Project Acronym:	**OPTIMIS**
Project Title:	**Optimized Infrastructure Services**
Project Number:	**257115**
Instrument:	**Integrated Project**
Thematic Priority:	**ICT-2009.1.2 – Internet of Services, Software and Virtualisation**

Monitoring Infrastructure User Guide

Activity 4: *Basic Service Operation*

WP 4.1: *Monitoring and Data Collection*

Due Date:		M34
Submission Date:		31/03/2012
Start Date of Project:		01/06/2010
Duration of Project:		36 months
Organisation Responsible for the Deliverable:		HLRS-University of Stuttgart (USTUTT)
Version:		1.0
Status		Final
Author(s):	Pierre Gilet	USTUTT

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	2012-04-04	Created document.	Pierre Gilet
0.2	2013-03-12	Updated document for Y3 of OPTIMIS project.	Pierre Gilet
1	2013-03-22	Final version	Pierre Gilet

Table of Contents

Index of Figures

Index of Tables

1 Introduction

This document is the user guide of the Monitoring Infrastructure software component.

1.1 Glossary of Acronyms

Acronym	Definition
D	Deliverable
WP	Work Package
MI	Monitoring Infrastructure
IPVM	Virtual Machine of the Infrastructure Provider

2 Monitoring Infrastructure User Guide

2.1 Release information

Java Component Name	Release Number	Release Date
MonitoringInfrastructure	3.0	2013-03-31

2.2 Functionalities

2.2.1 Submitting Data to the Monitoring Infrastructure

The MI uses both pull and push mechanisms to collect the values of monitored parameters and store them in the Monitoring database.

The Aggregator component provides the functionality to submit monitoring parameters to the MI. It is capable of submitting service parameters, physical parameters, virtual parameters and energy parameters. The submitted values are then forwarded by the Aggregator to RabbitMQ, and ultimately, they are inserted into the MI database.

In short, the component used to submit data, which leads then to the insertion of that data into the MI database, is the Aggregator one.

2.2.2 Extracting Data from the Monitoring Infrastructure

The Aggregator component also provides a functionality to extract the latest values stored as static variables in it and that were processed by it. This functionality can be used to retrieve the latest values that were processed by the Aggregator, without having to query anything from the MI database.

The MonitoringManager component provides the functionality to extract monitoring data from the MI database. It helps MI consumers retrieve monitoring data.

In short, the component used to get the latest values processed by the Aggregator is the Aggregator itself. And the component used to extract data from the MI database is the MonitoringManager one.

The next figure illustrates what the previous lines said (it is however a simplified view of the MI architecture).

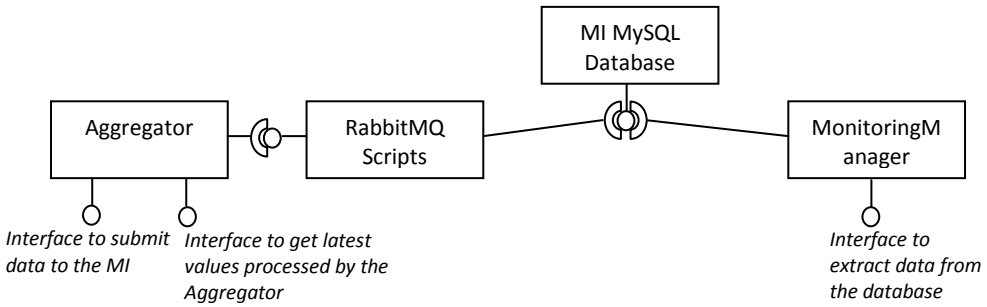

Figure 1 Aggregator and MonitoringManager Components

A client of the MonitoringManager is the Monitoring Website. That website provides a graphical representation of values stored in the database. Link: http://ip-adress-of-the-IPVM:8080/MonitoringManagerWeb/.

The Monitoring Website is also integrated within the IP Dashboard of OPTIMIS: http://ip-adress-of-the-IPVM:8080/IPManagerWeb.

2.2.3 List of Monitored Parameters

The MI is delivered with the following monitored parameters.

2.2.3.1 Physical Hosts

Monitored every minute:

Metric	Unit	Example
count_of_users	users	3 users
cpu_average_load		1.370,1.910,1.830
cpu_speed	MHz	2527.022
disk_free_space	MB	658887 MB
Downstream	Kbps	17.904 Kbps
free_memory	kB	61700 kB
hardware_error		Hardware error
status		Values: OK, CRITICAL, WARNING
Upstream	Kbps	2.624 Kbps

Monitored every 12 hours:

Metric	Unit	Example
disk_total_space	MB	903775 MB
mac_address		BC:30:5B:CE:97:CB
fqdn		optimis1
last_reboot		1317036644(10299240) // 1st figure: last reboot timestamp in UNIX time format (seconds elapsed since Jan 1, 1970), 2nd figure in brackets: uptime in seconds since last reboot.
No_of_cores	cores	16 cores
total_memory	kB	9170944 kB

Monitored every 24 hours:

Metric	Unit	Example
peak_time		2.128 Kbps 2011-12-01 12:12:00
trough_time		1.145 Kbps 2011-12-01 07:25:00

2.2.3.2 Energy Sensors

Monitored every minute:

Metric	Unit	Example
current	A	0.55 A
real_power	W	76.50 W
real_energy	kWh	265.86 kWh
apparent_energy	kVAh	434.27 KVAh
apparent_power	VA	127.50 VA
power_factor	%	78 %
xentop_cpu		Domain-0 10.2;instance-j 3.2;instance-l 0.0;instance-m 0.1;instance-v 0.0 // The figures given next to the domain names are the CPU(%) ones of xentop.

2.2.3.3 Virtual Machines

Monitored every two minutes:

Metric	Unit	Example / Note
mem_total	MB	
cpu_speed	MHz	
mem_used	%	
os_release		
cpu_user	%	Percentage of the CPU usage
vm_state		
disk_total	GB	
machine_type		
cpu_vnum		
bytes_out	bytes	
bytes_in	bytes	

2.3 Getting Started

2.3.1 Using the Software

In Eclipse or any similar IDE, use the RESTClient sub-component of the MonitoringInfrastructure project to call the RESTful web services of the Aggregator and MonitoringManager components.

2.3.1.1 getClient Class

This class is used to invoke the GET web services of the MonitoringManager component (i.e. to extract data from the MI database).

Here is an example with the getClient class and the getLatestReportForEnergy method. The method returns data from the infrastructure provider where the MonitoringInfrastructure component is installed.

```
package eu.optimis.mi.rest.client;
import eu.optimis.mi.monitoring_resources.MonitoringResourceDatasets;
public class TestGetClient {
    public static void main(String[] args){
        getClient client = new getClient("212.0.127.140", 8080,
"MonitoringManager/QueryResources");
     MonitoringResourceDatasets rs = client.getLatestReportForEnergy("optimis1");
     for (int i=0; i<rs.getMonitoring_resource().size(); i++){
     System.out.println(rs.getMonitoring_resource().get(i).getMetric_name());
     System.out.println(rs.getMonitoring_resource().get(i).getMetric_value());
     System.out.println(rs.getMonitoring_resource().get(i).getMetric_timestamp().toString());
         }
    }
}
```

List of methods available in the getClient class:

getLatestReportForEnergy getLatestReportForService getLatestReportForVirtual getLatestReportForPhysical	Retrieve only the most recent monitoring data from the database related to a specific resource id (identifying a service, a virtual machine or a physical machine).
getLatestReportForMetricName	Retrieve only the most recent monitoring data from the database related to a specific metric name and resource type (resource type can be 'service', 'virtual', 'physical' or 'energy').
getReportForPartService getReportForPartPhysical getReportForPartVirtual getReportForPartEnergy	Retrieve all the monitoring data from the database related to a specific resource type ('service', 'virtual', 'physical' or 'energy') and within a time window. The time format of the "from" and "to" input parameters is yyyyMMddHHmmss.
getReportForPartServiceId getReportForPartPhysicalId getReportForPartVirtualId getReportForPartEnergyId	Retrieve all the monitoring data from the database related to a specific resource id (identifying a service, a virtual machine or a physical machine) and within a time window. The time format of the "from" and "to" input parameters is yyyyMMddHHmmss.
getReportForPartMetricName	Retrieve all the monitoring data from the database related to a specific metric name, resource type (resource type can be 'service', 'virtual', 'physical' or 'energy') and within a time window. The time format of the "from" and "to" input parameters is yyyyMMddHHmmss.

Table 1 getClient Methods

2.3.1.2 GetAggregatorClient Class

This class is used to invoke the GET web services of the Aggregator component (i.e. to get the latest values stored in the Aggregator).

List of methods available in the getAggregatorClient class:

getCurrentReportForPhysical	Return the most recent monitoring data of resource type 'physical' still stored in the Aggregator component.
getCurrentReportForVirtual	Return the most recent monitoring data of resource type 'virtual' still stored in the Aggregator component.
getCurrentReportForService	Return the most recent monitoring data of resource type 'service' still stored in the Aggregator component.
getCurrentReportForEnergy	Return the most recent monitoring data of resource type 'energy' still stored in the Aggregator component.

Table 2 GetAggregatorClient Methods

2.3.1.3 postClient Class

This class is used to invoke the POST web service of the Aggregator component (i.e. to submit data to the MI).

Here is an example with the postClient class and the pushReport method that can be used to submit data to the MI.

```
package eu.optimis.mi.rest.client;


import eu.optimis.mi.monitoring_resources.MonitoringResourceDatasets;
import eu.optimis.mi.monitoring_resources.MonitoringResourceDataset;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;


public class testPOST {
   public static void main(String[] args) {


        String serviceID = "DemoApp"; // Mandatory if resourceType in ('service').
        String virtualResID = "123456"; // Mandatory if resourceType in ('service', 'virtual').
        String physicalResID = "optimis1"; // Mandatory if resourceType in ('service', 'virtual',
'physical', 'energy').
        String resourceType = "service"; // Allowed values: physical, energy, virtual, service.
        String collectorID = "dummycollector"; // Do not change this value.


        postClient post = new postClient("optimis-ipvm.atosorigin.es", 8080,
                        "Aggregator/Aggregator/monitoringresources/" + resourceType);
        Date now = new Date();
```

```
        now.setTime(System.currentTimeMillis());

        MonitoringResourceDataset md1 = new MonitoringResourceDataset(serviceID, virtualResID,
                        physicalResID, resourceType, collectorID, "num_of_cores", "4", "", now);

        MonitoringResourceDataset md2 = new MonitoringResourceDataset(serviceID, virtualResID,
                        physicalResID, resourceType, collectorID, "processor_speed", "3", "GHz",
now);

        MonitoringResourceDataset md3 = new MonitoringResourceDataset(serviceID, virtualResID,
                        physicalResID, resourceType, collectorID, "main_memory", "4096", "MB",
now);

        MonitoringResourceDataset md4 = new MonitoringResourceDataset(serviceID, virtualResID,
                        physicalResID, resourceType, collectorID, "storage", "40000", "GB", now);

        MonitoringResourceDataset md5 = new MonitoringResourceDataset(serviceID, virtualResID,
                        physicalResID, resourceType, collectorID, "swap_memory", "8192", "GB",
now);

        MonitoringResourceDataset md6 = new MonitoringResourceDataset(serviceID, virtualResID,
                        physicalResID, resourceType, collectorID, "network_bandwidth", "1000",
"MBit/s", now);

        List<MonitoringResourceDataset> mdList = new ArrayList<MonitoringResourceDataset>();
        mdList.add(md1);
        mdList.add(md2);
        mdList.add(md3);
        mdList.add(md4);
        mdList.add(md5);
        mdList.add(md6);
        MonitoringResourceDatasets mds = new MonitoringResourceDatasets();
        mds.setMonitoring_resource(mdList);
        System.out.println(post.pushReport(mds));
    }
}
```

2.3.1.4 Stop/Start the Monitoring Infrastructure

How to stop the MI:

- Log on to IPVM with the root user: `ssh root@ip-adress-of-the-IPVM`
- Edit the crontab list: `crontab -e`
- Comment out all the jobs coming from scripts located in /opt/optimis/MonitoringInfrastructure and its sub-directories.
- Save the crontab file.

How to start the MI:

- Re-enable the MonitoringInfrastructure jobs in the crontab file. The MI jobs will then restart automatically.

See the installation guide for more details. It explains how to install the MI and how to activate it.

2.3.1.5 Troubleshooting

A cronjob calling /opt/optimis/MonitoringInfrastructure/scripts/check_data_files.sh runs daily. It checks the health of the MI. For instance, it checks if there are too many XML files still waiting to be processed by RabbitMQ. It also checks if the RabbitMQ server is up and running. It checks if Tomcat is up and running, etc. You can look at the script directly to get the list of functionalities verified by it. If something wrong has been detected by the script, an email is sent automatically to the people flagged as MI administrators in check_data_files.properties.

To start troubleshooting, most problems come from the following causes:

- Tomcat is down or running erratically. You may need to kill the existing Tomcat daemon process still running on IPVM and restart it, possibly with different start options.
- The RabbitMQ server is down or running erratically. You may need to restart the RabbitMQ server daemon with less required disk space. See the README file in /opt/optimis/MonitoringInfrastructure/rabbitmq.
- The IPVM has a full disk space. You can check this by logging on to IPVM and running: `df -h`. In this case, you will need to stop IPVM, increase its disk space and restart it.
- The Nagios Server is down on the physical host used as Nagios Server. Or the HTTP daemon is down on that server. You will need to restart Nagios or the HTTP daemon there, and make sure that they are started automatically at reboot time (run: `chkconfig nagios on`, and run also: `chkconfig httpd on`).

- The PDU used to get power consumption information is down. Look at /opt/optimis/MonitoringInfrastructure/logs/run_get_metric.log for more details. If necessary, contact the administrator in charge of the PDU.
- The connection to the MI database is lost. Log on to IPVM. Look at the contents of /opt/optimis/MonitoringInfrastructure/share/database.properties and try to connect to the database with the mysql client: `mysql -h [DBVM-host] -u mmanager_usr –p[user-password] -D optimis_db`. Note that there is no space character between the "-p" switch and the password.
- Entries in /etc/hosts have been deleted. Those entries are necessary for the the virt-top monitoring executed by the script /opt/optimis/MonitoringInfrastructure/scripts/get_metric/xentop_scripts/check_optimis_xentop.sh. See installation guide and /opt/optimis/MonitoringInfrastructure/scripts/README for more details.

To troubleshoot issues in the MI, do the following:

- Log on to IPVM with the root user.
- Check the logs in /opt/optimis/MonitoringInfrastructure/logs.
- Run /opt/optimis/MonitoringInfrastructure/scripts/check_data_files.sh manually and look at the standard output.
- Stop the Monitoring Infrastructure (i.e. disable the cronjobs) as long as the problem is not fixed.
- Remove manually all the empty files possibly found in /opt/optimis/MonitoringInfrastructure/var/*/ (i.e. in the 4 sub-directories of the var directory).
- Remove manually all the temporary files coming from MI processes possibly found in /tmp.
- Run: `pgrep -f MonitoringInfrastructure`. This will list the MI processes still running. Kill them all: `pkill -f MonitoringInfrastructure`.
- Once the issue is fixed, reactivate the cronjobs of the MI.

2.3.2 Contributors

The following people at HLRS (University Stuttgart) designed and developed the Monitoring Infrastructure for OPTIMIS (they are listed alphabetically): Pierre Gilet, Gregory Katsaros, Anthony Sulistio, Tinghe Wang.