MonALISA Web Services User Guide

February 5, 2004

Chapter 1

Web Services for MonALISA

A simple Web Service is integrated with the MonALISA service, as well with the MonALISA Repository. The Web service, "MLWebService", provides an interface for publishing the monitoring data using a WSDL/SOAP technology. In this way, any client can connect and receive selected monitoring data.

1.1 Service description

The service offers a single port type with three operations: `getValues`, `getConfiguration` and `getLatestConfiguration`.

- the `getValues` operation interrogates the database and gets all the parameter values for the farm, the cluster, the node and the parameter name specified as arguments. The time when the parameter(s) was registered in the database must be between two moments of time (fromTime and toTime) also specified as operation input. The return type is a complex schema type, an array of results containing all the values taken from the database.
- the `getConfiguration` operation interrogates the database and gets all configurations of all farms that were registered in the database between two limits of time (from time and to time) given as input. The return type is a complex schema type, an array of configurations that were found in the database that matched the constraints .
- the `getLatestConfiguration` operation interrogates the database and returns the last configuration received in the database for a given farm. It receives as input a string, the farm name. It returns a complex schema type that represents the configuration.

1.2 Service implementation

The web service application was developed using Apache Axis (See <`http://ws.apache.org/axis`>). The interface of the service contains the following functions:

`Result[] getValues (String farmName, String clusterName, String nodeName, String para`

This function can be called in two ways:

- specifying negative values for times. For example, if the call is
  `getValues( "*", "*", "*", "*",-3600000,0)`
  the service will return all the values registered in the database in the last hour.
- specifying absolute values for time. For example, if the call is
  `getValues ("*", "*", "*", "*", 1060400000000, 1065000000000)`
  the service will return all the values registered in the database with the registration time between the two values specified in milliseconds.

The Result class is a Bean class and has the following description:

```
public class Result {
    private String farmName;      // the farm name that contains the parameters
    private String clusterName;   // the cluster name that contains the parameters
    private String  nodeName ;    // the node name that contains the parameters
    private String[] param_name ; // the parameters names
    private double[] param;       // the parameters values
    private long time;            // the absolute time in milliseconds when this value
                                  // registered in the database
    ............                  // get/set functions
}
```

`WSConf[] getConfiguration (long fromTime, long toTime);` The times specified for this function are absolute moments of time in milliseconds.

The WSConf is a Bean class and has the following description:

```
public class WSConf {
    private WSFarm wsFarm; // the farm that had this configuration
    private long confTime; // the time when this configuration was registered in
                           // the database
    ............          // get/set functions
}
```

the java class that describes a farm:

```
public class WSFarm {
    private String farmName;         // the name of the farm
    private WSCluster[] clusterList; // the clusters of this farm
    ..........                       // get/set functions
}
```

the java class that describes a cluster:

```
public class WSCluster {
    private String clusterName; // the name of the cluster
    private WSNode[] nodeList ; // the nodes contained in this cluster
    ..........                  // get/set functions
}
```

the java class that describes a node:

```
public class WSNode {
    private nodeName ;        // the node name
    private String paramList; // the list of parameters for this node
    .........                 // get/set functions
}
```

`WSConf[] getLatestConfiguration (String farm)` returns the latest configurations received in the database for all farms (farm=”*”) or returns the latest configuration for a specified farm.

Chapter 2

Clients Examples for MLWebService

An archive with Java and Perl examples of simple MLWebService clients example can be downloaded from <http://monalisa.cacr.caltech.edu/>. These examples shows you how to interrogate the web service from MonALISA and get monitoring data using the SOAP protocol.

2.1 MLWebService clients examples presentation

The client examples presented here can interrogate both the MLWebService from the Repository and the MLWebService from the MonALISA service. There are examples for **Java-Axis**, **WSIF** and **Perl**.

2.2 Examples archive structure

The sources of the clients examples are located in the `WS-Clients` directory. There are special subdirectories in it (`Java-Axis`, `Perl`, `Wsif`), each containing clients developed using different libraries (**Apache Axis**, **Soap:Lite** and **Apache Wsif**). Every client example calls a function of the MLWebService and is located in a directory having the name of the called function of the service. The source of every example is called **Client** (`Client.java` or `Client.pl`). There are special scripts in each directory for automating the installation of used libraries, the compilation and execution of each client:

- for the examples developed in Java (Axis or WSIF) each example contains the following scripts
  - the `generate_classes` script uses the WSDL2Java tool for generating the client used classes;
  - the `compile_classes` script compiles the client classes;
  - the `run_client` script executes the example.
- for the example developed in Perl, there were used special modules (**Soap::Lite** and **Time:HiRes**). This modules are automatically installed using the `install_soap_lite` and `install_time_hires` scripts located in the Perl directory.

For details see the `Readme` files from every example directory.

2.3 Examples developed using Apache Axis, Perl SOAP::Lite modules and Apache WSIF

Apache Axis is an implementation of the SOAP (Simple Object Access Protocol).
SOAP::Lite for Perl is a collection of Perl modules which provide a simple and lightweight interface to the Simple Object Access Protocol, both on client and server side.

WSIF (Web Service Invocation Framework) is a simple Java API for invoking Web services no matter or how and where the services are provided. WSIF is closely based upon WSDL (Web Service Description Language - See <http://www.w3.org/TR/wsdl>), so it can invoke any service that can be described in WSDL.

There are three types of clients, one that interrogates the `getConfiguration` function of the service and returns the configuration registered in the database in the last week, one that interrogates the `getValues` function and returns all the parameters and parameter values respecting specified constraints and another one that interrogates the `getLatestConfiguration` service function and returns the latest configuration received in the database for a given farm (See Section 1.2 for the definition of these functions).