MonALISA User Guide

version 1.0, September 14, 2003

The MonALISA framework provides a distributed monitoring service system using JINI/JAVA and WSDL/SOAP technologies. Each MonALISA server acts as a dynamic service system and provides the functionality to be discovered and used by any other services or clients that require such information.

MonALISA is entirely written in java. The distribution is packed in several self contain jar files. Practically it is required to have only java installed on the system hosting the MonALISA Service (NOT on the monitored nodes!). The way to configure and use the service is described in this guide. MonALISA also allows to dynamically add new monitoring modules, Filters or Agents.

Running a MonALISA service does not require any root privilegies, and we **strongly suggest** to do not use the service from a root account.

Java

For running a MonALISA service you need to have the java runtime environment (j2se 1.4.2 or higher) installed on one system that will run the Monitoring Service for an entire farm. For development of dedicated modules or agents the user should install the entire JDK.

Setting the environment to run java may look like this:

```
JAVA_HOME=$HOME/JAVA/jdk
export JAVA_HOME
export PATH=$JAVA_HOME/bin:$PATH
```

The monitoring Information

Collecting the monitoring information can be done in several ways using dynamically loadable modules.
It is possible to collect information using:

- SNMP demons
- Ganglia
- LSF or PBS batch quing systems
- Local or remote procedures to read `/proc` files
- User modules based on dedicated scripts or procedures.

SNMP

MonALISA has dedicated modules to collect values provided by snmp demons. Using snmp modules requires that the snmpd demons are installed and properly configured on the nodes or network elements (switches or routers) the user want to monitor.

You can test the values provided by the snmpd demon (based on how it is configured) by using:

- legacy ucd-snmp: `$snmpwalk [-p port_no] system_name community OID`
- net-snmp: `$snmpwalk -v2c -c community system_name[:port_no] OID`

Example:

```
$ snmpwalk -v2c -c public host_name:161 .1.3.6.1.2.1.2.2.1.10
```

In the previous example the query is performed on the host "host_name", using default settings for snmp (transport = UDP; port = 161; community = public). The output should look like this:

```
IF-MIB::ifInOctets.1 = Counter32: 1430
IF-MIB::ifInOctets.2 = Counter32: 966737519
```

For more information in how to configure and use SNMP: http://www.net-snmp.org/

MonALISA provides snmp modules to collect:

- IO traffic from nodes and network elements
- CPU usage
- System Load
- Disk IO traffic
- ...

Here are the OIDs that must be "exported" by the snmpd daemon in order to allow various dedicated MonALISA snmp modules to collect the data:

- **snmp_IO**:
  `incoming network traffic: .1.3.6.1.2.1.2.2.1.10`
  `outgoing network traffic: .1.3.6.1.2.1.2.2.1.16`

- **snmp_Load**:
  `Load5, Load10 and Load15:.1.3.6.1.4.1.2021.10.1.3`

- **snmp_CPU**:
  `CPU_usr, CPU_nice and CPU_idle:.1.3.6.1.4.1.2021.11`

- **snmp_MEM**:
  `MEM_free, Swap_MEM_Free:.1.3.6.1.4.1.2021.4`

- **snmp_Disk**:
  `FreeDSK, UsedDsk:.1.3.6.1.4.1.2021.9`

Kernel `/proc` files

Modules to collect the system monitoring information from the kernel are part of the MonALISA distribution. These modules are mainly design to be used on the node MonALISA service is running but they may also be used on remote systems via rsh or ssh.

Ganglia

Ganglia is a well known monitoring system which is using a multi cast messaging system to collect system information from large clusters. MonALISA can be easily interfaced with Ganglia. This can be done using the multicast messaging system or the gmon interface which is based on getting the cluster monitoring information in XML format. In the MonALISA distribution we provide modules for both these two possibilities. If the MonALISA service runs in the multicast range for the nodes sending monitoring data, we suggest using the Ganglia module which is a multicast listener. The code for interfacing MonALISA with Ganglia using gmon is `Service/usr_code/GangliaMod` and using the multicast messages is `Service/usr_code/GangliaMCAST`. The user may modify these modules. Please look at the **service configuration examples** to see how these modules may be used.

DataBase system

MonALISA comes with an embedded SQL Data Base (McKoiDB) which is used by default. If you are happy with this option nothing has to be done. If you would like to use an other DB system, you need to have the JDBC driver for it and to create initially the tables used by MonALISA. We tested the system with MySQL. The MonALISA distribution provides the drivers and the configuration scripts for the two DataBase systems. We also provide for convenience an archive which has all the scripts to install and configure MySQL to be used as a storage mechanism for monitoring values. To install and use MySQL it is not required root access. In `$MonaLisa_HOME/Service/Examples` there are simple scripts (in `SimpleEx_MySql/`) for using MonALISA with MySQL. The JDBC drivers can be found in `$MonaLisa_HOME/Service/lib/mysql-driver.jar`.

MonALISA Configuration

There are two main configuration files: a global configuration file (`$MonaLisa_HOME/Service/CMD/ml_env`) used by the scripts and the other one used by MonALISA itself (`$MonaLisa_HOME/Service/<YOUR_FARM_DIRECTORY>/ml.properties`).

Global configuration file (`$MonaLisa_HOME/Service/CMD/ml_env`)

- **MONALISA_USER** - the user name under wich the service is running. It will not start from other account or from the root account.
- **JAVA_HOME** - the path to your current JDK. The starting expects to find `JAVA_HOME/bin/java` to start MonALISA
- **SHOULD_UPDATE** - whether or not MonALISA should check for updates when is started. If this parameter is "true" when MonALISA is started first will check for updates and after that it will start. If set to "false" it will not check for updates. This parameter is also used to check for autoupdates when it is running. Please see Starting a Monitoring Service with Autoupdate
- **MonaLisa_HOME** - path to your MonALISA installation directory. Environment variables can also be used. (e.g `${HOME}/MonaLisa`)
- **FARM_HOME** - path to a directory where to your farm specific files resides. It's better to place this directory under Service directory. (e.g. You can use the variable `MonaLisa_HOME` defined above. `${MonaLisa_HOME}/Service/MyTest`. MonALISA comes with a simple example in `${MonaLisa_HOME}/Service/TEST`
- **FARM_CONF_FILE** - the file used at the startup of the services to define the clusters, nodes and the monitor modules to be used. It should be under `${FARM_HOME}` directory. (e.g `FARM_CONF_FILE="${FARM_HOME}/mytest.conf"`)
- **FARM_NAME** - the name for your farm. (e.g `FARM_HOME="MyTest"`). We would like to ask the users to use short names to describe the SITE on which they are running MonALISA.
- JAVA_OPTS - is an **optional** parameter to pass parameters directly to the Java Virtual Machine (e.g `JAVA_OPTS="-Xmx128m"`)

The MonALISA properties

The file `$MonaLisa_HOME/Service/<YOUR_FARM_DIRECTORY>/ml.properties` is specific for your farm configuration. (e.g lookup services to use, the jini group that your service should join, etc. You should find comments on in the file)

The configuration, the monitored parameters and the repetition time may be changed via de Administrative GUI.

Setup the configuration files for your site:

- Go to `"MonaLisa"/Service` directory and create a directory for your site (e.g MySite). You may copy the configuration files from one of the available site directory (e.g.: under `"MonaLisa"/Service/TEST`). You **must** include the folowing files in you new Farm (`ml.properties`, `db.conf.embedded` and `my_test.conf`)

- Edit the configuration file (`my_site.conf`) to reflect the environment you want to monitor.
- Edit `ml.properties` if you would like to change the Lookup Discovery Services that will be used or if you would like to use another DB System.
- You may add a `myIcon.gif` file with an icon of your organization in `"MonaLisa"/Service/ml_dl`

The **only** script used to start/stop/restart "MonaLisa" is ML_SER under this directory.

After you have done what is described in the "Configuration" section you can start using MonALISA:

```
Service/CMD/ML_SER start.
```

Authentication & Security

The Administration of the service can be done from a GUI and is done via a SSL connection. This requires the user to provide X.509 certificate to be imported into the key trust store of the service. Once the user's certificate becomes a trusted certificate for the service it allows the Administrator user to change the configuration from a GUI. This is required only if the user want to change the configuration of this service.

Running MonALISA behind a firewall

MonALISA can run behind a firewall if there are three open ports. These ports are dynamicaly assigned when MonALISA starts and can be configured in `$MonaLisa_HOME/Service/<YOUR_FARM_DIRECTORY>/ml.properties`. There are two parameters **`lia.Monitor.MIN_BIND_PORT`** and **`lia.Monitor.MAX_BIND_PORT`**. MonALISA will try to bind the first three available ports in this range, starting from lia.Monitor.MIN_BIND_PORT until lia.Monitor.MAX_BIND_PORT is reached.

Start a Monitoring Service with Autoupdate

This allows to automatically update your Monitoring Service. The cron script will periodicaly chech for updates using a list of URLs. When a new version is published the system will check its digital signature and then will download the new distrbution as a set of signed jar files. When this opration is completed the MonALISA service will restart atomaticaly. The dependecies and the configurations realted with the service are done is very similar way with the Web Start technology.

This funtionaly makes it very easy to maintan and run a MonALISA service. We recomand to use it!

In this case you should add "MonaLisa"/Service/CMD/CHECK_UPDATE to the user's crontab that runs MonALISA. To edit your crontab use:

```
$crontab -e
```

Add the folowing line:

```
*/5 * * * * /path_to_your_MonaLisa/Service/CMD/CHECK_UPDATE
```

This would check for update every five minutes. It is resonable value that this value should be >= five minutes. To check for update every 10 minutes add the folowing line instead of the one above.

```
*/10 * * * * /path_to_your_MonaLisa/Service/CMD/CHECK_UPDATE
```

To disable autoupdate you cand edit the ml_env file in `"MonaLisa"/Service/CMD` and set `SHOULD_UPDATE="false"`. It is no need to remove the script CHECK_UPDATE from your crontab.

Launch `"MonaLisa"/Service/CMD/ML_SER start`
MonALISA should check for updates now.

The Administration GUI

In order to connect to a MonALISA service as an administrator your X.509 certificate must first be imported into the MonALISA service keystore as a trusted entry. The MonALISA keystore is located under the SSecurity directory (`FarmMonitor.ks`). A script to import a certificate into the keystore can be found in the same directory. We also provide scripts to create a new keystore, export your certificate from the keystore and import into MonALISA Service keystore. Following is an example how to create a keystore with your private key and how to import your public key into the service keystore. The example will use the `keytool` command under `$JAVA_HOME/bin`. The scripts assumes that your `$PATH` environment variable contains `$JAVA_HOME/bin`.

1. First go to under "`MonaLisa`"`/Service/SSecurity`
2. Generate a private key. This will also generate a keystore(`MyAdmin.ks`) with your private key in it. The last parameter is an alias used to identify your private key in `MyAdmin.ks`. Please do not forget the password that you will give to your keystore. You will use it from the GUI Client to administer your farm.

   ```
   $./genKey MyAdmin.ks myadmin_private
   ```

3. Export now your public key into a file (`myadmin.cer`)

   ```
   $./exportCert MyAdmin.ks myadmin_private myadmin.cer
   ```

   The first parameter is the path to the keystore generate at step 2). The second parameter (also given at step 2) is the alias that identifies your private keystore and the last one is the file wich will hold your public key.

4. Import your public key into the MonALISA trusted keystore

   ```
   $./importCert myadmin_public myadmin.cer
   ```

   This will import the public key from `myadmin.cer` into the `FarmMonitor.ks`.

Use the security menu in the Global Client to authentificate yourself using the keystore which contains your private key. For all the farms which have your cerificate imported in the trust keystore you will atomatically get administravie rights. This allows to change the configuration, to add or stop modules for individual nodes or clusters as well as to re-start or stop the service.

Start a Global GUI

This program allows to discover all the registered centers with any set of Lookup Discovery services (started with jrun command) and allows any user to see real-time global values as well any measurement for each component in the system and its history.

```
cd Clients/Gui/
./jGlobal
```

The global GUI can be started from the MonALISA web page (http://monalisa.carc.caltech.edu) select Download -> Web Start client. This will guarantee that you will use always the updated version and it is also possible to install the client of your system very easily.

We strongly suggest using this option.

If you want to change the list of Lookup Discovery services to be used by the global client simply edit the `jGlobal` script or use the Discovery menu in the client window to add/remove lookup discovery services.

You also change the groups you are interested to monitor. Details in how to use the GUI are presented in the help file and in the Examples under MonALISA documentation.

Web Service Client WSDL/SOAP (java)

MonALISA provides also a Web Service interface. It allows any client to connect and receive selected values. A predicate mechanism can be used. Client examples code for java (using Axis or WSIF) and perl are provided as a separate package the user may download from the Web Services section. These Web Services clients allow the user to dynamically access monitoring information from any MonALISA monitoring services or global MonALISA repositories.

Writing into MDS or any other programs or database systems

The MonALISA framework allows to dynamically load additional modules for writing (or processing) the collected values. In `usr_code/MDS` is an example of writing the received values into MDS. This is done using a unix pipe to communicate between the dynamically loadable java module and the script performing the update into the LDAP server.

Please also read the ReadMe file from this directory.

* Another simple example which simply print all the values on sysout can be found on `usr_code/SimpleWriter`
* Another example to write the values into UDP sockets is in `usr_code/UDPWriter`

Writing new Monitoring Modules

New Monitoring modules can be easily developed. These modules may use SNMP requests or can simply run any script (locally or on a remote system) to collected the requested values. The mechanism to run these modules under independent threads, to perform the interaction with the operating system or to control an snmp session are inherited from a basic monitoring class. The user basically should only provide the mechanism to collect the values. to parse the output and to generate a result object. It is also required to provide the names of the parameters this module is collecting.

Examples to generate new modules are in `${MonaLisa_HOME}/Service/usr_code`. Please see the `ReadMe.txt` in this directory

Writing new Filters

Filters allow to dynamically create any new type of derived value from the collected values. Es an example it allow to evaluate the integrated traffic over last n minutes, or the number of nodes for which the load is less than x. Filters may also send an email to a list or SMS messages when predefined complex condition occur. These filters are executed in independent threads and allow any client to register for its output. They may be used to help application to react on certain conditions occur, or to help in presenting global values for large computing facilities.

Farm Configuration Examples

The MonALISA service is using a very simple configuration file to generate the site configuration and the modules to be used for collecting monitoring information. By using the administrative interface with SSL connection the user may dynamically change the configuration and modules used to collect data.

It is possible to use the build modules (for snmp, local or remote /proc file...) or external modules. We provide several modules which allow exchanging information with other monitoring tools. These modules are really very simple and the user can also develop its own modules. In what follows we will present a few simple examples in how to make the configuration for a farm.

Monitoring a Farm using snmp

The configuration file should look like this:

```
*Master
>citgrid3.cacr.caltech.edu citgrid3
monProcLoad%30
monProcStat%30
monProcIO%30
*ABPing{monABPing, citgrid3.cacr.caltech.edu, " "}
*PN_CIT
>c0-0
snmp_Load%30
snmp_IO%30
snmp_CPU%30
>c0-1
snmp_Load%30
snmp_IO%30
snmp_CPU%30
>c0-2
snmp_Load%30
snmp_IO%30
snmp_CPU%30
>c0-3
snmp_Load%30
snmp_IO%30
snmp_CPU%30
```

The first line (`*Master`) defines a Functional Unit (or Cluster)
The Second line (`>citgrid3.cacr.caltech.edu citgrid3`) adds a node in this Functional Unit class and optionally an alias.

The lines:

```
monProcLoad%30
monProcIO%30
monProcStat%30
```

are defining three monitoring modules to be used on the node "citgrid3". These measurements are done periodically every 30s. The `monProc*` modules are using the local `/proc` files to collect information about the cpu, load and IO. In this case this a master node for a cluster, were in fact MonALISA service is running and simple modules using the `/proc` files are used to collect data.

The line:

```
*ABPing{monABPing, citgrid3.cacr.caltech.edu, " "}
```

defines a Functional unit named "ABPing" which is using an internal module monABPing. This module is used to perform simple network measurements using small UDP packages. It requires as the first parameter the full name of the system corresponding the real IP on which the ABping server is running (as part of the MonALISA service) The second parameter is not used. These ABPing measurements are used to provide information about the quality of connectivity among different centers as well as for dynamically computing optimal trees for connectivity (minimum spanning tree, minimum path for any node to all the others...)

```
*PN_CIT
```

defines a new cluster name. This is for a set of processing nodes used by the site. The string "PN" in the name is necessary if the user wants to automatically use filters to generate global views for all this procesing units.

Then it has a list of nodes in the cluster and for each node a list of modules to be used for getting monitoring information from the nodes. For each module a repetition time is defined (%30). This means taht each such module is executed ones every 30s. Defining the repeating time is optional and the default value is 30s.

Monitoring a Farm using Ganglia gmon module

The farm configuration file will look like this:

```
*Master
>ramen gateway
monProcLoad%30
monProcIO%30
monProcStat%30
*PN_popcrn {IGanglia, popcrn01.fnal.gov, 8649}
*ABPing{monABPing, ramen.fnal.gov, " "}
*Internet
>tier2.cacr.caltech.edu
monPing%50
```

The first line (*Master) defines a Functional Unit (or Cluster).
The Second line (>ramen gateway) adds a node in this Functional Unit class.

In this case ramen is computer name and optionally the user may add an alias (gateway to this name).

The lines:

```
monProcLoad%30
monProcIO%30
monProcStat%30
```

are defining three monitoring modules to be used on the node "ramen". These measurements are done repeatedly every 30s. The monProc* modules are using the local /proc files to collect information about the cpu, load and IO.

The line:

```
*PN_popcrn {IGanglia, popcrn01.fnal.gov, 8649}%30
```

defines a cluster named "PN_popcrn" for which the entire information is provided by the IGanglia module. This module is using telnet to get and XML based output from the Ganglia gmon. The telnet request will be send to node `popcrn01.fnal.gov` on port 8649.
All the nodes which report to ganglia will we part of this cluster unit and for all of the them the parameters selected in the IGanglia module will be recorded. This measurement will de done every 30s.

The Ganglia module is localted in the `Service/usr_code/GangliaMod`. The user may edit the file and customize it. This module is NOT in the MonaLISA jar files and to be used, the user MUST add the path to this module to the MonaLISA loader. This can be done in ml.propreties by adding this line:

```
lia.Monitor.CLASSURLs=file:${MonaLisa_HOME}/Service/usr_code/GangliaMod/
```

```
*ABPing{monABPing, ramen.fnal.gov, " "}
```

This like defines a Functional unit named "ABPing" which is using an internal module monABPing. This module is used to perform simple network measurements using small UDP packages. The first parameter must be the full name of the system which corresponds to the real IP on which the ABping server is running. The second parameter is not used.

The next lines:

```
*Internet
>tier2.cacr.caltech.edu  caltech
monPing%50
```

Defines a new functional unit (Internet) having one node `tier2.cacr.caltech.edu` with the alias caltech for which a ping measurement is done by the monPing module every 50s.

Monitoring a Farm using Ganglia Multicast module

For getting copies of the monitoring data send by the nodes running the ganglia demons (using a multicast port) it is necessary that the system on which MonaLISA is running to be in muticast range for these messages.

Adding such a line:

```
*PN_cit{monMcastGanglia, tier2, " GangliaMcastAddress=239.2.11.71;
                                                GangliaMcastPort=8649"}
```

in the configuration file, will use the Ganglia multicast module to listen to all the monitoring data and then to select certain values which will be recorded into MonALISA. The service system will automatically create a configuration for all the nodes which report data in this way.

The PN_cit is the name of the cluster of processing nodes. Is is important for the cluster name of processing nodes to contain the "PN" string. This is used by farm filters to report global views for the farms.

The tier2 is the name of the system corresponding to the real IP address on which this MonALISA service is running. The second parameter defines the multicast address and port used by Ganglia.

The GangliaMcat module is located in the `Service/usr_code/GangliaMCAST`. The user may edit the file and customize it. This module is NOT in the MonaLISA jar files and to be used, the user MUST add the path to this module to the MonaLISA loader. This can be done in ml.propreties by adding this line:

```
lia.Monitor.CLASSURLs=file:${MonaLisa_HOME}/Service/usr_code/GangliaMCAST/
```

Getting Job related information from PBS

To monitor data provided by the PBS, you will need to add these line into the config file:

```
*JOBS
>tier2
PBSjobs{cmsim,ooHits}%30
```

The first line defines the name of the functional unit (JOBS). The second line defines the node (normally the current system) where the PBSJobs module will run. The module has a parameter containing a list o jobs for which information will be provided. The module will run every 30s. The code of the PBSJobs module is in `Service/usr_code/PBS/` and must be added to the MonaLISA class loader in similar way like the Ganglia modules.