Using EMS to Increase Availability in Mission Critical Environments

Deborah Hamilton
Hewlett Packard
19111 Pruneridge Avenue
Cupertino, CA 95014
(408) 447-3338
debbie_hamilton@hp.com
Fax (408) 447-1053

Abstract

As the business impact of turning away customers increases, more and more solutions are becoming mission critical. As a result, companies are becoming more concerned about increasing the availability of their applications and systems. One way to increase availability is to monitor critical system resources in order to detect faults as early as possible and possibly even prevent faults. In addition, monitoring of system resources allows companies to plan for when they will need to grow hardware and/or application resources to meet increasing demands.

The Event Monitoring Service (EMS) is a framework that allows EMS-compatible monitors to "plug-in" and monitor important system resources. There are EMS-compatible monitors for many system resources ranging from hardware (disks, memory, and CPUs) to kernel resources, database resources and MC/ServiceGuard resources. The EMS framework allows users and applications to configure monitors and set event notification mechanisms.

This paper describes how EMS works, what notification mechanisms are available, what kind of resources can be monitored and how to integrate EMS into an application. It also describes how EMS fits into fault management and system management solutions. This paper also gives a list of additional sources of information for EMS.

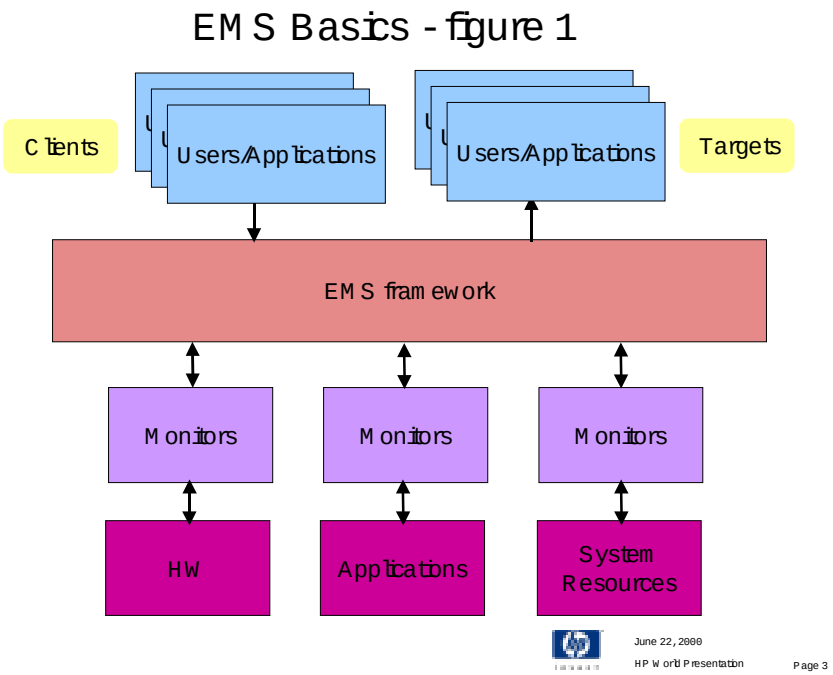
EMS BASICS

The Event Monitoring Service (EMS) is an event management framework that runs on and is provided free with HP-UX. EMS is also distributed free with the Diagnostic Media on the Independent Product Release. EMS serves as a coordinator of event information and provides a consistent event interface for providers (i.e. monitors), consumers (i.e. targets), and administrators (i.e. clients). Monitors plug into the EMS framework allowing users and applications to use a single interface to configure the monitors and set up event information and notification mechanisms. EMS starts and stops the monitors, gets and stores information used by the monitors, polls the monitors for event information if necessary and directs monitors to send events to the appropriate place. For example, an application or user can tell EMS that it wants to know when a disk fails so that the user/application can recover or replace the bad disk. EMS will let the monitor know that there is a request for disk failure information. If a disk failure does occur, EMS will determine who has requested event information and send out the notification to the user/application using the appropriate mechanism.

We refer to applications that set up requests for event information as *clients*, applications that receive event information as *targets* and applications that interface with the resource being monitored as *monitors*. The same application can be both a *target* and a *client*. Users can act as a *client* by using the EMS GUI to set up event notification requests. See figure 1 for a picture of clients, targets, monitors and the framework.

Targets can use the event information in any way they choose - for recovery and fault management (e.g. roll over to another node when there is a critical failure) or for proactive system planning (e.g. when CPU utilization is beyond a certain amount, add new processors). EMS conveys information to targets - it is up to the target to determine how to use that information.

EMS has a defined API for the monitors to use to interface with the EMS framework. There is a free development kit available for monitor writers to use for creating monitors. This kit includes example code for an EMS-compatible monitor. See the section on *additional sources of information* for the web page that provides a download of the development kit and provides further information.



Clients can configure the monitoring to fit their needs by selecting the devices or resources to monitor, identifying the event or threshold, choosing the mechanism of notification, providing comments to send with the event notifications, and by selecting the threshold intervals for polling. The following are available notification mechanisms:

- SNMP traps
- Email
- Syslog
- OPC messages
- Console
- TCP/UDP messages
- text log

This means that event information can be sent to OpenView/VantagePoint Operations, directly to any SNMP-capable management station, to a network application listening on a TCP or UDP port, to any e-mail address, locally to the console or to any system or regular log file.

Clients can also set up events with severity classifications. Clients can use this severity to distinguish between an event that is severe enough to cause failover verses one that is more of a warning or purely informational. For example, ServiceGuard uses the severity level to determine when to failover to another node.

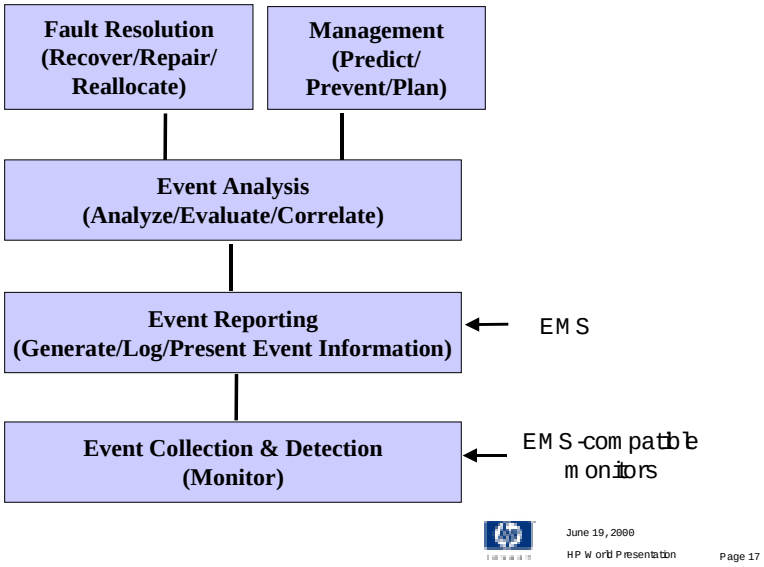
EMS supports multiple kinds of monitors:

- Monitors that write to a file that EMS checks. EMS provides event notifications based on changes to this file.
- Monitors that don't store state information. EMS keeps information on how frequently to poll the monitor for information and then determines when a state change has occurred and sends an event notification when appropriate.
- Monitors that store there own state information and are smart enough to send EMS notices of events.

In addition, the EMS GUI can be used for discovery of resources that are available for monitoring. Clients can define conditions that indicate when notification events should be sent. It can be at periodic intervals, when a component state changes or when a pre-defined threshold condition is met. EMS can either poll or wait for an event, depending on how the monitor works.

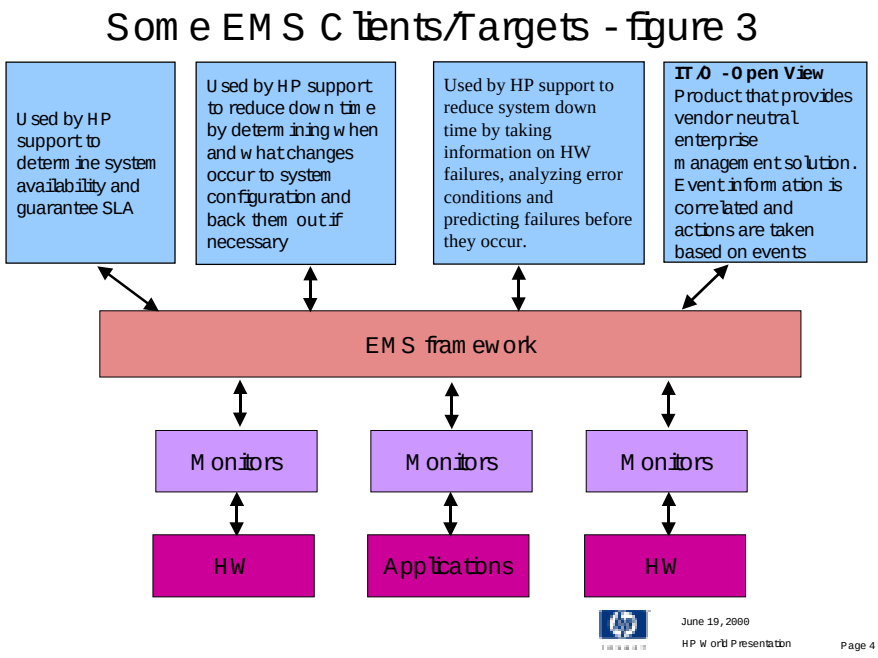
How EMS fits into Mission Critical Solutions

Logical components of Fault/System Management
Figure 2



Fault management and system management can be broken into five logical components: Event information collection, event reporting and distribution, event analysis, fault resolution and system management. See figure 2 for a diagram of these components. EMS provides event reporting and distribution functionality shown in this diagram. EMS-compatible monitors provide event collection and detection. Appendix A lists the system resources that can be monitored through currently available EMS-compatible monitors to provide the event collection and detection functionality in this picture. These monitors are available through multiple channels: some are available with diagnostics media, some are available from EMS, and some come with hardware or systems that use them.

There are multiple clients and targets that provide event analysis, fault resolution and management functionality using EMS. For example, OpenView/VantagePoint Operations uses EMS to get some event information. (It also gets event information through other mechanisms). OpenView uses that information to provide additional levels of functionality - event analysis, fault resolution and management. ServiceGuard also uses EMS to get event information and then uses that information for analysis and fault resolution (i.e. failing over to another node). Other EMS clients/targets include Network Node Manager, Tuxedo, and TopTools. See figure 3 for a picture of a few other EMS clients/targets along with a statement about the event analysis, fault resolution and management functionality they provide.



Using EMS with fault and system management solutions

Some fault and system management solutions have EMS integrated and require no modifications in order to take advantage of EMS event information. The user does not need to know how the event information is configured and distributed. Some solutions, however, require users to setup requests for event information and distribution. Additionally, some users want to write their own applications that take advantage of EMS event information and some users want to set up requests for event information directly through the EMS GUI so they know what is going on and can manually react if there is a problem. This section describes where to get more information for the users that want to take advantage of EMS event information.

VantagePoint Operations (OpenView)

EMS provides additional types of event information to VantagePoint Operations and therefore is a complement to VantagePoint Operations. EMS does not replace VantagePoint Operations functionality. VantagePoint Operations provides a more generic monitoring solution in addition to allowing actions to be taken based on events. VantagePoint Operations also has filtering and correlation capabilities. EMS is focused mostly on monitoring HP-UX resources and only serves as an event distribution mechanism. There are two ways to set up VantagePoint Operations to use EMS event information:

- Set up EMS notifications to be sent to VantagePoint Operations via OPC messaging (OPC messaging is supported by both VantagePoint Operations and EMS for communication)
- Set up the EMS-provided threshold monitor template (see the section below on "sources of addition information" to get the web page which provides the template). The template lets VantagePoint Operations know what to do when an EMS event is received.

Toptools

EMS events can be sent to the TopTools event repository. Users can view the events just like other Toptool events. In order to do this, the user goes through the EMS GUI to set up the requests for event information to be sent to TopTools using SNMP traps. See appendix B for a description on how to use SNMP traps as a mechanism for receiving EMS events.

ServiceGuard and ServiceGuard OPS Edition

ServiceGuard and ServiceGuard OPS Edition use EMS to determine the health of resources such as disks and networking devices. These solutions use notifications from EMS to help determine when to failover a mission critical application (ServiceGuard Package) from one node to another. The user can configure ServiceGuard to make packages dependent on EMS monitored resources and thus increase the information used to determine failovers. The basic EMS configuration is done as part of the ServiceGuard and ServiceGuard OPS Edition setup.

Any SNMP-capable management station

Any SNMP-capable management station can receive EMS events by setting up the event notifications to be sent via SNMP traps. In addition, the management station must be configured to receive the SNMP traps. See appendix B on how to do this.

Other applications

Other applications such as event browsers can take advantage of EMS event information by configuring EMS to put event notifications in syslog or text log. In addition, an application can use tcp/udp to receive EMS event notifications. This is described in the development kit (see the section on additional sources of information).

Plans for EMS

In the next year, EMS is focusing on supporting rapid deployment capabilities, the 11i HP-UX release and the IA-64 platform.

Additional sources of information

- For more information about Unix Fault Management, see the book "Unix Fault Management" written by Brad Stone and Julie Symons. This book provides information about many event management solutions and includes a description of EMS.
- For EMS information, a description on how to use and trouble shoot EMS and how to write monitors, or to get the monitor development kit or VANTAGEPOINT OPERATIONS templates, go to the web page - http://www.software.hp.com/products/EMS/index.html
- See appendix A for a list of existing EMS compatible monitors.
- See appendix B for information on setting up SNMP traps so that applications can receive event information from EMS.
- Documentation on EMS can be ordered from HP - (*Writing Monitors for EMS* - part number B7611-90002, *EMS Hardware Event Monitoring User's Guide* - part number B6191-90011, *EMS Hardware Monitors Reference Manual* - part number B6191-90012)
- For information on setting up EMS with ServiceGuard or VantagePoint Operations, or NNM (Network Node Manager) go to http://www.software.hp.com and select High Availability.
- For information on how to set up CA UniCenter to use EMS, contact the CA field organization.
- For information on HP-UX High Availability solutions (including EMS), go to http://www.docs.hp.com/hpux/ha.

Appendix A

Here are resources that can be monitored currently through EMS compatible monitors. The monitors are available through multiple channels: some are available with diagnostics media, some are available from EMS, and some come with hardware or systems that use them. Some of the monitors are free and others are sold.
(This is not a complete list)

- Mass Storage
  - Model 30/FC disk array, Model 10 & 20 SCSI disk arrays
  - Model 12H & 12 disk array
  - Autoraid disk array
  - Model C243XHA Fast-Wide SCSI disk array
  - HP SureStore E FC60 disk array
  - HA Storage System (enclosures)
  - SCSI disk drives
  - SCSI tape devices, libraries & autoloaders
- FibreChannel Hardware
  - Gigabit FC Switch
  - FC Arbitrated Loop Hub
  - FC SCSI Mux
  - FC Adapters
- System Hardware
  - Processor data cache (lpmc) (parity errors, deallocates a processor where appropriate and possible)
  - Core hardware (fans, power supplies, system bus, temp, etc)
  - Memory (memory errors, single bit error rates, page deallocation rates, page deallocation thresholds)
  - Hardware configuration changes
- Kernel/OS/software
  - System resources (status, number of users, job queue size, file system free space, system and process table full utilization, threshold, swap space, file system (full/capacity))
  - Kernel resources (nproc, nflocks, nfile, ncallout, shmmni, semmni, semns, msgmni msgseg, msgtq1)
  - Cluster, node and package status
  - DataBase (status, table size, allocated, used, usage, server, connects, uptime, started, allowed max connect, peak connects, commits, commits per sec, disk reads, disk reads per sec, logical reads, logical reads per sec, read cache hit rate, disk writes, disk writes per sec, logical writes logical writes per sec, write cache hit rate)
  - LVM (logical volume status, copies and summary, mirrored copies of data)
  - Disk (physical volume status, links, summary)
- Networking/IO
  - SCSI Adapters
  - LAN interface status
  - ATM interface
  - OSI interface
  - Hyperfabric interface

Appendix B

EMS 3.x SNMP Trap Specification

Written by Julie Symons

This document provides the specification for SNMP Traps generated by EMS monitors for EMS 3.x. It includes all the traps that can be sent by EMS, the varbinds (variables), descriptions of the possible values for each varbind, and the conditions under which they occur. Also included here are the actions that could be taken to get more information about given events. Templates can be developed using this specification to format SNMP traps into event messages.

Configuring EMS to Send SNMP Traps

When configuring EMS, you can select notification at each polling interval, when the monitored value changes, or when a certain threshold condition has been met. In the EMS GUI, these options appear as

- "At each interval" (for polling)
- "When value changes" (for change notification)
- "When value is <operator> <value>" (for threshold conditions).

Monitoring requests should be configured for SNMP notification ("Notify Via"). Trap severity is configured by the client (e.g. EMS GUI), but is actually set by the target receiving the trap, based on the trap id. EMS trap notification depends on the emsagent and snmpd processes. Traps are sent to destinations listed in the file /etc/SnmpAgent.d/snmpd.conf. Make sure this file contains the IP address of the management station so that SNMP traps will be sent to the intended target.

User data can be supplied when configuring the monitoring request. When using the EMS GUI, the comment field is used to input user data. User data is not sent in the trap, but the user data flag will be set if user data exists for this event.

Most monitors can be configured in the three methods shown above. A set of monitors, asynchronous monitors, have more restricted options. The asynchronous monitors can't be polled and do not maintain state. Thus, you can only configure threshold condition monitoring for these monitors.

Trap OIDs

EMS traps use the following Enterprise OID:

Enterprise ID: .1.3.6.1.4.1.11.2.3.1.7

Enterprise Name: iso.org.dod.internet.private.enterprises.hp.nm.system.general.7

Trap Number	Trap Name	**OID**	**Description**
1	EMS_NORMAL_OID	.1.3.6.1.4.1.11.2.3.1.7.0.1	Normal Event
2	EMS_ABNORMAL_OID	.1.3.6.1.4.1.11.2.3.1.7.0.2	Default Problem Trap
3	EMS_REBOOT_OID	.1.3.6.1.4.1.11.2.3.1.7.0.3	Monitor Restart (Reboot) Event
4	EMS_RESTART_OID	.1.3.6.1.4.1.11.2.3.1.7.0.4	Monitor Restart Event
5	EMS_NORMAL_SEV_OID	.1.3.6.1.4.1.11.2.3.1.7.0.5	Problem Trap with normal severity
6	EMS_WARNING_SEV_OID	.1.3.6.1.4.1.11.2.3.1.7.0.6	Problem Trap with warning severity
7	EMS_MINOR_SEV_OID	.1.3.6.1.4.1.11.2.3.1.7.0.7	Problem Trap with minor severity
8	EMS_MAJOR_SEV_OID	.1.3.6.1.4.1.11.2.3.1.7.0.8	Problem Trap with major severity
9	EMS_CRITICAL_SEV_OID	.1.3.6.1.4.1.11.2.3.1.7.0.9	Problem Trap with critical severity

There are two basic types of traps sent by EMS: event notification traps and monitor restart notification traps. For event notification, you will see traps 1, 2, 5, 6, 7, 8, or 9. All of these traps have the same structure. Traps 3 and 4 are restart notification traps. These two have the same structure.

EMS Event Notification Traps

This section explains the Event traps. These traps are sent when events occur. The following table has all of the varbinds (variables) for traps 1, 2, and 5 through 9. These traps are used for poll events, change events, threshold condition events (Problem and Normal), and error events.

Here is the suggested formatting for these traps:
EMS $11 Event: Value $5 for "$1" (Threshold: $3 $7)

Examples:
EMS Problem Event: Value DOWN for "/vg/vg00/lv/lvol1/status" (Threshold: != UP)
EMS Poll Event: Value 10 for "/system/numUsers" (Threshold: poll)
EMS Change Event: Value DOWN for "/vg/vg00/lv/lvol1/status" (Threshold: change)

Variable #	Variable Name	Type	Description
1	Resource Name	OCTET STRING	Full EMS Name of the Resource being monitored
2	RequestID	INTEGER	A unique identifier for the monitoring request
3	Operator	OCTET STRING	Operator indicates poll, change or threshold condition operator: >, >=, <, <=, ==, !=
4	Resource Type	INTEGER	Integer representing the resource type: 3009 = String 3010 = Sbit32 (signed 32 bit integer) 3011 = Ubit32 (unsigned 32 bit integer) 3012 = Sbit64 (future use) 3013 = Ubit64 (future use) 3014 = Float64 (64 floating point number) 3015 = Enumerated Type 3016 = Error
5	Resource Value	OCTET STRING	Quoted string representing resource value, should be converted using the Resource Type indicated in Variable #4 Not applicable if Resource Type is Error
6	Threshold Type	INTEGER	Integer representing the threshold type, this is usually the same as resource type. Integer representing the threshold type: 3009 = String 3010 = Sbit32 (signed 32 bit integer) 3011 = Ubit32 (unsigned 32 bit integer) 3012 = Sbit64 (future use) 3013 = Ubit64 (future use) 3014 = Float64 (64 floating point number) 3015 = Enumerated Type 3016 = Error Not applicable if operator is poll or change; or if Resource Type is Error Note: enumerated type is not supported as threshold type.
7	Threshold Value	OCTET STRING	Quoted string representing threshold value, should be converted using the Threshold Type indicated in Variable #6. Not applicable if operator is poll or change; or if Resource Type is Error

8	User Data Flag	INTEGER	Indicates that additional user data is available from EMS 0 = no user data 1 = user (comment) data exists
9	Monitor Data Flag	INTEGER	Indicates that additional monitor data is available from EMS 0 = no monitor data 1 = monitor data exists
10	Notification Trigger	INTEGER	Indicates whether normal or problem event 0 = normal 1 = abnormal (threshold is true) 5 = abnormal (severity is normal) 6 = abnormal (severity is warning) 7 = abnormal (severity is minor) 8 = abnormal (severity is major) 9 = abnormal (severity is critical)
11	Event Type	OCTET STRING	String representing event type: Poll (for "At Every Interval") Change (for "When value changes") Normal (for when threshold condition is not true) Problem (for when threshold condition is true) Error
12	NotifyID	INTEGER	Unique identifier for this particular event

Basically, there are five different types of events for which you'll get one of these traps. These are poll events, change events, threshold/problem events, normal events, and error events.

Poll Events

If you select notification "At every interval" (as shown in the EMS GUI), you will get a "Poll" event at each polling interval. For EMS 3.0, this will always be an abnormal trap. In the EMS GUI, you can configure the event severity. If you configure for "Normal" severity, you will get Trap 5, "Warning" will generate Trap 6, etc. If a client configured this without specifying the severity, Trap 2 (the default abnormal trap) will be sent. The Event Type will be Poll.

If the monitor has an error getting the resource value, an Error Trap of the configured severity will be sent.

Change Events

If you select notification "When Value Changes" (as shown in the EMS GUI), you will get a "Change" event when the value changes. For EMS 3.0, this will be an abnormal trap (except for the initial notification). In the EMS GUI, you can configure the event severity. If you configure for "Normal" severity, you will get Trap 5, "Warning" will generate Trap 6, etc. If a client configured this without specifying the severity, Trap 2 (the default abnormal trap) will be sent.

The first trap you receive when monitoring for when a value changes will be a Trap 1, a normal trap. This is the initial notification, your starting value. The Event Type will be Change.

Threshold Condition Events (Problem Events and Normal Events)

If you configure the monitoring request to be notified when a certain threshold condition is true, this is called a threshold condition. You will get a Problem event if the condition is true. If configured a certain ways, you may also get a Normal event when the condition is false.

With threshold monitoring, you have options for Initial, Repeat, and Return.

With the Repeat Option, you'll get a Problem trap of the configured severity repeatedly until the threshold condition is no longer true. This option is always set for asynchronous monitors because two critical events on the same resource may represent two different events.

With the Initial Option, you can have a trap sent immediately. The Trap Id is based on whether or not the threshold condition is true. If the threshold isn't true, Trap 1 with Event Type = Normal will be sent. Otherwise the trap of the configured severity will be sent.

With the Return Option, you will get a Trap 1 with Event Type of Normal when the threshold condition is no longer true.

For monitoring asynchronous events, repeat is the only option set.

You can configure the severity of Problem Events (when the threshold condition is true) in the EMS GUI. If you configure for "Normal" severity, you will get Trap 5, "Warning" will generate Trap 6, etc. If a client configured this without specifying the severity, Trap 2 (the default abnormal trap) will be sent. Trap 1 will always be sent for a Normal Event (either initial notification or a return notification).

With some monitors, the severity (or trap sent) is based on the resource value. If this is the case, Trap 5 (for resource value = 1), through Trap 9 (for resource value = 5) will be sent. This option shows up in the EMS GUI as "Map from value".

If the monitor has an error getting the resource value, an Error Trap of the configured severity will be sent. The notification sent after the error is based on the threshold condition. If true, problem trap of configured severity will be sent. If the condition is not true, Trap 1 with Event Type of Normal will be sent.

Error Events

An error event is generated when a monitor reports to EMS that it encountered an error trying to obtain the current resource value. This will be an abnormal trap of the configured severity.

When the monitor no longer reports this error, a notification will be sent regardless of whether the threshold condition is true and regardless of initial, repeat, or return options configured. The event type will be one of Problem, Normal, Change, or Poll for the event after the error event.

Getting addition information about the event

The user and monitor data flags within the trap indicate if additional event information is available. If either of these flags are set, the "resdata" command can be run on the source node to get user data (provided in the Comment field in the EMS GUI) and/or additional data about the event provided by the monitor.

This can be invoked as an automatic action, for example. Or you could just provide instructions to the user on how to get this information himself with a cut/paste line of the command filled in with the proper options and inputs.

The resdata command takes Resource Name, RequestID, and NotifyID as input. All of these input parameters are provided in the trap.

If user data only, the command is:

```
/opt/resmon/bin/resdata –r <Resource Name> -R <RequestID> -u
```

If monitor data only, the command is:

```
/opt/resmon/bin/resdata –r <Resource Name> -R <RequestID> -n <NotifyID>  -m
```

If both user and monitor data is available, the command is

/opt/resmon/bin/resdata –r <Resource Name> -R <RequestID> -n <NotifyID> -a

For simplicity, you could use the 3rd example for all three cases. If user data and monitor return data flags are both zero, resdata should not be run.

EMS Monitor Restart Notification

The monitor restart events are sent when EMS has restarted a monitor after a system reboot or when EMS has detected that a monitor has died. EMS will reconfigure all persistent monitoring requests at this time and generate a restart notification event to the configured targets. For SNMP targets, trap 3 or 4 is sent.

Here is the suggested formatting for these traps:
EMS Monitor Restart due to reboot: "$2" (command: $3, vendor: $4, version: $5, key: $7)
EMS Monitor Restart: "$2" (command: $3, vendor: $4, version: $5, key: $7)

Example:
EMS Monitor Restart: “Disk_Monitor” (command: /etc/opt/resmon/lbin/diskmond, vendor: Hewlett-Packard Co, version: A.00.00, key: 005431)

Variable #	Variable Name	Type	Description
1	Restart Type	OCTET STRING	REBOOT or RESTART
2	Monitor Title	OCTET STRING	Title of the monitor affected by the outage
3	Command	OCTET STRING	Command used to launch the monitor
4	Vendor	OCTET STRING	Vendor who supplied the monitor
5	Version	OCTET STRING	Version of the monitor
6	PID	INTEGER	Process ID of the monitor
7	Monitor Key	OCTET STRING	Unique identified for this monitor, to be used for running resdata to get list of resources affected by the monitor outage.

EMS provides for persistence of monitoring requests so that they are retained through a system reboot or can be reinstated if a monitor dies. During normal operation, EMS will check to see that monitors with persistent request are alive. If it has detected that a monitor has died (or the system has just been reboot), it will restart the monitor and sent an event indicating that the monitor has been restarted.

Trap 3 will be sent after reboot and the first variable will contain “REBOOT”. Trap 4 will be sent after EMS has restarted a monitor after it has detected that the monitor is no longer running.

Getting addition information about the event

In both cases, the resdata command can be invoked to get a list of resources affected by the monitor outage. This can be done as an action or can be given to users as instructions on how to get more information.

The resdata command should be run on the source node (the node generating the event) using the following command line options in order to get a list of resources affected by the outage:
/opt/resmon/bin/resdata –M <Monitor Key>