CrossGrid Developer Manual Guide

JIMS - the JMX-based Infrastructure Monitoring System

Task 3.3.3 - JIMS

Document Filename:	**CG-DeveloperManual**
Workpackage:	**Task 3.3.3 - JIMS**
Partner(s):	**CYF**
Lead Partner:	**CYF**
Config ID:	**cg-developermanual-v1.5.20**
Document classification:	**PUBLIC**

Abstract: This is the skeleton/example for a developer manual for the software developed within the CrossGrid project.

Delivery Slip

	Name	Partner	Date	Signature
From	Piotr Nowakowski	FZK	Nov 2004	
Verified By				
Approved By				

Document Log

Version	Date	Summary of changes	Author
0.1	Nov 3rd, 2004	First draft version	Piotr Nowakowski

Contents

Copyright Notice

This research is partly funded by the European Commission IST-2001-32243 Project CrossGrid.

1 Introduction

JIMS (The JMX-based Infrastructure Monitoring System) is designed to monitor grid infrastructure parameters, ie.:

1. parameters of worker node host platform:
   - CPU load, memory and disk usage
2. parameters of worker node network interfaces
3. network resources and its condition:
   - ICMP packets latency,
   - UDP packets latency,
   - throughput measured using UDP packets.

1.1 Abbreviations and Acronyms

```
JIMS - The JMX-based Infrastructure Monitoring System
JMX  - Java Management Extensions
SG   - SOAP Gateway
GDS  - Global Discovery Service
WN   - Worker Node, execution host in the terms of grid engines
CE   - Computing Element, master host in the terms of grid engines
SE   - Storage Element, host with NFS service for WNs and CE
NFS  - Network File System
```

1.2 References and Source Code

JIMS is available at following locations on FZK web server (see installation guide for details):

- http://savannah.fzk.de/cgi-bin/viewcvs.cgi/crossgrid/crossgrid/wp3/wp3_3-moninfr/wp3_3_3-jims/ - sources in CVS repository
- http://savannah.fzk.de/distribution/crossgrid/autobuilt/i386-rh7.3-gcc3.2.2/wp3/RPMS/ - RPMs
- http://gridportal.fzk.de/websites/crossgrid/cg-wp3-3/wp3_3_3-jims/docs/installguide.pdf - installation guide
- http://gridportal.fzk.de/websites/crossgrid/cg-wp3-3/wp3_3_3-jims/docs/developermanual.pdf - this developer guide

Current version of JIMS is 1.5.20.

2 Implementation Structure

2.1 Product Use Cases

JIMS monitoring system is available as a web service running on the CE host of the cluster at address of the following form:

```
http://hostname:7702/axis/services/SoapGateway
```

where hostname is the name of CE host.

JIMS API provides useful methods for accessing its SOAP Gateway interface by the means of SoapGatewayServiceLocator class. JIMS API is packaged in $CG_LOCATION/share/java/jims-client.jar in cg-jims-client RPM (See installation guide for details and RPM location). Simple usage scenario for obtaining Soap Gateway interface of JIMS is shown below:

```
String sgAddress = "http://" + host + ":" + port + "/axis/services/SoapGateway";
SoapGateway sg = new SoapGatewayServiceLocator().getSoapGateway(new URL(sgAddress));
```

Using SoapGateway interface there can be performed further operations in monitoring system using methods described in the *Interface* section. Method getAttributes() delivers monitoring parameters from all worker nodes (WN-s) in cluster:

```
String parameter = "Uptime";
String mBeanName = "Monitoring:class=SystemInformation";
Object [] attributes = sg.getAttributes(MBeanName, parameter);
```

Available parameters were included in Appendix A. After getAttributes() method execution there are returned values of given MBean attribute for all registered MBeanServers, i.e. Worker Nodes in particular.

There are three monitoring modules:

```
"Monitoring:class=SystemInformation"
"Monitoring:class=SNMPMirror"
"Monitoring:class=NetworkMetrics"
```

Using SoapGateway interface, there can be also invoked methods on chosen MBean. Below there is shown example of "measureICMPLatency", "measureUDPLatency" and "measureThroughput" methods invocation:

```
String sgAddress = "http://" + host + ":" + port + "/axis/services/SoapGateway";
SoapGateway sg = new SoapGatewayServiceLocator().getSoapGateway(new URL(sgAddress));

String[] argTypes  = { "java.lang.String" };
Object[] argValues = { "149.156.9.15"     };
String   networkMetrics = "Monitoring:class=NetworkMetrics";
String[] mbs = sg.scGetMBeanServers();

//interface NetworkMetricsMBean:
```

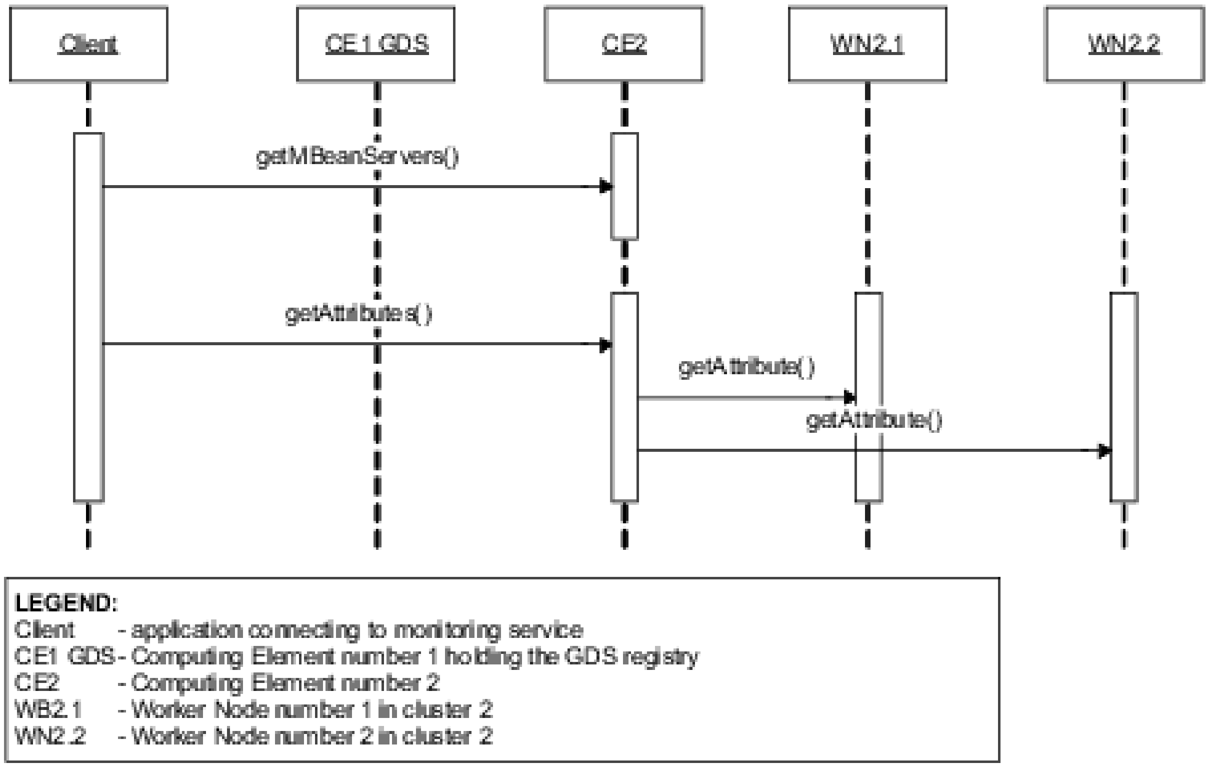
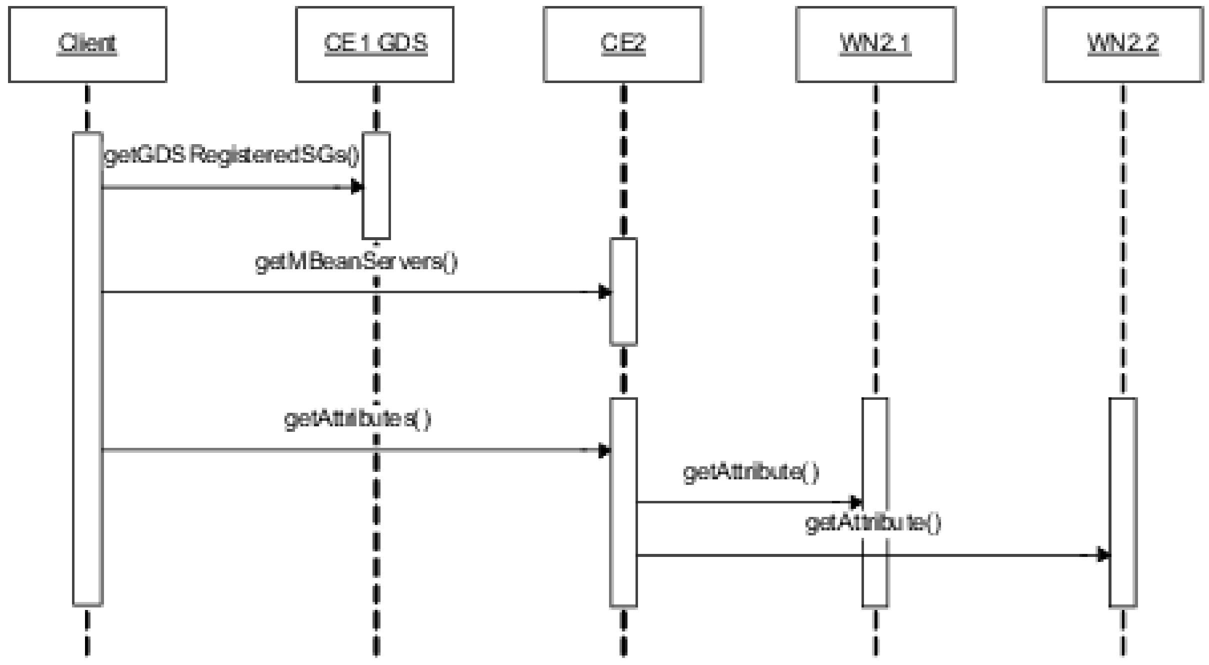

Client
CE1 GDS
CE2
WN2.1
WN2.2
getMBeanServers()
getAttributes()
getAttribute()
getAttribute()
LEGEND:
Client - application connecting to monitoring service
CE1 GDS - Computing Element number 1 holding the GDS registry
CE2 - Computing Element number 2
WB2.1 - Worker Node number 1 in cluster 2
WN2.2 - Worker Node number 2 in cluster 2

Figure 2.1: Reading attributes using JIMS WS interface in SOAP Gateway installed in CE

```
//double measureICMPLatency(String destinationHostIP);
double icmpLatency =
((Double) sg
 .invoke(
 mbs[i],
 networkMetrics,
 "measureICMPLatency",
 argValues,
 argTypes))
.doubleValue();

//interface NetworkMetricsMBean:
//double measureUDPLatency(String destinationHostIP);
double udpLatency =
((Double) sg
 .invoke(
 mbs[i],
 networkMetrics,
 "measureUDPLatency",
 argValues,
 argTypes))
.doubleValue();

//interface NetworkMetricsMBean:
//double measureThroughput(String destinationHostIP);
```



LEGEND:

Client	- application connecting to monitoring service
CE1 GDS	- Computing Element number 1 holding the GDS registry
CE2	- Computing Element number 2
WB2.1	- Worker Node number 1 in cluster 2
WN2.2	- Worker Node number 2 in cluster 2

Figure 2.2: Finding CE using Global Discovery Service installed in well known, chosen CE

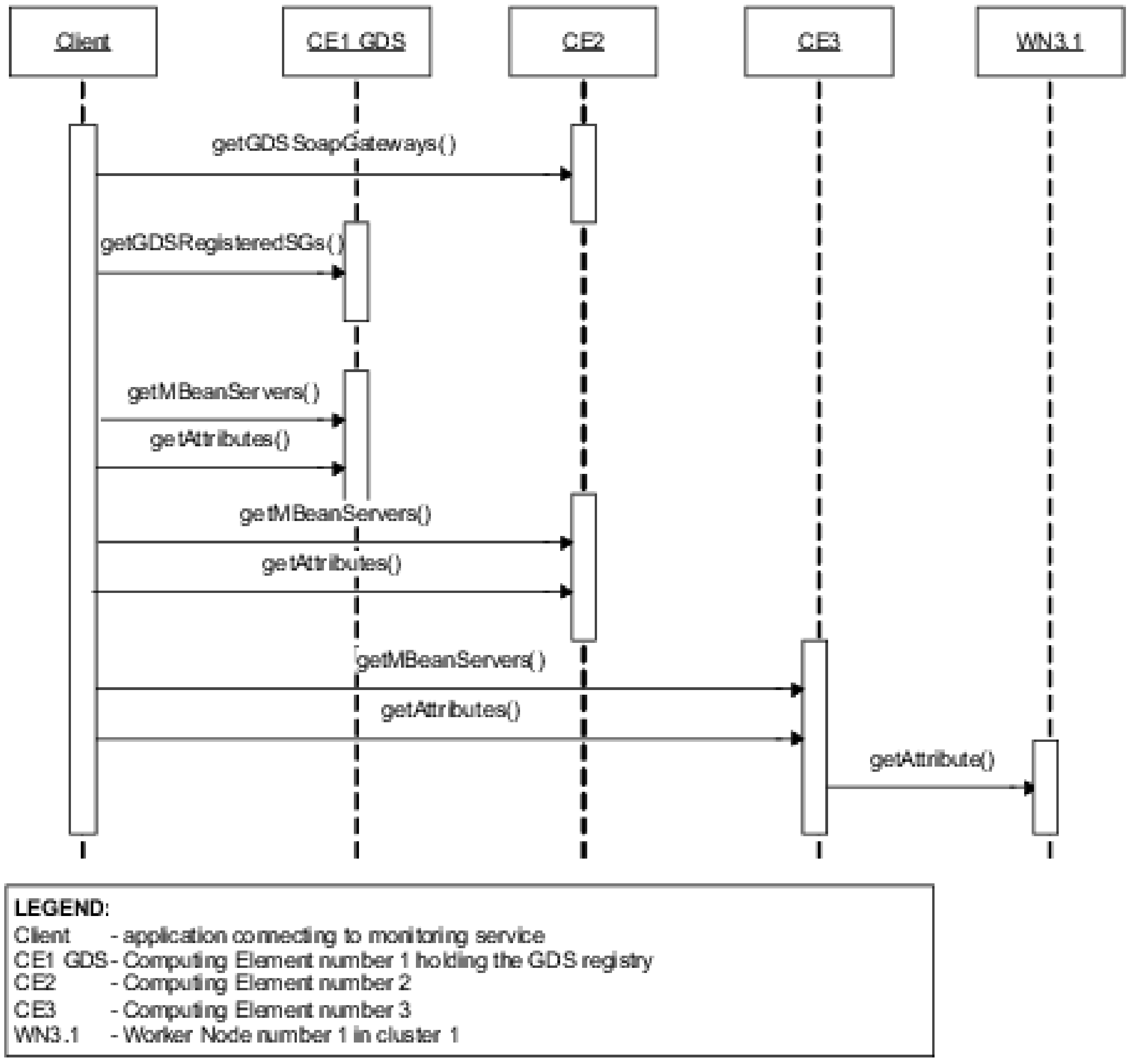

Client
CE1 GDS
CE2
CE3
WN3.1
getGDSSoapGateways( )
getGDSRegisteredSGs( )
getMBeanServers( )
getAttributes()
getMBeanServers()
getAttributes()
getMBeanServers( )
getAttributes()
getAttribute()
LEGEND:
Client - application connecting to monitoring service
CE1 GDS - Computing Element number 1 holding the GDS registry
CE2 - Computing Element number 2
CE3 - Computing Element number 3
WN3.1 - Worker Node number 1 in cluster 1

Figure 2.3: Finding GDS using any CE and reading parameters from another chosen CE

```
double throughput =
((Double) sg
 .invoke(
 mbs[i],
 networkMetrics,
 "measureThroughput",
 argValues,
 argTypes))
.doubleValue();
```

List of all methods available in NetworkMetrics module:

```
public double measureICMPLatency(String destinationHostIP);
public double measureUDPLatency(String destinationHostIP);
public double measureUDPLatency(String destinationHostIP, int attempts,
int requestPacketDataLength, int replyPacketDataLength);
public double measureThroughput(String destinationHostIP);
public double measureThroughput(String destinationHostIP, int attempts,
int requestPacketDataLength, int replyPacketDataLength);
public int getAttempts();
public int getUdpReceiveDataFieldSize();
public int getUdpSendDataFieldSize();
public void setAttempts(int attempts);
public void setUdpReceiveDataFieldSize(int udpReceiveDataFieldSize);
public void setUdpSendDataFieldSize(int udpSendDataFieldSize);
public int getUDPTimeout();
public void setUDPTimeout(int timeout);
public String getPingCommandExecutable();
public void setPingCommandExecutable(String pingCommandExecutable);
```

Default values of some parameters:

```
private int attempts = 10; // cycles send/receive
private int udpSendDataFieldSize = 1000; // [B]
private int udpReceiveDataFieldSize = 1000; // [B]
private int UDPTimeout = 600; // [ms]
```

In order to know the order of returned values, there should be invoked another method

```
sg.scGetMBeanServers()
```

or

```
sg.scGetWorkerNodes()
```

returning JMX RMI addresses of MBeanServers (these addresses include also the IP address) or the list of IP addresses of all registered Worker Nodes. The order of delivered attributes by getAttributes() is always the same and concerns the order provided by the scGetMBeanServers() and scGetWorkerNodes() method.

Below there is presented result using cg-jims-cli JIMS client requesting "Uptime" attribute from "Monitoring:class=SystemInformation" MBean. Values are read using getWorkerNodes() (the IP numbers) and the getAttributes() methods.

```
JIMS>get Uptime
[Monitoring:class=SystemInformation] [Uptime]:
[01] 149.156.9.15: 3024699.2
[02] 149.156.9.16: 3024763.2
[03] 149.156.9.17: 3024632.8
[04] 149.156.9.18: 3024565.5
[05] 149.156.9.19: 861434.0
[06] 149.156.9.20: 3024288.5
[07] 149.156.9.21: 3024299.8
[08] 149.156.9.22: 3024282.5
[09] 149.156.9.24: 3024238.8
[10] 149.156.9.26: 3023981.0
[11] 149.156.9.29: 3023996.5
[12] 149.156.9.42: 3023537.2
```

2.2 Product Component Model

To be determined.

2.3 Detailed Implementation Model

To be determined.

2.4 Product Interfaces

There is one commont interface for JIMS service called SoapGateway which exposes methods for accessing monitoring parameters and for performing other system management operations. The methods for monitoring system management start with sc and the let us for example obtain the current list of MBean Servers connected to the SOAP Gateway.

Management methods:

1. scGetMBeanServers - returns the n MBean Servers - only the ones representing WNs
2. scGetAllMBeanServers - the same as above, provided for backward compatibility

Methods not implemented in current release (planned to be implemented in the future because require proper definition in skeleton of WS binding implementation file to provide adequate operations on JMX objects):

1. scAddMBeanServer - adds MBeanServer to the list of registered servers in SG. This method is used to add the MBeanServer having it's current valid JMX RMI Connector address.
2. scRemoveMBeanServer - removes given MBeanServer from the list of registered servers in SG. Can be used to remove any registered MBeanServer from the list, including the ones discovered and registered by ActiveDiscovery class.
3. scSetDiscovery - administrative function enabling active discovery service at cluster level. If enabled (true), SG automatically periodically sends discovery requests and waits for responses from every existing and reachable WNs.

4. scSetDiscoveryTimeStamp - sets time stamp in discovery mechanism
5. scSetHeartBeatRetries - sets number of retries in heart-beat mechanism used for removal of not existing worker nodes in list of SG.

Methods allowing access to monitoring parameters:

1. invoke - invokes a method
2. isRegistered - checks whether the MBeanServer is registered
3. getAttributes - returns attributes from given MBeanServer
4. getAttributes - returns attributes from all registered MBeanServers
5. getAttribute - return one attribute from given MBeanServer
6. setAttribute - return one attribute from all registered MBeanServers
7. getAttributesWithTimeStamp - return attributes from all registered MBeanServers including time stamp at the end of the list indicating the time the measurements or attributes were taken
8. createMBean - creates MBean
9. createMBean - creates MBean
10. createMBean - creates MBean
11. createMBean - creates MBean
12. getMBeanCount - returns the number of MBeans registered in given MBeanServer
13. getMBeanInfo - returns the MBeanInfo interface of dynamic MBean (see JMX specification)
14. getMBeanAttributeNames - returns the names of MBean attributes
15. getMBeanAttributeTypes - returns the types of MBean attributes
16. getMBeanAttributeDescriptions - returns the descriptions of MBean attributes
17. getMBeanDescription - returns MBean description
18. getMBeanClassName - returns the name of the class of MBean
19. getMBeanConstructorDescriptions - returns descriptions of MBean constructors
20. getMBeanConstructorNames - returns names of MBean constructors
21. getMBeanOperationNames - returns names of MBean's operations
22. getMBeanOperationDescriptions - returns names of MBeans's operations' descriptions
23. getMBeanOperationReturnTypes - returns types returned by MBean operations
24. getMBeanNotificationNames - returns names of MBean notifications
25. getMBeanNotificationDescriptions - returns descriptions of MBean notifications
26. getObjectInstance - returns instance of MBean
27. isInstanceOf - compares the instance of MBean with given Java type
28. queryMBeans - returns the list of MBeans registered in given MBeanServer
29. queryNames - returns the names of MBeans registered in given MBeanServer
30. setAttributes - set attributes given by method parameters
31. unregisterMBean - unregister given MBean

3 Product Testing

4 Contact Information and Credits

5 Appendix A

Table 5.1: JIMS SystemInformation module parameters

Name	Type	Example value	Description
AverageIdle	long	101	CPU idle time 0.01[s]
AverageUser	long	84	CPU user time 0.01[s]
AverageNice	long	0	CPU nice time 0.01[s]
AverageSystem	long	17	CPU system time 0.01[s]
AverageIowait	long	0	CPU iowait time 0.01[s]
AverageIrq	long	0	CPU irq time 0.01[s]
AverageSoftirq	long	0	CPU softirq time 0.01[s]
NcpuIdle	long[]	n.a.	per CPU idle time 0.01[s]
NcpuUser	long[]	n.a.	per CPU user time 0.01[s]
NcpuNice	long[]	n.a.	per CPU nice time 0.01[s]
NcpuSystem	long[]	n.a.	per CPU system time 0.01[s]
NcpuIowait	long[]	n.a.	per CPU iowait time 0.01[s]
NcpuIrq	long[]	n.a.	per CPU irq time 0.01[s]
NcpuSoftirq	long[]	n.a.	per CPU softirq time 0.01[s]
AverageNcpuIdle	long[]	n.a.	per CPU idle time 0.01[s]
AverageNcpuUser	long[]	n.a.	per CPU user time 0.01[s]
AverageNcpuNice	long[]	n.a.	per CPU nice time 0.01[s]
AverageNcpuSystem	long[]	n.a.	per CPU system time 0.01[s]
AverageNcpuIowait	long[]	n.a.	per CPU iowait time 0.01[s]
AverageNcpuIrq	long[]	n.a.	per CPU irq time 0.01[s]
AverageNcpuSoftirq	long[]	n.a.	per CPU softirq time 0.01[s]
Idle	long	115529730	CPU idle time 0.01[s]
User	long	9404497	CPU user time 0.01[s]
Nice	long	446	CPU nice time 0.01[s]
System	long	1013177	CPU system time 0.01[s]
Iowait	long	0	CPU IO wait 0.01[s]
Irq	long	0	CPU irq time 0.01[s]
Softirq	long	0	CPU softirq time 0.01[s]
Itime	float	581874.8	CPU idle time [s]
Cline	java.lang.String	n.a.	kernel options
Cpuinf	java.lang.String	n.a.	CPU information
FileSystemStatistics	java.lang.String[]	n.a.	filesystem usage statistics
Iomap	java.lang.String	n.a.	IO map
L15m	float	0.96	CPU usage during last 15 minutes
L5m	float	1.0	CPU usage during last 5 minutes
L1m	float	1.0	CPU usage during last 1 minute
Maxmem	long	1006	installed physical memory
Maxswp	long	517	installed physical swap
Mem	long	12	free available physical memory
Membuf	long	78	physical memory used as buffers
Memch	long	765	physical memory used as cache
Memsh	long	0	physical memory used as shared memory
Model	java.lang.String	n.a.	CPU model and brand
Ncpus	long	2	number of CPUs
Ndisks	long	1	number of disks
Nproc	long	89	number of processes
Rproc	long	4	number of processes running
Swp	long	492	swap file usage [MB]
TimerPeriod	int	n.a.	period between average values measurements
Type	java.lang.String	n.a.	CPU options
Uptime	float	629740.06	CPU uptime
Ver	java.lang.String	n.a.	kernel version

Table 5.2: JIMS SNMPMirror module parameters

Name	Type	Example value	Description
hostIp	java.lang.String	127.0.0.1	
icmpInAddrMaskReps	java.lang.Long	0	
icmpInAddrMasks	java.lang.Long	0	
icmpInDestUnreachs	java.lang.Long	1933	
icmpInEchoReps	java.lang.Long	7	
icmpInEchos	java.lang.Long	10535	
icmpInErrors	java.lang.Long	0	
icmpInMsgs	java.lang.Long	12475	
icmpInParmProbs	java.lang.Long	0	
icmpInRedirects	java.lang.Long	0	
icmpInSrcQuenchs	java.lang.Long	0	
icmpInTimeExcds	java.lang.Long	0	
icmpInTimestampReps	java.lang.Long	0	
icmpInTimestamps	java.lang.Long	0	
icmpOutAddrMaskReps	java.lang.Long	0	
icmpOutAddrMasks	java.lang.Long	0	
icmpOutDestUnreachs	java.lang.Long	1388	
icmpOutEchoReps	java.lang.Long	10535	
icmpOutEchos	java.lang.Long	0	
icmpOutErrors	java.lang.Long	0	
icmpOutMsgs	java.lang.Long	11923	
icmpOutParmProbs	java.lang.Long	0	
icmpOutRedirects	java.lang.Long	0	
icmpOutSrcQuenchs	java.lang.Long	0	
icmpOutTimeExcds	java.lang.Long	0	
icmpOutTimestampReps	java.lang.Long	0	
icmpOutTimestamps	java.lang.Long	0	
ifAdminStatus	java.lang.Integer[]	n.a.	
ifDescr	java.lang.String[]	n.a.	
ifInDiscards	java.lang.Long[]	n.a.	
ifInErrors	java.lang.Long[]	n.a.	
ifInNUcastPkts	java.lang.Long[]	n.a.	
ifInOctets	java.lang.Long[]	n.a.	
ifInUcastPkts	java.lang.Long[]	n.a.	
ifInUnknownProtos	java.lang.Long[]	n.a.	
ifIndex	java.lang.Integer[]	n.a.	
ifLastChange	java.lang.Long[]	n.a.	
ifMtu	java.lang.Integer[]	n.a.	
ifNumber	java.lang.Integer	2	
ifOperStatus	java.lang.Integer[]	n.a.	
ifOutDiscards	java.lang.Long[]	n.a.	
ifOutErrors	java.lang.Long[]	n.a.	
ifOutNUcastPkts	java.lang.Long[]	n.a.	
ifOutOctets	java.lang.Long[]	n.a.	
ifOutQLen	java.lang.Long[]	n.a.	
ifOutUcastPkts	java.lang.Long[]	n.a.	
ifPhysAddress	java.lang.String[]	n.a.	
ifSpeed	java.lang.Long[]	n.a.	
ifType	java.lang.Integer[]	n.a.	
ipDefaultTTL	java.lang.Integer	64	
ipForwDatagrams	java.lang.Long	0	

Table 5.3: JIMS SNMPMirror module parameters - continued

Name	Type	Example value	Description
ipForwarding	java.lang.Integer	2	
ipFragCreates	java.lang.Long	403168	
ipFragFails	java.lang.Long	0	
ipFragOKs	java.lang.Long	0	
ipInAddrErrors	java.lang.Long	0	
ipInDelivers	java.lang.Long	23923951	
ipInDiscards	java.lang.Long	0	
ipInHdrErrors	java.lang.Long	0	
ipInReceives	java.lang.Long	39426814	
ipInUnknownProtos	java.lang.Long	0	
ipOutDiscards	java.lang.Long	0	
ipOutNoRoutes	java.lang.Long	0	
ipOutRequests	java.lang.Long	35086287	
ipReasmFails	java.lang.Long	0	
ipReasmOKs	java.lang.Long	7743936	
ipReasmReqds	java.lang.Long	23231248	
ipReasmTimeout	java.lang.Long	0	
ipRouteTable	[[Ljava.lang.String;	n.a.	
ipRoutingDiscards	java.lang.Long	0	
numberRetries	java.lang.Integer		
portNumber	java.lang.Integer		
readCommunity	java.lang.String		
snmpEnableAuthenTraps	java.lang.Integer	2	
snmpInASNParseErrs	java.lang.Long	0	
snmpInBadCommunityNames	java.lang.Long	0	
snmpInBadCommunityUses	java.lang.Long	0	
snmpInBadValues	java.lang.Long	0	
snmpInBadVersions	java.lang.Long	0	
snmpInGenErrs	java.lang.Long	0	
snmpInGetNexts	java.lang.Long	28	
snmpInGetRequests	java.lang.Long	50	
snmpInGetResponses	java.lang.Long	0	
snmpInNoSuchNames	java.lang.Long	0	
snmpInPkts	java.lang.Long	75	
snmpInReadOnlys	java.lang.Long	0	
snmpInSetRequests	java.lang.Long	0	
snmpInTooBigs	java.lang.Long	0	
snmpInTotalReqVars	java.lang.Long	70	
snmpInTotalSetVars	java.lang.Long	0	
snmpInTraps	java.lang.Long	0	
snmpOutBadValues	java.lang.Long	0	
snmpOutGenErrs	java.lang.Long	0	
snmpOutGetNexts	java.lang.Long	0	
snmpOutGetRequests	java.lang.Long	0	
snmpOutGetResponses	java.lang.Long	63	
snmpOutNoSuchNames	java.lang.Long	0	
snmpOutPkts	java.lang.Long	61	
snmpOutSetRequests	java.lang.Long	0	
snmpOutTooBigs	java.lang.Long	0	
snmpOutTraps	java.lang.Long	0	
sysContact	java.lang.String	Unknown	

Table 5.4: JIMS SNMPMirror module parameters - continued

Name	Type	Example value	Description
sysDescr	java.lang.String	n.a.	
sysLocation	java.lang.String	crossgrid	
sysName	java.lang.String	zeus05.cyf-kr.edu.pl	
sysServices	java.lang.Integer	0	
sysUpTime	java.lang.Long	63027852	
tcpActiveOpens	java.lang.Long	13517	
tcpAttemptFails	java.lang.Long	0	
tcpConnTable	[[Ljava.lang.String;	n.a.	
tcpCurrEstab	java.lang.Long	6	
tcpEstabResets	java.lang.Long	1891	
tcpInErrs	java.lang.Long	3	
tcpInSegs	java.lang.Long	12381515	
tcpMaxConn	java.lang.Integer	-1	
tcpOutRsts	java.lang.Long	1679	
tcpOutSegs	java.lang.Long	23035415	
tcpPassiveOpens	java.lang.Long	10805	
tcpRetransSegs	java.lang.Long	1689	
tcpRtoAlgorithm	java.lang.Integer	1	
tcpRtoMax	java.lang.Integer	120000	
tcpRtoMin	java.lang.Integer	200	
timeout	java.lang.Integer		
udpInDatagrams	java.lang.Long	11543839	
udpInErrors	java.lang.Long	6	
udpNoPorts	java.lang.Long	1379	
udpOutDatagrams	java.lang.Long	11770616	
udpTable	[[Ljava.lang.String;	n.a.	
writeCommunity	java.lang.String		

6 EDG License Agreement

This section should contain the EDG agreement, under which CrossGrid software is being licensed. If your software follows a different licensing pattern, replace this text with another license, appropriate for your software.

Bibliography

[QAP] WP5, CYRFRONET; **Quality Assurance Plan**; Evolving document