名启动服务时，不支持-javaagent参数。

解压包含`start`脚本的zip文件:

```
play clean dist && unzip dist/*.zip
```

在启动应用时增加`-javaagent`参数:

```
cd UNZIPPEDFOLDER;
chmod a+x start;
./start -javaagent:/path/to/tingyun-agent-java.jar
```

Play 2 development mode

在play的安装目录修改`framework/build`文件 (或者修改windows环境下的build.bat文件)

```
nano `which play`/../framework/build
```

在 java exec 调用之前配置JAVA_OPTS，增加-javaagent参数:

```
JAVA_OPTS="$JAVA_OPTS -javaagent:/path/to/newlens/tingyun-agent-java.jar"
```

如果在调用java exec这行不存在 `${JAVA_OPTS}` ，将 `${JAVA_OPTS}` 添加进去

dev模式启动(切勿使用 `play start`):

```
play run
```

Play 1 installation

在运行应用的时候增加-javaagent参数:

```
play run helloworld -javaagent:/path/to/tingyun-agent-java.jar
```

Resin installation

Resin pro 3.1+ production mode

找到conf目录下的resin.conf文件，增加 `<jvm-arg>-javaagent:/path/to/tingyun-agent-java.jar</jvm-arg>`，详情如下：

针对集群所有server增加探针：

```
<cluster id="cluster_id">
          ........
    <server-default>
          ...........
        <jvm-arg> -javaagent:/path/to/tingyun-agent-java.jar</jvm-arg>
          ..........
   </server-default>
          ..........
</cluster>
```

针对集群某台server增加探针：

```
<cluster id="cluster_id">
          ........
    <server id="" address="xxx.xxx.xxx.xxx" port="xxxx">
          ...........
        <jvm-arg> -javaagent:/path/to/tingyun-agent-java.jar</jvm-arg>
          ..........
   </server>
          ..........
</cluster>
```

Resin pro 4.0+ production mode

Resin pro 4.0+ 有两种安装方式

第一种：找到conf目录下的resin.properties文件，将jvm的配置打开，并增加 `-javaagent:/path/to/tingyun-agent-java.jar`，详情如下：

```
# Arg passed directly to the JVM
jvm_args  : -javaagent:/path/to/tingyun-agent-java.jar
# jvm_mode    : -server
```

第二种：增加 `<jvm-arg>-javaagent:/path/to/tingyun-agent-java.jar</jvm-arg>`，详情如下：

针对集群所有server增加探针：

```
```

```
<cluster id="cluster_id">
          ........
    <server-default>
          ...........
        <jvm-arg> -javaagent:/path/to/tingyun-agent-java.jar</jvm-arg>
          ..........
   </server-default>
          ..........
</cluster>
```

针对集群某台server增加探针：

```
<cluster id="cluster_id">
          ........
    <server id="" address="xxx.xxx.xxx.xxx" port="xxxx">
          ...........
        <jvm-arg> -javaagent:/path/to/tingyun-agent-java.jar</jvm-arg>
          ..........
   </server>
          ..........
</cluster>
```

Tomcat installation

目录:

Tomcat for Windows

大多数Windows用户都把Tomcat作为服务来运行，Tomcat提供了一个配置程序来指定该服务的JVM参数。

- 1、点击Start > Apache Tomcat (Version) > Configure Tomcat;
- 2、选择： Java;
- 3、在 Java Options 文本框中的行末增加:
  ```
  -javaagent:/path/to/tingyun-agent-java.jar
  ```
- 4、点击Apply按钮;
- 5、重启Tomcat服务

设置`-javaagent`参数时，请使用正斜杠作为路径分隔符，例如:

```
-javaagent:C:/tingyun/tingyun-agent-java.jar
```

对于Tomcat 6版本，在`-javaagent`参数之后需要有回车符。并且路径分隔符可以使用正斜杠和反斜杠。

如果您使用`catalina.bat`来启动Tomcat，请在该批处理文件的顶部增加以下内容:

```
SET JAVA_OPTS=%JAVA_OPTS% -javaagent:/path/to/tingyun-agent-java.jar
```

Apache Commons Daemon(jsvc)

Tomcat 6中自带的Apache Commons Daemon(jsvc)不支持`-javaagent`参数。关于该问题的描述请见：[daemon] JSVC does not support all the standard java 5.0 launcher options

在Apache Commons源代码仓库中已经包含了该问题的修复版本。参见:

- Apache源代码仓库
- Apache SVN源代码仓库

您也可以直接下载编译好的jsvc版本。该版本的jsvc支持通过`-X`前缀来设置`-javaagent`参数

在WebLogic上安装Java探针

目录:

在Linux和Mac下的安装

在Linux和Mac操作系统下安装听云Java探针的步骤:

- 1、在domain的bin目录下找到startWebLogic.sh文件:

```
WEBLOGIC_HOME/user_projects/domains/domain_name/bin/startWeblogic.sh
```

- 2、修改文件，在 `# START WEBLOGIC` 和 `echo "starting weblogic with Java version:"` 之间增加探针配置:

```
...
# START WEBLOGIC

export JAVA_OPTIONS="$JAVA_OPTIONS -javaagent:/path/to/tingyun/tingyun-agent-java.jar"

echo "starting weblogic with Java version:"
...
```

- 重启weblogic

在Windows下的安装

在Windows操作系统下安装听云Java探针的步骤:

- 1、在domain的bin目录下找到startWebLogic.bat文件:

```
WEBLOGIC_HOME/user_projects/domains/domain_name/bin/startWeblogic.bat
```

- 2、修改文件，在 `# START WEBLOGIC` 和 `echo "starting weblogic with Java version:"` 之间增加探针配置:

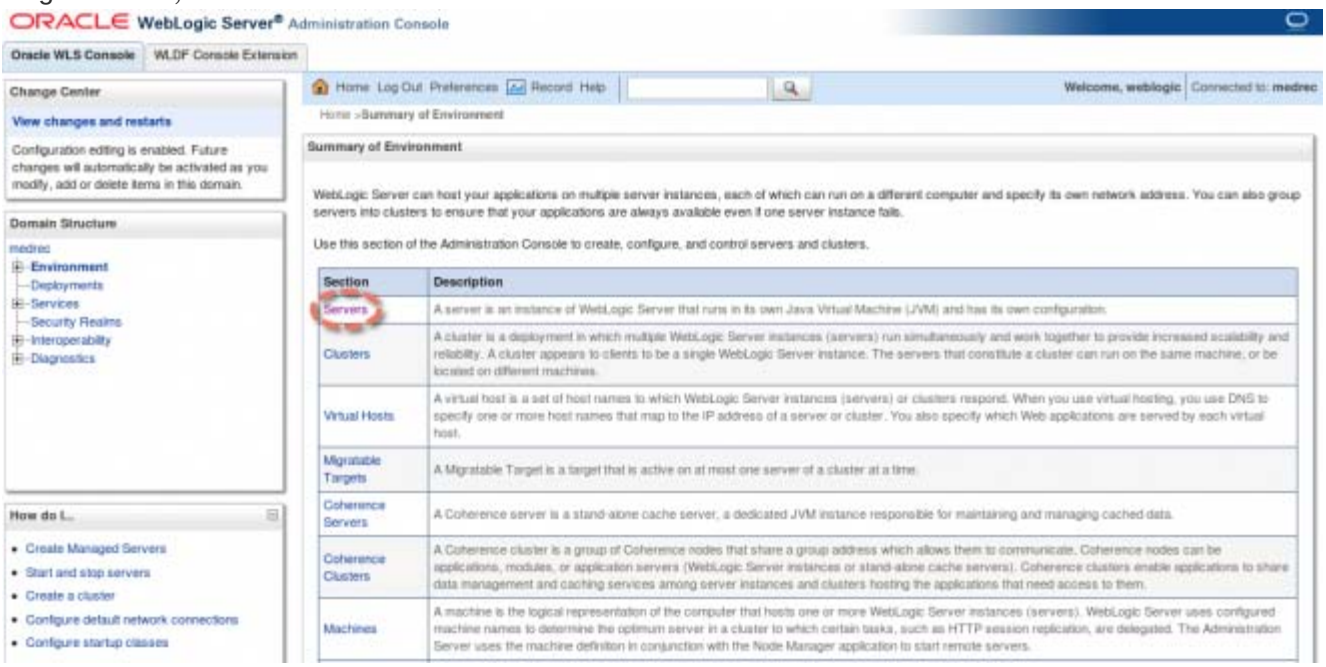
```
...
# START WEBLOGIC

set JAVA_OPTIONS="%JAVA_OPTIONS% -javaagent:C:\path\to\tingyun\tingyun-agent-java.jar"

echo "starting weblogic with Java version:"
...
```

- 重启weblogic

管理控制台安装

当使用Node Manager来启动和停止被管理的服务器时，采用以下的步骤来安装Java探针：

- 从管理控制台中，浏览：Environments > Servers > (select your server) > Server Start > Arguments；



- 2、在Arguments选项的文本框中增加以下内容：

```
-javaagent:"/path/to/tingyun-agent-java.jar"
```

TODO:截图

在WebSphere上安装Java探针

在WebSphere上安装Java探针

目录:

兼容性

听云Java探针支持WebSphere 7.0到8.5.5的所有版本，除了有部分8.5.0的版本与该探针不兼容。

在WebSphere上安装Java探针

1. 修改server.xml文件(不推荐)，文件路径大致为：*WAS_HOME*/AppServer/profiles/*app_server_name*/config/cells/*cell_name*/nodes/*node_name*/servers/*server_name*/server.xml

```
...
        <jvmEntries ... genericJvmArguments="-javaagent:/path/to/tingyun/tingyun-agent-java.jar"
... >
        ...
        </jvmEntries>
...
```

2. 通过管理控制台配置探针(推荐)：

- 1、登录管理控制台:https://ip:9043/ibm/consol
- 2、浏览：Servers > Application servers > (选择指定的Server);
- 3、选择：Configuration > Service Infrastructure > Java and Process Management > Process Definition > Additional Properties;
- 4、在Process Definition > Additional Properties下，选择Java Virtual Machine;
- 5、在Java Virtual Machine页面中，在Generic JVM arguments选项的文本框中，增加以下内容：

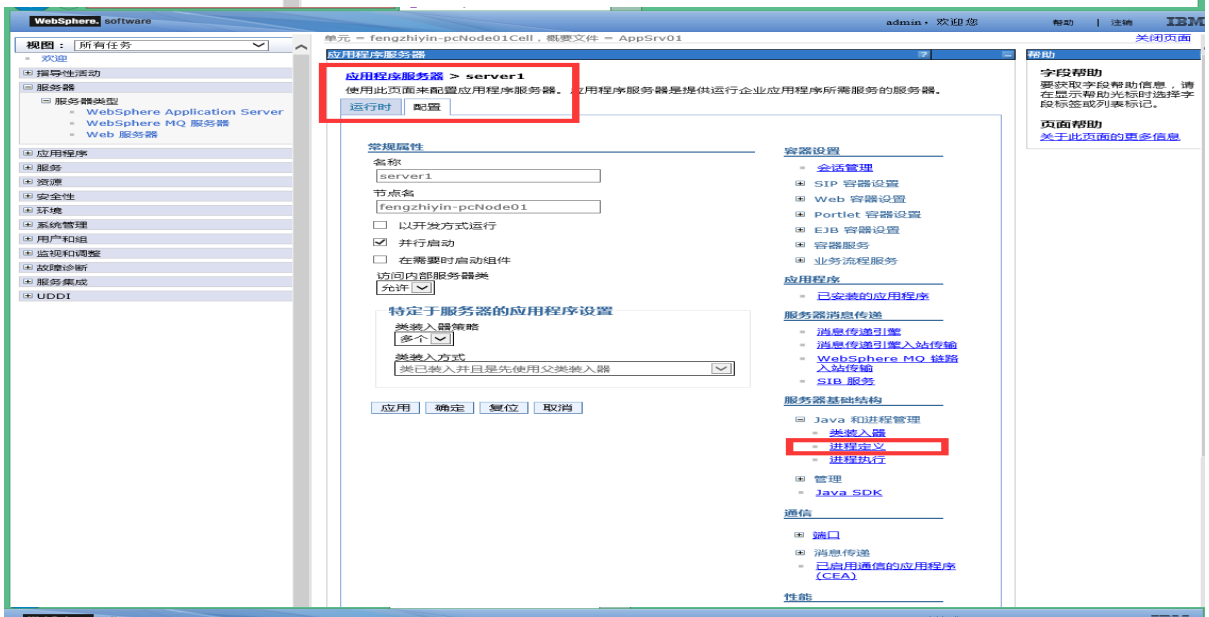
  ```
  -javaagent:"/path/to/tingyun/tingyun-agent-java.jar"
  ```

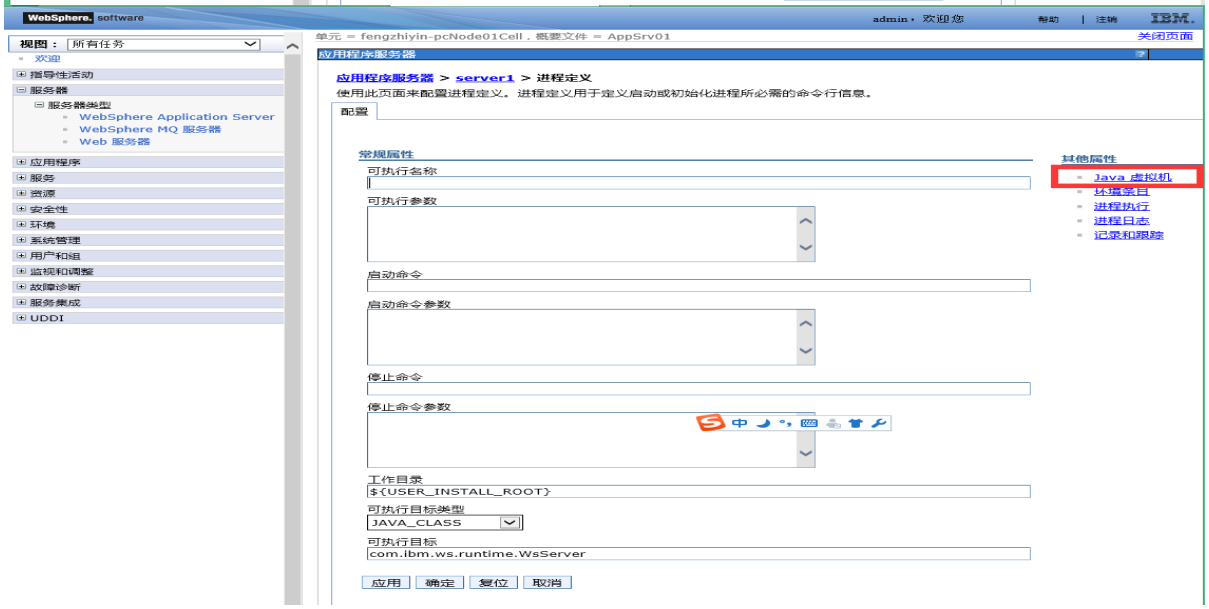
- 6、点击Apply按钮，然后点击Save按钮;
- 7、重启服务。

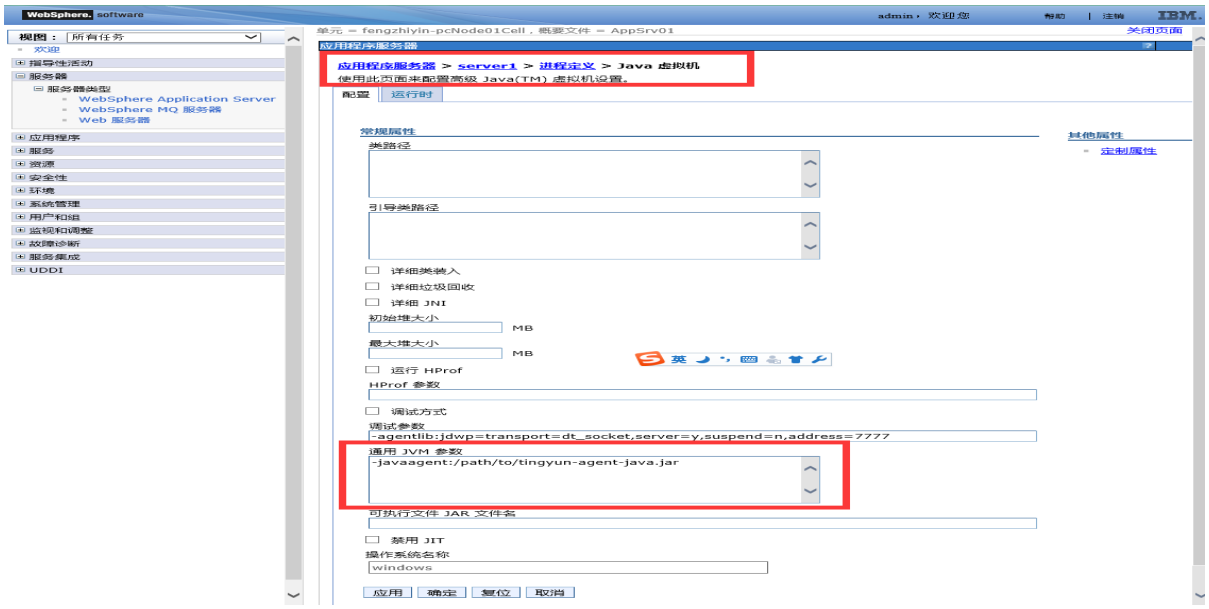
- > 提示：如果启用了 Java安全性(Java 2 Security), 需要授权Tingyun Agent通过JMX获取PMI指标。

- 参考以下步骤：


WebSphere. software
admin，欢迎您
帮助 | 注销
IBM.
视图： 所有任务
欢迎
指导性活动
服务器
服务器类型
WebSphere Application Server
WebSphere MQ 服务器
Web 服务器
应用程序
服务
资源
安全性
环境
系统管理
用户和组
监视和调整
故障诊断
服务集成
UDDI
单元 = fengzhiyin-pcNode01Cell，概要文件 = AppSrv01
关闭页面
应用程序服务器
应用程序服务器
使用此页面来查看环境中的应用程序服务器列表以及其中每个服务器的状态。还可以使用此页面来更改特定应用程序服务器的状态。
首选项
名称
节点
主机名
版本
您可以管理以下资源：
server1
fengzhiyin-pcNode01
fengzhiyin-pc
Base 8.5.5.3
总数 1
帮助
字段帮助
要获取字段帮助信息，请在显示帮助光标时选择字段标签或列表标记。
页面帮助
关于此页面的更多信息
命令辅助
查看上一个操作的脚本编制命令


应用程序服务器 > server1
使用此页面来配置应用程序服务器。
运行时
配置
常规属性
名称
server1
节点名
fengzhiyin-pcNode01
以开发方式运行
并行启动
在需要时启动组件
访问内部服务器类
允许
特定于服务器的应用程序设置
类装入器策略
多个
类装入方式
类已装入并且是先使用父类装入器
应用
确定
复位
取消
容器设置
会话管理
SIP 容器设置
Web 容器设置
Portlet 容器设置
EJB 容器设置
容器服务
业务流程服务
应用程序
已安装的应用程序
服务器消息传递
消息传递引擎
消息传递引擎入站传输
WebSphere MQ 链路入站传输
SIB 服务
服务器基础结构
Java 和进程管理
类装入器
进程定义
进程执行
管理
Java SDK
通信
端口
消息传递
已启用通信的应用程序 (CEA)
性能


应用程序服务器 > server1 > 进程定义
使用此页面来配置进程定义。进程定义用于定义启动或初始化进程所必需的命令行信息。
配置
常规属性
可执行名称
可执行参数
启动命令
启动命令参数
停止命令
停止命令参数
工作目录
${USER_INSTALL_ROOT}
可执行目标类型
JAVA_CLASS
可执行目标
com.ibm.ws.runtime.WsServer
应用
确定
复位
取消
其他属性
Java 虚拟机
环境条目
进程执行
进程日志
记录和跟踪



安全性（Java 2 Security）配置

如果您使用的是Java 2 Security或WebSphere管理安全性, 需要授予tingyun目录下所有jar文件的执行权限。可以通过修改java.policy文件启用授予全局安全性，也可以修改某个Server的server.policy文件只授权单个Server的权限：

授权所有Server

- 修改java.policy文件,文件路径大概为：

```
WAS_HOME/java/jre/lib/security/java.policy
```

- 将如下内容添加到java.policy中， `file:` 后面的路径必须指定到tingyun-agent-java.jar的目录，并确保最后存在 `-` 。

```
grant codeBase "file:/path/to/tingyun/-" {
        permission java.security.AllPermission;
        permission java.net.SocketPermission "*.networkbench.com", "connect,accept,resolve";
        permission java.lang.RuntimePermission "createClassLoader";
        permission java.lang.RuntimePermission "getClassLoader";
};
```

- 重启应用WebSphere

授权单个Server

- 修改server.policy文件,文件路径大概为：

```
WAS_HOME/AppServer/profiles/APP_SERVER_NAME/properties/server.policy
```

- 将如下内容添加到java.policy中， `file:` 后面的路径必须指定到tingyun-agent-java.jar的目录，并确保最后存在 `-` 。

```
grant codeBase "file:/path/to/tingyun/-" {
        permission java.security.AllPermission;
        permission java.net.SocketPermission "*.networkbench.com", "connect,accept,resolve";
        permission java.lang.RuntimePermission "createClassLoader";
        permission java.lang.RuntimePermission "getClassLoader";
};
```

- 重启应用WebSphere

> 提示：每个Server最终的安全性都取决于`java.policy`和`server.policy`的并集，切勿在多个文件配置同样的授权。

policy文件的配置格式及语法可参考Default Policy Implementation and Policy File Syntax

WildFlay installation for Java

Domain mode

修改domain/configuration/domain.xml里的<jvm-options />增加<option />:

```
<server-group name="main-server-group" profile="full">
  <jvm name="default">
    <jvm-options>
      <option value="-javaagent:/path/to/tingyun-agent-java.jar"/>
    </jvm-options>
  </jvm>
</server-group>
```

如果你在Windows使用，路径使用的是斜杠'/'。例如： `d:/tingyun/tingyun-agent-java.jar`

Standalone mode

操作系统	描述
Unix / Mac OS	在bin/standalone.conf文件的底部, 添加: `JAVA_OPTS="$JAVA_OPTS -javaagent:/full/path/to/tingyun-agent-java.jar"`

Windows	查找bin/standalone.bat文件中得如下内容:
	`rem Setup JBoss specific properties`
	在改行之后添加
	`set "JAVA_OPTS=-javaagent:C:/tingyun/tingyun-agent-java.jar %JAVA_OPTS%"`

无容器安装

```
java -javaagent:/path/to/tingyun/tingyun-agent-java.jar -jar xxx.jar
```

注：-javaagent 一定要放置在 -jar 之前。

升级探针

- 登陆到http://report.tingyun.com
- 下载最新的探针
- 使用zip文件里的tingyun-agent-java.jar替换当前使用的文件
- 重启Application Server

应用自动命名

Java Agent 通过应用名称归类应用性能数据。

在单一的JVM实例上，Java Agent将所有性能数据（Web应用过程、数据库、NoSQL、错误、JVM、后台任务等）汇总到tingyun.properties文件中通过`nbs.app_name`指定的应用名称上。修改tingyun.properties文件里`nbs.auto_app_naming`为true后, 应用将根据其context, filter, servlet, 或者request attribute自动命名为不同的名称。

自动命名适用于单个JVM上运行多个应用，Java Agent会将Web应用过程、数据库、NoSQL、错误、JVM数据上传到不同的应用。但是后台任务依然会被上传到tingyun.properties文件配置的应用名称上。.

备注: 所有配置改动都需要重启JVM才能生效。

目录

应用命名规则

当设置 `nbs.auto_app_naming=true` 时,Java Agent使用如下规则命名应用:

数据	命名规则
后台任务	配置在tingyun.properties里的默认名称
Web应用过程	优先级: • Request attribute (高优先级) • Servlet init parameter • Filter init parameter • Web app context parameter • Web app context name (display-name) • Web app context path (低优先级)

Request attribute

Request attribute `APPLICATION_NAME`的优先级最高，请尽早的再Web应用过程中设置改属性，如果多次调用，最后一次设置会覆盖之前的设置。

备注: 只有设置在ServletRequests上`APPLICATION_NAME` 才会生效。

根据request URI设置应用名称:

```
  protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletExcept
ion, IOException {
    ...
   String requestUri = httpServletRequest.getRequestURI();

   if (requestUri.startsWith("/your-owner-uri/")) {
       request.setAttribute("com.tingyun.agent.APPLICATION_NAME", "CustomApplicationName");
    }
```

```
    ...
```

Servlet init parameter

配置web.xml 中得 init parameters实现对应用的命名:

```
<servlet>
 <servlet-name>PayServlet</servlet-name>
 <servlet-class>com.example.PayServlet</servlet-class>
 <init-param>
   <param-name>com.tingyun.agent.APPLICATION_NAME</param-name>
   <param-value>OnlinePay</param-value>
 </init-param>
</servlet>
```

Java Agent通过调用如下方法来获取init-param内容

```
javax.servlet.ServletConfig#getInitParameter(String)
```

参数为`com.tingyun.agent.APPLICATION_NAME`.

如果有多个Servlet，第一个servlet的init-param生效. 如果没有指定init-param 使用默认的应用名称。

Filter init parameter

如果没有使用Servlet, 通过Filter的 init parameter 命名应用:

```
<filter>
  <filter-name>ProductFilter</filter-name>
  <filter-class>com.sample.ProductFilter</filter-class>
  <init-param>
    <param-name>com.tingyun.agent.APPLICATION_NAME</param-name>
    <param-value>ProductApplication</param-value>
   </init-param>
</filter>
```

Java Agent通过向`javax.servlet.FilterConfig#getInitParameter(String)`传递`com.tingyun.agent.APPLICATION_NAME`参数获取init-param内容

如果有多个filter，第一个filter的init-param生效。

Context parameter

使用context param命名应用:

```
<context-param>
  <param-name>com.tingyun.agent.APPLICATION_NAME</param-name>
  <param-value>MyWebApp</param-value>
</context-param>
```

Java Agent通过向`javax.servlet.ServletContext#getInitParameter(String)`传递`com.tingyun.agent.APPLICATION_NAME`参数获取context-param内容

Display name

配置web.xml文件的display-name元素命名应用:

```
<display-name>MyApplicationName</display-name>
```

Java 探针通过调用`javax.servlet.ServletContext#getServletContextName()`获取display-name内容。

Context path

如果没有配置display-name, 其他一些高优先级的属性也没有设置，Java Agent通过调用`javax.servlet.ServletContext#getContextPath()`获得context path，来命名应用过程。

Context path一般作request URI的一部分，且在最前面。例如：如果一个url类似这样：http://www.sample.com/tingyun-agent/testHttpClient:

- Request URI是 /tingyun-agent/testHttpClient.
- Context path 是 /tingyun-agent.
- Application 会被命名为 tingyun-agent.

听云Browser

Java Agent 通过对Web页面嵌入js代码监测页面性能数据。

应用Java Agent 实现页面监测功能，需要：

- 1、确认升级java agent到最新版本(1.3.0或者更高版本)；
- 2、将tingyun-agent-api.jar添加到web应用的classpath中；

备注:tingyun-agent-api.jar的Maven坐标：

```
<dependency>
  <groupId>com.tingyun</groupId>
  <artifactId>tingyun-agent-api</artifactId>
  <version>1.0.0</version>
</dependency>
```

目录

自动页面嵌码

Java Agent对tomcat(5-8),glassfish(3-4),jetty(7-9)支持自动页面嵌码

登陆听云Browser： `https://report.tingyun.com/browser` ，新建支持自动页面嵌码的应用。

jsp页面必须存在`head`标签，探针将会在head标签中嵌入监测页面加载时间的js代码，如：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head> tinyun script code.....</head>
 <body>...</body>
</html>
```

手动页面嵌码

Java Agent 对非tomcat、glassfish、jetty容器，支持采取tingyun api的方式进行页面嵌码

在页面中添加类似代码 `com.tingyun.api.agent.TingYun.getRUMHeader()` ；详情如下：

JSP页面的手动嵌码:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head>
 <%=com.tingyun.api.agent.TingYun.getRUMHeader()%>
 </head>
 <body>...</body>
</html>
```

Servlet等生成html方式:

```
response.setContentType("text/html;charset=utf-8");
```

```
......
PrintWriter out = response.getWriter();
out.write(com.tingyun.api.agent.TingYun.getRUMHeader());
......
```

Velocity等模板引擎方式:

后端代码:

```
......
VelocityEngine velocityEngine = new VelocityEngine(properties);
VelocityContext context=new VelocityContext();
context.put("tingyunRumCode",com.tingyun.api.agent.TingYun.getRUMHeader());
.....
```

*.vm中:

```
<html>
 <head>
 ......
 $tingyunRumCode
 ......
 </head>
 <body>...</body>
</html>
```

JSF的支持:

```
......
<head>
<%=com.tingyun.api.agent.TingYun.getRUMHeader()%>
</head>
......
```

配置SSL证书

- 备份默认证书
  `cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/cacerts_bak`
- 下载听云证书
  `cd /opt/tingyun;wget http://download.tingyun.com/certs/networkbench.cer`
- 合并证书
  `keytool -import -alias networkbench -keystore $JAVA_HOME/jre/lib/security/cacerts -file networkbench.cer`
  默认密码：changeit
- 重启Application Server

卸载探针

- 1)、将备份的启动脚本回退到原来的名称
- 2)、删除tingyun目录
- 3)、重启应用服务器

通过XML文件自定义监控

通过XML文件，可以在不修改代码的前提下监控指定方法的性能。XML文件需要指定具体的类和方法的描述。

在探针启动的时候加载定义的XML，实现对特定类的方法的嵌码。

目录

文件结构

- extension.xsd xml schema。
- sample.xml 示例。

sample.xml说明请参考文件说明或者xml自定义监控示例

验证

2种方式:

- 使用浏览器打开
- 使用一些在线的xml文件验证器

文件路径

Java Ageng在启动的时候读取和解析xml文件。有两种方式可以指定xml文件路径:

文件路径	步骤
创建extensions目录	1. 在tingyun-agent-java.jar同级目录创建extensions目录; 2. 拷贝sample.xml到extensions目录; 3. 确认在tingyun.properties文件里没有设置nbs.extensions.dir。
通过tingyun.properties指定已经存在的目录	1. 修改tingyun.properties文件，配置nbs.extensions.dir属性，指定到xml文件路径; 2. 确认指定的目录存在扩展名为.xml的文件存在。

验证文件是否被加载

为了验证文件是否被加载, 首先需要设置tingyun.properties文件的nbs.agent_log_level=debug。

1. 编辑tingyun.properties修改日志级别属性

   ```
   nbs.agent_log_level=debug
   ```

2. 启动或者重启Application Server

如果读取成功, logs/tingyun_agent.log 会打印类似如下内容:

```
Reading custom extension file /opt/tingyun/extensions/sample.xml
```

如果没有出现类似内容，可能是文件找不到，或者是xml文件检验失败，请检查文件路径以及文件合法性。

多个文件

如果存在多个xml文件

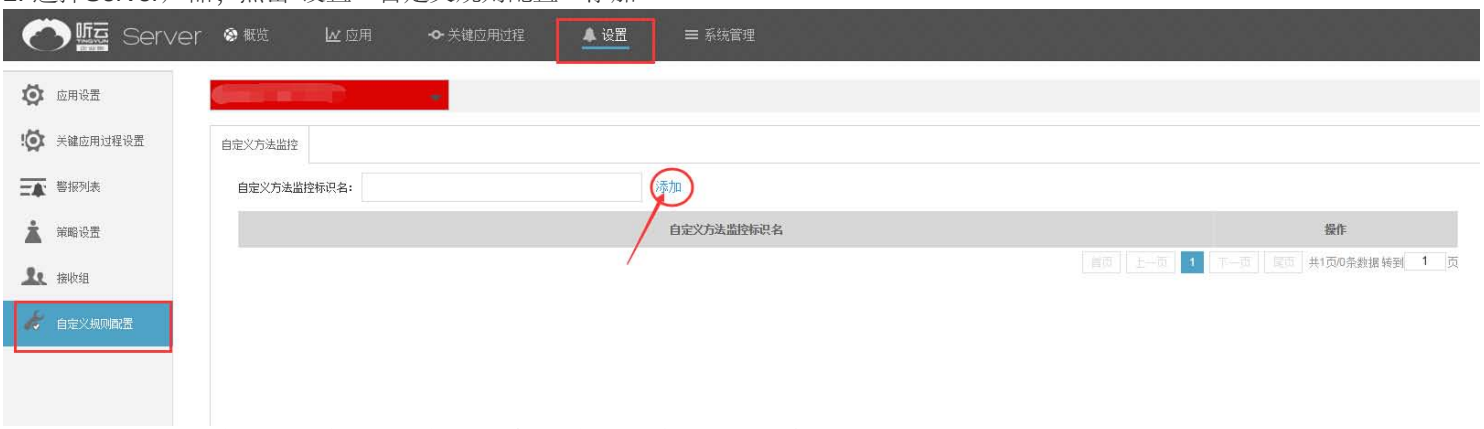
- 如果两个文件的extensionName相同，`version`数值大的生效，数值小的会被忽略
- 如果两个文件的extensionName相同，`version`也相同，则第一个被加载的文件会生效，其他文件不会生效。

通过报表端自定义监控

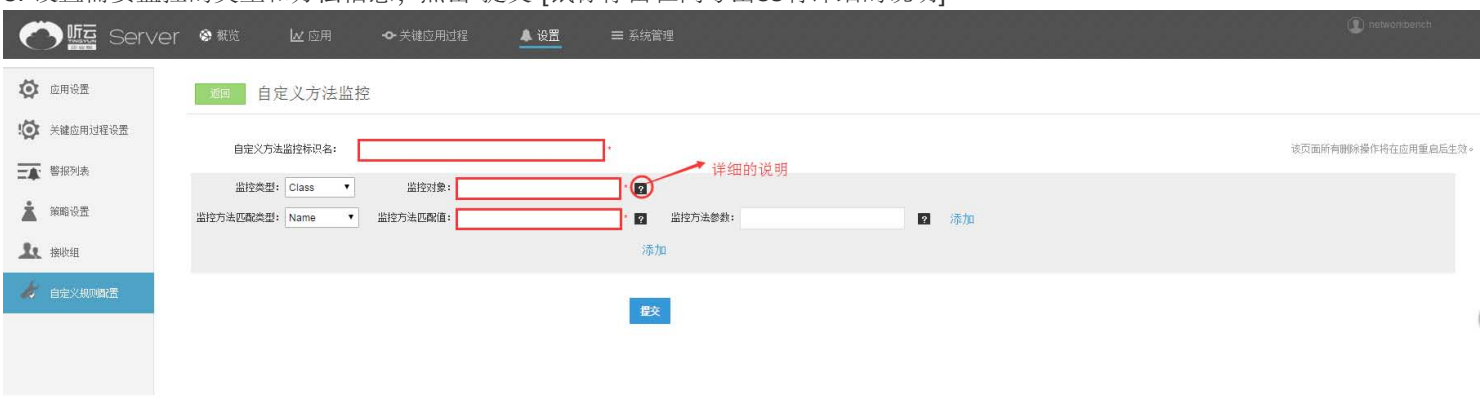
通过在报表端配置，可以在不修改代码，不重启容器的前提下增加监控指定方法的性能。

报表端配置：

- 1. 登录到http://report.tingyun.com/
- 2. 选择Server产品，点击“设置->自定义规则配置->添加”



- 3. 设置需要监控的类型和方法信息，点击“提交”[鼠标停留在问号出3s有详细的说明]



报表端配置元素说明

监控类型 和 监控方法类型必须均填写，方可起作用。

监控类类型相关

- 监控类型选择Class时，方框处填写Class全路径名(com.tingyun.custom.InstrumentClass)，监控填写的指定的Class类;
- 监控类型选择Interface时，方框处填写Interface全路径名，监控指定Interface的所有实现类;
- 当监控类型为BaseClass时，方框处填写BaseClass全路径名，监控指定的BaseClass类及其实现类;

监控方法类型相关

- 监控方法匹配类型选择Name时，方框处填写监控方法名，此时需填写方法参数：填写监控方法的参数全名，参数的顺序必须和方法的参数顺序一致，以英文逗号（,）分隔[注：基础类型可直接填写，如int则直接填写int即可]。
  - 如果需要监控指定方法名的的所有方法（无论是否有参数），此处不需填写。
  - 如果指定监控的方法没有参数，此处需要填写为 [] 即可。
- 当监控方法类型选择ReturnType时，方框处填写返回类型的全路径名，void返回值请选择按方法名进行自定义监控；此时对所有返回该ReturnType的方法均添加监控。
- 当监控方法类型选择Annotation时，方框处填写指定的Annotation全路径名，此时对所有添加该Annotation的方法均添加监控。

问题描述

在报表里看不到数据

解决思路

正常情况下，当你的应用有request后，约5分钟左右，数据既可以在报表中心查看。如果没有数据，请按照以下步骤查找问题：

1. 是否已经安装探针，如果没有安装，请参考第2节安装
2. 安装完探针是否重启了应用服务器，如果没有重启，请重启
3. 确保tingyun-agent-java.jar 和tingyun.properties在同一级目录
4. 执行ps -ef | grep tingyun 是否有内容，如果没有内容，请检查1、2、3步。
5. 在tingyun目录是否存在logs/tingyun_agent.log文件，如果不存在请检查1、2、3、4
6. 将日志级别改为debug，检查tingyun_agent.log文件内容，看是否有报错，如果有报错，根据错误提示做相应修改
7. 确定你的应用是否被禁用或者删除，如果出现如下日志，请检查应用配置，或联系我们的技术支持。 `TingYun Agent disabled for: Java Application`
8. 验证你的应用数据是否上传到正确的应用名称上，在tingyun_agent.log文件里会出现以下信息： `Agent 7864460@cib01122/Java Application connected to dc1.networkbench.com` 如果你将数据上传到多个应用上，则以上信息会出现多条。如果显示的不是正确的引用，请检查是否开启应用自动命名，如果开启自动命名请确保在tingyun.properties中配置nbs.app_name和web.xml的display-name相同

版权申明

北京基调网络股份有限公司

www.tingyun.com

总部地址：北京市朝阳区京顺路5号曙光大厦C座207室

邮编：100028

联系电话：010-84440086