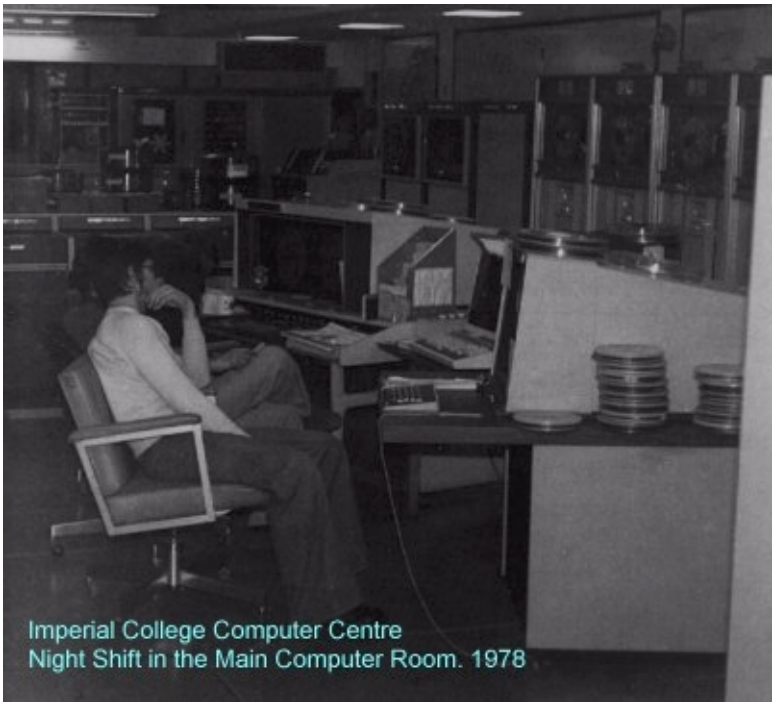
“How Automating the Console Gives You Power over Your Environment”

Kim F. Harris
Entrix Computing Ltd.
Lakeview Court, Ermine Business Park
Huntingdon, England
PE18 6XY
Tel. +44-1480-414131
Fax. +44-1480-414134
Email kim@entrix.co.uk

Introduction

Once upon a time, there were five people on a shift, three shifts a day, five days a week plus special work at weekends. There were three mainframe computers – each with a console and each with a console operator and the other two people did the work of loading tapes, punch cards, paper, etc. They were all very busy people and the CDC computers cost $2.5million each.

That was in 1978 and the place was Imperial College Computer Centre in London where I first learned my craft of computer operations and system management.



Don’t worry; this is not to be a nostalgia journey into the past – although like a lot of “old timers” I can easily be side-tracked into *that* occupation. It is useful, however, to look back at how we did things in those days to get some idea of the care we took in looking after our expensive charges because the cost of them being down or problems occurring was so astronomical as to be unacceptable.

“I.T. Managers are being asked to do more and more with less and less”.

This is a phrase from our own corporate advertising blurb which is within our own direct experience and that of our customers and other people we speak to.
There was a time when systems were being distributed geographically and in terms of the responsibility for their upkeep but it is our experience that these distributed systems are often coming back under the management of a central department. At the same time, financial pressures are applied to such an extent that the human resources available to manage all these systems are stretched to an intolerable degree. This also happens through frequent company acquisitions and mergers.

One of the first things to suffer is the health monitoring of the increasing number of systems upon which the business is extremely dependant.

This is a bad thing.

In the old days on our CDC mainframes we watched the consoles like hawks and considered how we could schedule work manually by forcing it into the processor until the compilation had completed and the core memory requirement reduced. We could put a large queue of jobs into the rollout queue ready to run in parallel on the dual processor system while we went for dinner. There was always someone on the console of each system and immediately a message flashed up we were on the ball to do something about it. As soon as a critical error occurred, we put one of the shift on the telephone to answer the calls from users which would come in about fifteen seconds after the system crashed. We prided ourselves on having the recovery in progress before the first ring of the phone - in other words, we did not wait for the users to call us to tell us the system was down.

I believe that the same should be true of operations today but it is a sad fact that in many ways, apart from the fact that systems are more reliable than they used to be, the service we provide is less well managed and the response to problems slower than was the case in 1975. We simply don’t have the time or resources to attend to such things in the manner we would like. Exactly the same applies to routine house-keeping. In the old days, we had a list of things to do each day and a log to enter up which recorded the action and results. How many sites do that today?

Manual intervention with scheduling and other functions of the operating system are, thankfully, no longer necessary and are not desirable. The ratio of cost of machinery to people has swung about completely. However, I would like you to think about the issue of taking care of the systems again. To ignore them is not necessarily a good thing but the economics of watching over anything between one system and hundreds does not make sense. The cost of systems being down is just as great as it always was – the difference is that today, with the pervasiveness of information technology into companies and organisations, a down system can stop the business in its tracks; not just give a PhD student an excuse to go to the pub.

The answer to the problem of how to give a better service, of course, is better management and the application of more resources to the problem. Unfortunately, more resources are not available so we have to do a better job of managing and automating the I.T. service we provide rather than just fire-fighting when problems occur.

If this paper does nothing else but remind you that housekeeping and monitoring tasks ensure a better service, then it will have done its job.

The Console as the Seat of Power

It is useless having critical messages going to the system console if nobody ever sees them!

Console messages are often important and that is why they are written to the system console. Unfortunately, the console as the seat of authority in an I.T. organisation is in some ways seen as rather outmoded and sending messages there is an unreliable way of getting an alert to the system manager. It used to be the case that an operator was seated there night and day glued to the screen and taking immediate action upon any message. I should know - I did my share of it in the 70s!

The console can be – just as it was twenty five years ago – a great point of concentration of control. This can be in both a physical sense and a logical sense. I am not suggesting that we go back to having shifts of five operators sitting there all the time getting square eyes watching a CRT, I am suggesting a practical approach to answering the question “how can I improve the service I offer to my customers”.

There are to parts to my proposition that with appropriate management and planning, the use of the console as a repository for alerts can re-enfranchise it as an effective system management tool.

The first part is to make the console a useful control point again. This entails looking at the messages which are sent and managing them.

The second part is to automate the process of monitoring, reacting to and further managing the messages. This automation process is considerably more applicable and cost effective in situations where there are multiple servers and, moreover, where those servers are located in multiple locations where the manager does not normally live.

Consider these issues:

1. Where does the information come from which would allow me to run my systems better?
2. Where does that information go and where should I look for it?
3. How can I gather it all together and sift it for what is urgent, what is important, and what do I need to act upon?
4. What can I automate and how?
5. What can I gain by monitoring and automating and how will the business benefit?

Sources of Information

1. Information which is written to the console by default. This includes logon errors, security alerts, operator requests, file system errors, application messages.
   In many cases this information is routinely of no great importance or interest and can be ignored. This chaff makes the wheat even more difficult to see. This can be particularly a problem if the system is running an application written for a bygone age when an operator could be relied upon to see the message and hopefully act upon it. Mostly this information is no longer necessary but the programmer hasn’t troubled to take out the messages.

2. Information which is written to log files and which you could usefully be aware of if only you had the time to look in the log files.

3. Problem alerts which with a little imagination and trouble you could send to a place where they would be picked up.

4. Last but by no means least are the answers to questions you can and should pose to the system periodically yourself. Can you reach a portion of a network or the Internet? Is a

job still running? Are we running out of disk space? Is a UNIX®[i] daemon or Window NT®[ii] service running?

Regular Console Messages

Consider which messages are to be sent to the console. Sometimes these are configurable. Sometimes they come from batch jobs or applications and should be removed or made more useful. Look at the console log file to see which things you should be monitoring for.

More Messages Available Free

A simple technique on HPUX and other UNIX systems for monitoring what is happening on the system is to set up a background process which tails the syslog file to the console. This is probably not something you would want to do if you were actually going to monitor it full time with a human being but it does contain messages which are indicative of the health of the system and which you would probably otherwise never see.

Here are some examples of messages which appear in our log files which are of interest but by default we don't see.

On an HP9000 in /var/adm/syslog/syslog.log which can be echoed to the console with the command tail –f /var/adm/syslog/syslog.log > /dev/console &

```
Dec 17 14:57:55 neptune ftpd[10639]: User webuser: Login incorrect
```
Could indicate an attempt to hack in

```
Dec 20 22:16:54 neptune named[305]: MAXQUERIES exceeded, possible
data loop in resolving (usmint.treas.gov)
```
Indicates a possible DNS problem

```
Dec 24 20:00:07 neptune vmunix: /: file system full
```
Definitely something you want to pay attention to

```
Jan  1 08:35:49 neptune syslog: su : + ttyp4 webuser-root
```
A record of someone using su to get root capability

```
Jan 11 14:22:45 neptune ftpd[25708]: connection from
sqsdream.demon.co.uk at Tue Jan 11 14:22:45 2000
Jan 11 14:22:46 neptune ftpd[25708]: FTP LOGIN FROM
sqsdream.demon.co.uk, sqs
Someone logging on to ftp. We only allow a small number of authorised users
so we might want to know when they come in.
```

On an HP9000 in /var/adm/sulog
This file logs when users use su to get root capability

```
SU 12/24 13:20 - ttyp2 vanmgr-root
```

Network logging on the HP9000 is a configurable feast. It is carried out by the nettl process which in turn is configured by the nettlconf program. Console logging of nettl messages may be turned on or off at various levels.

A Word On Automation and User Interfaces

Ah! How we yearn for simple commands..

Before proceeding with detailed discussion, I would like to just bring up a thought about the mode of interface used when managing systems.

One of the problems with automating the process of managing and health checking “modern” systems such as Windows NT and, to a lesser extent, NetWare, is that they use graphical user interfaces.

In the “old days” of MPE, VMS, DOS and UNIX, it was easy to type a command and see the result immediately. It was possible to put the output of the command into a file, analyse it, and make a decision as to whether it was good or bad. In this way programs and scripts could easily be written to automate the health checking process. With the advent of more “user friendly” systems where everything is mouse driven and in a window this was actually more difficult to do and in many ways checks had to be done manually.

It should be remembered, however, that there are often character mode equivalents of the GUI based tools. SAM (the administration user interface on HPUX) is only a front end to UNIX commands. In Windows NT, there are usually character mode command equivalents to management tools (such as the NET command to start and stop services). These commands for NT may be either inherent in the operating system or available as utilities from third parties or contained in the Windows NT Resource Kit from Microsoft.

Once a command and its results are available in character command mode then automation becomes possible - either through your own scripts or through a third party system management tool.

Elements of a Console Automation System

The Naming of Parts

The main functions of a console automation system are

- **Message Monitoring**
  To sort out the wheat from the chaff. Filter out the messages which can cheerfully be ignored and emphasise the ones which cannot. The message monitoring system should be able to at least look for specific strings including the use of wild cards. In the diagram below, the console is connected to the physical console port and all input and output from that port will be monitored, regardless of the state of the system.

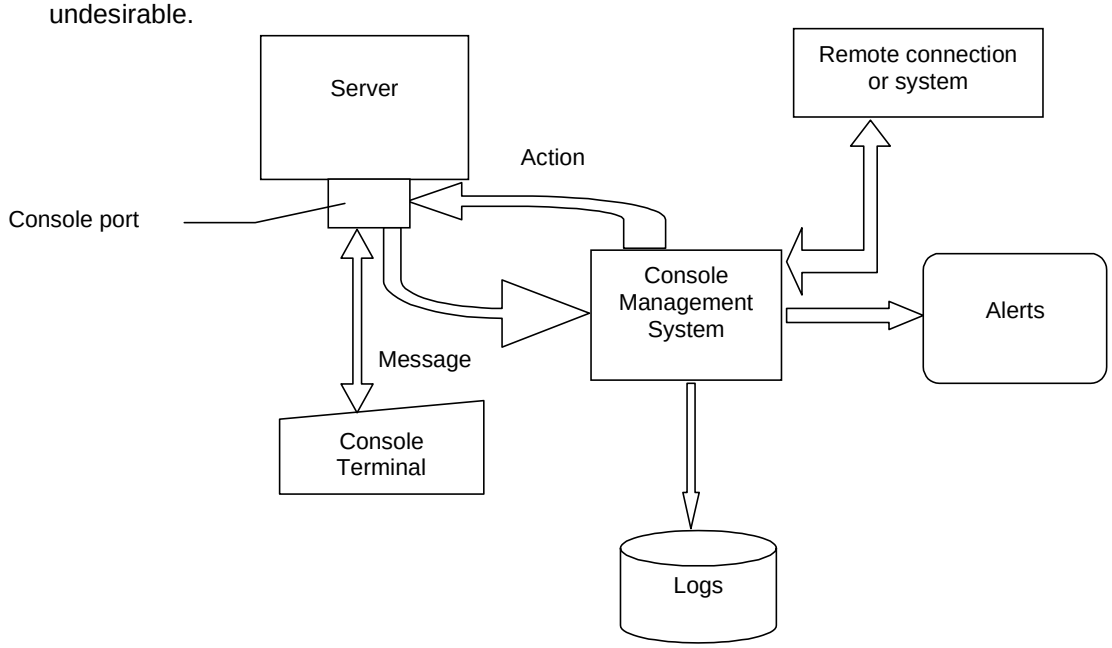
- **Alerting**
  Relay the message from the “real” system console where it will probably not be seen, to a central monitoring point or alert mechanism where it will be seen and action taken upon it. This alert mechanism would be typically a pager, an email, a voice message, a report or a referral to another system such as a telephony based escalation system or a central framework management system such as HP OpenView[iii].

- **To Take Action**
  Take action automatically upon receipt of the message.
  This may take the form of a command or series of commands back to the console or indeed to another console managed by the console management system or to run a script or program which will carry out a necessary recovery task. Part of that action will, of course, be to notify a person that such a recovery has taken place.

- **Remote Access and Framework Integration**
  Remote access to the console is a key part of the functionality of the management regime. Many of the benefits come from the ability of personnel to fix problems or investigate them from their own desks without having to go to the server room.
  That remote access capability should give “real” console capability – for example the ability to initiate and follow through a reboot or carry out tasks in single user mode.
  It should also allow integration with framework management systems (for example HP OpenView/ITO” to facilitate action on the console being taken by that framework system.

- **Logging**
  The system should give the ability to log both events which are detected and which are

acted upon by the system but also to log all input and output on the console ports. This is particularly important for systems which don't themselves log console activity (such as non-server devices) or where the overhead of doing so on the operating system is undesirable.



A typical schematic for a comprehensive console management automation system

All these operations should be possible concurrently on a workstation monitoring multiple system console ports providing consolidation along with automation.

Probing for Information

The operating systems will, as we have discussed, give you a certain amount of information free. However, to get further information and to check your systems out on a periodic basis, you will need to proble.
In an ideal world real human beings would enter the commands periodically and check the results. In the real world they either have no time to do that or they forget. We need to ensure that the job gets done regardless.

In our model of using the console as the fount of all knowledge, we want to relay that information to the console where it can be picked up by the console automation system and relayed. The alternative is to use a full blown automated system management tool – a story for another day.

We want the our methodology to be one of *exception management*. There is little point in sending messages that say everything is in good order. That would be just noise which we don't need. In the exception management environment we know that everything is good because we have not been told otherwise. Provided that we *are* checking that things are good then that scenario is fine.

The basic principle is simple enough: carry out a task periodically, check the results, if all is clear then carry on (perhaps logging result of the test) but if all is not as it should be send the message to the console where it can be acted upon.

There are two methods of carrying out these tasks. One is free and assumes that you have a person each console. It involves running a task in a scheduler which ends in the alert being sent to the console. The other method is to have the console management system carry out the task for you on the console itself. The latter is easier to do because you don't have to route the error messages, they are right there in the data stream being monitored by the system.

Let us take a simple example. We wish to check that the system is still connected to the Internet or that connectivity with another system is good. Using our console methodology, we can carry out a ping to a remote system and direct the result to the console.

If we are doing it on the console directly with the console management system, just issue the command:

```
ping 193.123.32.2 -n 5
```

If you are doing it in a script to be executed by cron, use the syntax

```
#! /bin/ksh
# Script to carry out a ping test and direct the result to the
console
ping 193.123.32.2 -n 5 > /dev/console
```

In either case, the console management system can detect the string "`100% packet loss`" and carry out the necessary alert. Note that the IP address above is fictional and will always result in failure.

The action taken by the console management system will depend upon its degree of sophistication. Ideally, an automated recovery procedure could be initiated but at the very least the system should be able to rais an alert by some method such as an email, a page, a voice message or the referral of the alert to some system which is watched at all times such as HPOpenView.

A similar simple mechanism could be used, for example, to check for disk space. The command `bdf > /dev/console` can be used to send the output to the console where the message manager can be used to detect whether there is a problem or not. For example, it can check whether the utilisation on any volume is greater than a given threshold in the output:

```
Filesystem          kbytes    used   avail %used Mounted on
/dev/vg00/lvol1     199610  171242    8407   95% /
/dev/vg00/lvol3    1629530  960712  505865   66% /usr
/dev/vg00/lvol4    1297994  307261  860933   26% /opt
```

These examples are very simple and are for HPUX but the same methodology may be used for other platforms and for other more sophisticated tests.

Benefits of Console Management

It would be possible to write at much greater length about the technicalities and possibilities of such a management regime but they rapidly become obvious and no doubt the reader will have his or her own ideas about what is important and how he or she would use such a system.

Ultimately what is most important is not the technicalities but the benefits to the business or organisation and the reasons for implementing such a methodology or system.

These are:

1. Improvement in availability and reliability of business critical servers through the detection of problems more quickly and the discovery of potential problems through more effective monitoring.

2. Improvement in availability of systems and time to repair through remote access to servers removing travelling time.

3. Staff savings through automation of routine monitoring and other system operations and management tasks.

4. Staff time saving through the ability to remotely manage servers.

5. The ability to improve physical security of data centers by reducing the requirement for routine physical access.

6. In some cases, the original physical console can be dispensed with. In cases where large numbers of servers are involved this can save real estate and maintenance costs on terminals.

The Author

Kim Harris has been involved with Hewlett-Packard systems since 1978 and has experience in system operations, management,and the design, authoring and deployment of applications and system utilities.

Kim has been a system management tools vendor in the community since 1985 and currently is the owner and Managing Director of Entrix Computing Ltd. based in England with an office in Salem, Oregon. Entrix specializes in open systems management services and tools.

---

[i] UNIX is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.
[ii] Window NT is a registered trade mark of Microsoft Corporation.
[iii] HP OpenView is a product and trade mark of Hewlett Packard Company.