11 11 TEX LATEX

Roxmon - A host monitoring system running in Roxen and Pike.

Linus Tolke linus@lysator.liu.se Rev $Date: 2000/02/08 22:10:36 $

Roxmon is an aid to monitor hosts and receive some notice when some condition is seen. It is intended to monitor and report error conditions and the provided macros are designed to help you in setting up your own monitoring.

Contents

1 Introduction

This document comprises all parts of the documentation of Roxmon. It starts off by describing what problem the project aims to solve and thus what void the Roxmon "product" intends to fill. Then it includes the information needed for the operator installing and configuring Roxmon and for the solution provider who wants to use Roxmon as a platform for his applications. Finally the design of Roxmon is described for the people taking part in the development of Roxmon itself.

1.1 Glossary

There are a lot of possibly confusing words used here. I order to establish some kind of order in the chaos this Glossary will be the terminology used in this document and in the Roxmon code. As with all other areas regarding the Roxmon project, please don't hesitate to suggest improvements or corrections using the bug report mechanism.

Alarm levels

Some of the Macros are normally Configured with level when they are supposed to report a problem. It could be when the disk is 90% full or then the process syslog is bigger than 1Meg. These configuration parameters are called Alarm levels.

Configuring, Configuration

The process of allocating Macros to different Agent instances depending on what services are run on that Host and to decide the Alarm levels and testing interval on each host.

Host

A computer. Every host can only run one instance of the Agent.

Installing, Installation

The process of adding a Host to the monitored network by installing the Agent and adding the Host to the Monitor.

Macro

I believe this is a controversial choice.

This is the Pike "program" that implements the checking of a condition. It is run within the Agent. Normally this just tests a condition, if it is met, it sends an alarm and goes to sleep, if not met, it goes to sleep immediatly. Then after a configurable amount of time it wakes up again and starts over. This could however be initiated by a program reporting things to the Macro or some more elaborate scheme.

Monitor

This is a part of Roxmon. It is the part that collects the alarms from all Agents in a big list, shows that list to the Administrator, and allows the Administrator to work with the list, acknowledging alarms and such. It is implemented in Roxen.

Pike

Pike is the name of the programming language that Roxmon (Agent, Macros and Monitor) is written in. It is also the name of the interpreter running the Agent and Roxen.

Roxen

Roxen is a Web Server product from *Idonex AB*. The Roxen Challanger version of Roxen that is used by Roxmon is available under the terms of GPL.

Site

A collection of Hosts spread over a geographical area so small that the Administrator normally walks or takes the elevator to reach all Hosts.

User

A person normally working on the Host. Ideally he will not notice Roxmon but only the improved availability of the network and services gained from introducing Roxmon.

1.2 The project and license

Roxmon is an open-source project run by Linus Tolke on his spare time. Everybody is allowed to copy and use the result of the project without any warranty or support under the terms of GPL.

1.3 Contributions and configuration

By design, Roxmon has a very thin line between what is Roxmon delivered items and what you add when you do the Configuration. This means that if you are Configuring Roxmon to handle some product I hope you will find time to contribute your work to the Roxmon project so that others can gain from it. The Roxmon project is open for additions for monitoring all kinds of services and products. The license agreement of the product you are monitoring might pose limitations that you will have to follow but in most cases that probably won't be any problem.

1.4 Big changes from the previous version

For small changes, see the ChangeLog file.

1.5 Where to obtain new versions of Roxmon

The *Roxmon project home page* <`http://roxmon.sourceforge.net/`> is located at the *SourceForge* <`http://sourceforge.net/`>.

1.6 Roxmon plan

This is how I currently the development of Roxmon will work in versions of the product:

- 0.0.x Initial versions.
- 0.1.0 First publicly available version.
- 0.1.x Buggfixes, more prepared macros available.
- 1.0.x Complete with simple development tools and help.
- 2.x More exotic features like support for several masters, handling SNMP queries and SNMP traps...

As partners/testers for the 0.0.x versions and alpha/betatesters for the 0.1.0-version I would like to see administrators for sites/organisations with:

- 3 - 20 Unix machines, all with the same Unix version (a version supported by Pike)
- All machines located geographically at one site
- Roxen Challanger already running internally
- Administrator with knowledge of Pike

As alpha/beta-testers for the 1.0-versions, I would like to see administrators for sites/organisations with:

- 5 - 100 Unix machines, with possibly different Unix versions
- All machines located geographically at one site

The 2.0 versions are targetted for organisations with:

- 5 - 1000 Unix machines
- Machines spread geographically
- Requirements for several monitoring stations for redundance reasons

2 Problem description

Roxmon is intended to be a tool for the administrator to help him quickly locate problems on any of the hosts in the network. The kind of problems detected can be anything from a user turning the computer off to disk full or network down.

The important thing is to provide the information on how each machine is doing to the administrator and give him a quick overview of the status of his network.

There are other, more mature, products on the market that solve the same problem but Roxmon has the following advantages:

- It is available under the terms of GPL.
- It is written entirely in Pike making the newly developed macros immediatly available on all hardware/OS-combinations where Pike is available.

3 Vision

Commercial vision

Hopefully this product will be useful for a wide range of system administrators working with OS and applications from the free world promoting the spirit of open source and free software. They will in turn contribute to make this the number one tool for this.

This will also promote the language Pike and the web server Roxen as application platform for a wider range of applications.

The language chosen (Pike) that is an intepreted language running on many platforms, will make it possible to seamlessly use the add-on applications on different platforms more or less without having to keep track of platform differences. This will simplify the exchange of applications and stimulate the project.

Technical vision

The tool building the infrastructure of the system is lightweight. This means that most of the intelligens will reside in the "configured" parts of the system. This infrastructure will be augmented with security solutions for authorisation and good handling of firewall requirements. This without having to modify the applications.

The applications are developed and structured in a way that makes them easy to find and configure the monitoring as you want it.

4 Installation, configuration and running

5 Roxmon internals

Since Roxmon is only a small project all this design information is kept in this document together with all other information. This could be confusing for the reader that doesn't need it but hopefully he can skip it without getting confused.

5.1 What you need to know to write a Macro

This section describes the parts of the design that you need to know in order to write a Macro for the Roxmon.

5.1.1 Alarm syntax

Every alarm has the following fields:

category

A string.

This is the area that the alarm regards or a service. The exact names used in each installation is chosen by the Administrator when Configuring a Macro. A suggestion is chosen by the Macro designer.

It is intended for the Monitor to be able to connect an alarm to one of the services in its diagram and give that service a color depending on the current status. The Administrator drawing the diagram and allocating names for the things on that diagram will have to take these names, located in the Macros, into account to be able to do this. Could be things like `authpriv`, `cron`, `kern`, `lpr`, `mail` ...

severity

A string.

One of `Emergency`, `Alert`, `Critical`, `Error`, `Warning`, `Notice`, `Info`, `Debug`, and `Unknown` that are listed here in falling order of importance. Other strings will be treated as `Unknown`. The `Unknown` severity is a sign of a Macro with bugs or not fully developed.

The severity is normally chosen by the Administrator when Configuring the Macro. A suggestion is chosen by the Macro designer.

alarm

A string.

A small sentence describing what the alarm is about.

details

A string.

This is any amount of text explaining or describing the condition around the alarm.

time

A timestamp.

Allocated automatically by the Agent.

hostinfo

A mapping.

Allocated automatically by the Agent.

This can be used to identify the host that generated the alarm.

START_TIME

An integer.

Allocated automatically by the Agent. Used to uniquely identify an alarm.

SEQUENCE_NUM

An integer.

Allocated automatically by the Agent. Used to uniquely identify an alarm.

The size of an alarm is best to keep less than 16k in order not to deteriorate communication performance.

5.2 Other design aspects of Roxmon

This is where all the rest of the design information of Roxmon is kept.

The ideal approach would be to sort the design details in a systematic order depending on how it is built, however I have chosen to first describe everything that is of any use to the Macro writer and then, in this chapter, the rest of the stuff and that will make this section a little less systematic. You will probably have to go back and forth between this chapter and the previous one (5.1 (What you need to know to write a Macro)).

5.2.1 Agent installation

The agent is installed in the directory /var/opt/Roxmon.

The pike binary and all its auxiliary files are installed in /var/opt/Roxmon/pike with subdirectories bin, include, modules...

Configurations are in directory /var/opt/Roxmon/config and a list of yet unhandled "alarms" are in /var/opt/Roxmon/alarms. The configuration is actually a collection of pike-scripts, macros and their configuration files that "run" within the agent.

```
/var/opt/Roxmon/
                pike/
                     bin
                     include
                     modules
                     ...
                config/
                alarms/
                run/
```

5.2.2 Monitor installation

The monitor is a Roxen module installed in the Roxen server.

5.2.3 Server connection

This chapter describes the communication between the Agent and the Monitor.

Functions initiated from the server

PING

Immediatly responded to by the Agent.

DOWNLOAD

Download request.

FETCH_ALARM_LIST, FETCH_ALARM

A query for alarms.

DELETE_ALARM

Reset an alarm.

UPDATE_ALARM

Update an alarm.

These functions are always answered immediatly (or as fast as possible).

Functions initiated from the client

ALARM

Deliver an alarm

ALARM_DELETED

Somebody else has deleted an alarm.

ALARM_MODIFIED

Somebody else has modified the alarm.

The functions from the client are all asynchronous, they are never acknowledged. The last two are only sent to the server connections that did not initiate the operation.

Protocol syntax All operations and responses delivered are mappings that we have made encode_value on. They are coded using a simple HOLERITH-code: <number of chars>H<data>. The data is an encoded mapping. The function name is the value of the field with the name "op". Arguments are field with other names. For responses the op field has the value "RESPONSE" and the original op is in the field with the name "response".

Details on all operations Here is the complete list of operations with arguments:

Function: "op":"PING" Response: "op":"RESPONSE" "response":"PING" other arguments are returned.

Function: "op":"DOWNLOAD" "name":<name of the function downloaded> (a string used as the filename). "contents":<the complete pike-script> (a string stored in the file). If the "contents" is the empty string, this means that we remove the file and disactivate the function. Response: "op":"RESPONSE" "response":"DOWNLOAD"

Function: "op":"FETCH_ALARM_LIST" Response: "op":"RESPONSE" "response":"FETCH_ALARM_LIST" "alarms":<an array(mapping) with all the alarms names>

Function: "op":"FETCH_ALARM" "alarm":<mapping with the alarm name> Response: "op":"RESPONSE" "response":"FETCH_ALARM" "alarm":<mapping with the alarm information> If there is no such alarm this has the value 0.

Function: "op":"DELETE_ALARM" "alarm":<mapping with the alarm name> Response: "op":"RESPONSE" "response":"DELETE_ALARM" "alarm":<mapping with the alarm name> (same)

Function: "op":"UPDATE_ALARM" "alarm":<mapping with the alarm information> Response: "op":"RESPONSE" "response":"UPDATE_ALARM" "alarm":<mapping with the alarm information> (same)

Async: "op":"ALARM" "alarm":<mapping with the alarm information>

Async: "op":"ALARM_DELETED" "alarm":<mapping with the old alarm information>

Async: "op":"ALARM_MODIFIED" "alarm":<mapping with the new alarm information>