One Pager: GlassFish v3 Monitoring

Table of Contents

1. Introduction

1.1. Project/Component Working Name

GlassFish v3 Monitoring

1.2. Name(s) and e-mail address of Document Author(s)/Supplier

Prashanth Abbagani: prashanth.abbagani@sun.com
Sreenivas Munnangi: sreenivas.munnangi@sun.com
Jennifer Chou: jennifer.chou@sun.com

1.3. Date of This Document

04/08/2009
07/29/2009: Sreeni added 4.1.6.2. on asadmin enable-monitoring/disable-monitoring

2. Project Summary

2. Project Summary

2.1. Project Description

During GlassFish v3 Prelude, basic infrastructure was provided for monitoring based on light weight probes. In this release the infrastructure will be enhanced to use BTrace for byte code instrumentation, enable the developers to instrument their modules using probes and listeners, expose the module's probes as DTrace probes, provide for adhoc client scripting, expose monitoring statistics as JMX mbeans and REST API.

2.2. Risks and Assumptions

- Risks: Availability of resources to complete the planned tasks.
- Assumptions:

3. Problem Summary

3.1. Problem Area

Enable monitoring of the complete system by encouraging all modules to provide probe and listener infrastructure so that one can monitor the entire transaction right from start to end in a uniform manner. Expose the same probes as DTrace probes for better integration with Solaris DTrace which provides for a uniform experience on Solaris platforms. For adhoc monitoring on other platforms (including solaris), provide client scripting based on javascript to start with. Similar to v2, expose the monitoring data as mBeans based on GMbal infrastructure. Also expose the monitoring data as REST for better alignment with cloud computing trend. In addition we need to support asadmin list and get along with asadmin monitor for backward compatibility.

3.2. Justification

Better alignment with well known monitoring tools like DTrace thus providing uniform monitoring experience. Enable the modules provide its monitoring probes and listeners thus decoupling v2 based centralized monitoring. Leverage the developer knowledge of scripting by providing client scripting for adhoc monitoring which has been a gap for a long time. Expose monitoring data through REST for alignment with web services, soa and cloud computing. Also expose the monitoring statistics as JMX/AMX mBeans, and as dotted names through CLI as done in v2.

4. Technical Description

4.1. Details

Main components of the monitoring infrastructure are Probe Provider, Probe Listener, DTrace, Client Scripting container, CLI, GUI, REST, and JMX/AMX which are discussed below.

4.1.1. Probe Provider and Probe Listener

These are the core components of monitoring architecture. Modules are the probe providers who emit probes/events based on certain actions. The probe listeners act as observers to the generated probes and creates/updates corresponding monitoring statistics. The statistics are then exposed through several clients depending on the need. The clients include DTrace, Client Script, REST, JMX, CLI, GUI.

Complete design details including architecture diagram are provided in Making a module monitorable in GlassFish v3.

4.1.5. DTrace

Initially the module providers are registered with Monitoring Infrastructure for all the probes exposed by the module. Subsequently, whenever the probe is fired, the probe infrastructure raises DTrace probe using BTraceUtils.raiseDTraceProbe() method. Since the DTrace is available on solaris platforms, a check is made for platform before raising the DTrace probe. BTrace uses native code to raise DTrace probe and this will be much

enhanced in jdk7. As we have to cater to jdk6 users, we plan to use BTrace.

The generated DTrace probes will appear in the format <provider>:<module>:<name> for example glassfish:web:servletStart. It should be possible to get the list of DTrace probes using dtrace -l command. However corresponding infrastructure seems to be available only in jdk7 and until that time, we could use asadmin list-probes which is needed for client-scripting.

4.1.5. Client Scripting

Script savvy developers can write scripts for adhoc monitoring, deploy and run it on the fly similar to DTrace. The advantage of client scripting is that it can be run on any platform. The client scripting container provides for handling the deployment and life cycle of scripts. Key feature of client script is that it notifies the client in an asynchronous manner by pushing the result back to client instead of client pulling it. Comet provides good support for pushing data back to clients in an asynchronous manner. The list of available probes can be obtained through GUI or using asadmin list-probes. Goals provides for entering the script, deploying and running the script and visualizing the results on the same page.

4.1.6. Command Line Interface (CLI)

As explained earlier, the asadmin get and list commands (with --monitor=true option to differentiate from the config get/list commands) will automatically expose the monitoring data from the monitoring tree. The monitoring tree hierarchy is populated by the statistic objects (in StatsProvider).

To support Client Scripting and DTrace, certain additional commands were suggested and discussed in the appropriate sections.

4.1.6.1. asadmin monitor

Using monitor pluggability, it is possible to monitor the data continuously based on an interval, similar to top command in unix.

This command displays commonly monitored statistics for Enterprise Server components and services. Monitoring output is displayed continuously in a tabular format; the --interval option can be used to display output at a particular interval (default 30 seconds).

```
C:\glassfishV3\v3-prelude\v3_prelude_release\glassfish\bin>asadmin monitor --type=webmodule server
asc  ast   rst   st    ajlc  mjlc tjlc   aslc    mslc      tslc
0    0     0     1     0     0    0      8       8         8

C:\glassfishV3\v3-prelude\v3_prelude_release\glassfish\bin>asadmin monitor --type=jvm server
                        JVM Monitoring
UpTime(ms)                      Heap and NonHeap Memory(bytes)
current                 min           max           low       high      count
11623062                8585216     167313408  0         0         38531072

C:\glassfishV3\v3-prelude\v3_prelude_release\glassfish\bin>asadmin monitor --type=httplistener server
ec    mt       pt      rc
2     20234 288.27 94
```

4.1.6.2. asadmin enable-monitoring/disable-monitoring

The monitoring funcationality including attaching btrace-agent is done based on the 'monitoring-enabled' attribute of 'monitoring-service' element. If monitoring-enabled is true then the btrace-agent is attached as part of startup. When monitoring-enabled is false, btrace-agent is not attached at startup time. However when user changes monitoring-enabled to true while the server is running, it should be possible to attach the btrace-agent and start monitoring functionality.

Purpose of this pair of commands is to provide enable /disable monitoring during run time without having to restart the server (Alternatively user should be able to use asadmin set command to enbale/disable the monitoring-enabled flag, but have to restart the server to take effect). It does attach btrace-agent based on the given pid and optionally sets the monitoring level for given modules.

enable-monitoring

```
enable-monitoring [--mbean=false] [--dtrace*=true] [--level web-container="LOW":ejb-
container="HIGH"] [--options="debug=true"] [--pid=<pid>]
```

- `enable-monitoring`
  sets the attribute 'monitoring-enabled' to 'true'
- `enable-monitoring --mbean=true --dtrace*=false`
  sets the attribute 'monitoring-enabled' to 'true', mbean-enabled to true and dtrace-enabled to false
- `enable-monitoring --options="debug=true" --pid=<pid>`
  sets the attribute 'monitoring-enabled' to 'true' and attaches btrace agent using --options
- `enable-monitoring --level web-container="LOW":ejb="HIGH"`
  sets the levels for given modules in addition to 'monitoring-enabled'

disable-monitoring

```
disable-monitoring --modules="web-container,ejb-container"
```

- `disable-monitoring`
  sets the attribute 'monitoring-enabled' to 'false'
- `disable-monitoring --modules="web-container,ejb-container"`
  this command will just set the levels for given modules to 'OFF' and it **does not change the value for 'monitoring-enabled'**

*- Available as a value-add feature, made available only to the paid customers.

Above also caters an important use case of adhoc monitoring, i.e. turning monitoring on in production while server is running, for ex. enable dtrace on the fly.

4.1.7. Graphical User Interface (GUI)
A monitoring UI page for each of the Monitorable Component listed in section 4.1.1 will be shown in the Admin Console. Please refer to Section 4.1 of the Admin GUI Functional Spec for more details.
GUI provides support for client scripting as described in 4.1.5

4.1.8. REST
To enable better integration with REST clients, web services, SOA and cloud computing, the monitoring data is exposed using REST infrastructure. The monitoring statistic objects are exposed appropriately using standard REST path notation. As part of the publish API of the Monitoring statistic tree, we also notify the REST infrastructure to expose these objects as REST automatically.

4.1.9. JMX/AMX
The infrastructure provide by GMbal and AMX will be used to expose monitoring data as mBeans. The monitoring statistic data objects will be annotated with GMbal annotations as described above to expose the corresponding monitoring statistics as mBeans. Unlike v2, the mBeans expose the statistics data as opendata instead of proper Statistic type. Need to explore GMbal infrastructure to see if it is possible to convert opendata back to Statistic type.

4.2. Bug/RFE Number(s)

// List any Bug(s)/RFE(s) which will be addressed by this proposed change.
// Provide links to the Issue tracker Bug(s)/RFE(s)where possible

4.3. In Scope

The modules which need to provide probes and listeners are WEB, Grizzly, EJB, JMS, IMQ, JPA, JTS, Jersey, ORB, Metro, JRuby, JVM

4.4. Out of Scope

// Aspects that are out of scope if not obvious from above.

4.5. Interfaces

4.5.1 Exported Interfaces

4.5.1.1. gfprobe-provider-client.jar

Interface	Stability	Former Stability (if changing)	Comments
org.glassfish.flashlight.provider.annotations. @ProbeProvider(providerName="glassfish", moduleName="web") @Probe(name="jspLoadedEvent") @ProbeParam("jsp") String jsp			

4.5.1.2. probe-provider.xml

```
<probe-provider name="glassfish" module="web" >
   <probe name="jspLoadedEvent"
     <class>com.sun.enterprise.web.jsp.JspProbeEmitterImpl </class>
     <method>jspLoaded</method>
     <signature>void (javax.Servlet,String,String)</signature>
     <probe-param type="javax.Servlet" name="jsp"/>
     <probe-param type="String" name="appName" />
     <probe-param type="String" name="hostName"/>
     <return-param type="void" />
     <location>RETURN/ENTRY/EXCEPTION</location>
   </probe>
</probe-provider>
```

4.5.1.3. META-INF/MANIFEST.MF

probe-provider-class-names : <provider-class-name[,provider-class-name]>
(and/or)
probe-provider-xml-file-names : <path of xml file name w.r.t root [, xml file name]>

4.5.1.4. probe-listener-api.jar

MonitoringFramework.register("jsp-container", "applications/app1/jsp1", new JspStatsProvider());
GlassFishConfigChange interface to receive callbacks about config changes.

4.5.1.4. Client Scripting

ScriptContainer.registerListener('web:webmodule::webModuleStartedEvent', params, 'webModStart');
ScriptContainer.print(String);
ScriptContainer.println(String);
asadmin list-probes <provider>:<module>:<name>, we plan to support regex in future.
asadmin deploy --type clientscript /home/jspLoaded.js
asadmin run-script --script=jspLoaded.js

4.5.1.4. DTrace

Probes will be exposed as <glassfish>:<module>:<event>

4.5.2 Imported interfaces

Interface	Stability	Exporting Project: Name, Specification or	Comments

		other Link.	
GMbal AMX REST Statistics BTrace			

4.5.3 Other interfaces (Optional)

Interface	Stability	Exporting Project: Name, Specification or other Link.	Comments

4.6. Doc Impact

User visible part of monitoring, CLI commands and GUI screens need to be documented. Also need to describe how the customer can implement monitoring in his (3rd party) module using our monitoring infrastructure.

4.7. Admin/Config Impact

There will be configuration information for the monitoring elements that needs to be persisted in the domain.xml. The CLI and GUI clients are written by monitoring and GUI teams for such configuration. There is a dependency on configuration pluggability for configuring the monitoring level and to set the enable status.

4.8. HA Impact

The monitoring data tree which is represented in memory will be lost if the machine becomes unavailable.

4.9. I18N/L10N Impact

// Does this proposal impact internationalization or
// localization?

4.10. Packaging & Delivery

Please refer to Monitoring design document.

4.11. Security Impact

Monitoring infrastructure uses OSGi/HK2 thus uses security provided for these containers.
Client Scripting exposed thru GUI need to be guarded appropriately by using secure protocols.
DTrace works similar to Solaris DTrace.

4.12. Compatibility Impact

Based on the available information, we will be compatible for dotted names which are exposed through asadmin list/get and asadmin monitor. As pointed in 4.1.9, GMbal exposes mBeans using opendata which is not fully compatible with v2. If this is not adequate then we need to provide a way to convert opendata to proper Statistic type.

4.13. Dependencies

<>Pl. see 4.5.2 on imported interfaces and 4.3 for Inscope components to provide monitoring functionality.

5. Reference Documents

- Making a module monitorable in GlassFish v3
- DTrace
- Pluggability Funtional specification
- REST infrastructure
- GMbal
- AMX
- CLI and AMX specification

6. Schedule

6.1. Projected Availability

Java EE 6 release.