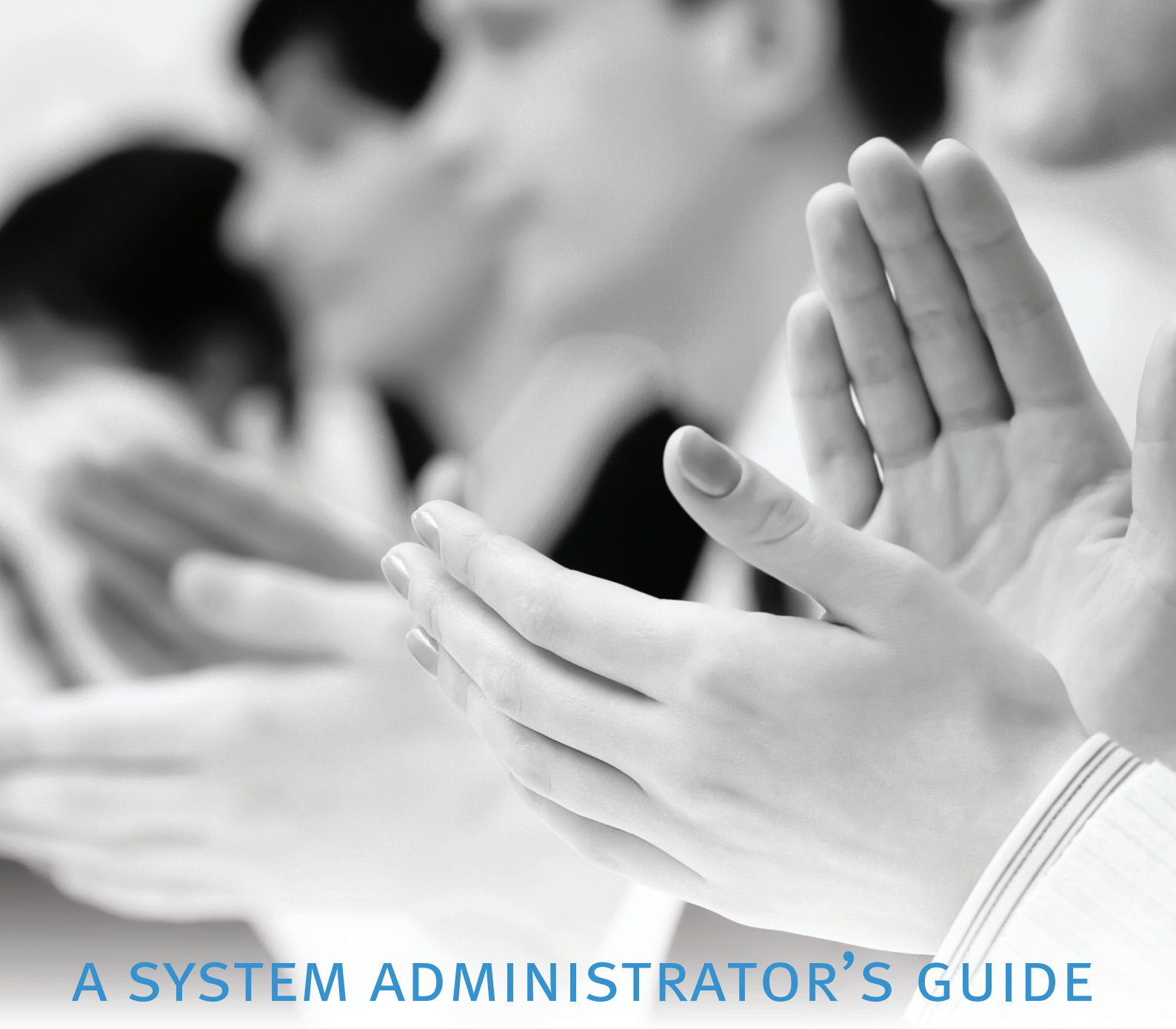


A SYSTEM ADMINISTRATOR'S GUIDE TO MONITORING AVAMAR

EMC Proven Professional Knowledge Sharing 2011

Kelly Mills
Solutions Architect
EMC Corporation
Kelly.Mills@EMC.com

Table of Contents

Executive Summary

Backup and recovery infrastructure is critical to the life of any business. Longer data retention requirements and legal regulations for data storage and retrieval contribute to the need for a robust but agile backup solution. Given strict recover time and recover point objectives that define many service level agreements, the backup team operates around the clock. This group must be conscious of costs and often work with constrained budgets. A monitoring solution needs to be dependable and cost effective.

EMC Avamar® is a software/hardware backup and recovery solution that takes advantage of data commonality in the environment and deduplicates the data at the source client and globally at the Avamar server. Avamar detects changes at the subfile level and uses local cache files to speed file comparison and processing. The result is fast client-side processing, low network bandwidth utilization (as only the changed blocks are sent to the Avamar server), and lower storage consumption. Although Avamar includes mechanisms for self-alerting, external monitoring should be added.

Nagios is a powerful enterprise-class monitoring system that enables organizations to quickly identify and resolve infrastructure problems. This Open Source tool—with a large community of developers and support—is easily customizable for your situation and add-ons can be created to meet your needs.

Syslog-NG is a leading Open Source solution for log collection and management. It provides a central logging set up that can be used to filter messages with tools such as SWATCH.

Introduction

This article will describe a general solution for monitoring the Avamar environment from a Systems Administrator viewpoint using Nagios and Syslog. It will highlight some of Avamar's built-in utilities and focus on how they can interact with external tools.

Audience

This article is intended for Avamar administrators with knowledge of UNIX/Linux administration. It is not intended as a step-by-step guide for installing and configuring each package but to make the user aware of the capabilities.

Avamar Architecture

Avamar is a disk-based backup solution. The Avamar software installs on top of EMC-provided hardware or qualified customer hardware. This article will focus on the EMC-provided data storage solution. Hardware configurations can consist of single node or multiple node implementations. A node is a physical server running the Red Hat Linux operating system. The two basic nodes are the Utility node and the Data node. The Utility node provides internal Avamar server processes and services, including the administrator server, jobs scheduling, authentication, and maintenance and web access. The Data node is dedicated to providing storage for the backups. In an Avamar single node implementation, the Utility and Data node functions reside on the same physical node. In a multiple node configuration, the Utility node is its own dedicated physical node. The amount of capacity and protection needed will determine the number of Data nodes.

Avamar Notification Features

Avamar system activity and operational status is reported as various event codes to the Utility node MCS service. Examples of events include maintenance activity status, failed logins, or client activations. Each event contains useful information such as the event code, date and time stamp, category, type, summary, and what generated the event.

Notification options can consist of email messages, pop-up alerts, Syslog, SNMP, and events sent directly to EMC support. For this article, we will use the Syslog feature to achieve our desired results. Notifications are configured as Profiles via the Avamar Console.

Setting up the SYSLOG environment

The building blocks for setting up the SYSLOG monitoring environment will be in this order:

1. Identifying a server to act as the Monitoring server
2. Setting up the SYSLOG-NG server
3. Setting up SWATCH to filter the SYSLOG-NG messages
4. Configuring SYSLOG on the Avamar nodes
5. Setting up the Avamar SYSLOG Profile

You will need a new or existing Linux/UNIX server in your environment that can be used as the central monitoring server. This sever can be a virtual machine or physical server. You need to install the base operating system such as Red Hat, Solaris, HPUX, or flavor of your choice. For details on installing the OS and packages, please consult the respective OS manuals.

Download a copy of SYSLOG-NG for your OS to your central monitoring server. You will need to disable the syslog service that comes with your OS before installing. Install the SYSLOG-NG package. You can edit the syslog-ng.conf file to customize it with you preferences. For example, you can have SYSLOG-NG create a directory for each host that sends it a message and name the log file using your standard naming convention. Here is an example:

```
destination d_rmt  { file("/pool1/syslog/$HOST/$YEAR$MONTH$DAY" owner(root) group(root) perm(0600) dir_perm(0700) create_dirs(yes)); };
```

```
log { source(s_udp); filter(f_all);      destination(d_rmt); };
```

Start the SYSLOG-NG process. Verify the process is running by using the “ps” command.

SWATCH, a free tool that uses pattern matching to filter log messages, will monitor log files as they are written in real-time. As patterns are matched, SWATCH will then carry out specific actions as defined in the configuration file. Those actions can be simple email notifications or execution of a script. Download a copy SWATCH for your OS platform and install as per the platform.

Now we must edit the syslog-ng.conf file to have it call SWATCH when a new message arrives and process it. Here is an example:

```
#swatch to read from stdin
destination d_swatch { program("/usr/local/sbin/swatch -c /export/home/swatcher/swatchrc --read-pipe=\"cat /dev/fd/0\""); };
```

```
log { source(s_udp); filter(f_all);      destination(d_swatch); };
```

After the syslog-ng.conf file has been updated, we will need to configure the SWATCH configuration file to look for certain patterns and send notifications. You can use regular expressions for pattern matching. Please see a sample Avamar SWATCH configuration in the Appendix. In this example, we are watching for the Avamar code “22605”, which is a Node Offline event. We use the keyword *watchfor*.

```
##OFFLINE NODES
watchfor =  /<Code> 22605/
```

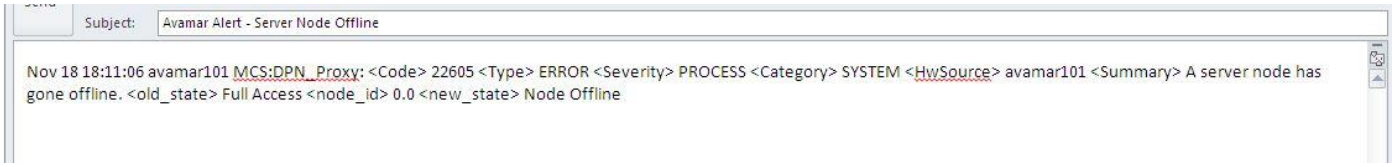
We then can use the keyword *mail* or *pipe* to act upon that match. In this example, we are using the *pipe* keyword to send the text of the message to a homegrown internal script called *notify* which then sends it along to a recipient.

```
pipe /export/home/report/notify "Avamar Alert - Server Node Offline" "user@email.com"
```

The complete stanza would look like this:

```
##OFFLINE NODES
watchfor =  /<Code> 22605/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Node Offline" "user@email.com"
```

Here is an example of the EMAIL alert message:



The SYSLOG-NG service needs to be restarted each time that the swatchrc file is updated. SWATCH does have the ability to point the swatchrc configuration file against an existing log file for testing.

Avamar now needs to be configured to take advantage of the central monitoring server.

We must add a line to the /etc/syslog.conf file on **ALL** the Avamar nodes (including the spare node on multiple node implementation). This allows us to capture all system messages regardless of if the Avamar application is running. It also provides a central location for archiving syslog messages. The entry in the syslog.conf file on each Avamar node needs to point to the central monitoring server. For example, if the central monitoring server had IP 1.2.3.4, it would look like this:

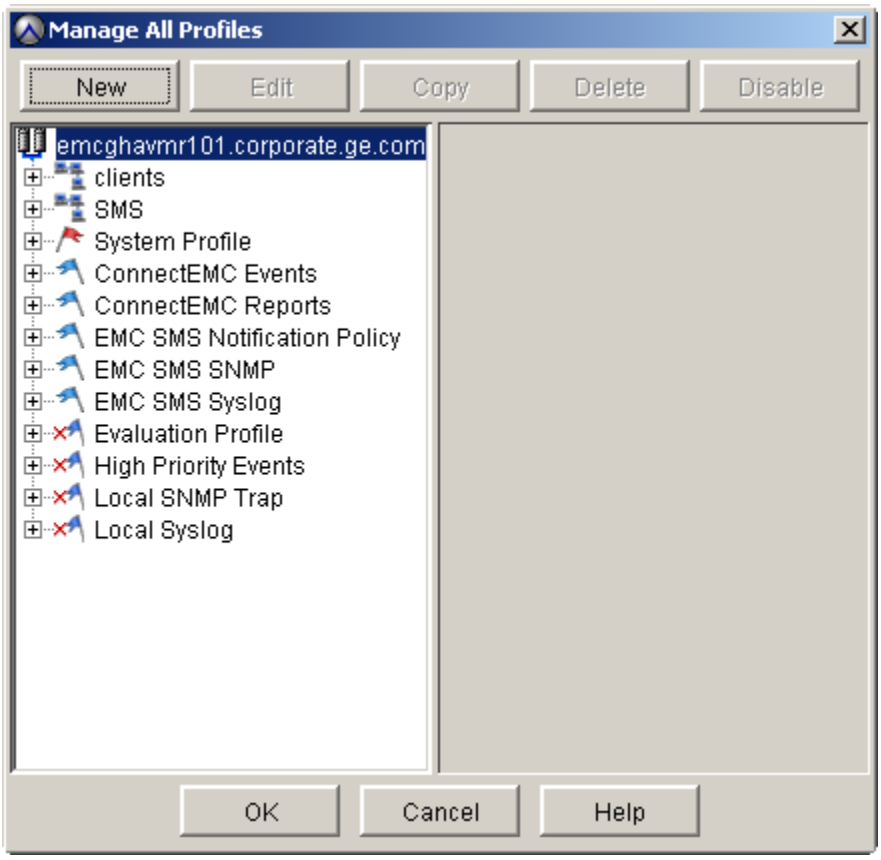
```
*.*      @1.2.3.4
```

You will then need to restart the syslog service on the Avamar node using the command "service syslog restart".

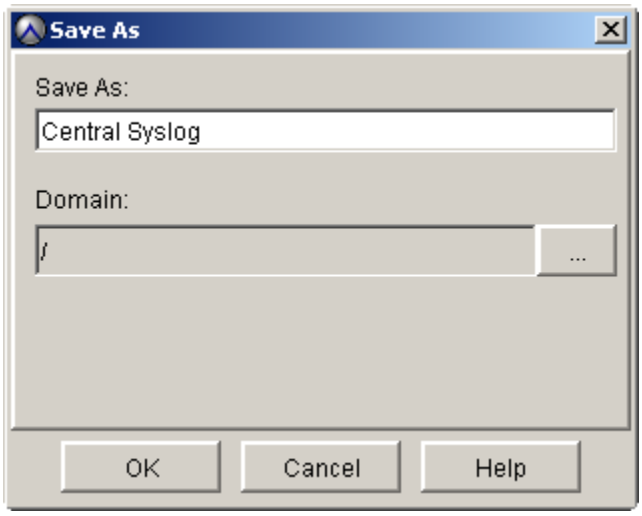
```
Shutting down kernel logger:                               [  OK  ]
Shutting down system logger:                               [  OK  ]
Starting system logger:                                    [  OK  ]
Starting kernel logger:                                    [  OK  ]
```

Back on the central monitoring server, in the location you configured for the host directories to be created, you should now see a directory for this Avamar node. That directory will contain a log file with a new entry.

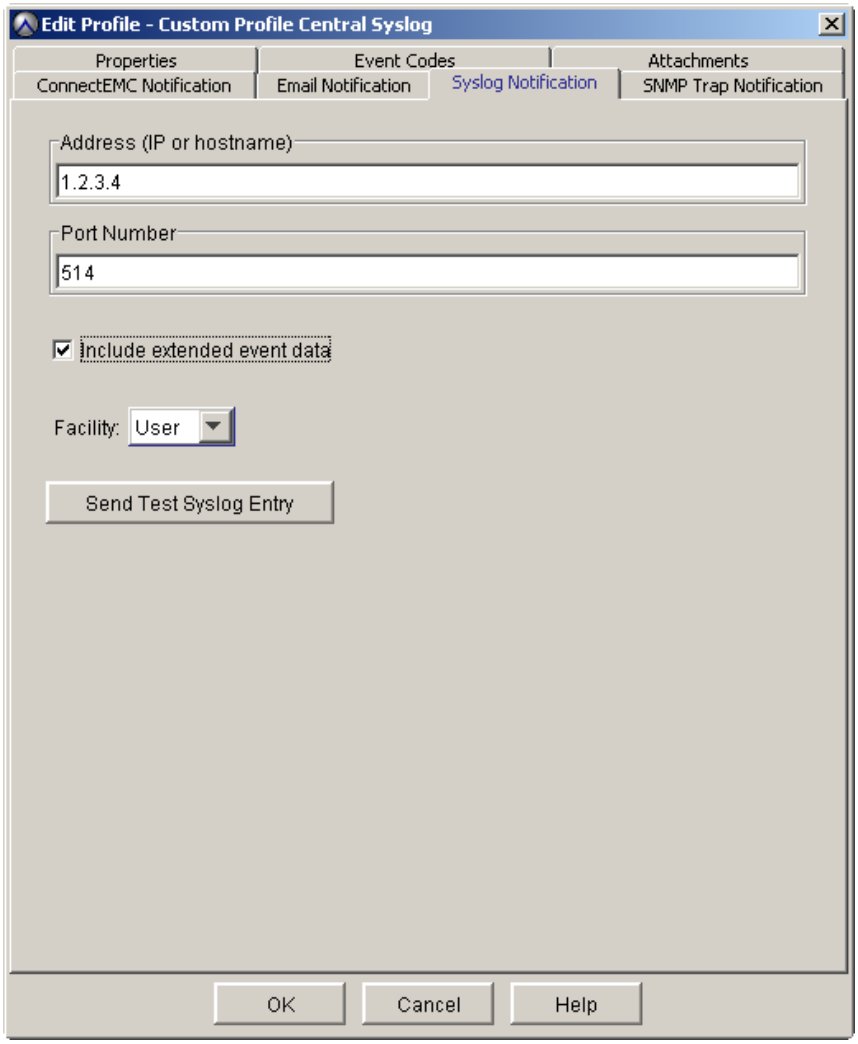
To capture Avamar application-generated syslog events, we need to create a profile in the Avamar console. Launch the Avamar console. From the Tools menu, click Manage Profiles.



Click on Local Syslog to highlight and then click Copy. Save the new profile.



Highlight the new profile and click Edit from the top menu. Click on the Syslog Notification tab.



Enter the IP address of the central monitoring server. The port will stay at the default of 514. Check the “Include extended event data” checkbox. This option causes the alert message to be delimited using tags for easier parsing by SWATCH. Click OK.

Highlight the new profile you created and then click Enable from the top menu.

A syslog central monitoring system is now in place. The benefits to this setup include:

- ALL default Avamar events are sent to central monitor for processing. This eliminates the need to pick and choose what alerts you want Avamar to send from the long list.
- In a multiple Avamar grid environment, you wouldn’t have to log in to each console and modify events that are no longer needed as this is done centrally.
- Syslog on the physical nodes will send server-based events outside of the Avamar application that might be specific to that hardware platform.
- In a multiple node configuration, the Spare node is now monitored using Syslog so that you can fix any issues before the need arises to use it.
- A central SWATCH configuration provides a quick way to add alerts or notification options.

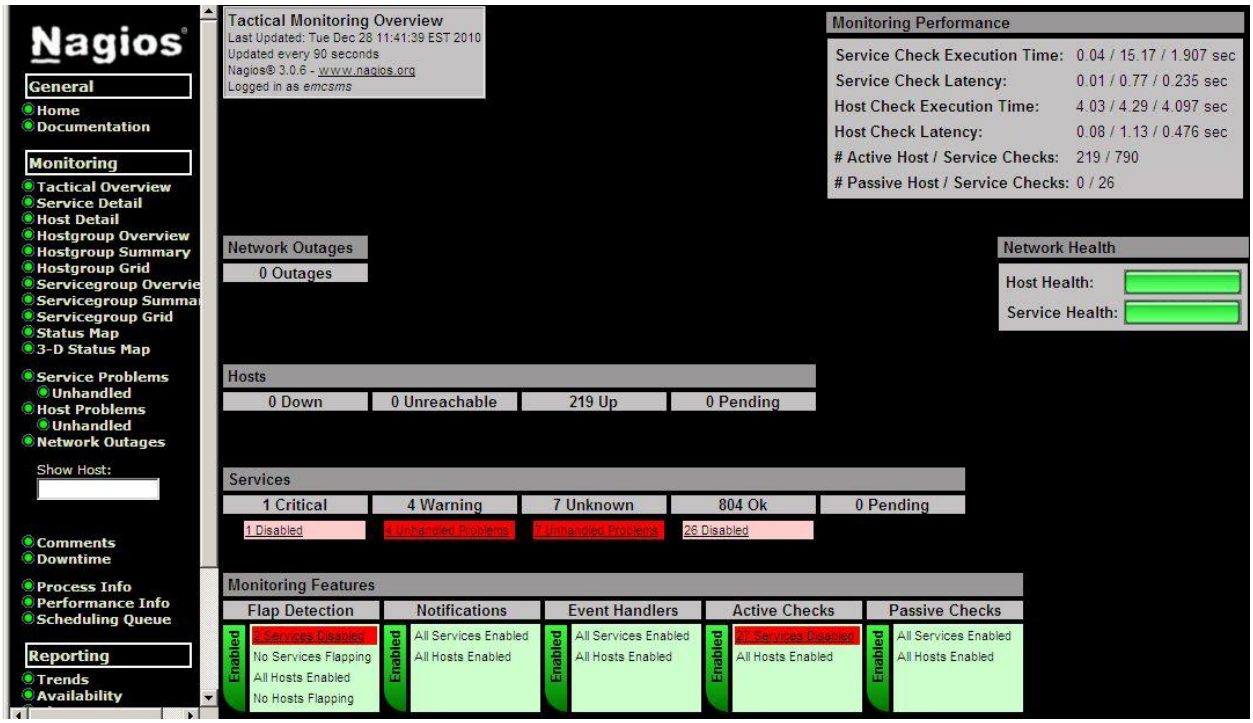
Nagios Overview

Nagios is a graphical monitoring application that can easily provide tactical overviews of your environment. You can monitor physical devices as well as process and services running on those devices. You can further customize Nagios by placing hosts and services into groups. Nagios provides methods for creating users and user groups. The notification section is highly configurable to meet your reporting needs. Nagios can monitor your environment in several ways.

1. Active Checks: Nagios will use protocols such as PING and HTTP to externally connect to devices. Nagios can also use SSH or NRPE to connect to a device and run a plugin (script).
2. Passive Checks: An outside process contacts Nagios.

Nagios comes with many built-in plugins and commands that will enable you to start monitoring servers very quickly.

Here is a screen shot of the tactical overview page:



Sample Nagios Configuration

Once you download and install Nagios on your central monitoring server, you will need to set up some configuration files. First define contacts and contact groups in the Nagios contacts.cfg and contactsgroup.cfg file respectively. This will allow you to notify certain individuals or groups of people. You can set time ranges for notification as well as what types of severity alerts should be sent to which parties.

In this example, we will set up Nagios to “ping” our Avamar nodes to determine if they are up. For simplicity you can add all your nodes into a file called avamar_hosts.cfg. This file defines ALL Avamar nodes.

A typical entry looks like this for each node:

```
define host{
use     linux-server
host_name       localhost
alias   New york
address         1.2.3.4
}
```

You can also create an avamar_group.cfg file that would group certain Avamar nodes together. For example, hostgroup “avamar-servers” would contain ALL nodes, whereas hostgroup “avamar-utility-servers” would only contain the utility nodes. A typical entry would look like this:

```
define hostgroup{
hostgroup_name  avamar-servers
alias           ALL Avamar Servers
members         localhost
}
```

Now we can create a service configuration file that defines what we want Nagios to check for. This file contains default settings such as name of the service, what checks are allowed, if notifications are enabled, time period that checks are allowed, the contact group, polling interval, and so on.

In this example, we created a file called avamar_service.cfg. This file contains our default settings for whenever this service is used. It also contains our “ping” check. Here is an example of that check.

```
# Define a service to "ping" the local machine

define service{
        use                             avamar-service
        hostgroup_name                  avamar-servers
        service_description             PING
        check_command                   check-host-alive
        }
```

As you can see, we are using the avamar-service default settings, we are applying this check against the hostgroup “avamar-servers”, the service description name will show up as PING in the browser, and last, we are calling the built-in command “check-host-alive”.

Once all configuration changes are made, the Nagios service needs to be reloaded using the /etc/init.d/Nagios reload command. If errors are detected, you can troubleshoot by calling the Nagios binary and pointing to the configuration file, i.e. /usr/local/Nagios/bin/Nagios –v ../etc/Nagios.cfg

Here is a screen shot of the PING service for our node “localhost”:

Service Status Details For Host 'localhost'

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	PING	OK	12-28-2010 12:32:27	532d 1h 58m 43s	1/4	PING OK - Packet loss = 0%, RTA = 0.21 ms

You can also drill down into the service for more detail:

Service State Information

Current Status:	OK (for 532d 2h 2m 2s)
Status Information:	PING OK - Packet loss = 0%, RTA = 0.23 ms
Performance Data:	rta=0.230000ms;100.000000;500.000000;0.000000 pl=0%;20;60;0
Current Attempt:	1/4 (HARD state)
Last Check Time:	12-28-2010 12:37:27
Check Type:	ACTIVE
Check Latency / Duration:	0.148 / 4.045 seconds
Next Scheduled Check:	12-28-2010 12:42:27
Last State Change:	07-14-2009 11:36:49
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	12-28-2010 12:38:45 (0d 0h 0m 6s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Service Commands

- Disable active checks of this service
- Re-schedule the next check of this service
- Submit passive check result for this service
- Stop accepting passive checks for this service
- Stop obsessing over this service
- Disable notifications for this service
- Send custom service notification
- Schedule downtime for this service
- Disable event handler for this service
- Disable flap detection for this service

Service Comments

Add a new comment Delete all comments

Entry Time	Author	Comment	Comment ID	Persistent	Type	Expires	Actions
This service has no comments associated with it							

Advanced Nagios Configuration

PLUGINS

Nagios can be set up to do very elaborate checks via plugins on the nodes. You might want to have Nagios check node capacity or GSAN status. The plugin on the node can return information to be displayed in the browser window and depending on the exit code, it can trigger the appropriate alert and notification. There are two ways that Nagios can connect to the server and run the plugin.

1. NRPE – this is a daemon process that is installed and runs on each server to be monitored
2. SSH – With the use of SSH keys, Nagios can connect to the server and run commands

In this example, we will connect to a Utility node using SSH and run a plugin to see if any of the nodes are above 60% utilization. On the Avamar Utility node, we have created a script called "nodeutil". The purpose of this script is to check each node and put Nagios into a warning state if the node utilization is between 60% and 62%. It will put Nagios into a critical state if the node utilization is 63% or higher. Here is what Nagios looks like after the check is run:

Node utilization Status	OK	12-28-2010 10:38:34	16d 3h 13m 11s	1/3	OK - all nodes below 60% utilization

We can also connect to the utility node and run other plugins to check Avamar-specific services, for instance, the status of the GSAN. The logic of your plugin needs to be aware that the GSAN status changes based on the maintenance job that is running and should account for that. This prevents false Nagios alerts. Here is a screen shot of more checks that you can monitor:

Service	Status	Last Check	Duration	Attempt	Status Information
Capacity Status	WARNING	12-28-2010 10:42:23	103d 2h 52m 48s	3/3	Online Nodes= 16 / Avg Capacity Used= 55.9312% / UTIL= 84%
Checkpoint	OK	12-08-2010 11:18:43	485d 4h 42m 29s	1/1	Checkpoint Complete
Dell Open Manage - HTTP	OK	12-28-2010 13:52:18	531d 14h 28m 6s	1/3	HTTP OK HTTP/1.0 200 OK - 2788 bytes in 0.064 seconds
EMS Status	OK	12-28-2010 13:27:20	531d 0h 24m 0s	1/3	EMS = up
Enterprise Manager - HTTP	OK	12-28-2010 13:34:52	496d 23h 34m 29s	1/3	HTTP OK HTTP/1.1 200 OK - 14693 bytes in 0.099 seconds
GSAN Status	OK	12-28-2010 13:42:45	35d 2h 59m 58s	1/3	GSAN = ready
Garbage Collect	OK	12-09-2010 09:05:26	19d 4h 50m 51s	1/1	Garbage Collect Completed
MAINTENANCE Status	OK	12-28-2010 13:52:23	531d 0h 18m 40s	1/3	MAINT = enabled
MCS Status	OK	12-28-2010 13:27:20	531d 0h 16m 0s	1/3	MCS = up
Node utilization Status	OK	12-28-2010 10:38:34	16d 3h 17m 43s	1/3	OK - all nodes below 60% utilization
PING	OK	12-28-2010 13:48:04	297d 7h 27m 5s	1/3	PING OK - Packet loss = 0%, RTA = 0.43 ms
SCHEDULE Status	OK	12-28-2010 13:52:24	126d 23h 53m 53s	1/3	SCHED = up
SSH	OK	12-28-2010 13:52:29	531d 2h 29m 34s	1/3	SSH OK - OpenSSH_3.9p1 (protocol 2.0)

PASSIVE CHECKS

Passive checks can also be used to update Nagios about events. In this example, we will tie together the SYSLOG/SWACTH setup used earlier to contact Nagios. Recall that SWACTH can run a script if a pattern is matched. Nagios has a built-in command that you can call and assign variables. In this case we have SWACTH call a homegrown script called “alert_nagios” that looks at the error message to determine the node name and then calls the Nagios command “submit_check_result” .

Here is our SWATCH configuration to check for Garbage Collect failure:

```
##Garbage Collect Failed
watchfor = /<Code> 4202/
                pipe /export/home/report/notify "Avamar Alert - Failed garbage collection" "user@mail.com"
                pipe /export/home/report/alert_nagios "Garbage_Collect" "2" "Garbage Collect Failed"
```

Here are the “alert_nagios” script contents:

```
#!/usr/bin/bash -x

#this is a nagios alert script to be called by other programs such as swatch

#arg 1 = service
#arg 2 = code
#arg 3 = label

#stick input into variable
MESSAGE=`cat`


# lets see if we can get the hostname from the messgae
NAME=`echo $MESSAGE | awk '{print $4}'`

SERVICE=$1
CODE=$2
LABEL=$3

#run it
#$NAME $SERVICE $CODE "$LABEL"

/usr/local/nagios/libexec/eventhandlers/submit_check_result $NAME $SERVICE $CODE "$LABEL"
#
```

Here is what Nagios looks like after the check runs:

	Service	Status	Last Check	Duration	Attempt	Status Information
	Garbage_Collect	CRITICAL	12-09-2010 06:29:59	19d 6h 40m 39s	1/1	Garbage Collect Failed

You can also update Nagios the same way to show that a check was successful.

Here is our SWATCH configuration to check for Garbage Collect success:

```
##Garbage Collect Complete
watchfor = /<Code> 4201/
                echo normal
                #pipe /export/home/report/notify "Avamar Info - completed garbage collection" "user@email.com"
                pipe /export/home/report/alert_nagios "Garbage_Collect" "0" "Garbage Collect Completed"
```

Here is what Nagios looks like after the check:

Conclusion

Monitoring Avamar includes not only the application itself but the devices that make up the Avamar infrastructure. Syslog and Nagios are free tools that provide limitless options for building a solid central monitoring solution. These tools extend the packaged Avamar features to create a comprehensive monitoring system. The added capabilities and automated coverage help alleviate the time you spend worrying about backups.

Appendix

SAMPLE SWATCH AVAMAR CONFIGURATION

```
##AVAMAR ALERTS ################
##OFFLINE NODES
watchfor =  /<Code> 22605/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Node Offline" "user@email.com"
                throttle 60:00
##Checkpoint Overdue
watchfor = /<Code> 22408/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Checkpoint of Server is overdue" "user@email.com"
                pipe /export/home/report/alert_nagios "Checkpoint" "2" "Checkpoint Overdue"
##Checkpoint Failure
watchfor = /<Code> 4302/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Checkpoint of Server Failed" "user@email.com"
                pipe /export/home/report/alert_nagios "Checkpoint" "2" "Checkpoint Failed"
##Checkpoint Complete
watchfor = /<Code> 4301/
                echo normal
                pipe /export/home/report/notify "Avamar Info - Checkpoint of Server is complete" "user@email.com"
                pipe /export/home/report/alert_nagios "Checkpoint" "0" "Checkpoint Complete"
##HFS check Failed
watchfor = /<Code> 4004/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Failed HFSCHECK" "user@email.com"
##Garbage Collect Failed
watchfor = /<Code> 4202/
                echo normal
                #pipe /export/home/report/notify "Avamar Alert - Failed garbage collection" "user@email.com"
                pipe /export/home/report/alert_nagios "Garbage_Collect" "2" "Garbage Collect Failed"
watchfor = /<Code> 4203/
                echo normal
                #pipe /export/home/report/notify "Avamar Alert - Failed garbage collection" "user@email.com"
                pipe /export/home/report/alert_nagios "Garbage_Collect" "2" "Garbage Collect Failed"
##Garbage Collect Complete
watchfor = /<Code> 4201/
                echo normal
                #pipe /export/home/report/notify "Avamar Info - completed garbage collection" "user@email.com"
                pipe /export/home/report/alert_nagios "Garbage_Collect" "0" "Garbage Collect Completed"
##Server generated event
watchfor = /Failure reported/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Event Reported" "user@email.com"
                throttle 60:00
##Server generated event
watchfor = /Failure detected/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Event Detected" "user@email.com"
                throttle 60:00
##Server generated event
watchfor = /eth0 NIC Link is Down/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Event Detected" "user@email.com"
                throttle 60:00


##Server generated event
watchfor = /Device failed/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Event Detected - Device Failed" "user@email.com"
                throttle 60:00
##Server generated event
watchfor = /Memory device status is critical/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Event Detected - Device Failed" "user@email.com"
                throttle 60:00
##Server generated event - disk suspended/degraded
watchfor =  /<Code> 22632/
                echo normal
                pipe /export/home/report/notify "Avamar Alert - Server Event Detected - Device Failed" "user@email.com"
                throttle 60:00
```