# Monitoring Framework Documentation

The monitoring framework is divided into 3 distinct parts:

1. Gathering Subsystem
2. Action Subsystem
3. Driver

This documentation describes the programming rationale behind the three (3) different subsystems.

# Gathering Subsystem

Any OSCAR component which wants to be monitored or can generate monitoring data such as heartbeat, IPMI, pbs_mom e.t.c simply provides a perl module in the Gather directory of the OCA framework. This module must contain 2 subroutines: open and update. The open is called during initialization and will be expected to return a hash containing configuration information of that particular module/plug-in while the update subroutine is called when a monitoring data update is required.

When the gathering subsystem is initialized it creates the monitoring database with create method in "MN_Framework.pm" and parses all the "**configuration hash**" returned by the open subroutine of each module and creates an appropriate column in the monitoring database. With the Oscar_Monitor table initialized, any component can then call the update subroutine of Gather with its "**data hash**" expect its data to be made available through the OSCAR database.

### Structure of the Configuration Hash

```
sub open
{
    return my %configuration = (
                    'component' => 'Ganglia',
                    'function' => 'Heartbeat'
); }
```

The above shows an illustration of the open subroutine which returns a configuration hash.

**Component:** This refers to the name of the component from which the data is derived.

**Function:** As its name implies, the function of the component is required. This is because one component could be used to generated different types of data hence to distinguish them we need a short description for function.

**Structure of the Data Hash**

The monitoring framework expects a hash containing monitoring information to be returned when the update subroutine is queried. In the data hash 2 fundamental elements are required:

**Column**: Specifies what column in the table to update. Each column of the table is simply a concatenation of both the component name and its function that were returned by the open subroutine. The structure is:

"component_name"_"function"

**Hostname**: The hostname serves as a hash key for every node specific data. For example if we wanted to enter heartbeat data for node "oscarnode1.xeno" we would:

$my_hash{"oscarnode1.xeno"} = "node_up"

Therefore each hostname is associated with its corresponding monitoring data.

## Action Subsystem

The action subsystem(rules system) is very simple to extend and interface with. For any component to harness its power, it needs two components; A module in the Action subdirectory of the monitoring framework and its rules in the same subdirectory. As elaborated in the monitoring framework specification, the rules of any component is just a script whose name reflects the rule name to be applied as present in the monitoring database. Similarly, the monitoring component module should contain an open subroutine which receives two parameters; Rule name to apply and any associated monitoring data stored in an array.

## Driver Subsystem

Developing a driver for the monitoring framework requires importing the Action, Driver and Gather Modules. The role of the driver is straightforward; connecting the other two fundamental subsystems by ensuring the monitoring database is updated while responding to any changes/updates to the database. Each driver will be required to initialiaze both frameworks by calling their respective open subroutines. To extract data from the monitoring database requires calling the retrieve method in the Driver.pm module with one argument, column name. The scripts for the driver subsystem are stored in the scripts directory of OSCAR_HOME.

## Testing/Installation

To test this framework with an existing OSCAR installation apply the provided patch in your OSCAR home directory (typically “/opt/oscar/”) then run the provided C3 driver test script (located in OSCAR_HOME/scripts/MN_C3_driver). If all works as intended, your C3 configuration file should be updated automatically if a node goes offline or online. Please note that your C3 file will only be updated when ganglia recognizes the node as offline.

## About this Document

This document describes a monitoring framework which is required to develop monitoring plugins for OSCAR. For more current information about this framework visit the OSCAR website. Additional information can be found here on my blog.