OpenVMS Technical Journal V12, January 2009

Table of Contents

OpenVMS Technical Journal V12

Intelligent Scheduling using fuzzy logic in applications

Sushruta C. Nadig

In an enterprise environment, multiple applications performing specific tasks on the production systems can cause huge performance bottlenecks, reducing system efficiency and productivity. To reduce these bottlenecks, some tasks, such as those that consume CPU, perform lot of I/O, or require time to complete, can be scheduled in intervals when the load on the system will be smaller. Job scheduling can be done based on the type of job, CPU consumption, priority, and other factors. But to do all these tasks automatically, an intelligent scheduler is required.

An automated system that can schedule activities intelligently and will provide improved performance without human intervention would be helpful for most system administrators – and fuzzy logic can do it.

The problem with current schedulers

Every application is designed for a specific task – defragmentation tool, backup solution, database application, operating system scheduler dispatching jobs. Scheduling is mostly done based on user-specified times or by straightforward rules, such as backing up, running defragmentation every day, after certain data is written, etc.

In an operating system, jobs can be scheduled based on the FIFO basis, priority, CPU time, etc. But these application or operating system schedulers are rule-based and do not have intelligence embedded that will decide when to schedule, what to schedule, and how to schedule.

Automating: scheduling by adding intelligence

Automated scheduling triggers jobs by monitoring certain conditions or parameters in an IT environment. Unlike the traditional method of scheduling, sometimes there could be occasions where regular jobs do not need to be executed. Typically, we would require the following things for automating scheduling:

- **Determine the inputs for automation.** While setting up the automated scheduling, the input parameters for any application will have to be determined. For example, CPU utilization and I/O activity on a system and disk can be the input parameters for triggering a defragmentation process.

- **Analyze the input and make a decision.** For example, if a disk is highly fragmented, it is ideal to trigger defrag when CPU utilization on the system is lowest and I/O activity is also low.

  In any scenario, there are many possible outcomes to consider before any decision is made. A decision table can be formed by considering all the values for inputs. This is similar to what the human brain does while deciding something: Multiple conditions will be considered and an action will be decided upon.

- **Based on the decision, trigger the action that is expected.** Generally, the result of automating schedules is the running of specific jobs. Based on the desired outcome, we can customize the decision table. For example, the system will either trigger defragmentation and stop ongoing processes, or the system will not trigger defragmentation.

  The parameters can be monitored periodically or continuously and, based on the data, a decision can be made. The inputs can be very specific or generic. For example:

  Specific input: If CPU load is equal to 22%, then execute Backup.

  Generic input: If CPU load is low and I/O is high, then trigger fewer jobs. Low, high, and fewer are all linguistic variables that can take a range of values. While the human brain can handle this generic input, a system cannot handle it directly. However, by applying fuzzy logic, inexact variables can be handled and action taken.

Fuzzy logic, fuzzy sets, rules, and inference model

Fuzzy logic is a superset of conventional Boolean logic and extends it to deal with new aspects such as ambiguity and uncertainty. The basic elements of fuzzy logic are linguistic variables, fuzzy sets, and fuzzy rules. Fuzzy logic provides a method to assign values between 0 and 1.

The linguistic variables' values are words – specifically, adjectives such as small, little, medium, and so on. The fuzzy set generalizes the concept of a classical set, allowing its elements to have a partial membership. To assign this partial membership, separate functions will be written that are called membership functions. The membership function of a fuzzy set corresponds to the indicator function of the classical sets. For example:

The set Low ranges from 0 to ∞; any point from 0 to ∞ can become a member of the set. Membership functions determine how much each point belongs to the set. The value 0 will be the lowest of all the points in the range specified. Therefore, 0 is assigned the value of 1 and ∞ is assigned the value of 0. All the points between 0 and ∞ will be decimal values between 0 and 1.

Membership functions assign these values to the number of points we would like to have. This operation normalizes all inputs to the same range and has a direct effect on system performance and accuracy.

Fuzzy rules form the basis of fuzzy reasoning. They describe relationships among imprecise, qualitative, linguistic expressions of the system's input and output. Generally, these rules are natural language representations of human or expert knowledge and provide an easily understood knowledge representation scheme.

A typical conditional fuzzy rule assumes a form like the following:

IF CPU load is "Low" AND IO is "High" THEN trigger applications that do not do IO.

"CPU load is 'Low' AND IO is 'High'" are the inputs; "trigger applications that do not do IO" is the consequence.

The inputs specified might not be completely true or false, and their degree of truth may range from 0 to 1. We compute this value by applying the membership functions of the fuzzy sets labeled "Low" and "High" to the actual value of the input variables "CPU load" and "IO". After that, fuzzy logic is applied to the conclusion, depending on the conclusion model. Generally there are two types of models for deciding the inputs:

- Output is direct and precise; this is also called a singleton output membership function.
- Output is ambiguous and needs de-fuzzification for taking action.

Fuzzy logic in scheduling

The following steps are used to automate scheduling using fuzzy logic. This example shows how to automate scheduling for a defragmentation tool. Fuzzy logic is used here only to determine the input sets, which will be fragmentation index and CPU load. The output is binary: defrag or stop.

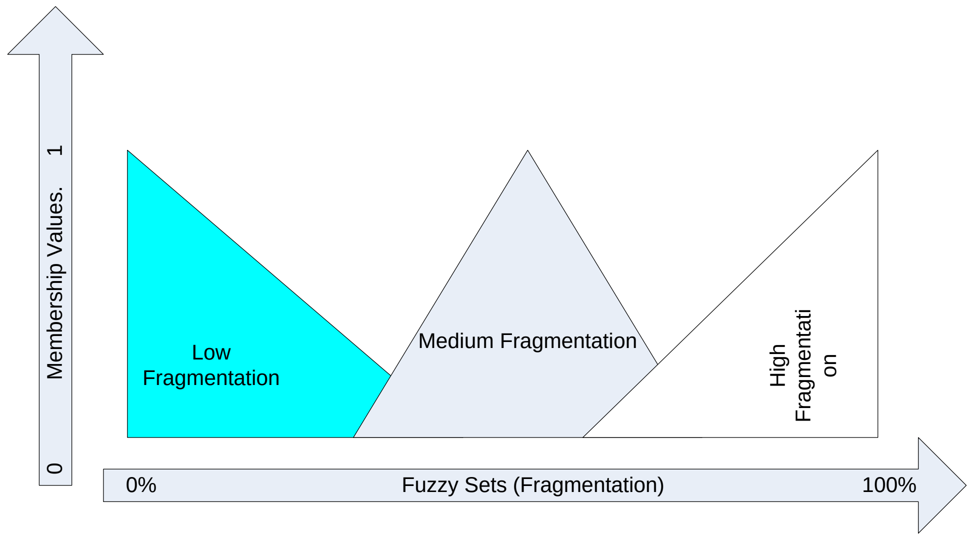
- **As mentioned above, determine the inputs for the system you are trying to automate.** Defragmentation is a very high CPU-consuming activity and can cause other applications to stall. If an automated scheduler can be associated with the defrag tool, that will take care of triggering defrag operation when the system is less occupied and when the disk is fragmented above a certain level. We can consider CPU load and fragmentation level as inputs for this system.

- **Check whether the input holds uncertainty. If yes, fuzzy logic can be applied.** While triggering defragmentation, rules such as "if CPU is Low and fragmentation is High then trigger defrag" will be used. Therefore, we need to fuzzify values for CPU load and fragmentation index.

- **Form fuzzy sets for the inputs decided.** In this scenario, Low, Medium, and High are the linguistic variables that can be associated with CPU load. The same variables can be applied to the fragmentation index. Before we proceed further, we need to decide upon the limits for the fuzzy sets we want to create.

  Assigning 0 to ∞ is an ideal way of representing the fuzzy set "Low". But to be realistic, we will have to set the boundaries for the fuzzy sets we are creating.

Figure 1. Typical division of fuzzy sets



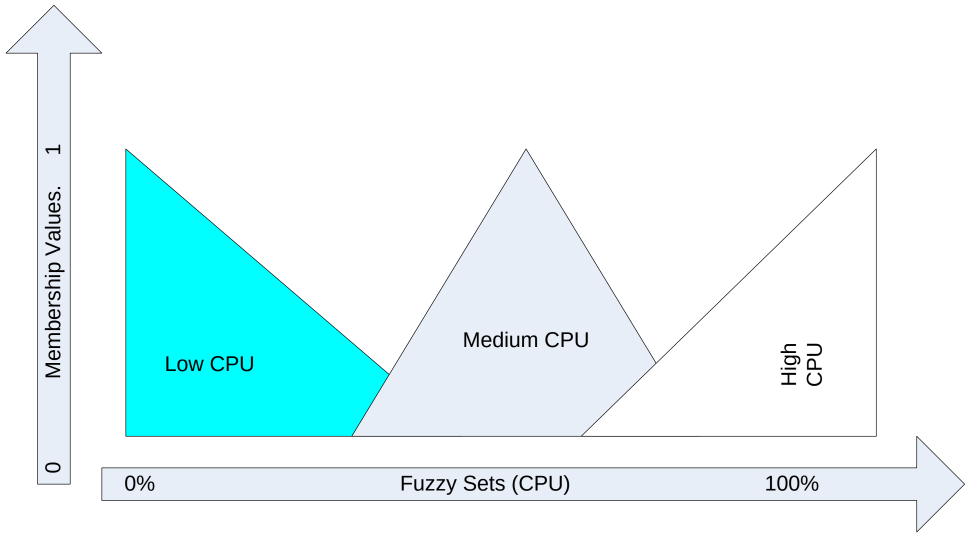


Figure 1 shows a typical division of fuzzy sets for the inputs decided in this example. The triangles in the figures indicate the ranges each set is occupying. For example, fuzzy set "Low" for the CPU ranges from 0 to 45. This range is decided based on what kind of input we are working on, to make it more realistic.

- Assign membership charcateristics for all the values the inputs can take and associate that value to the fuzzy set it belongs to.

  Consider CPU load as the input. If the CPU load on the system is 45%, map the value to the sets divided above and see where it belongs. 45% can belong to both the Low and Medium fuzzy sets. If a value corresponds to two fuzzy sets, check for how much it belongs to a particular set. A value of 45 in this example can be 2% of the set Low but 10% of the set Medium, so it can be considered an input corresponding to the set Medium. Work out this association for every input.

- Form a decision table to compare the inputs and take appropriate action according to that table. Based on the classification of inputs from the fuzzy sets, a decision would be made by considering both the inputs, as discussed above. This decision can be changed based on the outcome of the results from the tests carried out.

The idea is the decision should not be generic – it should be specific. We are eliminating the traditional way of scheduling based on the time-ordered queue. We have tried implementing this for DFO, and have found that it works well in improving performance.

Decision	**Low CPU load**	**Mid CPU load**	**High CPU load**
Low fragmentation	Execute defrag	Execute defrag	Not to execute
Medium fragmentation	Execute defrag	Not to execute	Not to execute
High fragmentation	Execute defrag	Not to execute	Not to execute

This completes the steps to automate scheduling defragmentation using fuzzy logic. In the examples above, there are more implications that would have to be handled during implementation. However, this is the basic approach to automating scheduling for applications using fuzzy logic. The same model can be used for applications such as backing up, OS scheduling, cleaning jobs, etc.

For more information

http://paper.ijcsns.org/07_book/200603/200603A13.pdf

OpenVMS Technical Journal V12

Utilizing /RETAIN=ERROR to make life easier

Bruce Claremont, Senior Consultant

Overview

The /RETAIN=ERROR qualifier is a useful tool to identify problem batch and print jobs. This paper discusses how to deploy the qualifier to your advantage as part of OpenVMS system maintenance and monitoring.

It is surprising how few OpenVMS sites implement /RETAIN=ERROR on their queues –particularly batch queues. At certain sites, a review of log files reveals that some jobs have not successfully run to completion for years. Since the logs aren't checked, no one is aware of the problem, which often leads to painful and expensive long-term data corruption.

One of the easiest ways to impress users is to contact them when a job goes bad – before they become aware of it. Proactive maintenance of this sort will help maintain a smooth-running system. The goal is less work for administrators, and /RETAIN=ERROR can help you achieve it.

How it Helps

/RETAIN=ERROR retains job information on a queue if the job ends abnormally. The primary cause of an abnormal termination for a batch job is an error within the job. Retaining the job information tells us what job failed and what parameters were used to invoke it. Reviewing the log file tells where the job failed, which helps identify the root cause of the problem.

For print jobs, retaining the job information when there is an error can help administrators identify an invalid PRINT statement, corrupted print file, or problem with the printer. As with batch jobs, knowing a problem exists is the first step towards eliminating the problem.

Retaining Log Files

Retaining batch job log files is strongly recommended. The /NOPRINT and /KEEP qualifiers ensure a log file is retained after a job completes. OpenVMS's default behavior is to delete a log file if a job completes successfully unless /KEEP or /NOPRINT are specified.

It's good practice to retain several versions of a log file. Having a complete log file from a valid run to compare to a log file from a failed run is a useful troubleshooting tool. An appropriately tuned PURGE command following the SUBMIT command ensures log files do not pile up unnecessarily.

```
$ SUBMIT /NOPRINT /LOG=LOGS: DOWNLOADS:5_MINUTE_CHK.COM
$ PURGE /KEEP=12 LOGS:5_MINUTE_CHK.LOG
```

Figure 1: Retain the last 12 log files for the procedure 5_MINUTE_CHK.

```
$ SUBMIT /NOPRINT /LOG=PAYLOGS: PAYROLL:MONTH_END.COM
$ PURGE / BEFORE="TODAY-365-00" PAYLOGS:MONTH_END.LOG
```

Figure 2: Retain the last year's worth of Month End payroll logs. How does this retain 12 months of logs? Should AFTER include a parameter?

/RETAIN=ERROR set up for a queue

Configuring a queue to enable /RETAIN=ERROR is easy. Simply add the /RETAIN=ERROR qualifier to the INITIALIZE command when creating the queue.

```
$ INITIALIZE /QUEUE /RETAIN=ERROR /BATCH PAYROLL_BATCH
```

Figure 3: Create a queue with /RETAIN=ERROR.

You can also add the /RETAIN=ERROR qualifier to an existing queue with the START or SET queue commands.

```
$ START /QUEUE /RETAIN=ERROR SYS$BATCH
```

Figure 4: Adding /RETAIN=ERROR on queue start up.

```
$ SET QUEUE /RETAIN=ERROR SYS$BATCH
```

Figure 5: Adding /RETAIN=ERROR to a running queue.

Using /RETAIN=ERROR at the queue level is a great way to monitor jobs. It allows you to know when any job fails so the root cause can be determined and the problem eliminated. However, /RETAIN=ERROR usage can be refined to the job level if so desired.

/RETAIN=ERROR set up for a job

/RETAIN=ERROR can be specified for individual jobs. However, remember that /RETAIN settings on generic and execution queues to which the job is submitted can override the local job setting. As OpenVMS Help says:

> "Although you can specify job retention options for your own jobs, the job retention option you specify may be overridden by the job retention option of the queue on which your job executed. If you submit or print a job to a generic queue, the generic queue's job retention setting may also override the job retention option you specify. This section describes how job retention is determined.
>
> An execution queue's job retention setting takes precedence over a generic queue's job retention setting; however, if the job's completion status does not match the job retention setting (if any) on the execution queue, then the generic queue's job retention setting attempts to control job retention. If the job's completion status does not match the job retention setting (if any) on the generic queue, then the user-specified job retention setting is used. Jobs submitted directly to execution queues are not affected by job retention settings on generic queues.
>
> If the execution queue's retention setting applies, the job is retained on the execution queue. Likewise, if the generic queue's retention setting applies, the job is retained on the generic queue. If the user-specified setting applies, the job is retained in the queue to which it was submitted."

Look at the SUBMIT command description in the HP OpenVMS DCL Dictionary for a comprehensive description of /RETAIN precedence rules.

Using /RETAIN=ERROR at the job level allows you to monitor specific jobs. If the queues upon which the jobs run do not have a conflicting /RETAIN policy, the job will be retained if it completes unsuccessfully.

```
$ PRINT /RETAIN=ERROR /QUEUE=LASER_21 RPT:FS3302.LIS
```

Figure 6: Print job FS3302 is retained on queue LASER_21 if it fails to print.

```
$ SUBMIT /RETAIN=ERROR /LOG=LOGS: /KEEP INVENTORY:INV906.COM
```

Figure 7: Job INV906 will be retained on queue SYS$BATCH if it fails to complete successfully.

Checking Job Status

Presuming your user account has the correct privileges or access controls, checking for retained jobs is as easy as issuing a SHOW QUEUE statement. Use the /BY_JOB_STATUS=RETAIN qualifier to limit the display to retained jobs.

```
$ SHOW QUEUE /BY_JOB_STATUS=RETAIN SYS$BATCH
```

Figure 8: Display retained jobs on SYS$BATCH queue.

It makes sense to have OpenVMS check for retained jobs automatically and notify the administrator of any found. A DCL procedure called BATCHQ_CHECK.COM is included in the article *Using OpenVMS to Meet a Sarbanes-Oxley Mandate Part 2: The DCL* that accomplishes this task.

Conclusion

The /RETAIN=ERROR feature provides another simple method to proactively monitor an OpenVMS system while helping to provide end users and management with a smooth-running system. Start using it today to identify and correct problem jobs on your OpenVMS system.

About the author: Mr. Bruce Claremont has been working with OpenVMS since 1983. Mr. Claremont has extensive programming, project management, and system management experience. He founded Migration Specialties in 1992 and continues to deliver OpenVMS and application migration services along with hardware emulation solutions. More information about Migration Specialties products and services can be found at *www.MigrationSpecialties.com*.

For more information

More on DCL development:

- *Using OpenVMS to Meet a Sarbanes-Oxley Mandate Part 2: The DCL*
  http://www.migrationspecialties.com/pdf/Using%20OpenVMS%20to%20Meet%20a%20Sarbanes-Oxley%20Mandate2.pdf
- *Simplification Thru Symbols*
  http://h71000.www7.hp.com/openvms/journal/v10/simplificationthrusymbols.html
- *Simplifying Maintenance with DCL*
  http://h71000.www7.hp.com/openvms/journal/v9/simplifying_maintenance_with_dcl.html

- F$GETQUI to the Rescue
  http://h71000.www7.hp.com/openvms/journal/v11/F$GETQUI.html

For the latest issue of the OpenVMS Technical Journal, go to: http://www.hp.com/go/openvms/journal.

OpenVMS Technical Journal V12

Access Restrictions in FTP

Ananth Shenoy

Currently, FTP service (provided by TCP/IP services for OpenVMS) does not provide a method to restrict access of user operations to a directory or set of directories. This has been inconvenient for system administrators and is risky for the system.

An FTP client (non-anonymous) who logs in can read all world-readable files, delete all world-deletable files, etc. A client's operation cannot be restricted to the login directory or a set of pre-defined directories. System administrators can use ACLs (access control lists) to restrict FTP user access, but ACL setup can be a difficult process. Also, ACL-based access restriction can be quite slow and can reduce the performance of other applications on the system. There have been other options to achieve the same goal, such as the use of the concealed logical, but those methods are even more cumbersome.

Keeping all these pain points in mind, a simple method of restricting user access to a particular set of directories was developed and implemented in TCP/IP v5.6 ECO3 and above.

FTP Anonymous Light

FTP Anonymous Light is a simple method for restricting user access to a particular set of directories. A system administrator who wants to restrict an OpenVMS user's FTP access to a particular set of directories should set TCPIP$FTP_ANONYMOUS_LIGHT for that user. Setting this parameter restricts the FTP operations of that user to a particular set of directories pointed to by TCPIP$FTP_ANONYMOUS_DIRECTORIES. For any particular target user, TCPIP$FTP_ANONYMOUS_LIGHT can be defined in LOGIN.COM. If FTP access must be restricted for all users, the parameter can be defined system-wide. FTP Anonymous Light users have to specify the correct password in order to log in. By contrast, when truly anonymous users are prompted to send their identity, any password is accepted.

The system administrator can also optionally set TCPIP$FTP_ANONYMOUS_WELCOME, and a message will be displayed upon successful login.

This example shows how FTP Anonymous Light works:

```
"TCPIP$FTP_ANONYMOUS_DIRECTORY" = "TCPIP$ENETINFO1:[UCX]"
   = "TCPIP$ENETINFO1:[UCX_AXP]"
   = "TCPIP$ECO:"
   = "TCPIP$PATCH:"
   = "COMMON_SYSDISK:[FAL$SERVER]"
   = "TCPIP$INTERNAL:"
"TCPIP$FTP_ANONYMOUS_LIGHT" = "1"
"TCPIP$FTP_ANONYMOUS_LOG" = "SYS$LOGIN:TCPIP$FTP_ANONYMOUS.LOG"
"TCPIP$FTP_ANONYMOUS_WELCOME" = "FTP Anonymous Light demo"
```

Commentary is marked in bold.

```
ftp plane.tcpip.zko.hp.com
220 plane.tcpip.zko.hp.com FTP Server (Version 5.6) Ready.
Connected to plane.zko.hp.com.
Name (plane.zko.hp.com:shenoy):
331 Username shenoy requires a Password
Password:
230-FTP Anonymous Light demo
230 Guest login OK, access restrictions apply.
FTP> cd sys$system
550 insufficient privilege or file protection violation
```

>>> This directory not in TCPIP$FTP_ANONYMOUS_DIRECTORY, hence access is restricted

```
FTP> cd tcpip$eco
250-CWD command successful.
250 New default directory is TCPIP$ENETINFO1:[TCPIP$ENGINEERING_CHANGE_ORDERS]
```

>>> This directory is present in TCPIP$FTP_ANONYMOUS_DIRECTORY, hence access is allowed

```
FTP> cd sys$login
250-CWD command successful.
250 New default directory is WORK4$:[SHENOY]
FTP> bye
221 Goodbye.
```

A sample log file looks like this:

```
type SYS$LOGIN:TCPIP$FTP_ANONYMOUS.LOG
20-JUN-2008 05:21:45.64 Anonymous Light User:shenoy from Host:16.116.92.100
20-JUN-2008 05:22:39.61 Anonymous Light User:shenoy status:00010001 CWD
dir:TCPIP$ENETINFO1:[TCPIP$ENGINEERING_CHANGE_ORDERS]
20-JUN-2008 05:23:13.49 Anonymous Light User:shenoy status:00010001 CWD
dir:WORK4$:[SHENOY]
20-JUN-2008 05:23:19.15 Anonymous Light User:shenoy status:00000000 RETR
file:WORK4$:[SHENOY]A.TXT;30
20-JUN-2008 05:23:26.07 Anonymous Light User:shenoy logged out
```

Even if the system administrator does not specify it, SYS$LOGIN will always be added to TCPIP$FTP_ANONYMOUS_DIRECTORY. This means that Anonymous Light users will always have access to their SYS$LOGIN.

In some instances, the system administrator may not want a user to access his or her SYS$LOGIN. To set this up, the system administrator should define TCPIP$FTP_ANONYMOUS_NOSYSLOGIN for that particular user. This parameter is particularly useful when a user has changed their directory in LOGIN.COM and the system administrator does not want to allow access to SYS$LOGIN.

Access restrictions by FTP Operations

The FTP Anonymous Light feature restricts user access to a particular set of directories. To further increase the system administrator's flexibility, a new set of parameters can be defined to restrict user operations.

The FTP server will check for the existence of four parameters. If each is defined, the FTP server will reject all:

TCPIP$FTPD_NOLIST - LIST and NLST commands

TCPIP$FTPD_NOREAD - RETR commands

TCPIP$FTPD_NOWRITE - STOR, STOU, APPE, RNFR, RNTO, DELE, MKD, and RMD commands

TCPIP$FTPD_NODELETE - DELE and RMD commands

These new access restrictions will apply in addition to any restrictions implied by the protections of the underlying files, directories, volumes, and devices.

If TCPIP$FTPD_NOLIST is defined, the use of wildcards will not be allowed in FTP operations. This is necessary in order to prevent FTP users from obtaining a list of the files in the directory simply by attempting to retrieve or delete all the files.

Below is a table of FTP client commands and the parameters used to control their operation:

Client command	**FTP Logical**
directory	TCPIP$FTPD_NOLIST
view	TCPIP$FTPD_NOREAD
put	TCPIP$FTPD_NOWRITE
get	TCPIP$FTPD_NOREAD
append	TCPIP$FTPD_NOWRITE
rename	TCPIP$FTPD_NOWRITE
create	TCPIP$FTPD_NOWRITE
delete	TCPIP$FTPD_NOWRITE, TCPIP$FTPD_NODELETE

So, for example, if a system administrator does not want a user to be able to delete files through FTP, TCPIP$FTPD_NODELETE can be set for that user.

This is a simple illustration of the usage of the above parameters:

```
 "TCPIP$FTPD_NODELETE" = "1"
 "TCPIP$FTPD_NOLIST" = "1"

$ ftp plane.tcpip.zko.hp.com
220 plane.tcpip.zko.hp.com FTP Server (Version 5.6) Ready.
Connected to plane.zko.hp.com.
Name (plane.zko.hp.com:shenoy): shenoy
331 Username shenoy requires a Password
Password:
230-FTP Anonymous Light demo
230 Guest login OK, access restrictions apply.
FTP> dir *
200 PORT command successful.
550 Cannot execute LIST command, Access denied.
```

>>> Here dir command is not allowed, because of wildcard present in command and TCPIP$FTPD_NOLIST is defined

```
%TCPIP-E-FTP_NOSUCHFILE, no such file *
FTP> del a.txt
550 Cannot execute DEL command, Access denied.
```

>>> Here del command is not allowed, because of logical TCPIP$FTPD_NODELETE is set

```
FTP> bye
221 Goodbye.
```

These parameters can be used in conjunction with FTP Anonymous Light to restrict user access via FTP, helping to mitigate a risk to the system that has been problematic for system administrators.

OpenVMS Technical Journal

Alternatives to HP SNA Server for OpenVMS HP Integrity servers

Ravindhar Uppada & Avinash Bilumane Lingappa, IBM Interconnect Engineering

Alternatives to HP SNA Server for OpenVMS HP Integrity servers
Ravindhar Uppada & Avinash Bilumane Lingappa

Executive summary

HP SNA Server for OpenVMS Alpha (SNA Server) is a layered software product that allows appropriately configured OpenVMS Alpha or VAX systems to be part of an IBM Systems Network Architecture (SNA) network. After installing the SNA Server and one or more SNA access routines, you can perform the following functions:

- Access IBM application programs or other system resources
- Act as 3270 display station
- Exchange data files and documents with an IBM host
- Implement distributed application programs that run between the OpenVMS Alpha and IBM systems

HP SNA Server for OpenVMS will not be ported to HP Integrity servers. However, you can use the following communication options for an OpenVMS Integrity server system to become part of an IBM SNA network:

- HP SNA Access Server for Microsoft® Windows® (Access Server)
- HP DECnet SNA Gateway for Synchronous Transport (ST)
- HP DECnet SNA Gateway for Channel Transport (CT)
- Mainframe Gateway for OpenVMS (MGO)

You can also continue to run HP SNA Server for OpenVMS Alpha or VAX systems and move the applications on to Integrity servers. The applications can access the SNA Server through DECnet or TCP/IP. This solution can be used only if you would like to retain the existing Alpha or VAX systems to run the SNA Server.

Refer to Table 1 in Appendix A for the SNA gateways compatible with OpenVMS applications. Table 2 shows the requirements for communication between OpenVMS systems and the mainframe.

Intended audience

This paper is intended for anyone using HP SNA solutions based on OpenVMS systems. It is also intended for HP customer support representatives.

HP SNA Access Server for Windows (Access Server)

The Access Server has the following components that are implemented as part of Microsoft Windows Server services:

- APPC service, which supports the OpenVMS SNA LU6.2 applications
- Basic service, which supports the layered OpenVMS SNA applications that are not LU6.2-based, such as LU types 0, 1, 2, and 3

The APPC and Basic services provide the means for OpenVMS SNA applications to use Microsoft Host Integration Server (HIS) for connecting to IBM SNA-based systems.

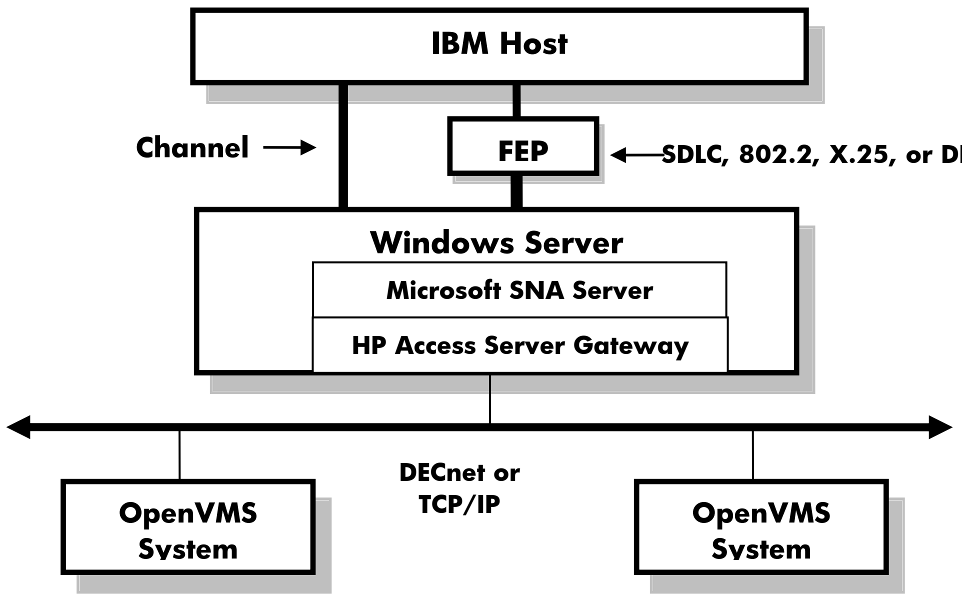
The Access Server is a member of the HP SNA Gateway product family. It allows you to exchange information and share resources between configured OpenVMS systems in DECnet and TCP/IP environments in a bidirectional manner. This server works in conjunction with the Microsoft HIS and an IBM system in an SNA environment.

It provides a means for OpenVMS access routines and custom applications developed using the HP SNA APPC/LU6.2 Programming Interface for OpenVMS to work across the Microsoft HIS with little or no modification to the applications.

The Access Server enables the Microsoft HIS to function as an HP SNA Gateway between TCP/IP or DECnet networks and an IBM SNA network.

The minimum hardware requirement is a Windows system with 64 MB memory and a network adapter supported by Microsoft HIS. Figure 1 illustrates the network configuration using Access Server Gateway.

Figure 1. Network configuration using HP Access Server for Windows



Features

The Access Server provides the following features:

- Will support a total of 2,000 simultaneous OpenVMS access routine connections, which can be a combination of access routines over DECnet and TCP/IP
- Offers flexible communication between networks, providing both hierarchical and peer LU communication
- Supports connections for LU types 0, 1, 2, and 3 or LU type 6.2 connections between IBM application programs on an IBM SNA network and multiple operating systems (Windows and OpenVMS) running on DECnet or TCP/IP networks
- Supports APPN network configuration for program-to-program connection between IBM applications and OpenVMS

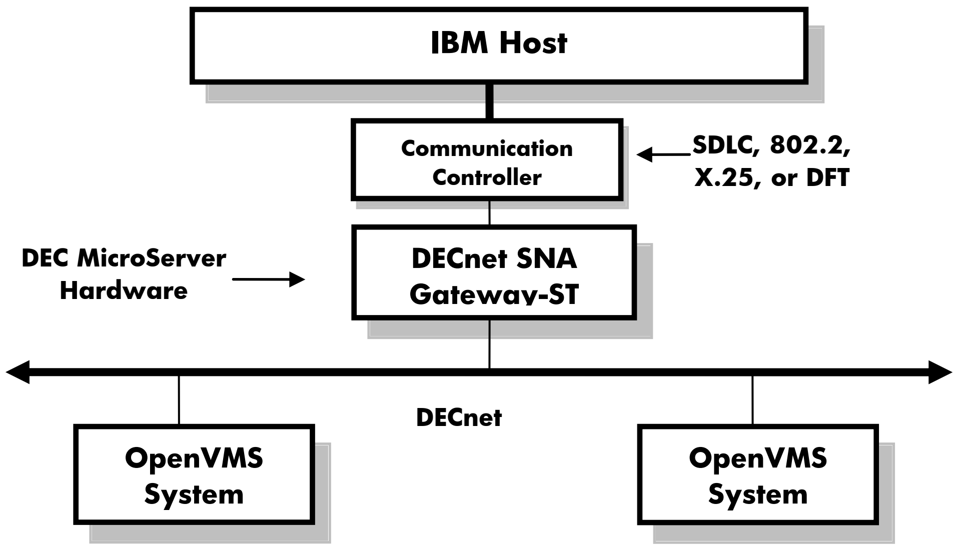
HP DECnet SNA Gateway for Synchronous Transport (ST)

HP DECnet SNA Gateway-ST is a member of the DECnet SNA Gateway product family. This family consists of hardware and software products that allow you to exchange information and share resources between the appropriately configured OpenVMS systems in a DECnet environment and IBM systems in an SNA environment.

Gateway-ST can be used to connect the DECnet and SNA networks. It can be used by a wide variety of HP-supplied access routines running on OpenVMS Integrity servers, and in Alpha, VAX, and Tru64 UNIX operating environments. Gateway-ST is managed and configured using the DECnet SNA gateway management software that is included with HP DECnet SNA Gateway-ST.

Figure 2 illustrates the network configuration using DECnet SNA Gateway for Synchronous Transport.

Figure 2. Network configuration using DECnet SNA Gateway-ST



Gateway-ST is part of both the DECnet and SNA networks. Architecturally, it is a DECnet Phase IV end-node implementation to DECnet and a Physical Unit (PU) Type 2.0 to SNA. It provides bidirectional network access between DECnet and SNA networks.

The Gateway-ST works with either of the DEC MicroServer hardware platforms. The DEC MicroServer (DEMSA-S*) provides connectivity for up to four synchronous lines; DEC MicroServer-SP (DEMSB-S*) provides a single synchronous line connection.

Note:
The term "DEC MicroServer" refers to a hardware device.

Features

The following functionalities provided by Gateway-ST are comparable to those of the SNA Server software:

- Gateway-ST appears to the SNA network as a PU Type 2.0 node
- Gateway-ST supports SNA synchronous communications using the SDLC protocol

The number of links supported depends on the hardware unit that is used. Using the DEC MicroServer (DEMSA-S*) hardware supports the following:

- Four SDLC line connections
- 128 concurrent SNA Logical Unit sessions per PU

Using the DEC MicroServer-SP (DEMSB-S*) hardware supports the following:

- One SDLC line connection
- 32 concurrent SNA Logical Unit sessions per PU

Restrictions

Deployment of the DECnet SNA Gateway-ST communication option depends on the availability of hardware – DEC MicroServer hardware is mandatory. For details regarding DEC MicroServer support, contact an HP representative through a standard support channel. The gateway hardware devices are no longer offered from HP, so this gateway is an alternative only if the gateway already exists.

HP DECnet SNA Gateway for Channel Transport (CT)

HP DECnet SNA Gateway for Channel Transport is a member of the DECnet SNA Transport product family. This family consists of hardware and software products that allow you to exchange information and share resources between appropriately configured OpenVMS systems in a DECnet environment and IBM systems in an SNA environment.

Gateway-CT can be used to connect the DECnet and SNA networks. It can be used by a wide variety of HP-supplied access routines running on OpenVMS Integrity servers, and in Alpha, VAX, and Tru64 UNIX operating environments. The Gateway-CT is managed and configured using the DECnet SNA gateway management software that is included with HP DECnet SNA Gateway-CT.

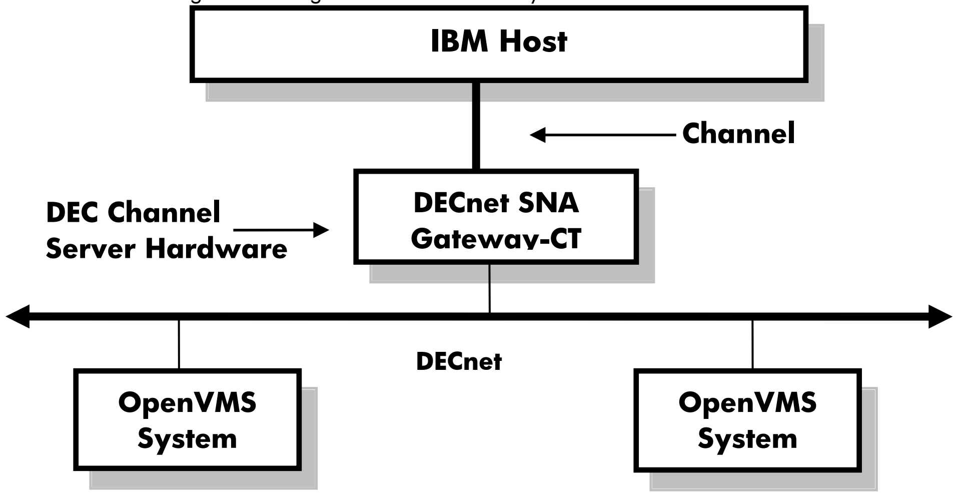
Gateway-CT is part of both the DECnet and SNA networks. Architecturally, it is a DECnet Phase IV end-node implementation to DECnet and PU Type 2.0 node to SNA. It provides bidirectional access between the DECnet and SNA networks.

The Gateway-CT software is supported on the DEC ChannelServer (DESNA-Ax) and DEC ChannelServer II (DESNB-Ax) hardware platforms. Both hardware systems connect directly to Ethernet local area network (LAN) and IBM S/370 Byte Multiplexer, Block Multiplexer, or Selector channel. When used with Gateway-CT, both systems provide SNA connectivity to any DECnet system in a DECnet network.

Note:
The term "DEC ChannelServer" refers to a hardware device.

Figure 3 illustrates the network configuration using DECnet SNA Gateway-CT.

Figure 3. Network configuration using DECnet SNA Gateway-CT



Features

Gateway-CT provides the following features:

- Appears to the SNA network as a PU Type 2.0 node
- Supports only channel connectivity with SNA

Restrictions

Deployment of the DECnet SNA Gateway-CT communication option depends on the availability of hardware – DEC ChannelServer hardware is mandatory. For details regarding DEC ChannelServer support, contact an HP representative through a standard support channel. The gateway hardware devices are no longer offered from HP, so this gateway is an alternative only if the gateway already exists.

Mainframe Gateway for OpenVMS (MGO)

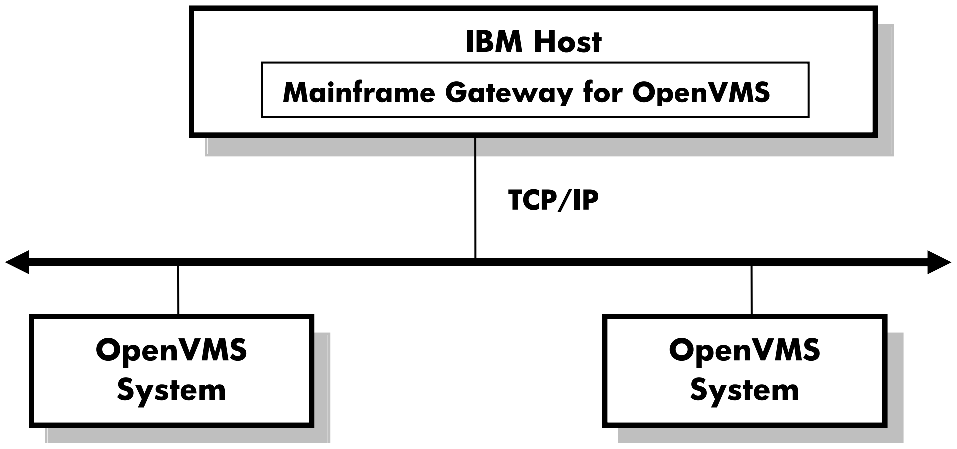
The Mainframe Gateway for OpenVMS can serve as an alternative to any of the HP SNA Gateway products. This software-only solution, developed by Data Access Incorporated (DAI), executes on an IBM mainframe. DAI offers a wide variety of IBM mainframe-related software products and services. This product is designed to work with HP products, but it is the sole property of DAI. For more information on DAI gateways, go to:

http://www.data-access-inc.com/html/gateways.html

MGO can be used as an alternative to the SNA gateways (CT, ST, Access Server, or SNA Server) used with OpenVMS application programs. It is a software solution only, and no new or additional hardware is required for implementing this communication option. No additional new software is required for installation on OpenVMS systems.

After the MGO software is installed and configured on the IBM mainframe, MGO appears to the OpenVMS systems as another HP SNA gateway. Figure 4 illustrates the network configuration using MGO.

Figure 4. Network configuration using MGO



MGO is associated with a number of SNA Logical Units (LUs), like any other HP SNA Gateways. Each OpenVMS application program uses one or more of these LUs to communicate with the other components in the SNA network. MGO requires no changes in the OpenVMS application programs; configuration changes are only required on the OpenVMS systems.

Features

MGO provides the following features:

- Compatible with all OpenVMS applications using any HP SNA Access routines
- Is IP-based and implemented on the IBM mainframe; can communicate with OpenVMS Alpha or Integrity server systems
- Can be used to replace any current HP OpenVMS SNA Gateway product

MGO provides the following advantages over other communication systems:

- SDLC and X.25 communication lines dedicated to OpenVMS systems are no longer needed.
- Separate SNA and IP networks are not required.
- HP hardware gateways are not required.
- Faster data transfer speeds are provided.
- It is a reliable system since no hardware – such as Gateway-CT, Gateway-ST, or 3745 IBM front-end processor – is required.
- TCP/IP is used instead of DECnet or SDLC.
- Yearly service and support charge is low.

Appendix A: compatibility and requirements

Table 1 lists the various SNA gateways compatible with OpenVMS applications using any HP SNA Access routines. Table 2 lists the communication requirements between an OpenVMS system and a mainframe.

Table 1. SNA gateways compatible with OpenVMS applications

SNA Access routine name	**MGO**	**Access Server**	**DECnet SNA Gateway-ST**	**DECnet SNA Gateway-CT**
SNA 3270 Terminal Emulator for OpenVMS	Yes	Yes	Yes	Yes
SNA APPC/LU6.2	Yes	Yes	Yes	Yes
SNA Data Transfer Facility for OpenVMS	Yes	Yes	Yes	Yes
SNA Application Programming Interface for OpenVMS	Yes	Yes	Yes	Yes
SNA 3270 Data Stream Programming Interface	Yes	Yes	Yes	Yes
SNA Printer Emulator for OpenVMS	Yes	Yes	Yes	Yes
SNA Remote Job Entry for OpenVMS	Yes	Yes	Yes	Yes
DECWindows DECnet SNA 3270 Terminal Emulator for OpenVMS	No	Yes	Yes	Yes

Table 2. Communication requirements between an OpenVMS system and a mainframe

Communication options and solution	**MGO**	**Access Server**	**DECnet SNA Gateway-ST**	**DECnet SNA Gateway-CT**
OpenVMS to gateway communication	TCP/IP	DECnet and TCP/IP	DECnet only	DECnet only
Gateway to mainframe communication	SNA within mainframe	Various	SDLC	Channel
Solution based on:	• Software only • Runs on mainframe itself • No additional hardware required	• Software and hardware • Runs on Windows	• Software and hardware • Runs on DEC ChannelServer	• Software and hardware • Runs on DEC MicroServer

Calling OpenVMS native routines from Java

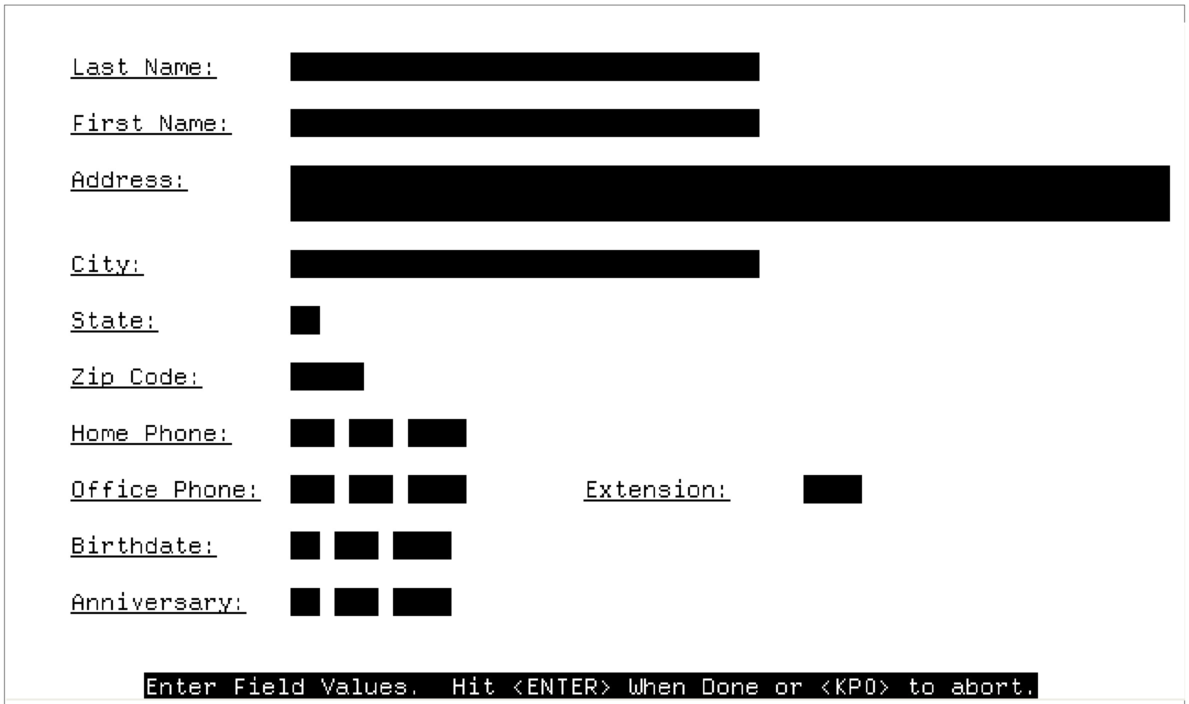
Tim E. Sneddon

Overview

Developing a Java application that calls native routines can be tedious. Along with the native routines and the Java application, there is also the need to develop a Java Native Interface (JNI) layer. This article presents a collection of tools and libraries that remove the need to develop the JNI layer—allowing Java to call native routines and manipulate data structures directly.

The "legacy" application

For this tutorial, a simple native application written in PL/I has been selected as the target of an "upgrade". The application is an address/phone book program, called PHONE[1] (not to be confused with the OpenVMS utility of the same name). It uses the SMG$ API to draw the screen and an RMS index file to store each phone book record. Figure 1 shows the record entry screen.



```
Last Name:
First Name:
Address:
City:
State:
Zip Code:
Home Phone:
Office Phone:                          Extension:
Birthdate:
Anniversary:

     Enter Field Values.  Hit <ENTER> When Done or <KP0> to abort.
```

Figure 1. PHONE data entry screen

The intention is to be able to access the phone book data file from Java so a new GUI can be built. By using Java, the options for future development are expanded to allow development of either a Swing-based (GUI) interface or a web-based front end (using Tomcat). However, the most important requirement is to allow the original application to continue being used with little or no change.

Setting up the run-time library

In order for Java to call a native routine it must be in a shareable image. The first step is to identify the modules that are necessary for accessing the data file. In this case, it is simple: the DATABASE.PLI module contains all routines related to file access. From this module, the information about its entry points are as follows:

- OPEN_PHONEBOOK—This routine opens the phone book datafile.
- CLOSE_PHONEBOOK—This routine closes the phone book datafile.

[1] For those with a PL/I compiler this program can be found in the examples directory for the Kednos VAX and Alpha PL/I compilers.

- GET_A_RECORD—This routine reads a record with the specified last name and returns a pointer to that record.
- GET_A_MONTH_RECORD—This routine reads a record for an event that occurs in the specified month. It returns a pointer to the record.
- GET_A_DATE_RECORD—This routine reads a record for an event that occurs on the specified date. It returns a pointer to the record.
- WRITE_A_RECORD—This routine writes the specified record into the phone book database.
- DELETE_A_RECORD—This routine deletes the specified record from the phone book database.

```
$ PLI PHONE
$ PLI DATABASE
$ PLI SCREEN
$ LINK PHONE,DATABASE,SCREEN
$ IF (F$SEARCH("PHONE.DAT") .EQS. "") THEN -
$ CREATE/FDL=PHONE.DAT
```

Example 1. Original PHONE build procedure

This information is then used to build a shareable image symbol vector that details the entry points to the linker. Example 1 shows the original build procedure for the self-contained application, which is one single executable. Example 2 shows the new build procedure (changed sections are highlighted in yellow).

```
$ PLI PHONE
$ PLI DATABASE
$ PLI SCREEN
$ LINK/SHARE=DATABASE_RTL.EXE DATABASE,SYS$INPUT/OPTION
SYMBOL_VECTOR = ( -
 OPEN_PHONEBOOK = PROCEDURE, -
 CLOSE_PHONEBOOK = PROCEDURE, -
 GET_A_RECORD = PROCEDURE, -
 GET_A_MONTH_RECORD = PROCEDURE, -
 GET_A_DATE_RECORD = PROCEDURE, -
 WRITE_A_RECORD = PROCEDURE, -
 DELETE_A_RECORD = PROCEDURE -
 )
$DEFINE/NOLOG DATABASE_RTL SYS$DISK:[]DATABASE_RTL
$ LINK PHONE,SCREEN,SYS$INPUT/OPTION
DATABASE_RTL/SHARE
$ IF (F$SEARCH("PHONE.DAT") .EQS. "") THEN -
$ CREATE/FDL=PHONE.DAT
```

Example 2. New PHONE build procedure (BUILD_PHONE.COM)

It is now possible to rebuild the application with the user interface in PHONE.EXE and the phone book file interface in DATABASE_RTL.EXE.

Building the Java layer

Once the RTL has been set up and shown to work correctly with the existing data file, the next step is to generate the Java definitions of the entry points and record structures. This is done using the Java back end for the SDL compiler.

SDL stands for Structure Definition Language. It is a language and compiler for taking language-independent record, constant, and entry-point definitions and generating language-specific include/header files.

Normally this header file would have to be built by hand. However, for PL/I developers there is a /SDL qualifier on the PL/I compiler that allows PL/I source modules to be translated to SDL. Example 3 demonstrates using the PL/I SDL generator and its output.

The newly generated SDL source file can then be processed by the Java back end to generate a Java module that defines the same information in a way that can be used by the J2VMS package. A severely trimmed example of the Java class generated from DATABASEDEF.SDL (shown in Example 2) can be seen in Example 4.

```
$ PLI/SDL=(MODULE=DATABASEDEF,OUTPUT=DATABASEDEF.SDL) DATABASE.PLI
$ TYPE DATABASEDEF.SDL
module DATABASEDEF;
entry "DELETE_A_RECORD" parameter(
 character length 65 reference
 ) returns boolean;
entry "WRITE_A_RECORD" parameter(
 pointer value
 );
entry "GET_A_DATE_RECORD" parameter(
 character length 5 reference,
 boolean value,
 boolean value
 ) returns pointer;
  ⋮
aggregate "ENTRY" union;
 "ENTRY_STRING" character length 266;
 "FIELDS" structure;
 "NAME" union;
 "FULL_NAME" character length 65;
 "PIECES" structure;
 "LAST" character length 32;
 "SPACE" character length 1;
 "FIRST" character length 32;
 end "PIECES";
 end "NAME";
 "ADDRESS1" character length 60;
 "ADDRESS2" character length 60;
 "CITY" character length 32;
 "STATE" character length 2;
  ⋮
end_module DATABASEDEF;
```

Example 3. Generating the SDL definitions from PL/I source (DATABASEDEF.SDL)

The two SDLJAVA_* logicals are required to give the Java back end information necessary for building the Java source file correctly. The SDLJAVA_PACKAGE logical details the Java package name giving the source module a place in the class hierarchy. SDLJAVA_LIBNAME defines the name of the run-time library (as required by LIB$FIND_IMAGE_SYMBOL) where J2VMS will locate the routines.

Incidentally, the steps up to now make it possible to use the SDL definitions with a host of other languages including C, Fortran, Ada, and BLISS—so not only can you expose your code to Java, but it is available in the common language environment.

```
$ DEFINE/USER SDLJAVA_PACKAGE "com.kednos.jphone"
$ DEFINE/USER SDLJAVA_LIBNAME "DATABASE_RTL"
$ SDL/ALPHA/LANGUAGE=JAVA DATABASEDEF.SDL/VMS_DEVELOPMENT
$ TYPE DATABASEDEF.JAVA
//*****************************************************************************
// Created: 15-Dec-2008 11:24:14 by OpenVMS SDL EV2-3
// Source: 05-DEC-2008 22:59:24 SYSPROG:[TSNEDDON.SCRATCH.SWING]DATABASEDEF.SDL
//*****************************************************************************
package com.kednos.jphone;
import vs.VMSparam;
import vs.SystemCall;
import vs.FieldDescriptor;
public class DATABASEDEF { // IDENT
private static SystemCall nullclass;
private static final String libname = "DATABASE_RTL";
private static SystemCall delete_a_record_return;
public static int delete_a_record(VMSparam[] args) {
 if (delete_a_record_return == nullclass) {
 delete_a_record_return = new SystemCall("DELETE_A_RECORD",libname);
 }
 return delete_a_record_return.call(args);
}
 ⋮
public static final int S_ENTRY = 266;
public static final FieldDescriptor entry_string = new FieldDescriptor(0,0,0,0);
public static final FieldDescriptor ENTRY_STRING = entry_string;
public static final int S_ENTRY_STRING = 266;
public static class _0 extends FieldDescriptor { // FIELDS
public _0() { super(0,0,0,0); }
 ⋮
public static final _0 fields = new _0();
public static final _0 FIELDS = fields;
public static final int S_FIELDS = 266;
}
```

Example 4. Generating Java class from SDL (DATABASEDEF.JAVA)

Wrapping it all up in a Java jacket

Now that the relevant modules have been shifted into a run-time library and the Java declarations have been generated, it is time to build the Java application. Figure 2 shows a screen shot of the Java equivalent of the data-entry screen shown in Figure 1. However, the focus of this article is the use of the J2VMS package and not the Swing toolkit, so the remainder of this article will focus on the interface to the phone book data file.

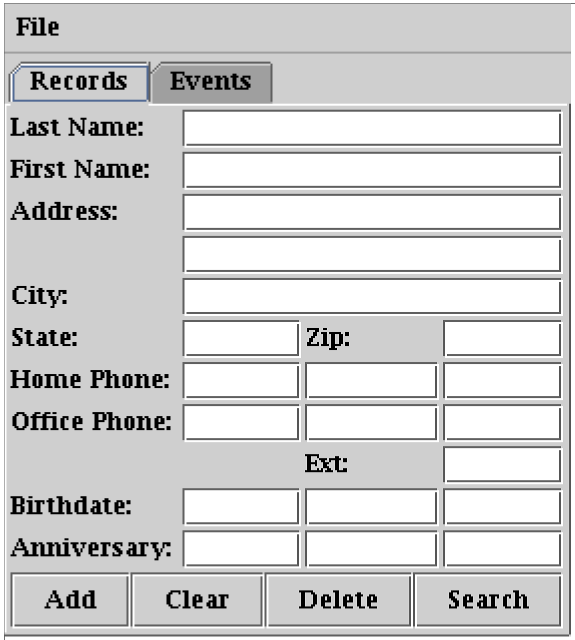


Figure 2. jPhone data entry screen

J2VMS is a Java package that provides a collection of classes that help Java feel more like a native OpenVMS language. It provides support for easily calling native routines and manipulating native data structures, and provides an interface to common OpenVMS argument-passing mechanisms.

To give the run-time library a Java "feel", it is hidden behind the class `com.kednos.jphone.Database`. Each of the native routines is given an equivalent method in the `Database` class. These are shown in Table 1. To make manipulating the phone book data record easier, the `Database` class extends the J2VMS class `vs.VmsStruct`. This means that the instance of `Database` becomes the record buffer also. Each of the methods operates on this buffer.

Java method	Native routine
Database	OPEN_PHONEBOOK
finalize	CLOSE_PHONEBOOK
getRecord	GET_A_RECORD
getMonthRecord	GET_A_MONTH_RECORD
getDateRecord	GET_A_DATE_RECORD
deleteRecord	DELETE_A_RECORD
writeRecord	WRITE_A_RECORD

Manipulating the record is facilitated via the inherited `get` and `put` methods from `vs.VmsStruct`. It would be possible to go one step further and create a set of get/set methods that manipulate each of the record fields. However, it is a reasonable amount of extra work for little (if any) gain. Example 5 demonstrates the field `ADDRESS1` being updated with the contents of the `address1` object (a text input field from the GUI). The object `db` is an instance of the `Database` class.

```
db.put(DATABASEDEF.FIELDS.ADDRESS1,
       DATABASEDEF.FIELDS.S_ADDRESS1, address1.getText(), ' ');
```

Example 5. Updating the ADDRESS1 field (Database.java)

To prevent problems with threading, the instance of the DATABASEDEF class (the SDL-generated "header" file) is declared static and used as a mutex. In Example 6, the code for the method `getRecord` is shown. It demonstrates the use of the `rtl` object as a mutex. The file access statements used in the native PL/I routines rely on the internal I/O buffers maintained by the PL/I run-time library. PL/I assumes that applications are single threaded, which conflicts with the Java environment. By using the mutex, the RTL will only be called by one thread at a time. To ensure that the active record is not lost, it is copied into the storage allocated by the `Database` class using the routine LIB$ LIB$MOVC3 before the mutex is released.

Conclusion

In conclusion, by using the PL/I SDL, SDL tools, and the J2VMS package, the task of writing a Java application that makes use of native routines and data structures has been greatly simplified. The work in creating a custom JNI layer has been completely avoided. Instead, all calls to native code and data manipulation are done from Java, making the application easier to understand and maintain. Using these tools allows more time to be spent on writing the new application, rather than working on the existing part that already works.

```
public class Database
      extends VmsStruct
{

// Own storage
//
private static boolean        isOpen = false;

private LibRoutines           lib = new LibRoutines();
private static DATABASEDEF    rtl = new DATABASEDEF();
⋮
public boolean getRecord(String name,
                         boolean first)
{
 int              pointer;
 boolean                result = false;

 /* The native run-time library we rely on for accessing the
 * phone book file is not thread-friendly and makes use of
 * PL/I internal file buffers. To prevent acess problems
 * we synchronize access to 'rtl'.
 */
 synchronized(rtl)
 {
      pointer = rtl.get_a_record(new VMSparam[] {
                         new ByRef(trunc(name,
                                     DATABASEDEF.FIELDS.S_NAME)),
                         new ByVal(first ? 1 : 0)
                         });

      if (pointer != 0)
      {
       /* To make sure we don't lose the record we've just
       * fetched (through a call from another instance) we
       * copy the record buffer into our own internal
       * storage.
       */
       lib.lib$movc3(new VMSparam[] {
                   new ByRef(DATABASEDEF.S_ENTRY),
                   new ByVal(pointer),
                   new ByRef(getTarget())
                   });

       result = true;
      }
 }

 return(result);
}
⋮
}
```

Example 6. Fetching a record (Database.java)

For more information

Tim Sneddon can be contacted via email at tsneddon@kednos.com.

For additional information, including the full source of the example application and kits for all software mentioned in this article, go to:

- www.kednos.com/kednos/Integration

For further information on creating shareable images, see the *HP OpenVMS Linker Utility Manual* at:

- http://h71000.www7.hp.com/doc/83final/4548/4548PRO.HTML

For more information regarding Java, see:

- http://h18012.www1.hp.com/java/alpha/
- www.kednos.com/kednos/Integration/Java

OpenVMS Technical Journal V12

LAN troubleshooting techniques

Ajo Jose Panoor, OpenVMS LAN Engineering

Anand Ramaswamy, OpenVMS LAN Engineering

Richard Stockdale, OpenVMS LAN Engineering

When setting up a LAN, there are a few techniques that can be useful for troubleshooting. This article will cover the following points:

1. LAN configuration—Know what LAN devices are on the system
2. LAN application configuration—Know what applications are using each device
3. LAN counters—Provide information about each LAN device
4. LAN buffer management—Know what buffers should be maintained to avoid errors
5. LAN trace—Know how to interpret the trace logs
6. LAN setup issues—Common setup issues in the customer environment

LAN configuration

First, it is important to determine what LAN devices are configured on the system. This can be done in a few different ways:

- **Looking at devices on the console**—On Alpha systems, the console command `SHOW CONFIG` displays the system configuration. For any devices that the console firmware does not recognize, the display simply lists the PCI Device ID, PCI Vendor ID, PCI Subsystem ID, and PCI Subvendor ID. You can then search for these IDs in SYS$SYSTEM:SYS$CONFIG.DAT to see if VMS could configure the device. On i64 systems, the EFI console displays bootable devices as the system runs through self-test. You can look at the devices in the Boot Configuration menu. The EFI utility `VMS_SHOW` on the VMS EFI partition displays most of the LAN devices as well.

- **Using LANCP on a live system**—LANCP is invoked by the command `$ RUN SYS$SYSTEM:LANCP`, via `$ MC LANCP`, by defining LANCP as a foreign command, or by defining DCL$PATH to include SYS$SYSTEM then issuing the command `$ LANCP`. The LANCP command `SHOW CONFIGURATION` lists the LAN devices configured on the system.

- **Using the System Dump Analyzer (SDA) on a live system**—SDA is invoked by the command `$ ANALYZE/SYSTEM` on a live system. The `CLUE CONFIGURATION/ADAPTER` command displays the I/O configuration. The SDA command `LAN CONFIGURATION` displays LAN device data similar to the LANCP `SHOW CONFIGURATION` command.

- **Analyzing a crash dump**—SDA is invoked by the command `$ ANALYZE/CRASH SYS$SYSTEM:SYSDUMP.DMP` for a crash dump. The `CLUE CONFIGURATION/ADAPTER` command displays the I/O configuration. The SDA command `LAN CONFIGURATION` displays LAN device data similar to the LANCP `SHOW CONFIGURATION` command.

LAN application configuration

The next step is to determine the characteristics of each application (user) running on each LAN device. This can be accomplished with these commands:

- **LANCP command**—`SHOW CONFIGURATION/USERS`:
- **SDA command**—`SHOW LAN`:
- **SDA command**—`SHOW LAN/DEVICE=EIA6` or `SHOW LAN/CLIENT=DECNET`:

LAN counters

SDA includes LAN-specific commands that provide information about each LAN device and LAN application. The following list summarizes some of the general SDA commands that are related to LANs:

- **`SHOW LAN/FULL`**—Displays detailed information about each LAN device
- **`SHOW LAN/ERROR`**—Displays LAN errors
- **`SHOW LAN/COUNTERS`**—Displays the LAN device and application counters
- **`SHOW LAN/WEIRD`**—Displays LAN anomalies
- **`SHOW LAN/QUEUES`**—Displays the LAN queue structures
- **`SHOW LAN/TIME`**—Displays the time of major LAN events
- **`SHOW LAN/VCI`**—Displays VCI structure information

Additional qualifiers limit the display to a medium type or specific device. These qualifiers are `/DEVICE=devname`, `/CLIENT=clientname`,`/UNIT=unitname`, and `/CSMACD`. The qualifier `/CSMACD` selects Ethernet devices only. The other three—`/DEVICE=devname`, `/CLIENT=clientname`,`/UNIT=unitname`—are synonymous. You can specify a client name with /DEVICE or /UNIT, and similarly with device and unit name. Client names include SCA, DECNET, LAT, MOPRC, IP, DIAG, ELN, BIOS, LAST, USER, ARP, RARP, MOPDL, LOOP, BRIDGE, DNAME, ENCRY, DTIME, LTM, AMDS, DECNETV, PATHWRK, IPV6.

The LAN-specific SDA extension image is SYS$LIBRARY:LAN$SDA.EXE. This image implements a set of LAN commands which mirror some of the display functions of LANCP, including configuration information, counters, and trace information. Additional displays expand on the data available by the SDA command `SHOW LAN`.

The LAN SDA commands are:

- **`LAN HELP`**—Displays help for the SDA LAN extension. LAN commands are implemented in the SDA extension LAN$SDA.EXE.
- **`LAN CONFIGURATION`**—Displays LAN device configuration information. Normally, it is similar to the LANCP SHOW CONFIGURATION command, but may vary from release to release.
- **`LAN COUNTERS`**—Displays LAN device counters, similar to the LANCP SHOW DEVICE/COUNTERS command.
- **`LAN INTERNAL_COUNTERS`**—Displays LAN driver internal counters, similar to the LANCP SHOW DEVICE/INTERNAL_COUNTERS/DEBUG/ZERO command.

- **`LAN QUEUES`**—Displays the request and pending queues maintained by the LAN driver and the LAN common routines for a device.
- **`LAN RINGS`**—Displays the ring structures maintained by a LAN driver for communicating transmit and receive information to the device.
- **`LAN TRACE`**—Displays LAN driver trace information collected for a LAN device.
- **`LAN TRACE/CONTEXT`**—Displays LAN driver trace context for a LAN device, consisting of the definition of the trace mask bits.
- **`LAN TRACE/REVERSE`**—Displays LAN driver trace information collected for a LAN device in reverse order, from latest to earliest.

An additional qualifier limits the display to a specific device: `/DEVICE=devname`, where `devname` is the name of a LAN device.

Note that not all device-specific commands are implemented for all devices, such as `LAN INTERNAL_ COUNTERS` and `LAN RINGS`.

LAN Control Program (LANCP)

The LAN Control Program (LANCP) is the main mechanism within OpenVMS for controlling and displaying information about LAN devices. LANCP is invoked by the command `$ RUN SYS$SYSTEM: LANCP`, via `$ MC LANCP`, by defining LANCP as a foreign command, or by defining DCL$PATH to include SYS$SYSTEM then issuing the command `$ LANCP`. Help is available within the program and in the *HP OpenVMS System Management Utilities Reference Manual*.

LAN buffer management

For most configurations, the LAN driver and LAN common routines maintain a reasonable number of buffers that minimize the possibility of lost packets.

During transmit, a relatively small number of buffers is sufficient to attain the performance capabilities of a LAN device. Each LAN driver sets up the transmit path according to the hardware specifications and arranges the number of transmit buffers and transmit segments without user input. The only way packets can be lost on transmit is if a transmit error occurs.

During receive, sufficient buffers must be available to the LAN device to support the incoming packet rate. The LAN device has some buffering, and the receive buffers that are given to the LAN device constitute the remaining buffering. The LAN driver attempts to filter and deliver receive packets to applications as fast as they are copied to receive buffers by the device. When the LAN driver doesn't keep up with the receive stream, buffering becomes exhausted and packets are discarded by the device. Packets are also discarded because the packet was corrupted or damaged, resulting in a CRC error.

When a receive packet is discarded, the reason is reflected in the LAN counters:

> **CRC error**—The packet was damaged during transmission, so the calculated CRC does not match the CRC in the packet. To address these errors, look at the network cable running from the LAN device to its termination point at the switch, hub, or other LAN device (for point-to-point links). Also be sure to verify the duplex setting, as a duplex mode mismatch can result in

CRC errors. A software bug in the LAN driver can also affect the transmit or receive operation of the LAN device, resulting in CRC errors.

Length error—The length of the packet doesn't match the expected packet length. It was either damaged in transmit, even though the CRC is good (some switches regenerate CRCs), or the application that constructed the packet included the wrong length. To address these errors, determine which application is generating the errors and fix it. LAN driver tracing can be used to trace received packets and locate the source of the errors.

Data overrun error or system buffer unavailable error—The LAN device exhausted its internal buffering and was unable to avoid losing an incoming packet. This can happen if device access to memory is slower than the incoming data rate or if the device has no available buffers for the receive data. A data overrun error is generally synonymous with a system buffer unavailable error. How these counters are maintained for a LAN device is driver- and device-specific. To address these errors, supply more receive buffers as described below. If the errors are the result of DMA performance issues, you may need to relocate the LAN device to another slot, perhaps on a PCI where the LAN device can get the bandwidth it needs. If the LAN device is a low-performance EISA or ISA NIC, you can replace it with a higher-performance adapter.

User buffer unavailable error—For QIO applications, such as DECnet Phase IV, the LAN driver discards receive packets if the application has not provided a buffer, or the number of buffers that the application is allowed to accumulate is exceeded. To address these errors, increase the receive buffer count for the application. For DECnet Phase IV, set the line receive buffers to a larger value. For this and other QIO applications, the LAN QIO parameter that is significant is NMA$C_PCLI_BFN—i.e., the number of user buffers, which has a maximum allowed value of 255. This value multiplied by the device buffer size is charged to the process BYTLM quota.

Supplying more receive buffers

To supply more receive buffers to a LAN device, you can adjust the minimum and maximum buffers using LANCP commands. An application can include these parameters in a `setmode set_mac` request, parameters `NMA$C_PCLI_MINRCV` and `NMA$C_PCLI_MAXRCV`. The permitted range for these parameters is 32 to 1536 buffers.

For example, for device EWC, let's set the minimum receive buffers to 512 and maximum receive buffers to 768. We must first define the values in the permanent device database so the values get set automatically when LANACP is started. Then the current values can be adjusted:

```
$ MC LANCP define device/min_buffers=512/max_buffers=768 ewc
$ MC LANCP set device/min_buffers=512/max_buffers=768 ewc
```

See the LAN chapter in the *HP OpenVMS I/O User's Reference Manual* for more details.

LAN trace facility

LAN driver trace support was introduced to the LAN drivers in V7.1-2 and V7.2. The trace support is intended for debug and analysis of LAN drivers and devices. Tracing is enabled and disabled via LANCP without needing to reboot the system. Tracing is not free—the trace buffers in the driver use non-paged pool (from 65 k to 32mb depending on the size of the trace buffer selected), and recording trace data costs some performance (the cost depends on what is being measured).

Trace facility usage

The trace facility is documented in the LANCP chapter of the *HP OpenVMS System Management Utilities Reference Manual.*

As discussed earlier, you can view the trace data using the SDA command `LAN TRACE`. On the running system, you can view the trace data using the LANCP command `SHOW DEVICE/TRACE`, which displays the trace context followed by the trace entries in order from earliest to latest.

Trace data display columns

The trace data display heading columns are as follows:

- **Sequence #**—The sequence numbers start at 1, increment to 4 billion, then go back to 0. If there is a gap in the sequence numbers being displayed, an asterisk follows the number. The number of gaps is given at the end of the trace display. There is only a gap when the trace data is being collected faster than it can be displayed.
- **EntryTimeStamp**—System uptime in seconds.
- **Time**—The system time of the event. This time matches the time values that are recorded by the LAN drivers and displayed by SDA. This time may be inconsistent across CPUs, since time differences between CPUs are not accounted for. You can use the CPU number displayed in the Misc column to understand the time differences.
- **Description**—Text description of the event.
- **XmtOut**—Number of transmits to the device outstanding at the time of the event. Since this is an 8-bit value, it is limited to 255.
- **RcvOut**—Number of receive buffers owned by the device at the time of the event. Since this is an 8-bit value, it is limited to 255.
- **Misc**—An additional byte of information.
- **AddlData**—3 additional longwords of information specific to the particular driver. The first is the packet size for transmit and receive entries; the second and third are part of the packet data that is displayed for the transmit and receive data entries.
- If the entry is a "transmit issued" or "receive done" event and the trace mask includes the bits for transmit data or receive data, the entry following the event is the packet that was transmitted or received, displayed in dump format, however much of it has been traced.

LAN setup issues

Ethernet, including Fast Ethernet, Gigabit Ethernet, and 10-Gigabit Ethernet, is the most common LAN technology used on Alpha and i64 systems. Early Alpha systems used Turbo Channel, FutureBus+, and XMI-based Ethernet Network Interface Cards (NICs). The next-generation Alpha systems introduced EISA, ISA, PCMCIA, and PCI NICs. The latest generation Alpha and current i64 systems support PCI and PCI-X based NICs. This section focuses on these NICs. NICs are also known as LAN adapters, LAN controllers, or LAN devices. Some LAN adapters are embedded on a system module and are called LAN on motherboard (LOM) implementations.

Different technologies and mediums have been used throughout the life of Ethernet. While these are fully documented in IEEE specs, the list in table 1 attempts to summarize the different types in order to clear up any confusion over terminology.

Table 1. Ethernet technology types

Technology	Medium	Speed	Duplex mode
10Base-5	AUI	10 Megabits/second	Half duplex
10Base-2	BNC	10 Megabits/second	Half duplex
10Base-T	UTP	10 Megabits/second	Full or half duplex
100Base-TX	UTP	100 Megabits/second	Full or half duplex
100Base-FX	Multimode fiber	100 Megabits/second	Full or half duplex
1000Base-T	UTP	1000 Megabits/second	Full or half duplex
1000Base-SX	Multimode fiber	1000 Megabits/second	Full or half duplex
10GBase-SR (10000Base-SX)	Multimode fiber	10 Gigabits/second	Full duplex

Auto-negotiation, parallel detection, and auto sense

The most common difficulty in setting up a LAN is selecting the correct speed and duplex mode. In an attempt to simplify this problem, the industry has introduced features like auto-negotiation (NWAY), parallel detection, and auto sense. These features are quite helpful when they are understood and used correctly.

Auto-negotiation is a handshake between two link partners (a link partner is one end of a point-to-point connection) using Fast Link Pulse (FLP) bursts to negotiate the highest common operating mode possible. If an auto-negotiating device determines that its link partner is not capable of auto-negotiation, it can utilize parallel detection. The parallel detection function is an auto-negotiating device's means to establish links with non-negotiating, fixed-speed devices. If the auto-negotiating device sends out FLPs but only receives Normal Link Pulses it knows it's connected to a 10 Megabit/second device. Conversely, if the auto-negotiation device sends out FLPs and receives FLPs but the FLP burst does not form a specific code word, then the link partner is a 100 Megabit/second device. Parallel detection does not determine the duplex mode; in general, it will automatically select half-duplex mode. Auto-negotiation and parallel detection are mainly performed by hardware but may be initiated and monitored by the LAN device driver. For Gigabit, the content of the FLP data has been extended to allow for 1000Base-T half and full duplex.

Auto sense is a method used by LAN devices that do not support auto-negotiation to determine the speed of the link partner. Auto sense is also used to select between AUI, BNC, and twisted pair modes for those LAN devices that support all three. While auto-sensing the speed, the LAN device is configured to run at 10 Megabits/second half duplex, and the LAN driver checks for a valid link. If a valid link is not detected, the driver sets the LAN device to 100 Megabits/second half duplex. Auto sense cannot determine the duplex mode, so the LAN driver normally selects half duplex. The same technique is used to determine the medium type.

Setting the Ethernet operating mode

A system manager can select the operating mode of the NIC in two ways. On Alpha systems, a console environment variable can be set at the SRM console. For example, a DE500-BA is identified by the console as an EW device with the console environment variable EWx0_MODE, where x is the controller letter number. When the LAN driver initializes the LAN device, it uses the console setting to set up the device. On Alpha systems and i64 systems, the device characteristics can be set using LANCP.

Some users understand the different operating modes and how they are set but are sometimes unclear as to which device supports which mode. Most Alpha SRM consoles don't inform the user that they have set the mode incorrectly. The LAN drivers do not inform the user of a bad setting, but they will usually choose a default setting if an incorrect setting was selected. One method of determining how the device has been set is to use the LANCP command `SHOW DEV EWA/CHARACTERISTICS`.

Ethernet setup problems

Users who understand the different modes of operation and also which NIC supports what mode can still run into setup problems. Most setup problems occur because the user has set the NIC operating mode incorrectly for the type of device it is attached to. Nearly all users connect their NIC to a hub or a switch and, less often, they connect one NIC to another NIC. It's important to understand that a hub, or repeater, is most often a fixed-speed, half-duplex device with all ports existing in the same collision domain. A switch, or bridge, is normally a dual-speed device supporting full or half duplex, auto-negotiation, and parallel detection. Most switches are manageable and the user can often disable these features. Each port of a switch represents a separate collision domain. A switch will only forward frames to a specific port, whereas a hub will replicate the frame on all ports. Multicast and Broadcast frames will of course be forwarded to all ports of a switch, subject to any filtering imposed on the switch by a network manager.

The following recommendations will help avoid setup problems with switches and hubs:

- When connecting a NIC that supports auto-negotiation to a switch that supports auto-negotiation, ensure that auto-negotiation is enabled for both the NIC and the switch.
- When connecting a NIC that does not support auto-negotiation to a switch that does support auto-negotiation, ensure that the NIC is set to half-duplex mode.
- When connecting a NIC to a 100Base-TX hub, ensure that the NIC is set to 100Base-TX half duplex.
- When connecting a NIC to a 10Base-T hub, ensure that the NIC is set to 10Base-T half duplex.
- When connecting a NIC to another NIC, ensure that both NICs are set to the same mode.
- When using a 100Base-FX-based NIC, ensure that both link partners have the same duplex mode set.

If these recommendations are not followed, problems will most likely result—the most common problem being the wrong duplex mode has been selected. When the wrong duplex mode is selected, it's possible that the link partners will continue to exchange frames, though the majority of them will be lost. This will force the application to handle lost frames, resulting in poor application performance. Examining the LAN counters will most often reveal this issue. For more information, refer to the section "LAN counters".

The following situations can lead to the wrong duplex mode being selected, as well as other problems:

- **A NIC is set to 100Base-TX full duplex and is attached to a switch that is auto-negotiating**—In this case, the device driver for the NIC has disabled auto-negotiation because it was told to do so when the user selected 100Base-TX full-duplex mode. The switch will attempt to negotiate but will fail since the NIC isn't participating. The switch will use parallel detection. The switch will most likely select the correct speed but will select half-duplex mode even though the NIC is running full-duplex mode.

- **A NIC is set to auto-negotiate and is connected to a hub**—In this case, the behavior is different for different families of NICs. For example, the Tulip/DE500 family does not perform parallel detection because of hardware limitations. Generally, the LAN device driver will attempt to auto-negotiate endlessly and the link will never come up. This situation can also be detected from the LAN counters. The number of auto-negotiation resets will increment at about once every 3 seconds and carrier check failures will also increment. (The Intel® 82559/DE600 family does perform parallel detection when auto-negotiation fails and will select the correct speed and duplex mode. This means that when using the Intel 82559/DE600 NICs the above suggestions for hubs don't necessarily have to be followed.)

- **A NIC is set to a speed that is different from the speed of the hub it is attached to**—This is not a good situation. In one case, a NIC set to 100Base-TX half duplex was connected to a 10Base-T hub. For some hubs, this would bring down every network connection within the collision domain. The reverse situation, a NIC set to 100Base-TX half duplex connected to a 100Base-TX hub, is less catastrophic, though the link will mostly likely remain down. In either case, the number of carrier check failures in the LAN counters will be incrementing. The LAN driver may also log transmit timeouts.

For Alpha systems, there are problems that can occur before VMS is actually involved, mainly at the SRM console level (>>>). These problems won't affect the LAN counters and the user will have to take care to observe the messages at the console.

- **A NIC is set to auto-negotiate but auto-negotiation fails**—This can happen for several reasons. Some are described above, but the real problem in this case is that the console LAN driver may default to 100Base-TX full duplex, causing the same catastrophic problems mentioned above.

- **The console environment variable for the NIC operating mode may change**—This has happened on some platforms after performing an SRM console firmware upgrade. It usually sets the mode to Twist (10 megabits half-duplex).

For i64 systems, the console setting is assumed to be auto-negotiation. The EFI firmware does auto-negotiation or parallel detection, and the VMS boot driver and runtime driver are expected to attempt to configure the device using auto-negotiation and parallel detection.

For more information

For more information, please contact the authors:
ajo-jose.panoor@hp.com
anand.r@hp.com
richard.stockdale@hp.com

OpenVMS Technical Journal V12

Installing and configuring a wiki engine on OpenVMS

Rishi Singhal, OpenVMS Debugger team

This article provides a brief introduction to wikis, the advantages of a wiki for knowledge management, and a step-by-step approach to configuring a wiki engine (such as PmWiki) on an i64 system.

Introduction

As Wikipedia explains (http://en.wikipedia.org/wiki/Wiki), a wiki is a collection of web pages designed to enable anyone who accesses it to contribute or modify content, using a simplified markup language.

"Content management systems will always have their place in the publishing world, but they've never been the best tools for business collaboration. A simple open-source app called the wiki may soon rule the knowledge management roost." – Ezra Goodnoe, InternetWeek 2005

A Wiki provides the following advantages:

1. A rich text editor, and the ability to add files and images
2. Content search
3. A collaborative environment for document sharing
4. Page revision history is available for audit and CM
5. Access from anywhere with a web connection

Experimenting with wikis on OpenVMS

Almost a year ago I started experimenting with different wikis on OpenVMS, such as MediaWiki, TikiWiki, DokuWiki, etc. There were issues with all of them. For example, MediaWiki had a prerequisite for a PHP version that was not available on OpenVMS.

Still searching for a lightweight wiki, I came across PmWiki, which is free software under the GNU GPL, uses flat file (http://www.pmwiki.org/wiki/PmWiki/FlatFileAdvantages), and is very easy to install and configure.

Installing and configuring PmWiki

Requirements

- OpenVMS v8.3 running on an i64 system
- CSWS_PHP version 1.3 based on PHP 4.3.10
- CSWS v2.1-1 based on Apache

Installation

- Download pmwiki-2.2.0-beta65.zip from:
  http://www.pmwiki.org/wiki/PmWiki/Download
- Copy the kit to SYS$COMMON:[APACHE] (assuming ODS-5 system disk; can be copied to any other ODS-5 disk)
- Unzip the kit:
  $ unzip pmwiki-2^.2^.0-beta65.zip

Configuring

1. Modify SYS$COMMON:[APACHE.CONF]MOD_PHP.CONF to reflect access path for PmWiki. Add below line (after the line containing - Alias /php/ "/apache$root/php/scripts/")

```
  Alias /wiki/ "/apache$common/pmwiki-2_2_0-beta65/"
```

2. Restart apache and open a web browser pointing to the pmwiki.php script on the server:

   http://node.domain.com/wiki/pmwiki.php

3. PmWiki will then analyze your system configuration and provide instructions (if needed) for creating the wiki.d/ directory, which will be used to hold the pages created for your site.
4. The following directories will be created:

   Directory SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65]

   wiki_d.DIR;1   wikilib_d.DIR;1

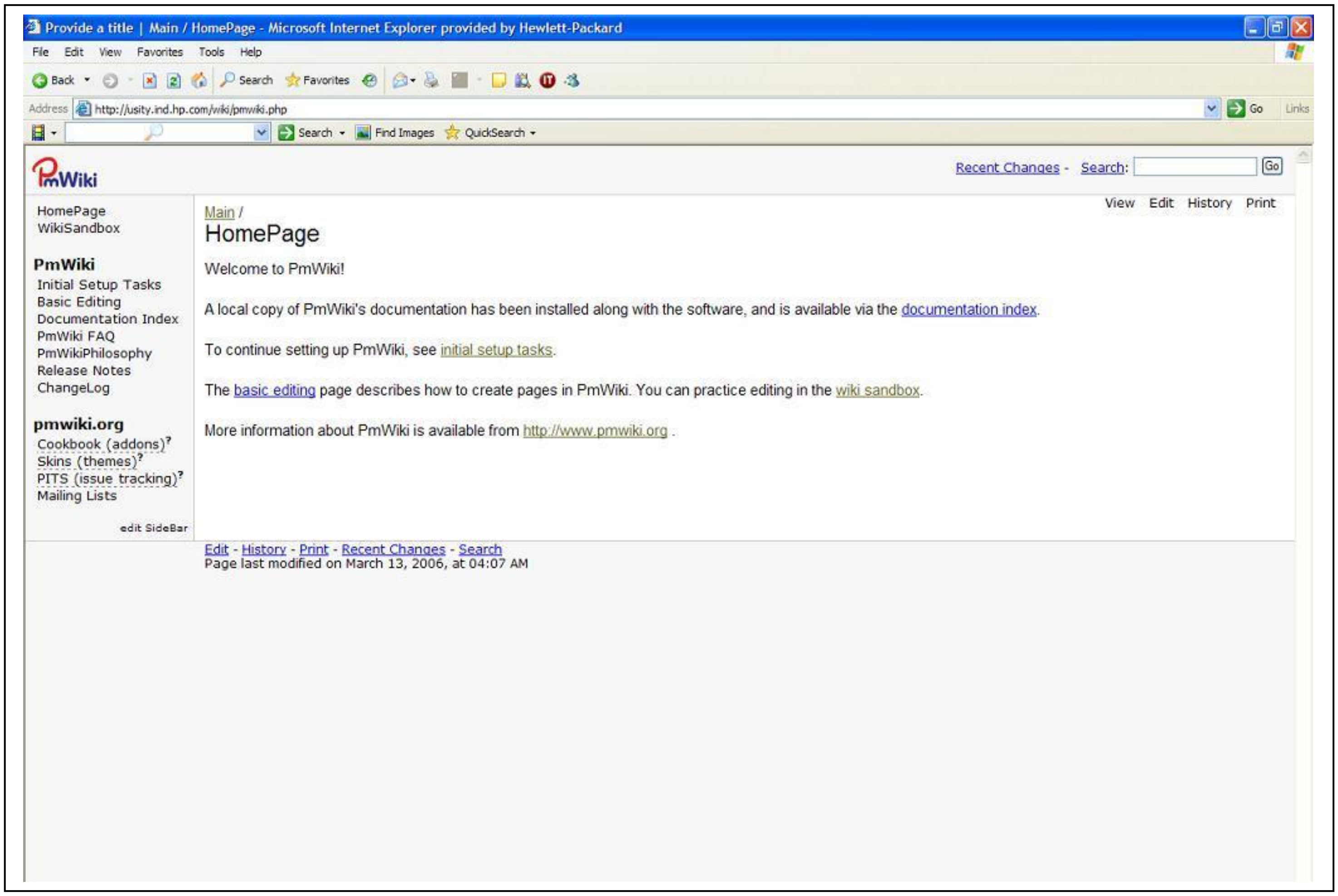
5. The names of these directories are expected to be wiki.d.dir & wikilib.d.dir, so rename them.
6. When you first install PmWiki, the SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php file does not exist. Copy the sample-config.php file (present in SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.docs] directory) to SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php and use it as a starting point.

```
$copy SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.docs]sample-
config.php SYS$COMMON:[APACHE.pmwiki-2_2_0-
beta65.local]config.php
```

7. Modify (uncomment or add) SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php. For example, to add Logo, Title, Password, and Enabling upload uncomment the lines as below:

```
#Provide a title to your wiki

$WikiTitle = "Provide a title";

#If you want to have your own logo copy it to the

#SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.pub.skins.pmwiki]
#directory

$PageLogoUrl = "$PubDirUrl/skins/pmwiki/urlogo.png";

#Provide  password for the administrator

$DefaultPasswords['admin'] = crypt('onesecret');

# Enable upload

$EnableUpload = 1;
```

Your wiki is ready to go, so refresh your http://node.domain.com/wiki/pmwiki.php page. You should see a page similar to this:



Main /

HomePage

Welcome to PmWiki!

A local copy of PmWiki's documentation has been installed along with the software, and is available via the documentation index.

To continue setting up PmWiki, see initial setup tasks.

The basic editing page describes how to create pages in PmWiki. You can practice editing in the wiki sandbox.

More information about PmWiki is available from http://www.pmwiki.org .

Edit - History - Print - Recent Changes - Search
Page last modified on March 13, 2006, at 04:07 AM

Tips and tricks

- While configuring PmWiki, if this is encountered:

  Fatal error: Call to undefined function: preg_match() in /apache$common/pmwiki-2_2_0-beta65/pmwiki.php on line 43

```
Two PHP extensions present in SYS$COMMON:[APACHE.PHP]PHP.INI have to be uncommented

extension=php_pcre.exe

extension=php_session.exe
```

- To restrict upload by the privileged few, modify SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php as

```
  #Set password for upload

 $DefaultPasswords['upload'] = crypt('secrettwo');
```

- If you want to track the modifications history, modify SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php as

```
  #While posting if you want the author name to be specified
  #compulsorily

  $EnablePostAuthorRequired = 1;
```

- To restrict editing by the privileged few, modify SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php as:

```
  #Not everyone will be allowed to edit. All the pages are
  read only otherwise

  $DefaultPasswords['edit'] = crypt('secretthree');
```

- To provide links to sharepoint (Microsoft® Windows®) use this format:

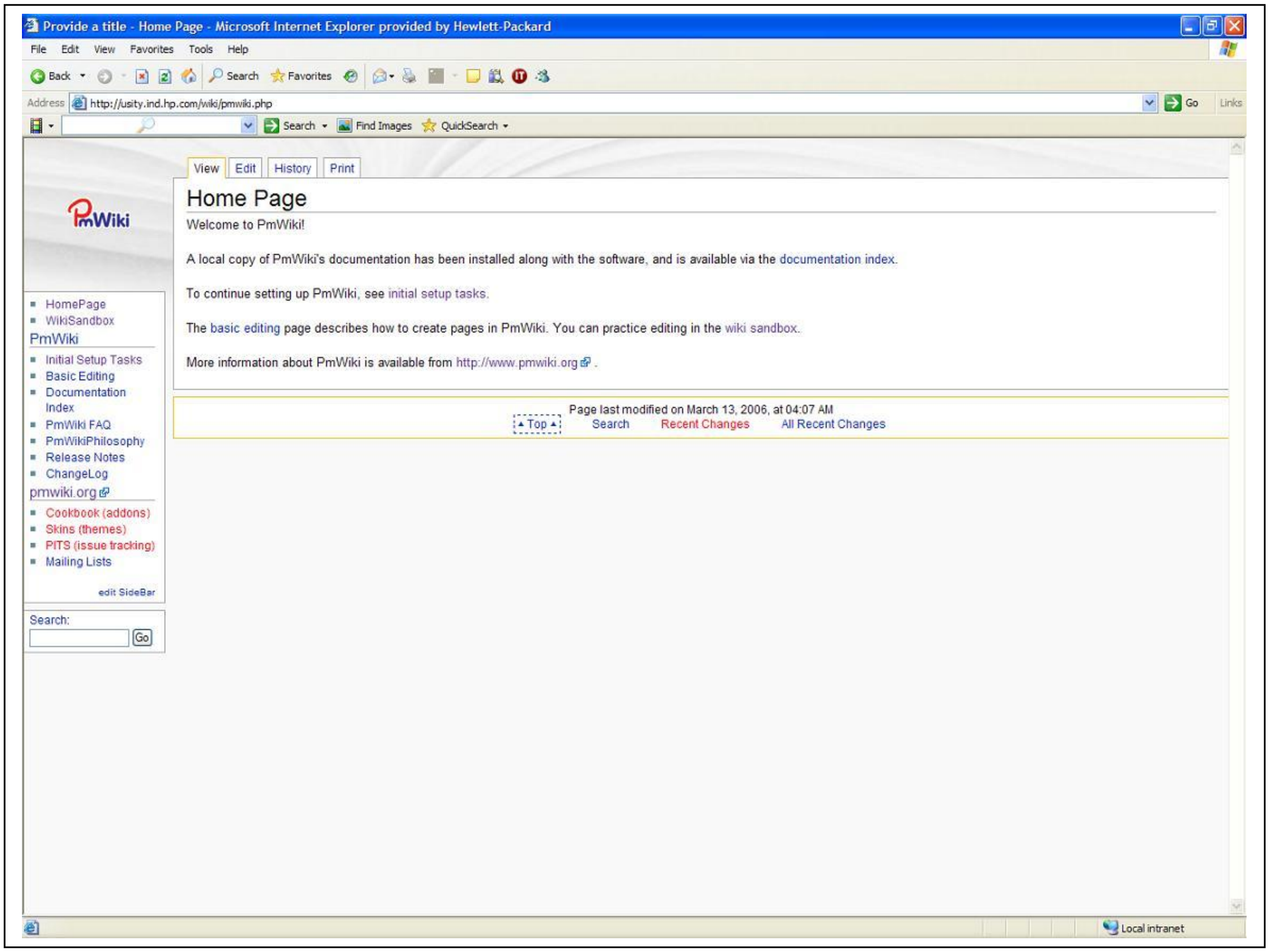
  %newwin%[[file:///&#92;\share\file.txt]]

- To use the same formatting as you have typed in the wiki editor, put your contents as: [@ formatted contents @]

- To use a different skin (goto http://www.pmwiki.org/wiki/Cookbook/Skins), such as monobook (makes PmWiki look like MediaWiki/Wikipedia):

  Download monobook.zip, extract it into SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.pub.skins] and add the following code to your local configuration file (SYS$COMMON:[APACHE.pmwiki-2_2_0-beta65.local]config.php):

```
$Skin = 'monobook';
```



Home Page

Welcome to PmWiki!

A local copy of PmWiki's documentation has been installed along with the software, and is available via the documentation index.

To continue setting up PmWiki, see initial setup tasks.

The basic editing page describes how to create pages in PmWiki. You can practice editing in the wiki sandbox.

More information about PmWiki is available from http://www.pmwiki.org .

Page last modified on March 13, 2006, at 04:07 AM

For more information

Contact the author at rishi.singhal@hp.com.

For more information about PmWiki, go to: www.pmwiki.org/wiki/PmWiki/PmWiki.