

# Managing Mission-Critical VMSclusters

Johan Michiels – Nand Staes – Gerrit Woertman – Jan Knol

## Abstract

In this paper we present the VMS cockpit concept. The cockpit is a dedicated VMS system that centralizes all management operations. The main task of this cockpit is to monitor the entire OpenVMS production environment and to take care of event notification in a uniform way. The cockpit assists the system manager 24 hours per day and automates routine tasks.

## Introduction

The main responsibility of the VMS system manager is to make sure that his OpenVMS environment always delivers the performance and availability levels the business demands. To achieve this, he needs tools that support and automate his job, and that cover all aspects of the production environment. In many ways this can be compared with the cockpit of a plane or boat.

To build such a cockpit, HP Consulting & Integration has made the toolkit "CockpitMgr for OpenVMS" available to customers. This toolkit has a number of unique and valuable features every VMS system manager needs. The present paper gives a technical overview of those tools.

The CockpitMgr toolkit has already been deployed at many important OpenVMS customer sites, where it is well appreciated. CockpitMgr runs entirely on OpenVMS, provides a solution for the entire production environment, and substitutes for the well-known POLYCENTER products.

## 1. History

In the early nineties, Digital launched POLYCENTER, a set of products and services that simplify the management of systems, networks and storage. Implementing POLYCENTER solutions enabled the Information Systems professional to provide enhanced services to end users, while controlling the management costs.

The most practical and cost-effective way to deploy these POLYCENTER products was achieved by implementing a dedicated system running the full function component of each product, while the agent components were running on the different VMS production systems. We called this dedicated system, running exclusively management software, "the cockpit".

Having an in-depth experience with POLYCENTER products on Digital's in-house VMS systems, we started delivering "cockpit services" to our customers. The objectives were to:

- Use POLYCENTER to simplify VMS systems and network management.
- Provide some integration among the different products. As each product had its own user interface and notification utilities, there was a clear demand to have at least one uniform method for event notification.
- Add some functionality that was missing in the products.

Later on, customers requested new functionality, and it was getting more and more difficult to deliver this within the framework of the existing products. Additionally, technology and cluster configurations changed over the years, demanding new monitoring capabilities. So we decided to gradually write our own monitoring tools, to be able to respond in an easy, flexible and professional way to the requirements of our customers.

We kept the original cockpit concept, and continued with the dedicated management system centralizing all management operations. As the cockpit is also the daily working platform for the VMS system manager, we do believe that ideally this system should be running OpenVMS.

Implementation of a cockpit is done with two main goals in mind:

- Do more with less staff. Routine management tasks should be automated as much as possible, giving the system manager time to work on real problems.
- “Beat the phone call”. A system manager is often called by an end user, complaining about some malfunction in the system or in an application. The cockpit attempts to detect problems or potential problems as soon as they occur, so that the necessary actions can be taken immediately. In many cases, things can be fixed before the end users run into it, or end users can be warned about the situation.

All tools we developed during the past 12 years are bundled in a kit “CockpitMgr for OpenVMS”. The tools run fully independently of any other management product.

The purpose of this document is to make an inventory of the most common needs of a VMS system manager, and to show how the CockpitMgr toolkit provides an answer to those requirements.

## 2. Cockpit Architecture

### 2.1. Agent

CockpitMgr provides several agents, each monitoring certain items related to system, network and storage. Those applications run either on the cockpit itself (local agent), or on a VMS system that is monitored and managed by the cockpit (remote agent). A remote agent must be able to communicate in some way with a server or collector on the cockpit.

The different agents are described in chapter 3.

### 2.2. Event

Any detected unexpected behavior, and any subsequent change in this behavior, results in the creation of an event. Such an event can be considered as a set of items, including:

Node name	The DECnet node name or TCP/IP hostname
Event name	A descriptive name for the event
Subsystem	The item of the system being referred to (disk, process, shadow set...etc)
Timestamp	Date and time the event occurred
Text	A comprehensive message text
Priority or severity	Critical, major, minor, warning, clear or indeterminate
Source	Indicates which agent generated the event

### 2.3. Event Notification System

A central application on the cockpit keeps track of all events. This application is called the "Event Notification System" (ENS).

The ENS is the kernel of the cockpit:

- It receives the events from the local or remote agents
- It can perform processing on events. E.g. it can replace the error code in the event text by a more comprehensive message or it can change the event priority.

- It can perform event correlation where one event can clear one or more other events to indicate that a problem has been solved.
- It keeps all event data in memory, and stores the events on disk for reporting purposes.
- It dispatches the events to the different notification utilities.

#### 2.4. Event Notification

A notification application can request all or selected events from the ENS, and send the information to one or more persons by different means.

Typical event notification can:

- Display events in the event console.
- Forward events to an enterprise manager.
- Trigger execution of any procedure, either on the cockpit itself or on the monitored system.
- Send a message to a pager or a cellular phone.

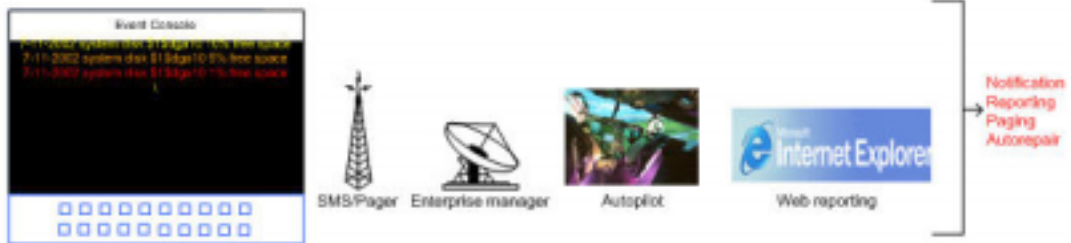
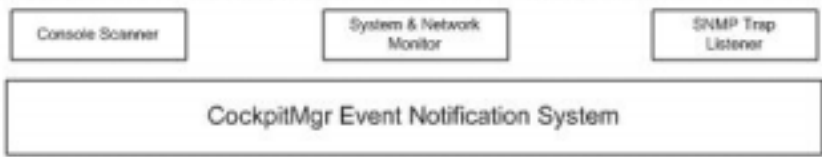
An API is available, which makes it easy to develop additional event notification routines.

Event notification is discussed in chapter 4.

#### 2.5. Event transfer

Events must be transferred between agents and the ENS, and between the ENS and notification applications.

Communication between the ENS and the monitoring and notification utilities occurs through the intra-cluster communication (ICC) system services. ICC is very suitable for large and fast data transfers, and has been available in OpenVMS since V7.2.



### 3. Monitoring

There are four important modules running on the cockpit. Those applications feed events directly into the ENS:

- A console manager, capable of scanning console output and searching for pre-defined series of characters (see section 3.1).
- A System Monitor, which communicates over DECnet or TCP/IP with a System Agent running on the managed OpenVMS systems (see section 3.2).
- A Network Device and SAN Switch Monitor, which uses SNMPgets to obtain information from the MIB agent (see section 3.3).
- An SNMPtrap Listener, which receives and interprets SNMPtraps from various sources (see section 3.4).

Other agents running on a managed OpenVMS system are:

- A performance monitor (see section 3.5).
- A security audit listener (see section 3.6).
- A log file browser (see section 3.7).

Those remote agents communicate information to the cockpit by sending OPCOM messages (to be captured by console manager) or by sending SNMPtraps (to be received and interpreted by the SNMPtrap Listener).

### 3.1. Console Management

OPA0, the OpenVMS system console, has a special function in the operations. The console is:

- The device to be used to boot and halt the system.
- The device to be used to perform OS installations and upgrades.
- The device to which the OS writes many important messages, and consequently one of the first things to check when doing troubleshooting.

In the past many VMS customers installed a console management product. Typical well-known products are:

- VAXcluster Console System (VCS), engineered by Digital.
- POLYCENTER Console Manager (PCM), the follow-up product of VCS.
- CA's Unicenter Console Manager, which is the same product as PCM, available under different license schemes.
- TECSYS ConsoleWorks, which is well known to the owners of AlphaServer GS series.

Unfortunately, only a few customers are really using all capabilities of those products. Too often the usage is limited to connecting remotely to the system's consoles.

This section describes what a console management product can signify to a system manager.

The next version of the CockpitMgr toolkit, to be released early 2003, will contain its own built-in console management functionality. But because of the long lasting history with PCM and ConsoleWorks, the integration modules for those products with CockpitMgr will continue to be supported.

#### 3.1.1. Functionality

A console management product must provide the following functionality:

1. Connect to console.

Every VMS system has an OPA0 console port, to which a video terminal is connected. The system manager connects to this terminal to boot and halt the

system, and to perform system upgrades. This implies that he should be physically present in the computer room to have access to the console.

To provide remote access to a system console, the RS232 console line is plugged into a port of a terminal server. The console management product then creates a LAT or Telnet connection to the port on the terminal server. Access to the consoles of the OpenVMS production systems can be obtained via the console management utility. Actually, any RS232 port can be connected and accessed via the console management utility.

2. Log all console messages in a log file for further use.

The first VAXes were delivered with a hardcopy console; later a small printer was connected to the console video terminal. The goal was to keep a log of all console messages, for further investigation if needed.

A console management product is capable of storing all console output on disk. This logging remains available as long as required, and can be used for reference later on.

3. Scan console output for dedicated strings.

Logging all console output in a file is one thing, but certain console messages require the immediate attention of a system manager. Therefore, a console management product must be able to scan the console output for a number of specific strings. When such a string is detected, the system manager must be notified, i.e. an event is generated in the ENS.

#### 3.1.2. Scan profiles

From the operational point of view, the scan capabilities are the most important. Lists of specific text strings, the so-called scan profiles, need to be created, maintained and validated for new versions of the operating systems and applications.

CockpitMgr comes with several scan profiles, allowing users/system managers? to detect and react on messages originating from the OpenVMS operating system, shadow server, cluster connection manager, VAX and Alpha hardware, and several layered products such as SLS, ABS, Scheduler ... etc. There is also a scan profile available for storage controllers configured into the console manager.

### 3.1.3. Remarks

- A console manager must be installed on a standalone OpenVMS system, not on a member of a VMScluster. It does not make sense to install it on one of the cluster members. If the cluster enters a quorum hang, and you need access to the consoles to investigate and to take action, your console management product will not be available at this point in time. So it is certainly not a good idea to use the quorum system in a disaster-tolerant cluster configuration as cockpit system.
- It is recommended that you configure storage controllers such as the HSJ, HSZ and HSG family of array controllers as well, as this will deliver the most complete monitoring. Console messages of those devices always contain an error instance code, which can be automatically translated into a comprehensive message by the CockpitMgr Event Notification System.

### 3.2. System Monitor

Most system managers periodically perform a number of elementary checks on their systems:

- Are the systems reachable over DECnet or TCP/IP?
- Are all processes running?
- Are all disks mounted, and are those disks running out of free space?
- Are my batch and print queues correctly started?
- Are there any devices with hardware errors?

Many system managers developed DCL procedures to automate those checks, and notification of any anomaly is often made by a terminal broadcast or by sending VMSmail.

Many people believe system management is no more than performing such elementary checks. Certain so-called enterprise management applications in particular do not provide anything more than frequent pings, and the checking of process availability, free disk space and hardware errors.

The CockpitMgr System Monitor takes care of verifying an OpenVMS production system on a continuous basis and in all its aspects. It is fully cluster-aware, and any item can be checked during specific periods of the day or week. It is also capable of triggering corrective actions immediately. This CockpitMgr module significantly increases system management productivity and reduces staff resource usage.

#### 3.2.1. Components

The CockpitMgr System Monitor has three main modules:

- The Configuration Utility. To be used on the cockpit, it allows definition of what to monitor, on which system or cluster, and during which period of day and week.
- The System Monitor. Reads the cockpit configuration file and sends instructions to the Agent running on the managed systems. Also takes care of feeding all events in the ENS.
- The System Agent. Runs on every OpenVMS production system. Receives instructions on what to monitor from the System Monitor, and returns status information.

Communication between System Monitor and Agent is established at regular time intervals, and occurs via either DECnet or TCP/IP. A comprehensive event is generated when no communication can be established between cockpit and managed system, or when a connection is aborted.

### 3.2.2. What is monitored

The System Monitor can instruct the Agent to check the following:

- Are there any changes in the error count of CPU, memory, disks, buses and communication devices?
- Are the OPAO system console settings OK? Console Management only makes sense when the system console is broadcast enabled, has a maximum terminal width to avoid line wrapping, and when it has been enabled as operator terminal for selected OPCOM classes. Necessary corrections to the terminal settings can't be made if somebody has created an interactive session on the console, so in this case a warning is also generated.
- Availability of processes. The Agent checks the existence of selected processes, on a single cluster member or cluster-wide. Processes are defined by the process name and (optionally) the UIC. The process name may contain wildcards in which case the minimum number of occurrences can also be specified. If a process is missing, a corrective procedure can immediately be started.
- Disk states. The Agent checks to see whether selected disks are correctly mounted. Disks are defined with either their physical name, or by the logical volume name. If a disk is not correctly mounted because it does not exist, it is not mounted at all, it is mounted foreign, it is in mount-verification, or it is member of a RAID set, a warning is issued.
- Disk free space. Four free space thresholds can be specified per disk, corresponding with warning, minor, major and critical event priority levels. If a threshold has been exceeded, a cleanup procedure can automatically be started.
- State of batch queues. In most cases a batch queue is "started", indicating that the queue is "idle", "busy" or "available". But sometimes a batch queue may have been voluntarily stopped, e.g. if the queue shouldrun

jobs only at night. The Agent checks whether the status of a batch queue corresponds with the expected status for that moment, and optionally corrects this.

- State of print queues. The Agent checks whether selected print queues are correctly started, and if none of them are currently stalled.
- Shadow Set problems. The Agent checks whether a shadow set has the correct number of expected members. If a member is missing, or if a disk is unexpectedly found to be member of a shadow set, a warning is issued. Start and completion of shadow copy and merge operations are also signaled.

### 3.2.3. System Monitor key features

- Monitoring of each node or individual item can be restricted to certain periods of the day or week. E.g. if an application is stopped every day at 20:00 to take a backup, and is restarted at 22:00, the system manager should not receive any message related to the stopped processes during that period.
- Any problem with a monitored node or item can be sent to one or several pagers. Paging can be initiated with a certain delay, giving the automatic repair or an operator the time to take action. If the issue gets solved before the expiration of the delay period, the page call is canceled.
- The cluster concept is fully present. Disks and shadow sets that are cluster-wide mounted are monitored on the cluster level. Messages will not be duplicated for each individual cluster member. Process existence can be checked cluster-wide. As CockpitMgr supports only clusters with a single queue file, batch and print queues are also monitored on cluster level.
- Configuring a new system in the configuration file can be done within minutes. A utility scans the new system, and generates all the commands to configure the monitoring for that system. The utility can also be used for systems that have only been partially configured; in this case only the commands for the missing items are generated.

- It is easy to extend the System Agent. Any DCL procedure or application is capable of delivering external events to the Agent, which will transfer this information to the cockpit.

### 3.3. Network and SAN Switch Monitor

VMScluster configurations changed drastically during the last years. Two trends can be considered:

- Configurations are made disaster-tolerant. Systems are located in two physically separated datacenters, and consolidated into one VMScluster. The two datacenters are interconnected by high-speed network links, and volume shadowing is deployed between the two datacenters. As such, operations can be resumed within minutes after the loss of one datacenter, and this without the need to restore data.

Such a disaster-tolerant configuration implies that the network is now an integral part of the VMScluster. Network devices must be operational at all times, and all physical connections must function properly. Appropriate monitoring of network devices and connections is consequently an absolute requirement.

- Storage is drifting away from the system, and is now located in a SAN. Monitoring of the SAN switches and the cabling is a must to be sure that at any time fully redundant paths are available between host and disks.

This section describes how network devices, SAN switches, and physical connections can be monitored.

#### 3.3.1. SNMP

The Simple Network Management Protocol (SNMP) enables a management station to query other network components or applications for information concerning their status and activities. Such a query is known as an SNMPget. The items that can be polled are called "managed elements".

#### 3.3.2. Network Monitor

The CockpitMgr toolkit extensively uses SNMPgets for the monitoring of selected network devices, SAN switches and their port states. Configuration is very easy and does not require any MIB expertise and knowledge on SNMP technology. Only the following information is required to configure a device to monitor:

- The TCP/IP host name.
- The community name: this is some kind of password to be used to gain access to the MIB tree structure. Usually this is "public".
- The TCP/IP port number (is generally 161).
- The type of the host. The supported types are listed below.
- The list of port numbers that need to be monitored.

CockpitMgr supports only selected device types. It is obvious that more types will be added whenever these appear in VMS environments.

#### 1. VMScluster configurations with FDDI as cluster interconnect.

The following network devices are frequently used:

- The DECconcentrator 900MX, a 6 port FDDI concentrator,
- The GIGAswitch/FDDI, a high-speed crossbar switching fabric.
- The DECbridge900EF and VNswitch900EF Ethernet to FDDI switches.

The CockpitMgr Network Monitor checks for selected ports on those devices:

- The physical state of the FDDI ports. The port states indicate that a link is operational. Changes in the port state often indicate a physical connection problem.
- The Spanning Tree Protocol (STP) state of the Ethernet ports.

Additional monitoring is performed on GIGAswitches, including the state of both fans and power supplies, the status of the links, and the status of the slot cards. The monitor takes hunt groups into account.

#### 2. VMScluster configurations with Ethernet, Fast Ethernet or Gigabit Ethernet as cluster interconnect.

Several kinds of Ethernet switches need to be monitored here. Any switch that supports the standard bridge MIB can be monitored, and especially the STP port state of selected ports is checked.

Extra monitoring capabilities are added for the VNswitch900EX, the VNswitch900XX and the Cisco Catalyst series.

3. The CockpitMgr Network Monitor provides additional SNMP-based monitoring capabilities, with the aim of making the network monitoring of the VMS production environment more complete:

- The reachability of any device that supports SNMP can be checked.
- The functioning of the modules in a MultiSwitch 900 (DEChub900) can be checked.
- The state of ports on any repeater that supports the standard repeater MIB can be monitored.
- The state of ports on a terminal server (access server) can be monitored. This is especially useful in combination with a console manager.
- The battery capacity and battery status of an UPS that supports the standard UPS MIB can be checked.

### 3.3.3. Fibre Channel Switches

Storage Area Networks enable organizations to uncouple the application (server) side from the information (data storage) side. Fibre Channel Switches enable enterprise class scalability and high performance connectivity between servers and StorageWorks storage systems.

Fibre Channel Switches run a MIB agent. CockpitMgr checks the following managed elements on such a switch:

- The operational state of the SAN switch.
- The value of the last diagnostic result.
- The status of the temperature, fan and power-supply sensors.
- Physical and operational status of the ports; the physical status may indicate a problem with one of the cables.

### 3.4. SNMPtrap Listener

Many SNMP agents on network devices and SAN switches can be configured to transmit messages to well-known addresses in response to specific events. Those unsolicited messages, called SNMPtraps, enable quick and possibly automatic reactions to specific conditions.

CockpitMgr provides an SNMPtrap Listener, which receives SNMPtraps from various sources. Each trap is analyzed, and the trap data is compared with a repository of known SNMPtraps. If the trap is recognized, a comprehensive event is generated in the ENS.

SNMPtraps may also be generated by certain applications running on OpenVMS systems. A typical application is the StorageWorks Command Console (SWCC) Agent. This application runs on OpenVMS and supervises the storage subsystems. The application is capable of sending SNMPtraps to any listener.

The SNMPtrap listener is also useful to achieve integration with the SAN appliance. The SAN appliance is a hardware and software solution to configure and manage a complex SAN. If the SAN appliance detects a problem, it can send an SNMPtrap to the cockpit. As such the SAN appliance provides the required alarming in the cockpit related to the complete SAN infrastructure.

### 3.5. Performance monitoring

OpenVMS systems occasionally may experience performance slowdowns. This performance degradation is typically due to requirements placed on one or more system resources, like CPU, memory and the I/O subsystem.

When an end user perceives an abnormal slow system response, he usually calls the system manager and asks to investigate. This search starts very often by executing some basic DCL commands to make a first diagnosis:

- SHOW SYSTEM: are there any processes in a special wait state?
- SHOW MEMORY: how about memory, pool and pagefile utilization?
- MONITOR SYSTEM: are there any processes using lots of CPU or making a high number of I/O requests? And how large is the compute queue?
- MONITOR PROCESS/TOPCPU: are there any processes that might be looping?
- MONITOR DISK: is there an I/O bottleneck on one or more disks?
- MC LANCP SHOW DEVICE/COUNTERS: is everything OK with the network?

The goal of the CockpitMgr Performance Monitor is exactly to perform those actions on a permanent basis. If some abnormal behavior in the system's performance is detected, it warns the system manager, who can investigate and correct before things escalate and production is impacted.

Performance-related research certainly benefits from the use of additional products such as DECamds, Availability Manager and ECP. Those products are available from and supported by VMS engineering at no extra cost.

#### 3.5.1. The CockpitMgr Performance Data Collector

The Performance Data Collector records OpenVMS system data for subsequent processing by the Performance Analyzer. This data is required to detect certain conditions, and to generate alarms. E.g. when checking whether the CPU utilization exceeds a certain threshold, we use the average CPU utilization during a given interval.

To collect performance data on CPU utilization and Direct I/Os, we currently still use the undocumented system service EXE\$GETSPI. This system service is also used by the MONITOR utility. It is our intention to use in the future the API of the new TDC product ("The Data Collector").

The Data Collector stores performance data in an indexed sequential file. Sampling rate is two minutes.

- CPU utilization (total and per mode).
- The I/O operation rate and I/O request queue length for each disk.
- The free memory.
- Non-paged pool size and utilization.
- Free and total space in page and swap files.
- Active and available number of CPUs.
- For each LAN controller, we also collect the throughput, and the number of "initially deferred packets sent," "single collision packets sent," "multiple collisions packets sent," and "total packets sent."

The collected data can be visualized in a motif-based graph application on a daily basis. It is not our goal to develop a sophisticated graphing application; ECP provides this functionality, but the graph utility is only provided to have an easy way to check the collected performance data.

### 3.5.2. CockpitMgr Performance Alarming

The CockpitMgr Performance Alarming is a process which:

- Is activated every two minutes.
- Checks a number of items against the configuration parameters.
- If a condition is met, it generates an OPCOM message or an SNMPtrap.
- If a condition was met, and later on it appears that the problem is solved, a new OPCOM message or SNMPtrap is generated.

The following items are currently checked:

- Are all available CPUs also active? A CPU may have been stopped manually or due to a hardware problem.
- Does the percentage of free space in the currently installed swap files drop below a given threshold?
- Does the percentage of free space in the currently installed page files drop below a given threshold?
- Has non-paged pool been extended from its original value?

- Are there any processes that used several hours of CPU time?
- Does memory utilization exceed a given threshold?
- Does the CPU utilization exceed a given threshold? Thresholds may be specified for total CPU, and CPU utilization in interrupt mode, MP synchronization, kernel mode, executive mode, supervisor mode, user mode and compatibility mode.
- Are any processes idle? A process is considered to be idle if it did not use any CPU and did not do any I/O during a given interval. You can specify which action needs to be automatically taken on an idle process that runs under a certain user name, has a specific process name, and runs a given image name (or a combination of those three ). Possible actions are from simple notification, to forced image exit, stop process, or execution of a DCL procedure.
- Are any processes looping? A process is considered to be looping if it uses during a period of 2 minutes at least 25% of one CPU, and does not do any I/O within that period. You can specify which action needs to be automatically taken on a looping process that runs under a certain user name, has a specific process name, and runs a given image name (or a combination of those 3). Possible actions are from simple notification, to forced image exit, stop process, or execution of a DCL procedure.
- Are any processes in a special wait state?

Future releases of CockpitMgr will contain more monitoring and alarming capabilities.

### 3.6. Monitoring Security Events

OpenVMS is considered a secure operating system. This does not mean that a system manager should not develop and implement a security plan. Security is more than building a fortress around corporate data. It is also finding the best way to give the right people lawful, predictable and reliable access to the right information at all time.

Monitoring security events is part of an overall security strategy. CockpitMgr provides a way to obtain flexible monitoring of the security of the information entrusted to your VMScluster.

#### 3.6.1. OpenVMS Auditing

The AUDIT\_SERVER process records security-relevant activity as it occurs on the system. Such activity is everything that has to do with user access to the system or to a protected object within the system, and includes:

- Logins, logouts, and login failures.
- Changes to the user authorization, rights list and network proxy files.
- Access to protected objects such as files, devices, global sections, queues ... etc.
- Changes to the security attributes of protected objects.

The AUDIT\_SERVER normally writes security event messages to two places:

- The security audit log file. Security events are written in binary format to this cluster-wide log file. Each record contains details related to the event, and can be examined with the ANALYZE/AUDIT utility. Specifying which security events are written to the audit log file is done with the SET AUDIT/AUDIT command (audit events).
- Security operator terminals. These are terminals that are enabled to receive OPCOM security class messages. This allows the system manager to monitor users' activity in real time. Specifying which security events are written to such terminals is done with the SET AUDIT/ALARM command (alarm events).

Our experience with security auditing and alarming is:

- The audit log file should be periodically reviewed for suspicious activity, but only few customers actually do this.

- The ANALYZE/AUDIT utility is especially used when external elements indicate that the security of the system and its data may have been compromised. The information in the audit log file may help the system manager to reconstruct events leading up to attempts to compromise the system security.
- Enabling a terminal as security operator terminal mainly happens when doing troubleshooting on privilege violations on protected objects.
- The OPAO system console is often enabled as security operator terminal. A security alarm is in fact nothing more than an OPCOM message containing a “dump” of the original binary record, and spans easily over 15 lines. The console manager utility unnecessarily logs all this in a file, and must scan all console output for specific text strings.

### 3.6.2. The CockpitMgr Security Audit Listener

CockpitMgr addresses the concerns mentioned above, and provides a solution to achieve continuous but simplified monitoring of security events. The system manager will more quickly detect suspicious security events, which then can be further examined with the ANALYZE/AUDIT utility. The OPAO system console can additionally be disabled as security operator terminal, yielding smaller console log files and less processing.

The operator terminal and the security audit log are the primary destinations for security event messages. As an additional feature of the security auditing facility, a listener mailbox can be created to receive a binary copy of all security-auditing messages.

The CockpitMgr Security Audit Listener is an application that creates and reads this mailbox, processes the auditing information, and generates comprehensive one-line event messages that are sent to the cockpit.

A few examples of typical security messages as presented to the system manager, are:

- Operator modified SYSUAF record of user SMITH: PGFLOQUOTA, BYTLM
- JONES accesses (R+W) file NET\$PROXY.DAT: successfully
- PETERS dialled in on \_VTA10 (DCS:.25001381112)
- WILLIAMS modified active SYSGEN parameters

### 3.7. Logfile Browser

Many OpenVMS systems often do a substantial amount of batch processing. E.g. banks run at night the so-called "end-of-day jobs", and incorrect termination of those job streams have serious implications on the operations next day. Even when a scheduler application has been deployed, the final status of one job does not necessarily imply that no errors occurred. The best way to verify correct termination is to check the log file.

CockpitMgr provides a simple log file browser, with following features:

- Per job, you can specify the set of character strings to scan for. Wildcards can be used. E.g. if the batch job executes only DCL commands, it might be sufficient to scan for the strings "-W-", "-E-" and "-F-".
- Log files can be scanned while the job is still running, in which case the browser keeps track of the part of the file already scanned. Browsing a log file can also be postponed until the job is completed.
- Each time a pattern match is detected, an OPCOM message is generated or an SNMPtrap is sent to the cockpit.
- The interval between 2 consecutive scans can be set by a parameter.

It is obvious that the usage of the Logfile Browser is not limited to batch log files. If the Logfile Browser can get shared access to a file, it can search it for those specific character strings.

#### 4. Event Notification Utilities

CockpitMgr provides several event notification and reporting capabilities:

- Displays events in an Event Console (Motif application).
- Sends messages to a pager or a cellular phone.
- Forwards event messages to an Enterprise Manager.
- Provides detailed reporting capabilities, optionally via a web interface.

Further, you can pass an event to the Automatic Pilot, which can initiate any procedure on either the cockpit or on the OpenVMS system that originated the event.

Finally, CockpitMgr provides an API that allows you to create your own event notification routines.



## 4.2. Notification to Pager or Cellular Phone

The CockpitMgr Event Console is without any doubt the main utility to check for new events. But a system manager does not always have direct access to this event console, and sometimes there are events that require his immediate attention.

Additionally we see that more and more companies attempt to stop or limit the second and third shifts. After normal working hours, limited or even no operator support is present on-site, and a duty role has been implemented.

To make sure that a system manager gets immediately informed on really important events, CockpitMgr provides the possibility of sending messages to a pager or a cellular phone.

### 4.2.1. The CockpitMgr Pager Engine

The CockpitMgr Pager Engine takes care of sending selected event messages to pager and cellular phone. It has the following properties:

- It checks if the message should be sent or not at this point in time. Pagers can be disabled (e.g. during the holiday of the owner), or pagers should only receive messages during certain periods of the day.
- The pager request can be accepted with a given delay. If the problem gets solved within this period, the pager request is canceled.
- If for some reason the message is not sent correctly (e.g. due to a modem problem), one or more retry attempts can be made.

### 4.2.2. Sending a message to a cellular phone

To send a message (a.k.a. an SMS) to a cellular phone, CockpitMgr uses a Cellular Engine Terminal from Siemens (the M20 or the TC35T). Such device is a compact GSM modem that can be used for transfer of data, voice, SMS and faxes. It can easily be connected to a terminal port of the cockpit system, or to a port of a terminal server.

There are several advantages of using a Cellular Engine Terminal:

- It is inexpensive
- The message is sent very fast to the SMS service center
- It works universally with all the telecom operators
- Cellular phones are commonly used

#### 4.2.3. Sending a message to a pager

Sending a message to a pager can be done by using a modem or via X25. CockpitMgr software modules have been validated to send paging requests to several telecom operators. Unfortunately, we have experienced that most operators implemented the communication between a client and the paging server in slightly different ways. Validating the CockpitMgr Pager Engine for another telecom operator often means a software modification.

Sending a message via a modem is rather slow, while X25 software is fast and reliable, but rarely available on the cockpit system. On the other hand, pagers may have better coverage than cellular phones in certain countries.

#### 4.3. Integrating the Cockpit with an Enterprise Manager

Many customers implement a so-called Enterprise Management Framework, where one application provides total enterprise management. Typical well-known enterprise managers are HP OpenView, CA Unicenter, BMC Patrol and Tivoli. Some of those solutions also provide some level of support for OpenVMS.

Just like a network manager who swears on the utilization of CiscoWorks, and a database administrator who runs Oracle Enterprise Manager to simplify everyday database management tasks, the high-end expert VMS system manager considers the cockpit as his dedicated daily work environment.

Nevertheless it might be useful to integrate the cockpit with the company's enterprise manager because

- Operators supervising multiple platforms only have to check one console for events, and
- Applications running on OpenVMS can be included in the enterprise business process views.

CockpitMgr currently provides a very simple, one-directional way of integration between the cockpit and an enterprise manager. As any enterprise manager has a built-in SNMPtrap Listener, it is very easy to send events via SNMPtraps. The cockpit always uses the same trap data format to send SNMPtraps containing full event details, such as event name, node, time stamp, priority, source and text. As such it is fairly easy to format those SNMPtraps within the enterprise management console, and to take additional actions such as icon coloring.

In the future, we consider developing a more robust integration with enterprise managers, especially with HP OpenView:

- Events will transfer between cockpit and enterprise manager by two applications communicating via TCP/IP sockets. The cockpit will keep track of the events sent to OpenView.
- Integration will be bi-directional, giving the possibility to take ownership and to delete events from the OpenView Event Console.

#### 4.4. Reporting

CockpitMgr provides a reporting utility, which generates detailed and customized reports.

When generating a report, you may specify:

- A time interval
- Node name(s)
- Event name(s)
- Priorities

If the Compaq Secure Web Server for OpenVMS, based on Apache, has been installed on the cockpit, the reports can be generated and displayed in your preferred web-browser. Access to those web pages can be protected by means of the HTACCESS mechanism.

CockpitMgr also does reporting on system downtime.

#### 4.5. Autopilot

CockpitMgr is capable of triggering procedures on any events. Those procedures can be executed either:

- On the OpenVMS system on which the event originated. This enables immediate automatic repair functionality, without intervention of the system manager.
- On the cockpit itself. This capability allows you to easily add new notification mechanisms. E.g.: this feature makes it possible to integrate the cockpit with the company's call handling system.

## 5. Conclusion.

'CockpitMgr for OpenVMS' bundles the experience of many VMS system managers into one toolkit. The challenge to those system managers always has been to do more with less staff, while the service to the end-user needed to be improved.

'CockpitMgr for OpenVMS' is today the most complete toolset in the industry, supporting the VMS system manager in the day-to-day operations.

The product has been created by VMS system managers, for VMS system managers. And it runs on OpenVMS.