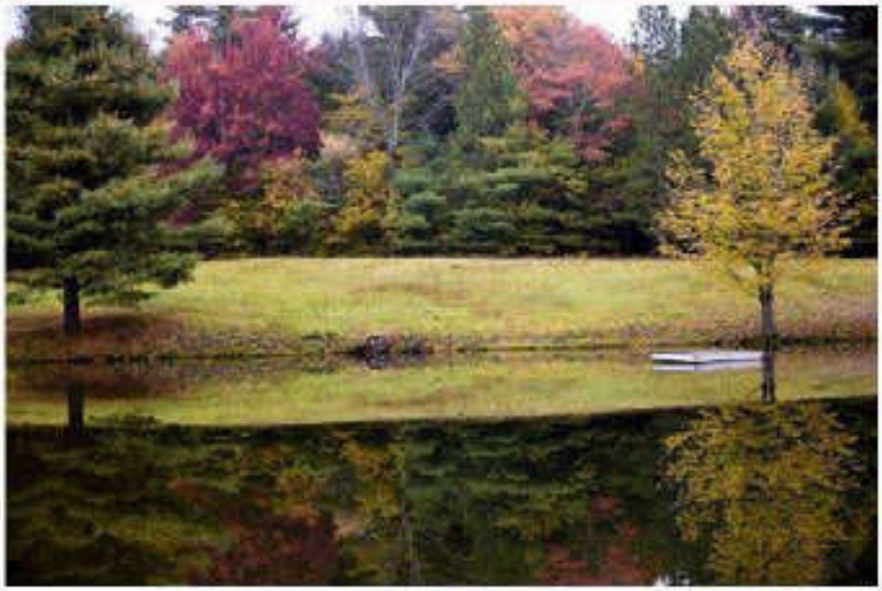
# OpenVMS Technical Journal

V14, March 2010



# Table of Contents

OpenVMS Technical Journal V14

# The Oracle Rdb Run-Time Code Generator for the OpenVMS Itanium Platform

Norman Lastovica, Senior Managing Engineer, Oracle Corporation

## Introduction

The Oracle Rdb database engine generates platform-specific executable code subroutines at run time. On VAX systems, VAX executable instructions are generated. On Alpha systems, Alpha executable instructions are generated. When Oracle Rdb was ported to the HP Integrity platform, the ability to execute run-time created subroutines was required as well. This paper discusses background of the original interpretation implementation with a later transaction to native Itanium instruction generation.

## Code Generation

When a user's request (such as the SQL statement "SELECT * FROM CUSTOMERS WHERE CITY = 'ESPOO' OR CITY = 'SALIDA' ORDER BY LAST_NAME") is passed to the database engine, a number of executable subroutines are created, at run-time, to perform various request-specific functions. These functions may include copying data fields, performing null-field handling, doing data field comparisons, and so on. This run-time request-specific code is an integral part of Oracle Rdb's database engine and helps to provide high levels of performance.

On VAX systems, such subroutines would contain VAX instructions (such as MOVC3, RET, MOVL and so on). When Oracle Rdb was ported to the OpenVMS and Tru64 environments for the Alpha platform, the code generation capabilities were extended to create Alpha instructions (such as CMOV, LDA, STQ, and so on). The logic of the subroutines, for the most part, is the same between the platforms; just the executable instructions, the register usage, and the system calling standard are different.

The following example contains a sequence of run-time generated instructions on a VAX system. Note the CISC architecture of the VAX computer with use of complex instructions that contain multiple operands along several addressing modes along with relatively high code density.

```
00FB1B18:   .WORD   ^M<R2,R3,R4,R5,R6,R7>
00FB1B1A:   MOVL    B^04(AP), R6
00FB1B1E:   MOVAB   @#00FB1B68, R7
00FB1B25:   MOVC5   S^#00, (SP), (SP), S^#05, B^01(R7)
00FB1B2C:   MOVL    B^69(R6), R0
00FB1B30:   ROTL    S^#08, R0, R0
00FB1B34:   BICL3   #FF00FF00, R0, R1
00FB1B3C:   XORL2   R1, R0
00FB1B3F:   ROTL    S^#10, R0, R0
00FB1B43:   XORB2   #80, R0
00FB1B47:   XORL3   R1, R0, B^02(R7)
00FB1B4C:   CLRB    B^01(R7)
00FB1B4F:   BBC     S^#02, W^00EF(R6), 00FB1B5C
00FB1B55:   XORB2   S^#01, B^01(R7)
00FB1B59:   CLRL    B^02(R7)
00FB1B5C:   RET
```

The next example contains a sample of run-time generated code on an Alpha system. Note that the Alpha is a more traditional "RISC"-style architecture where instructions are simpler, fixed size, and that the memory reference instructions either read or write memory, but do not atomically read and update memory in a single instruction.

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
01A060AC  A00C8138      LDL             R0,#XF8138(R12)
01A060B0  A02C808C      LDL             R1,#XF808C(R12)
01A060B4  44010400      BIS             R0,R1,R0
01A060B8  F4000004      BNE             R0,#X0000004 -> 1A060CC
01A060BC  A00C8064      LDL             R0,#XF8064(R12)
01A060C0  A02C813C      LDL             R1,#XF813C(R12)
01A060C4  400105A0      CMPEQ           R0,R1,R0
01A060C8  F400000D      BNE             R0,#X000000D -> 1A06100
01A060CC  A00C8110      LDL             R0,#XF8110(R12)
01A060D0  A02C8090      LDL             R1,#XF8090(R12)
01A060D4  44010400      BIS             R0,R1,R0
01A060D8  F400000F      BNE             R0,#X000000F -> 1A06118
01A060DC  A36B0144      LDL             R27,#X00144(R11)
01A060E0  220C8118      LDA             R16,#XF8118(R12)
01A060E4  223F001F      LDA             R17,#X0001F(R31)
01A060E8  224C8068      LDA             R18,#XF8068(R12)
01A060EC  233F0003      LDA             R25,#X00003(R31)
01A060F0  A75B0050      LDQ             R26,#X00050(R27)
01A060F4  A77B0058      LDQ             R27,#X00058(R27)
01A060F8  6B5A4000      JSR             R26,(R26)       "ots$strcmp_eqls"
01A060FC  E4000006      BEQ             R0,#X0000006 -> 1A06118
01A06100  201F0001      LDA             R0,#X00001(R31)
01A06104  A75E0008      LDQ             R26,#X00008(SP)
01A06108  A7BE0010      LDQ             FP,#X00010(SP)
01A0610C  23DE0020      LDA             SP,#X00020(SP)
01A06110  6BFA8001      RET             R31,(R26),1
01A06114  00000000      CALL_PAL        HALT
01A06118  47FF0400      CLR             R0
01A0611C  C3FFFFF9      BR              #XFFFFFF9 -> 1A06104
```

## Interpretation engine

Oracle Rdb was ported to run on the Microsoft Windows NT environment running on Intel x86 and Alpha processors (this product was, however, never released for production use). At that point in time, in order to rapidly complete the porting effort for the Intel x86 platform, an interpretation engine was created that could interpret those portions of the Alpha instruction set generated at run-time by Oracle Rdb. This approach allowed a single piece of code (the interpretation engine) to be written and, more importantly, debugged without having to change the instruction generation machinery within the Oracle Rdb database engine (which continued to generate subroutines using the Alpha instruction set).

Over time, an expanded set of “rich” instructions were added to the code generation capabilities on the Intel x86 platform. These instructions were intended to perform more complex actions as one “pseudo” instruction, replacing, in some cases, a large number of Alpha instructions in the code stream. Execution of these “rich” instructions could be more optimized as compared to individually executing long sequences of individual instructions. Approximately 150 of these “rich” instructions were eventually implemented.

Though not used on Alpha and VAX systems (the supported platforms for Oracle Rdb), this interpretation engine remained part of the Oracle Rdb source code and lay “dormant” for many years.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Itanium Emerges

With the advent of OpenVMS for the Integrity Server platform, Oracle chose to port the Rdb database engine to the Integrity Server for the OpenVMS operating system. Though native language compilers were available (primarily, in the case of Oracle Rdb, BLISS, C++, C and MACRO32), there was no immediate capability for the Oracle Rdb engine to create executable instructions for the Itanium architecture.
At this point, the interpretation engine was pressed in to service again. Most of the code had not even been compiled in over 10 years. But with a bit of effort (mostly correcting issues related to improved C compilers with enhanced detection for latent bugs), it was able to successfully execute once again. Debugging effort was required to get it working completely properly but it did prove to be a valuable tool that allowed a significantly more rapid production delivery and deployment of Oracle Rdb Release 7.2 on the Itanium platform.

## Performance

Overall, we anticipated that, while the performance of the interpreted code would never be as good as a native executable code subroutine, the Integrity system, as a whole, would perform at least comparable to "equal" Alpha systems. This was the case for the vast majority of applications and systems that we analyzed. CPU, memory and IO performance tended to provide a balanced system that performs very well when running customer applications. A few applications, however, spent a significant amount of time executing the run-time generated code and these applications were, in some cases, significantly slower than we, and our customers, would have preferred.
In particular, one major customer application was generally as good or better performing on Itanium systems than on Alpha systems. But several significant queries of the application were both frequently executed and much slower once migrated from Alpha to Itanium. Analysis revealed that most of the additional CPU time was spent in the interpretation engine while running particular parts of the application.

A major effort was spent in analysis and tuning of the interpretation engine itself. This tuning yielded performance improvements of over 20% in some cases. This was, however, not nearly enough (and, regrettably, not even in the same order of magnitude required). Further analysis indicated that there was likely no way to make the interpretation model execute fast enough to meet our customers' needs in all cases.

## A New Direction

It was felt that the investment required to enhance the Oracle Rdb database engine to add another set of code generation capabilities (in addition to VAX, Alpha and "rich" instructions) for native Itanium would consume significant resources for development and testing and likely could not be completed in time for this particular customer's production deployment schedule. There were too many locations in the code that would be required to be changed to produce instructions for yet another architecture. Our experience with the port to Alpha indicated that there would be substantial human resources required to produce and debug the resultant code.

© Copyright 2010 Hewlett-Packard Development Company, L.P

Based on this analysis, the concept of "compiling", at run time, a complete subroutine from a mixture of Alpha and "rich" instructions in to native Itanium executable code was born. The design that we arrived at is not dissimilar to the JAVA machinery's "Just In Time" (aka JIT) compiler available on many platforms: Input a stream of generic and platform-independent instructions and create platform-specific executable code which is expected to perform much better than interpreting the "pseudo" instructions.

Initial prototypes were developed to create and call an executable stream of Itanium instructions. The success of these tests supported the idea that it was viable for Oracle Rdb to be able to create native subroutines and to call such code at run time while on the Itanium platform.

## High Level Design

The basic operation for what we originally called the "interp compiler" (based on the idea that this was a compiler to replace the interpretation engine) was to pass a pointer to a complete subroutine of compiled Alpha and "rich" code and then attempt to compile it completely in to a native Itanium instruction subroutine. If the compilation was successful (all instructions were able to be compiled) then a pointer to a procedure descriptor for the generated routine was returned with the low bit set (i.e., an odd value). If, however, the compilation could not be complete (if instructions were found that were not able to be compiled), the original routine address was returned (with the low bit clear as the routine had been originally allocated on a longword boundary).
Later, when the subroutine was to be called, the low bit of the routine's address was first evaluated. If clear, the existing interpretation engine was called to execute the subroutine. If the routine address was odd (indicating that the low bit was set), the routine was called directly (after clearing the low bit) to be executed "native".
In this way, the "interp compiler" could start small (only able to compile a few instruction types) and then grow (by adding the ability to compile more and more instructions and addressing modes and so on) all while the database engine continued to operate correctly (presumably as more and more subroutines could be compiled, execution performance would continue to improve). This made it possible to continue to execute and test Oracle Rdb while the "interp compiler" was being actively developed. Without this model it was have been a much slower process in that the "interp compiler" would have had to be entirely complete before we could even begin to test it.

## Itanium Architecture

Significant attributes of the Itanium architecture that pertain to the "interp compiler" include:

- 128 bit "bundle" containing 3 instructions that will all be executed
- Multiple execution unit type combination selected via a "template" within a bundle
- Predicate registers that control if an instruction will have an effect or not

© Copyright 2010 Hewlett-Packard Development Company, L.P

Producing code for the Itanium architecture is a fair measure more complex than, for example, code generation on the Alpha architecture. A significant set of rules and requirements are imposed in terms of which instruction type may be used in which bundle slot depending on the specified template, the use of "stops" to indicate that the results of prior instructions are required by following instructions, and so on. A larger number of registers provides major benefits in regards to having more scratch registers available for intermediate results. And the use of predicates can, in some cases, drastically reduce the number of branches taken which can, in turn, improve performance by reducing "wasted" processor cycles due to "bubbles".

Additional steps were required after creating code. Because the Itanium instruction cache (I cache) and data cache (D cache) are not synchronized, after new executable code is created, the hardware must be notified by flushing the instruction cache for the memory addresses of the newly created code. This was accomplished most easily by calling the SYS$PAL_IMB system service specifying the starting address and the length of the generated code. The system service invalidates each I cache line as needed and ensures that the data and instruction caches are correctly synchronized prior to attempting to execute the new instructions.

## Starting Simple

The Oracle Rdb "interp compiler" is implemented as a routine written in BLISS (the primary implementation language utilized by Oracle Rdb for both ease of development and resultant product performance). Generation of instructions is accomplished though a set of macros that implement primitive operations that are generally produced as one or more instructions within one or more bundles. The original project goal was to have macros that would create one instruction per bundle. Over time, more and complex macros were created to perform different functions and to create bundles with more instruction sequences to help produce faster and denser code streams.

A simple macro might, for example, produce a single ld4 instruction (to fetch 4 bytes from memory) alone in a single bundle (nop instructions would occupy the remaining two slots). Another level of complexity might be a single macro to create a pair of memory load and store instructions in a bundle. A more complex macro may implement a call sequence where output parameters are created, registers are saved, a procedure descriptor read, the routine called, and then after the call registers are restored. This sequence would require a modest number of bundles to implement.

## Branches

One area of complexity is forward and backward branches within the code. The input subroutine may contain both "rich" and Alpha instructions that change the flow of control via conditional and unconditional branches. Branches are self-relative within the context of the input routine. To preserve the correct branch destination, a table is constructed that contains the address of the bundle containing the branch along with the original branch offset. Another table is maintained to associate the original instruction location along

© Copyright 2010 Hewlett-Packard Development Company, L.P

with the location of the generated code. After code generation is complete, branches are "fixed up" to adjust the destination offset to the correct destination bundle.

For performance, the Itanium architecture includes "hints" for most types of branch instructions. These hints allow a compiler to indicate additional information to the hardware in regards to how branches are expected to execute. The hardware, in turn, may use this information to predict how the flow of control is expected to operate and can allocate resources more efficiently and, ultimately, execute the whole of the code stream faster. For example, one such branch "hint" type would be "dynamic, predicted not taken". This hint implies that the compiler expects that the conditional branch will generally not be taken but the hardware should allocate prediction resources (such a history of branches taken or not taken at this location).

Based on both performance testing and research papers evaluated, the "interp compiler" utilizes these branch hints in the generated instruction stream. Unconditional branches are specified as "static, taken", most conditional branches are specified as "dynamic, predicted not taken". Exceptions to these rules are backward (typically involved in a loop) branches which likely are specified as "dynamic predicted taken".

## Exception Handling

In order to allow OpenVMS exception handling mechanisms to function properly, the "interp compiler" must "register" each generated routine with the operating system. This registration includes identifying any "unwind" information specifically regarding the routine's first and last instruction, the length of the routine's prologue and registers and stack usage. Because the created code will be both created and executed in the processor's executive mode, a kernel mode image exit handler is utilized to un-register the generated code during image run down. Without having such unwind information registered with the operating system, exception handling is not possible; otherwise an exception from the generated code, or code that is called by the generated code, cannot be handled and results in, depending on the mode and context, either process or image termination.

The OpenVMS calling standard uses a variant of the common Itanium standard which includes a moderately complex set of rules for representing unwind information. This scheme includes a compressed variable-length fields and a dense structure. Within the interp compiler, this "signature" information is produced at the end of executable code creation for each routine.

## Simple Code Sequence Examples

The following code sequence shows the original "rich" instruction (indicated by longword address and content fields at the left part of the line) CLR_Q (clear quadword) along with its single operand followed by the generated Itanium instructions (indicated by quadword content and instruction addresses) created for the "rich" instruction.
The operation's addressing mode is evaluated as an offset from the global register r2 (this register maps to Alpha register R12 within Oracle Rdb). The offset is created by adding

© Copyright 2010 Hewlett-Packard Development Company, L.P

8000 to the value 0040 and then sign extending from 16 to 64 bits. Next, r0 (which is always read as the value zero) is written to the destination address, thus clearing it.
It would obviously be possible to combine these two instructions in to a single bundle. However, the construction of the interp compiler is such that the addressing steps are evaluated first and then the operation steps are produced. While it would be viable to perform a second pass to combine the instructions in to a single bundle, it has not yet been a high priority for execution optimization.
An additional concern for such optimization is that significant amounts of time could be spent in the interp compiler that could exceed the potential benefits for performance improvements of the generated code. In this case, for example, the stall caused by the memory reference will dramatically overshadow any other optimizations possible for the two instructions which still require a stop between them (as the first updates r14 which is used as input to the second).

```
s037E54B4  04000157      CLR_Q
037E54B8  D0000040                dst    quad*
                                { .mfi
013807E80380  0000000080366190        add         r14 = 3F8040, r2
000008000000  0000000080366191        nop.f       000000
000008000000  0000000080366192        nop.i       000000 ;; }
                                { .mfi
008CC0E00000  00000000803661A0        st8         [r14] = r0
000008000000  00000000803661A1        nop.f       000000
000008000000  00000000803661A2        nop.i       000000 ;; }
```

In the next example, the MOV_Q (move quadword) “rich” instruction has two operands (source and destination address information). The source location is indicated as an offset (00000050) from a register (Alpha register R16 which is translated as Itanium register r32; the first input parameter to the routine as specified in the OpenVMS calling standard). The destination is an offset (0040) from register r2 (strictly, the offset is FFFF8040 from register R30). The interp compiler detects that both source and destination addresses are likely to be at least quadword aligned and produces a single ld8 instruction to read the source quadword and a single st8 instruction to write to the destination.

```
037E54C4  04000145      MOV_Q
037E54C8  00000010                src    char*   (R16)
037E54CC  00000050                offset long
037E54D0  D0000040                dst    quad*
                                { .mfi
0108020A0380  00000000803661C0        add         r14 = 0050, r32
000008000000  00000000803661C1        nop.f       000000
000008000000  00000000803661C2        nop.i       000000 }
                                { .mfi
013807E803C0  00000000803661D0        add         r15 = 3F8040, r2
000008000000  00000000803661D1        nop.f       000000
000008000000  00000000803661D2        nop.i       000000 ;; }
                                { .mmi
0080C0E00400  00000000803661E0        ld8         r16 = [r14] ;;
008CC0F20000  00000000803661E1        st8         [r15] = r16
000008000000  00000000803661E2        nop.i       000000 ;; }
```

## More Complex Examples

In the following example generated code, the “rich” instruction MOV_NB_BR_CLR is used to move a null bit (an indication of a database field within a row containing a value)

© Copyright 2010 Hewlett-Packard Development Company, L.P

to a byte and then branch if the bit was clear (indicating in this case that the field was not null). Note that there are 4 operands to the “rich” instruction. The interp compiler turns this “rich” instruction in to 8 Itanium instructions stored in three bundles.

The first two instructions add the offset 00CA to r32 (the first input parameter to the subroutine) and then fetch a byte from the resultant location. The next two instructions move the offset 3F8030 to r14 and then add r30 to r14 to result in the output address of the null bit.

The fifth and sixth instructions first extract the bit specified in the first operand of the rich instruction and then test the bit to determine if it is set or clear. The extr.u instruction extracts one bit from the specified position (4 in this case) and stores the result starting at bit 0 in the register r15. In the next instruction (tbit.z), the predicate register p6 will be set if the null bit is equal to zero and will be cleared if the null bit is not equal to zero.

Finally the resultant null byte is stored. If the null bit is clear (indicating that the database field has a value), a branch is to be taken. The branch displacement is a sign-extended 21-bit value indicating a number of longwords. Here, it is a forward branch of 4 longwords. In the instruction stream, if predicate p6 is true (which indicates that the null bit was not set), a relative branch is taken. Otherwise, if predicate p6 is false, the branch is not taken and execution continues at the first instruction of the following bundle.

```
037E546C  0400019E      MOV_NB_BR_CLR
037E5470  00000004                bitNum ubyte
037E5474  000000CA                nulByt ulong
037E5478  D0000030                res    ubyte*
037E547C  FFE00004                brOff  ulong   (037E5490)
                                { .mmi
01080A0943C0  00000000803660D0          add         r15 = 00CA, r32 ;;
008000F003C0  00000000803660D1          ld1         r15 = [r15]
013807C60380  00000000803660D2          mov         r14 = 3F8030 ;; }
                                { .mii
010001E1C380  00000000803660E0          add         r14 = r14, r30
00A400F103C0  00000000803660E1          extr.u      r15 = r15, 04, 01 ;;
00A038F00180  00000000803660E2          tbit.z      p6, p7 = r15, 00 }
                                { .mfb
008C00E1E000  00000000803660F0          st1         [r14] = r15
000008000000  00000000803660F1          nop.f       000000
008400009006  00000000803660F2   (p6) br.cond.dptk.many 0000040 ;; }
```

Within the database environment, string operations (moving, changing and comparing) are common. The following example demonstrates the compiled code for the CMP_S “rich” instruction which is used to compare two fixed the length strings. The first operand is the number of bytes to compare. The second operand is the address of the first string and the third operand is the address of the second string. CMP_S returns either -1, 0 or 1 to the return status register (r8 which maps to Alpha R0) depending on the relationship (less than, equal, greater than) of the two strings.
The loop count application register ac.lc is used in conjunction with the br.cloop (branch counted loop) instruction to implement the main loop construct. Within the body of the loop, two bytes are fetched with a post increment of the source registers. Then the cmp.eq instruction is used to compare the values of the bytes for equality. Predicate register p7 is set if the comparison detects inequality. Un-equal values result in a branch out of the loop. Otherwise (in the case of the bytes being equal to each other), a

© Copyright 2010 Hewlett-Packard Development Company, L.P

backwards branch is taken by the br.cloop instruction to the prior bundle to fetch the next bytes.

When the bytes are known not equal, they are compared to each other with the cmp.lt instructions. If the strings are equal (when the loop executes to completion and no different bytes had been detected), r8 remains as zero. If the last bytes fetched are not equal (indicating that the loop did not complete and a difference was found), r8 is set to either -1 or 1. Note that within the final two bundles, the comparisons are done in parallel (the instructions can execute simultaneously because they do not depend on each other) and then the two moves are executed in parallel. The moves to r8 can be executed simultaneously because at most one of them will produce a result because predicates p6 and p7 are mutually exclusive – in no case will both be set. It is possible that neither is set (when the strings are equal) and r8 will remain 0.

```
037F22FC  0400013B      CMP_S
037F2300  0000001F                srcLen uword
037F2304  D0000288                src1   byte*
037F2308  025B5950                src2   byte*
                                { .mfi
000008000000  0000000080372090        nop.m       000000
000008000000  0000000080372091        nop.f       000000
00005413C000  0000000080372092        mov.i       ar.lc = 1E }
                                { .mfi
01382FE10380  00000000803720A0        add         r14 = 3F8288, r2
000008000000  00000000803720A1        nop.f       000000
000008000000  00000000803720A2        nop.i       000000 }
                                { .mlx
000008000000  00000000803720B0        nop.m       000000
000000000009  00000000803720B1        movl        r15 = 00000000025B5950 ;;
00C596CA03C0 }
                                { .mmi
00A000E02400  00000000803720C0        ld1         r16 = [r14], 001
00A000F02440  00000000803720C1        ld1         r17 = [r15], 001
000008000000  00000000803720C2        nop.i       000000 ;; }
                                { .mbb
01C039120180  00000000803720D0        cmp.eq      p6, p7 = r16, r17
008600003007  00000000803720D1   (p7) br.cond.dpnt.many 0000010
0091FFFFE140  00000000803720D2        br.cloop.sptk.few 1FFFFF0 ;; }
                                { .mii
010800000200  00000000803720E0        mov         r8 = r0
018001120180  00000000803720E1        cmp.lt      p6, p0 = r16, r17
0180010221C0  00000000803720E2        cmp.lt      p7, p0 = r17, r16 ;; }
                                { .mfi
013FFFCFE206  00000000803720F0   (p6) mov         r8 = 3FFFFF
000008000000  00000000803720F1        nop.f       000000
012000002207  00000000803720F2   (p7) mov         r8 = 000001 ;; }
```

In the case of the SET_T (set text) instruction, one or more bytes of a constant value are written to memory starting at a specified location. The interp compiler attempts to optimize these memory writes by performing overlapped operations and performing as few writes as possible by promoting the size of the memory reference based on the minimum alignment of the read and write stream pointers. Two pointers are used, offset by 8 bytes, to allow multiple st8 instructions to be executed in parallel. Post-increment instruction modes are used to update the output pointers in order to avoid additional instructions that would otherwise be required in order to increment the pointers. “Tail” writes of one, two or four bytes are used to complete the sequence.

© Copyright 2010 Hewlett-Packard Development Company, L.P

If the byte count for the fill was larger, a loop would have been generated to perform the fill. In addition, the interp compiler produces, as needed, code to perform one, two or four byte writes prior to the loop and then again after the loop in order to align the output pointer on an 8 byte boundary so that as few memory writes as possible are created.

```
037E5550  04000151      SET_T
037E5554  00000020              src    byte
037E5558  D0000060              dst    char*   #XFFFF8060(R12)
037E555C  0000001F              dstLen word
                              { .mfi
013807EC0380  0000000080366340        add         r14 = 3F8060, r2
000008000000  0000000080366341        nop.f       000000
000008000000  0000000080366342        nop.i       000000 ;; }
                              { .mlx
010800E10400  0000000080366350        add         r16 = 0008, r14
008080808080  0000000080366351        movl        r15 = 2020202020202020 ;;
00C2002403C0 }
                              { .mmi
00ACC0E1E400  0000000080366360        st8         [r14] = r15, 010
00ACC101E400  0000000080366361        st8         [r16] = r15, 010
000008000000  0000000080366362        nop.i       000000 ;; }
                              { .mmi
00ACC0E1E200  0000000080366370        st8         [r14] = r15, 008 ;;
00AC80E1E100  0000000080366371        st4         [r14] = r15, 004
000008000000  0000000080366372        nop.i       000000 ;; }
                              { .mmi
00AC40E1E080  0000000080366380        st2         [r14] = r15, 002 ;;
00AC00E1E040  0000000080366381        st1         [r14] = r15, 001
000008000000  0000000080366382        nop.i       000000 ;;
```

## Accessing Unaligned Data

Both the Alpha and Itanium systems impose a severe performance penalty when the processor attempts to perform an unaligned memory reference. An unaligned reference, for example, would be to attempt to fetch a longword (4 bytes) from a virtual address where the two lowest bits are not clear (i.e. not aligned on a 4 byte boundary). And the penalty on OpenVMS Itanium systems is significantly higher than it is on Alpha systems. Thus, avoiding alignment faults has an even greater benefit (for all processes on the system) on Itanium systems.

The “interp compiler” attempts to detect memory references that are not naturally aligned and produces a longer code sequence to perform the memory read or write operation without the overhead of an alignment fault.

For “rich” instructions, the assumption is made that register addresses are naturally aligned on quadword (8 byte) boundaries. Offset values can be then evaluated to determine if the resultant memory address is aligned or not. When an unaligned reference is predicted, a sequence of instructions can be generated to avoid the fault. For example, a load of a quadword that is located on a longword boundary an be accomplished by fetching the two longwords and then merging them together with the “mix4.r” instruction:

```
                              { .mmi
013807CF8E40  000000008033E180        mov         r57 = 3F807C ;;
000008000000  000000008033E181        nop.m       000000
01000393CE40  000000008033E182        add         r57 = r30, r57 ;; }
                              { .mmi
00A083908900  000000008033E190        ld4         r36 = [r57], 004 ;;
008083900380  000000008033E191        ld4         r14 = [r57]
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
000008000000  000000008033E192        nop.i       000000 ;; }
                              { .mfi
000008000000  000000008033E1A0        nop.m       000000
000008000000  000000008033E1A1        nop.f       000000
00F88241C380  000000008033E1A2        mix4.r      r14 = r14, r36 ;; }
```

The following sequence (adapted from analysis of code generated by the HP GEM compiler backend) is a longword store where the destination is predicted to not be naturally aligned. The least significant bit of the address (presented in r3) is tested. If it is set, the address is byte aligned and p7 is set; otherwise the address is word aligned and p6 is set. In the case of byte alignment, a single byte is stored and the address is incremented (thus aligned on a word boundary) and the output value is shifted 8 bits to the right. A word is then stored and the output is shifted right 16 bits. Finally, if the original address was word aligned, the final word is written, otherwise the final byte is written. This sequence results in either two (for word alignment) or three (for byte alignment) memory writes.

```
00A072000180     0611                tbit.z  pr6, pr7 = r3, 0
00AC0031004E     0621          (pr7) st1     [r3] = r8, 1
00A5B882020E     0622          (pr7) shr.u   r8 = r8, 8
00AC40310080     0630                st2     [r3] = r8, 2
00A578840200     0631                shr.u   r8 = r8, 16
008C40310006     0640          (pr6) st2     [r3] = r8
008C0031000E     0641          (pr7) st1     [r3] = r8
```

## Optimizations

In some situations, the generated Itanium code sequences will execute faster than corresponding sequences on Alpha. For example, in cases of filling or comparing a relatively few bytes of memory, the code generated for Itanium includes a sequence of memory stores or fetches in-line while the Alpha code calls to the operating system routines OTS$FILL or OTS$CMP variants. The overhead of the call in some instances will be greater than the actual memory references.

In other cases, the Itanium instruction set provides instructions that perform operations that require a sequence of instructions on the Alpha platform. For example, the “mux1@rev” instruction can be used to reverse the order of bytes within a quadword. Within Oracle Rdb on the Alpha platform, this operation is accomplished in a series of independent shift and mask instructions. This byte reversal is, for example, used when constructing index keys so performance is an important consideration as this may be a commonly executed sequence.

Optimization does tend to be a repetitive and, at least based on our observations, a never-ending process. Over time, code sequences are compressed and improved with a goal of reducing latency in regards to the CPU clock rate and memory access latencies.

For example performance analysis both “by eye” and by processor cycle sampling lead to reductions in code steams by often “combining” addressing operands in to a single bundle as in the following example; initially the operands (moving the address values to r14 and r15) would have required two bundles.

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
06CE1B68  04000148      MOV_B
06CE1B6C  D00002D0                src    byte*
06CE1B70  D00010F1                dst    byte*
                               { .mfi
01382FEA0380  00000000805326F0         add         r14 = 3F82D0, r2
000008000000  00000000805326F1         nop.f       000000
01390FEE23C0  00000000805326F2         add         r15 = 3F90F1, r2 ;; }
                               { .mmi
008000E00400  0000000080532700         ld1         r16 = [r14] ;;
008C00F20000  0000000080532701         st1         [r15] = r16
000008000000  0000000080532702         nop.i       000000 ;; }
```

## Instruction Execution Frequency

As part of a performance analysis sub-project, we created an instrumented interpretation engine that sent, via an OpenVMS mailbox, instruction execution information from all processes on a system to a separate collector process that captured instruction execution counts during a portion of an Oracle Rdb regression test run. The following table includes the top 20 instructions and the number of times each instruction was executed. In the table, the indication "RICH" indicates a "rich" instruction and "EVAX" indicates an Alpha instruction.

| Instruction Mnemonic | Execution Count |
|---|---|
| RICH_MOV_B | 60,974,337 |
| RICH_MOV_NB_BR_CLR | 60,537,063 |
| RICH_MOV_L | 47,099,034 |
| RICH_MOV_Q | 42,723,231 |
| RICH_B_BR_SET | 32,023,634 |
| RICH_BRANCH | 23,386,664 |
| RICH_MOV_S | 21,233,064 |
| EVAX_LDA | 21,169,625 |
| EVAX_BIS | 17,947,775 |
| RICH_MOV_W | 16,446,120 |
| RICH_CMP_L | 16,401,014 |
| EVAX_JSR | 15,925,377 |
| RICH_MOV_B_BR_SET | 14,723,867 |
| RICH_EXE_ACTION | 14,336,902 |
| RICH_MOV_NBIT_BR_SET | 14,138,224 |
| EVAX_BR | 13,827,371 |
| RICH_OR_B_BR_SET | 10,937,403 |
| RICH_CNV_SORT_N | 8,293,658 |
| RICH_STALL | 7,429,742 |
| EVAX_LDAH | 6,818,905 |

This data was, in turn, used as a guide for which instructions should be first considered for increased optimization by the interp compiler. The idea is that an instruction executed several times an hour has a marginal impact on performance as compared with an instruction executed thousands of times per second.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Database Performance Improvements

The creation and optimization of the Oracle Rdb "interp compiler" has been an iterative affair. Initial performance improvements from the interp compiler allowed applications running on Itanium systems to run at least as fast as on Alpha systems. Further optimizations (including reducing memory references, eliminating unneeded "stops", avoiding alignment faults, and so on) have dramatically improved code quality and yielded even better performance. In some cases, application performance has improved by a factor of 3 due to the interp compiler generating native instructions.

And Oracle continues to measure and analyze performance of the Oracle Rdb database product family on the HP OpenVMS operating system for the Integrity Server platform. An extensive set of regression tests are continuously run in our development environment to help ensure correctness of the generated code. We are also in constant contact with our customer based to help understand their performance challenges. This input helps us decide where to focus our optimization efforts to everyone's benefit.

## Models and Examples Followed

A number of different resources were referenced in regards to code generation. In addition to the (voluminous) Intel documentation of the Itanium architecture, we also utilized the compiler machine code listings from high level language compilers on OpenVMS (for example, BLISS, C and MACRO32 which all use GEM code generator and the C++ compiler which uses an Intel code generator).

Both the OpenVMS debugger and system dump analyzer include the ability to format an instruction stream which helped significantly when we were learning the intricate details of the Itanium architecture.

The OpenVMS listings include the MACRO2000 facility which implements the MACRO32 compiler. This was used in many cases as a template for code generation for complex alpha instructions (such as ZAP and MSK). The internet also proved to be an excellent resource for example instruction streams and discussions of Itanium performance in regards to the use of the architecture.
The Intel and OpenVMS documentation was referenced extensively while we were creating the unwind information tables for generated code. And the OpenVMS calling standard manual was invaluable in regards of register usage rules.

## Credit and Thanks

A large number of people devoted a great many hours to this project of developing the Oracle Rdb "Just In Time" code generator for the Itanium systems. It is not possible to remember or credit everyone who was involved. But special thanks and recognition are due to engineering members of both HP and Oracle including: John Reagan, Jeanie Leab, Guenther Froehlin, Greg Jordan, Christian Moser, Burns Fisher, Ian Smith, Martin Ramshaw, and Richard Bishop.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## For more information

The Oracle Rdb web site is accessible on the internet at www.oracle.com/rdb. For more information about the Intel Itanium architecture and instruction set, visit www.intel.com. For more information about the HP OpenVMS system, visit www.hp.com.

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# OpenVMS Power Management

Prithvi Srihari, Burns Fisher and K Veena

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Abstract

Power utilization of data centers is becoming a critical factor, often making a crucial impact to the total cost of ownership. It also can make a serious impact on the environment of our increasingly resource-conscious world. In this context, this article focuses on the Power Management facilities provided by HP OpenVMS for Integrity servers version 8.4. While Power Management features are not new to OpenVMS, the strategies employed by various versions differ. This article elucidates these and helps the administrator make a decision as to when and how to use each power setting.

## Introduction

Power usage is increasingly becoming a critical issue in modern data centers. This paper provides information on how the Power Management features on the OpenVMS Operating System can be leveraged to reduce power consumption and informs the reader on changes to the Power Management features for HP OpenVMS for Integrity servers Version 8.4.

## Rationale for Power Management

While Power Management for computers is not a new problem, it was for a long time confined to computers with very limited power resources, especially those that depend on batteries. Laptops, personal digital assistants, and networked sensors are obvious examples. However, in today's world, the power consumed by data centers is also becoming a critical issue. The motivating factors for these are:

**The Increasing Costs of Power Consumption of Servers:** The costs of cooling servers—which includes the power costs, the cooling costs as well as the infrastructure costs—has gone up much higher than the costs of the servers themselves as shown in Figure 1 [1]. Server cost has remained constant, though, if anything, they are going down. Infrastructure costs alone already exceeded the cost of the server in 2004. The combined cost of the Infrastructure and Energy (I&E), which includes the cooling needs, exceeded the cost of the server in 2001.

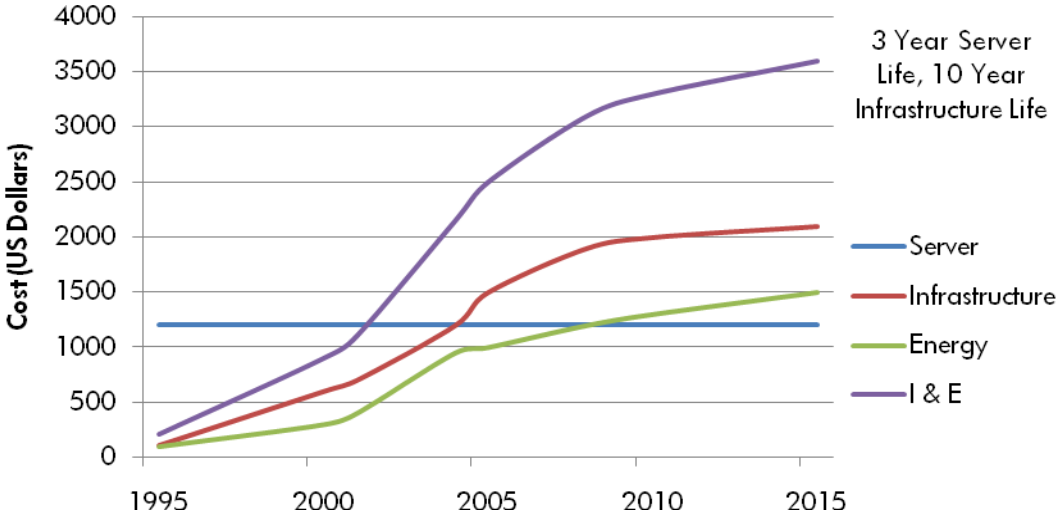



**Figure 1: Approximate Annual Amortized Costs in the Data Center for a 1U Server.**

© Copyright 2010 Hewlett-Packard Development Company, L.P

The Regulatory Emphasis on Power Efficiency: Governments worldwide are increasingly focusing on encouraging companies to produce energy-efficient products and servers are no exception. Efforts are now on to bring certain classes of servers within the ambit of the US Energy Star program [2]. In future, we may expect customers to actively choose products with better energy efficiency over those that are less energy efficient.

## HP's Approach for Power Management—How OpenVMS fits In

HP's Green Business Technology initiative provides leading and innovative capabilities for the data center, across the workplace, and for green IT practices and policies. A part of this initiative is HP's strategy to achieve energy efficiency for the enterprise, which is called 'HP Thermal Logic.' HP provides an extremely broad portfolio of energy-efficient systems, software, and services to help customers build new next-generation data centers or extend the capacity and life of existing ones [3]. HP Thermal Logic's goals are to:

a) **Reduce** total energy use
b) **Reclaim** 'trapped' energy capacity
c) **Extend** the life of the data center

| | |
|---|---|
| **Software** | Insight control environment with dynamic power capping: 3x capacity increase |
| | OS/Application power management |
| | Insight dynamics – VSE: Forecast future power savings |
| | Storage thin provisioning/dynamic capacity mgt |
| **Hardware** | HP Performance Optimized Datacenter (POD) |
| | HP BladeSystem |
| | Power optimized HP servers |
| | Low power options: Processors, memory, SSD drives: up to half the power consumption |
| **Energy savings from the component to the data center** | |

**Figure 2: HP Thermal Logic**

As illustrated in Figure 2, HP Thermal Logic encompasses both software and hardware, since power efficiency and management span a wide spectrum. This paper focuses on the 'OS/Application power management' layer, specific to OpenVMS Power Management. The power usage in a typical data center is composed primarily of the cooling load, followed by the IT (server) load, with UPS power contributing much less, as shown in Figure 3 [4]. Most of the power used by servers turns into heat that must be removed by cooling. So reducing server power consumption serves to reduce cooling power. Consequently, the first step in minimizing the power bill is to use the power management capabilities provided by servers.

© Copyright 2010 Hewlett-Packard Development Company, L.P

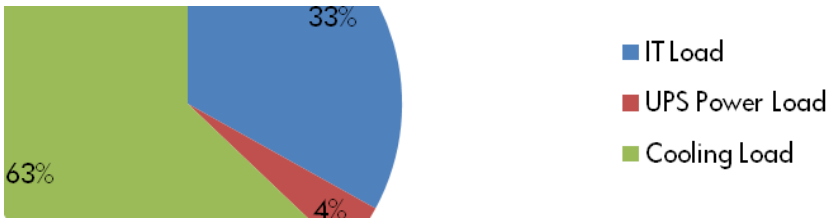



**Figure 3: This Chart shows that the Drivers of Power Usage in the Data Center are the IT Load and the Cooling Load.**

In the case of OpenVMS, these power management features primarily make use of the power savings features provided at the processor level.

## Itanium Processor Power Features

Various Intel Itanium processors implement subsets of the Advanced Configuration and Power Interface (ACPI) Processor Power and Performance States. These are implemented as controllable power states to facilitate power management. The most common of these are the 'LIGHT_HALT' state and the 'Power/Performance States' [5].

## The LIGHT_HALT State

This state corresponds to the ACPI $C_1$ Processor Power State [6]. It reduces power by stopping instruction execution, while maintaining cache and translation lookaside buffer coherence. Effectively, the processor is 'suspended,' not taking part in any scheduled activity. The processor enters this state when the firmware instruction PAL_HALT_LIGHT is called. The processor transitions from this state to the normal state in response to any unmasked interrupt.

© Copyright 2010 Hewlett-Packard Development Company, L.P

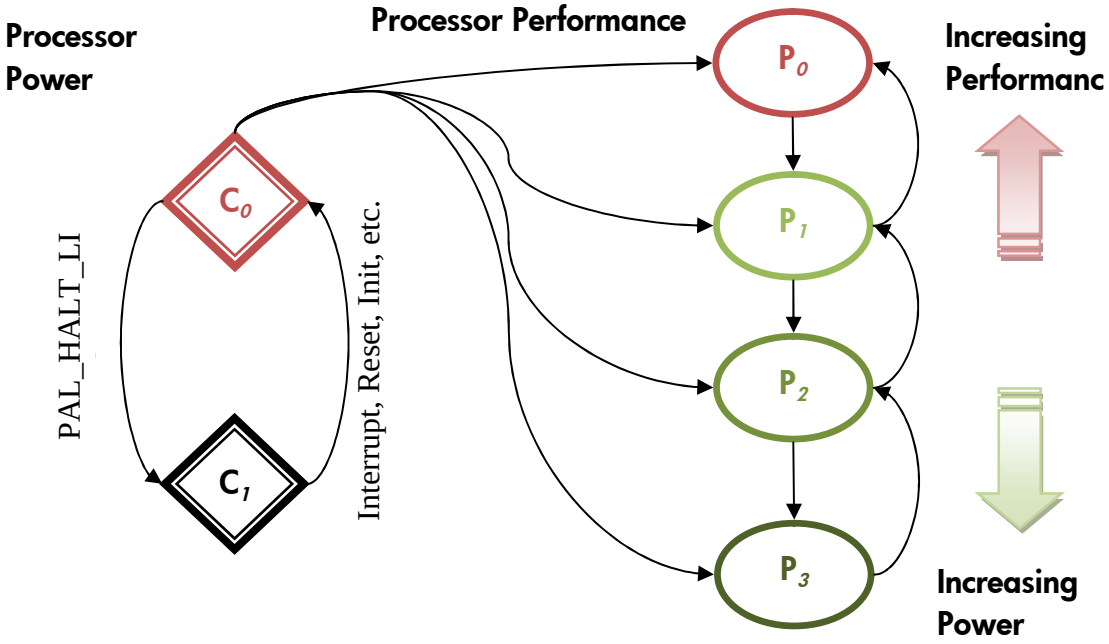



**Figure 4: A Simplified View of the Power States of an Itanium Processor with two Power and four Performance States**

These states correspond to the ACPI Processor Performance States. The states range from $P_0$ (maximum performance, least power savings) to $P_N$ (maximum power savings, least performance). The number of performance states available, *N*, varies with the model of the processor. For example, Figure 4 assumes *N* to be 3. As the performance state ranges from $P_0$, $P_1$, $P_2$,..., $P_N$, the power savings increase and the performance reduces. In all these states, however, the processor is still 'active,' albeit at a slower speed.

## HP Integrity Platform Power Options

Processor power states enable power savings at the processor level. At the system level, the administrator can configure any of four modes of operation, described below. These modes are configured from the system firmware and internally control the processor power states.

## HP Static High Performance Mode

The system's processors always operate at the highest power and performance state. In this mode, neither the firmware nor the operating system (in our case, OpenVMS) will ever program the processors to run in a lower power or performance state. This mode is useful for business operations when high system computing performance is critical and power savings is not a constraint. Effectively, running the system in this mode is as good as running a system that does not support power management features. For this reason, this mode may also be used as a baseline of power consumption data with power management.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## HP Static Low Power Mode

The system's processors operate continuously at the lowest possible power state. In this mode, the processors run continuously in the low-power state to save the maximum power possible, whilst maintaining system usability. This mode is useful for business environments where power availability is constrained. It can also be used in emergency situations when power usage has to be reduced urgently, while maintaining a reduced level of computing operations. However, it might affect the system performance if the workload has high processor utilization.

## HP Dynamic Power Savings Mode

Processor power use is adjusted on-the-fly to match performance levels to the application load. There is a slight performance loss when using this state, which is generally small but may be significant with some workloads. HP OpenVMS for Integrity servers version 8.4 has a new algorithm to manage this mode. The new algorithm switches the processor into LIGHT_HALT state when it is idle. Because there is some delay in restarting a processor that is in the LIGHT_HALT state, the algorithm tries to reduce interrupt latency by noting how often the processor is required to leave idle. If that frequency exceeds a threshold, the processor will not be allowed to enter the LIGHT_HALT state until the processor again leaves idle less often. This is to ensure that the response time of the system is quick to a burst of interrupts. Each processor in a system is monitored and adjusted independently. This mode allows the processors to operate responsively when high processor performance is needed and in a low power state otherwise.

## OS Control Mode

Dynamic power management for the system is managed by the Operating System through a policy mechanism chosen at the OS level. The power management scheme may vary depending on what the OS that is running on the system implements as its policy. For example, currently HP-UX 11i v3 treats this mode in the same way as HP Dynamic Power Savings Mode [7]. In our specific case of OpenVMS, OS Control Mode allows the OS Administrator methods for configuring the three modes mentioned earlier (Static High Performance, Static Low Power, and Dynamic Power Savings) from OpenVMS. The three sub-modes available under OS Control with OpenVMS are:

| OpenVMS Sub-Mode | Corresponding Firmware-Level Mode |
|---|---|
| OpenVMS Static High Performance | HP Static High Performance |
| OpenVMS Static Low Power | HP Static Low Power |
| OpenVMS Dynamic Power Savings | HP Dynamic Power Savings |

If no mode is chosen, by default, OpenVMS Dynamic Power Savings Mode is selected. These give the OS Administrator the flexibility to select, at run-time, a mode which suits the particular load being run without having to seek recourse to the firmware interface. Importantly, it allows the Administrator to easily automate mode changes using the standard OpenVMS job automation tools such as application programs, DCL scripts, job queues, etc. For example, it could be useful to keep a system on Dynamic Power Savings

© Copyright 2010 Hewlett-Packard Development Company, L.P

Mode during the day to provide reasonable system performance, but switch to Static Low Power Mode at night, when transaction traffic is less, in order to save power. A DCL script could easily be written to automate such a schedule.

## OpenVMS Power Management Features—Power Controlling Methods

Configuring the HP Static High Performance, HP Static Low Power, and HP Dynamic Power Savings Modes is done through the system firmware. This can be accomplished using either the Management Processor (MP) Console or the HP Integrated Lights-Out (iLO) Web Interface. Configuring the OpenVMS Static High Performance, OpenVMS Static Low Power, and OpenVMS Dynamic Power Savings Modes is done using an OpenVMS system service or through OpenVMS sysgen parameters. In either case, viewing the resulting power consumption is done using the system firmware interfaces (MP Console or iLO Web Interface).

## MP Console

At the MP console, the PM command (Power Management) is used to switch the power state, as shown in Figure 5. This command requires MP login access.

```
MP: CM> PM
Current System Power Mode: OS Control Mode
Power Regulator Menu:
     D – Dynamic Power Savings Mode
     L – Static Low Power Mode
     H – Static High Performance Mode
     O – OS Control Mode
Enter menu item or [Q] to Quit: L
Power mode will be set to Low Power.
   Confirm? (Y/[N]): Y
   Please wait…
       ➢  Power mode has been successfully changed.
```

**Figure 5: Power Setting from the MP Console.**

The new mode takes effect immediately, without the need to reboot the system.

## iLO Web Interface

The iLO web interface provides the same functionality as the MP console. To change the power state, the following steps have to be followed (in order):

1. From the main menu that appears at the top of the iLO web-page, ‘Virtual Devices’ has to be selected.
2. The sub-option ‘Power Regulator’ has to be selected.
3. The four available power modes will appear, as shown in Figure 6.

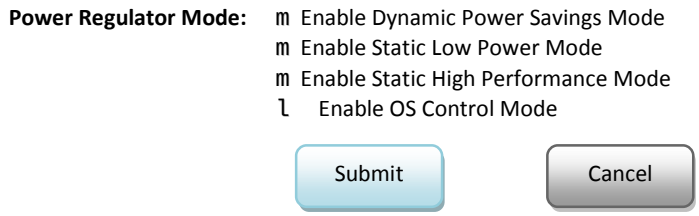



**Figure 6: Power Setting Menu in the iLO Web Interface.**

© Copyright 2010 Hewlett-Packard Development Company, L.P

4. On choosing one of the options and then clicking ‘Submit,’ the change takes effect.

Alternately, the power setting at the firmware level can also be controlled by using HP Intelligent Power Manager (HP IPM).

## OpenVMS System Service

After selecting OS Control at the firmware level, the OpenVMS system service, sys$power_control can be used to change the power state. This allows the Administrator to change the power state at the OpenVMS level programmatically. The syntax of the system service is:

```
int sys$power_control(
      unsigned __int64 request, unsigned __int64 *previous
);
```

…where request is used to choose the mode in the following manner:

| Request | Mode Chosen |
|---|---|
| POWER$C_HIGH_PERF | OpenVMS Static High Performance |
| POWER$C_LOW_POWER | OpenVMS Static Low Power |
| POWER$C_EFFICIENCY | OpenVMS Dynamic Power Savings |

…and previous is a return parameter, providing the power state that was existing before service was called.

This system service returns SS$_NORMAL if successful and an error otherwise. It requires WORLD privilege.

## OpenVMS Sysgen Parameters

OpenVMS also provides dynamic sysgen parameters to modify the power state. These parameters are CPU_POWER_MGMT and CPU_POWER_THRSH. CPU_POWER_MGMT is used primarily to change the power state. Its range of values and their meaning are:

| CPU_POWER_MGMT Value | Mode Chosen |
|---|---|
| 0 | OpenVMS Static High Performance |
| 1 | OpenVMS Static Low Power |
| 2 | OpenVMS Dynamic Power Savings |

CPU_POWER_THRSH is used only in dynamic mode. It specifies the number of interrupts per 10-millisecond interval beyond which idle power savings will be turned off. The default value is 50. The higher this number, the more power is saved and the higher average interrupt latency the system will experience while processors are idle.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## An Example Configuration

The system used is a HP BL870c with 4 cores, each of which is a 1.59 GHz Intel Montvale. The blade was lightly loaded to start with. The system's power was measured in the Static Low Power, Static High Performance, and Dynamic Power Savings Modes with processors being progressively loaded, the non-active cores being idle. The resulting power consumption graph is shown in Figure 7.

One can see the change in power consumption based on the mode chosen and the number of cores loaded. The difference between the power consumed at Low Power and High Performance Modes when all 4 cores are idle is about 11%. This tells us that if a system is found to be idle for long periods of time, it would be a good candidate to be switched to Low Power Mode. On the other hand, a system in which all 4 cores are active, the power gains between the two modes is about 4.5%. Hence, the power saving in this case is much less compelling. Of course, if a system is heavily loaded, its performance requirements may preclude switching it to Low Power Mode.

An interesting comparison is the Dynamic Mode. When system is lightly loaded, this mode provides power savings comparable to Low Power Mode. When the system is in heavy use, it provides very little power savings while giving the performance of High Performance Mode. Dynamic Mode may well be a very good mode to use when load characteristics vary between heavy and light usage (such a day-night scenario). Finally, it is anticipated that future processors will be capable of more power savings.

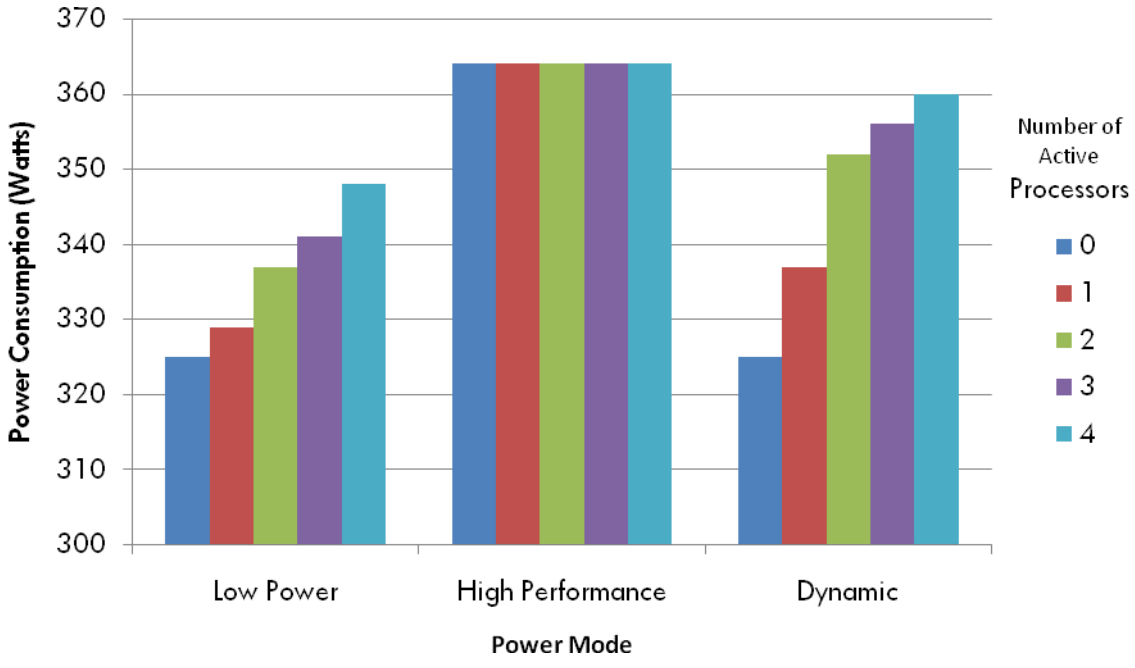



**Figure 7: An Example of the Variation of the Power Consumed for each Power Mode**

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Conclusion

In this article, the need for Power Management was discussed. The various power states provided by HP Integrity servers and HP OpenVMS for Integrity servers version 8.4 to aid in Power Management were explored. Power savings can be controlled at the OpenVMS level and power control can be dynamic. Administrators can configure power management strategies in response to changing power constraints and load profiles. Power control can be achieved using a variety of convenient interfaces. OpenVMS power management features align with HP's overall green initiatives and help customers save money without sacrificing performance while helping to ensure a cleaner and greener world.

## References


[1] Belady, Christian L., 'In the data center, power and cooling costs more than the IT equipment it supports,' Electronics Cooling Magazine, February 2007.

[2] United States Environmental Protection Agency, 'ENERGY STAR Program Requirements for Computer Servers Preliminary Draft Version 1.0 Tier 2,' September 24th 2009.

[3] Hewlett-Packard, 'HP Thermal Logic,' Internet: http://www.hp.com/go/integritythermallogic/ accessed November 16th 2009.

[4] Belady, Christian L., Wade Vinson, 'Power-Hungry Data Centers Must Develop a Plan for Curbing Costs, Increasing Efficiency,' Enterprise Networks & Servers, February 2007.

[5] Intel Corporation, 'Intel Itanium Architecture Software Developer's Manual Volume 2: System Architecture Revision 2.2,' January 2006.

[6] Hewlett-Packard, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd, Toshiba Corporation, 'Advanced Configuration and Power Interface Specification Revision 3.0b,' October 10th 2006.

[7] Hewlett-Packard, 'Going Green with HP-UX 11i v3–Power Management Techniques to Reduce Costs and Environmental Impacts,' March 2009.


© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# OpenVMS – Paged Dynamic Memory Fragmentation Causing Performance Problems

John Hockett, HP Services Technology Consultant

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Introduction

When customers added additional users to their environment, they started receiving complaints related to response time performance problems. The number of complaints increased as the time since the last reboot increased. After a reboot, the performance was acceptable for part of a normal production day. The problems were seen only by users who were running a large custom application. Typically, users running other applications and DCL commands did not see any performance problems. The users were experiencing random pauses of 3 to 5 seconds, and often multiple processes were seen in a MUTEX wait state for short intervals as shown in Figure 1.

```
$ SHOW SYSTEM
OpenVMS V7.3-2 on node L 26-OCT-2005 13:46:02.97
Pid   Process Name State Pri I/O     CPU   Page flts  Pages
:
2892F00C _TNA6942: MUTEX 6 58122 0 00:00:31.32 36814  814
2885C06C _TNA8017: MUTEX 6   491 0 00:00:03.33   592  714
2890C9D2 146811    MUTEX 6    87 0 00:00:00.83   123  112
288F7203 _TNA8023: MUTEX 5    81 0 00:00:01.96   125  116
2890BA06 _TNA8024: MUTEX 6    87 0 00:00:00.72   129  116
2890920F RA_127    MUTEX 3     0 0 00:00:00.00    20   30 S
288F6211 MS_66     MUTEX 6    15 0 00:00:00.00   123  134 S
2891BA13 MAINT_CAR MUTEX 6     0 0 00:00:00.00    20   30 S
2891AB04 _TNA7911: MUTEX 6 10016 0 00:00:10.04  7191 1412
28932E42 _TNA8009: MUTEX 6 32810 0 00:00:14.61  1231  763
:
```

Figure 1: SHOW SYSTEM

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Analysis

After reviewing the collected T4 data from node L, it was determined that each time an MWAIT spike (Blue) occurred, there was also a similar Kernel Mode spike (Red) just before and throughout the MWAIT spike.
**Note:** A process in a Mutex wait state is considered an MWAIT process. See Figure 2 below.

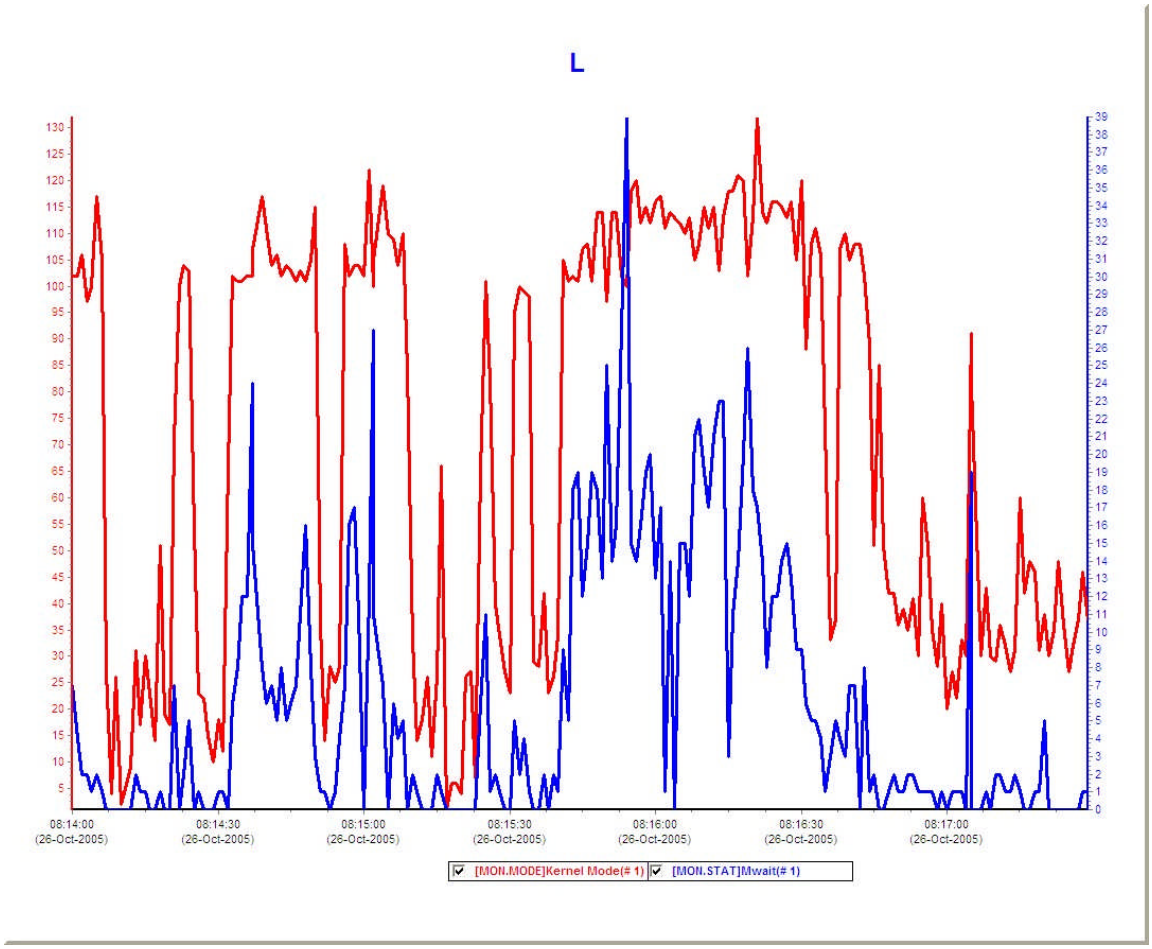



Figure 2: KERNEL mode and processes in MUTEX WAIT

© Copyright 2010 Hewlett-Packard Development Company, L.P

A process running in Kernel Mode owns a MUTEX causing the number of other processes waiting for the same MUTEX to spike. Normally, the time spent in Kernel Mode holding a MUTEX is very short, but there is evidence to show that some Kernel Mode operations are taking longer than expected, and thus causing the other processes, waiting for the same MUTEX, to spike at times.

In the following example (Figure 3), process 288E3D5A ran continually from 10:11:54.174601 to 10:11:58.000763 with most PC samples in EXE$DEALLOCATE_C+0001C or EXE$DEALLOCATE_C+00020, and an occasional LOGICAL_NAMES+00288. During this time, other processes waiting for the same MUTEX are placed in a MUTEX wait state.

```
$ ANALYZE/SYSTEM
SDA> PCS LOAD
SDA> PCS START TRACE
     wait 10 minutes
SDA> PCS STOP TRACE
SDA> SET OUTPUT PC.DAT
SDA> PCS SHOW TRACE
SDA> PCS UNLOAD
SDA> EXIT
$ EDIT PC.DAT
     :

Timestamp              CPU PC       IPL Pid      Routine
Module
---------------------- --- -------- --- -------- ----------
---------------------- ------
31-OCT 10:11:54.000216  03 80124F74  3  00000000
SCH$CALC_CPU_LOAD_C+00464
PROCESS_MANAGEMENT+00000F74
     :
31-OCT 10:11:54.174601  03 8003FCFC  2  288E3D5A
EXE$DEALLOCATE_C+0001C
SYSTEM_PRIMITIVES_MIN+00013CFC
      :  1866 total PC samples in this collection from the
same PID
         that contained:
          EXE$DEALLOCATE_C 1808 samples
          LOGICAL_NAMES 58 samples
31-OCT 10:11:58.000763  03 8003FD10  2  288E3D5A
EXE$DEALLOCATE_C+00030
SYSTEM_PRIMITIVES_MIN+00013D10
     :
```

Figure 3: System PC samples summarized

© Copyright 2010 Hewlett-Packard Development Company, L.P

The System Dump Analyzer (SDA) MTX tool is used to determine which MUTEX has the highest usage. In Figure 4, the MUTEX with the highest usage is the LNM MUTEX (logical name MUTEX).

```
$ ANALYZE/SYSTEM
SDA> MTX LOAD
SDA> MTX START TRACE
SDA> MTX SHOW TRACE/SUMMARY

Mutex Trace Information:
------------------------------------------------------------
--
           Read  Read Lock Write  Write Lock
           Locks   Waits   Locks    Waits  Wait
Unlocks
Mutex      /sec    /sec    /sec     /sec    %
/sec
---------- ------ -------  ----- -------  ------  ----
--
LNM        1664.7 602.6    145.2   188.6   43.7%
1810.0
IODB        382.4   4.1    324.0    50.8    7.8%
706.4
CEB           0.0   0.0     13.6     0.0    0.0%
13.6
PGDYN         0.0   0.0    159.6     0.3    0.2%
159.5
GSD           0.0   0.0     52.4     2.2    4.2%
52.4
CIA           0.0   0.0      0.1     0.0    0.0%
0.1
ORB           0.0   0.0      0.0     0.0    0.0%
0.0
VOL          74.0   0.0      0.0     0.0    0.0%
74.0
OBJCLS       17.2   0.0      0.0     0.0    0.0%
17.2
RSDM          0.0   0.0      1.7     0.0    0.0%
1.7
???          53.7   0.0      0.2     0.0    0.0%
53.9
           ------ -------  -----  ------  ------  ----
--
           2192.1 606.8    696.8   241.9   29.4%
2888.8
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
...
Mutex         Rate/sec  Operation          Callers PC
Module                     Offset
----------  ----------  ---------------   -----------------------------------
--------------------------------

LNM             2157.3  Lock Read          801FFFAC  LOGICAL_NAMES+07FAC
LOGICAL_NAMES             00007FAC
LNM              289.9  Lock Write         801FD7BC  LOGICAL_NAMES+057BC
LOGICAL_NAMES             000057BC
LNM              109.2  Lock Read          801F8FA8  LNM$SEARCH_ONE_C+00068
LOGICAL_NAMES             00000FA8
LNM               23.4  Lock Write         801F9314  LNM$LOCKW_C+00024
LOGICAL_NAMES             00001314
LNM               20.6  Lock Write         801FF284  LOGICAL_NAMES+07284
LOGICAL_NAMES             00007284

IODB             144.5  Lock Read          800F5EF8  IO_ROUTINES+23EF8
IO_ROUTINES               00023EF8
IODB             127.1  Lock Read          800F6920  EXE$DVI_FREEBLOCKS_C+00650
IO_ROUTINES               00024920
IODB             126.3  Lock Write         800ED164  IO_ROUTINES+1B164
IO_ROUTINES               0001B164
IODB             125.7  Lock Write         800EAEC8  IOC_STD$CREATE_UCB_C+00F68
IO_ROUTINES               00018EC8
IODB             111.4  Lock Read          800F5E68  IO_ROUTINES+23E68
IO_ROUTINES               00023E68

CEB               13.6  Lock Write         80156B34  EXE_STD$CHKWAIT2_C+00DF4
PROCESS_MANAGEMENT        00032B34

PGDYN            109.6  Lock Write         8003F604  EXE$ALOPAGED_C+00074
SYSTEM_PRIMITIVES_MIN     00013604
PGDYN             50.3  Lock Write         8003FB54  EXE$DEAPAGED_C+00094
SYSTEM_PRIMITIVES_MIN     00013B54
```

Figure 4: MUTEX usage

After observing the system following a reboot, it was evident that the Paged Dynamic Memory (PAGEDYN) free list fragmentation increased the longer the system was up. As the fragmentation increased, the Kernel Mode time and the number of processes in MUTEX wait state increased. Once the PAGEDYN freelist exceeded 10,000 free blocks, the pauses started becoming noticeable to the users.

The high LNM MUTEX utilization was presented to the customer and they were able to identify that their applications used the Job Logical Name Tables quite extensively. Working with the customer, we determined that their production applications frequently execute thousands of Logical Name creations and deletions in the Job Logical Name Table. The data structures that support the Job Logical Name Tables (LNMx) are allocated from PAGEDYN.

PAGEDYN has always been organized as a singly linked list.

© Copyright 2010 Hewlett-Packard Development Company, L.P

Each time new LNMx structures are to be allocated from PAGEDYN, the LOGICAL_ NAMES code acquires the LNM MUTEX and also requests the memory allocation code to traverse the PAGEDYN linked list, one data structure at a time until the proper sized LNMx structure can be allocated from PAGEDYN. The LOGICAL_ NAMES code then adds the LNMx to an existing Job Logical Name Table or creates a new Job Logical Name Table.

When a LNMx is to be returned, the LOGICAL_NAMES code must also acquire the LNM MUTEX and remove the LNMx structure from the Job Logical Name Table and call the memory deallocation code to traverse the PAGEDYN linked list one data structure at a time until the proper place is determined to insert the returned LNMx structure.

If PAGEDYN has become fragmented into several pieces, this can cause users to experience performance problems while the singly linked list is being traversed. During the time that it takes to allocate or return a LNMx, other processes that want to allocate or delete logical names or logical name tables go into a MUTEX wait state.

After reviewing the customer's system, it was determined that PAGEDYN was very fragmented. In the example shown in Figure 5, there are 34,826 fragments. Most of the fragments are less than 64 decimal bytes in length:

```
$ SHOW MEMORY/POOL/FULL
     :
Paged Dynamic Memory
Current Size (MB)         143.04 Current Size (Pagelets)
292960
Free Space (MB)            84.24 Space in Use (MB)
58.79
Largest Var Block (MB)     81.30 Smallest Var Block
(bytes)   16.00
Number of Free Blocks      34826 Free Blocks LEQU 64
bytes    33533
```

Figure 5: SHOW MEMORY output

Therefore, a potential allocation or deallocation of an LNMx (or some other packet from PAGEDYN) could require, in a worst case scenario, "walking" a list of 34,000 packets before the allocation or deallocation request could be completed.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Solution

To eliminate the performance problem, OpenVMS engineering designed and provided new versions of SYSTEM_PRIMITIVES*.EXE,*.STB

This new code implements PAGEDYN lookaside lists similar to the lookaside lists that have been implemented in Nonpaged Dynamic Memory for years. This new code is implemented in the MEMORYALC routine of the SYSTEM_PRIMITIVES image. The MEMORYALC routine does the actual allocation and deallocation of Paged Dynamic Pool from the new lookaside lists.

Like non-paged pool, the PAGEDYN lookaside lists start out empty. As packets are deallocated, the size is checked and the packet is returned to the appropriate list. If the packet is larger than RSVD_EXEC_1 (prior to VMS 8.4) or PAGED_LAL_SIZE (VMS 8.4 and later), the packet is returned to the variable paged pool. When an attempt is made to allocate a packet less than or equal to RSVD_EXEC_1 or PAGED_LAL_SIZE, the appropriate PAGEDYN lookaside list containing packets of that size is checked. If a packet is found, it is removed and returned to the caller without having to acquire the PGDYN MUTEX. If a packet is not found, the PGDYN MUTEX will be acquired and the packet will be allocated from the variable free list as had been traditionally done.

When RSVD_EXEC_1 or PAGED_LAL_SIZE is non-zero, PAGEDYN lookaside lists are enabled and the maximum packet size to use on the paged pool lookaside list is established. It can range between 1 - 2560 decimal bytes (the packets themselves have 16 byte granularity, 16, 32, 48, 64, etc). For most systems, setting RSVD_EXEC_1 or PAGED_LAL_SIZE to 512 bytes is more than adequate. The most utilized PAGEDYN lookaside lists are 80 to 208 bytes. Also, as part of further performance enhancements, the PGDYN MUTEX is not used if a packet is found on or returned to one of the PAGEDYN lookaside lists.

If the requested PAGEDYN lookaside lists is empty or the packet request is larger than the SYSGEN parameter setting, the PGDYN MUTEX is acquired and memory is allocated from the PAGEDYN singly linked variable size free list as it was done in past implementations.

If there is a PAGEDYN shortage, the new code will reclaim memory from the PAGEDYN lookaside lists and return it to the singly linked variable size freelist.
This new code can be enabled in SYSTEM_PRIMITIVES images dated after 14-NOV-2008 by setting the SYSGEN parameter RSVD_EXEC_1 to a nonzero value. When RSVD_EXEC_1 is 0, the default value, this feature is disabled. When RSVD_EXEC_1 is nonzero (1 - 2650), it establishes the size of the largest paged pool lookaside list to use. For example, setting RSVD_EXEC_1 to 512 would create multiple lookaside lists for packets sized up to 512 decimal bytes and should be adequate for most systems.

Remedial kits for VMS 7.3-2, V8.2, V8.2-1, V8.3, and V8.3-1H1 containing a SYSTEM_PRIMITIVES.EXE dated on or after 14-NOV-2008 will contain this new functionality.

© Copyright 2010 Hewlett-Packard Development Company, L.P

The following remedial kits or a later version of these kits contain this new functionally:

- VMS831H1I_SYS-V0400
- VMS83I_SYS-V0900
- VMS821I_SYS-V0900
- VMS83A_SYS-V1100
- VMS82A_SYS-V1200
- VMS732_SYS-V1800

The Paged Dynamic Lookaside Lists are expected to be fully implemented with the 8.4 release of OpenVMS. It will be enabled by setting the SYSGEN parameter PAGED_LAL_SIZE to a nonzero value. The default PAGED_LAL_SIZE setting of zero sets the default behavior as it has been for decades, a singly linked variable size free list.

For more information, refer to the release notes for each of the above mentioned kits. The kits are available at the following websites:
http://www.itrc.hp.com or ftp://ftp.itrc.hp.com/

## Acknowledgement

Thanks to Mark Morris for the design, implementation, and description of the changes to the SYSTEM_PRIMITIVES images. Thanks to Kevin Jenkins, Tom Cafarella, and Jean Pierre Corre for their assistance with the initial problem definition. Finally, thanks to Mark Morris, Rob Eulenstein, Ted Saul, and John Fisher for their review of this article.

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# SYSMON for OpenVMS Systems

Muthuvel Balasubramanian

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Introduction

One of the challenges that the customers face with the legacy OpenVMS environments is to have their OpenVMS servers running business applications monitored real-time and have the incidents fixed as soon as they occur. The difficulties here are to have a monitoring system deployed at the first place or if one exists already, to have their existing monitoring systems integrated with the up-to-date solutions to keep current with the technology developments. The situation worsens when the customer is running very old Versions of OpenVMS servers where in, there are no tools currently available in the market that can be deployed because of compatibility issues.

SYSMON (System Monitor) utility is designed to address these challenges.

## What is SYSMON?

SYSMON is a DCL (Digital command language) based solution that works on all OpenVMS versions and all supported hardware architectures. Currently, this solution is successfully deployed on three customer environments.

## Features

The following are the features of SYSMON:

- Built on client server model where one server will be acting as a server while the rest of the systems, the clients, will be reporting the incidents to the server. The server in turn is also being monitored by another system (secondary server) to notify in case the server itself goes down.
- Automatic failover of SYSMON primary server to the secondary server should the primary fail.
- Highly scalable and customizable.
- Automatic status tracking of incidents and automatic closure of the incidents when the issue is resolved.
- Can be installed and set up on the fly. This means, it does not require any down time of the system.
- Automatic filtering of duplicate incidents.
- Real-time monitoring interface to view the list of open issues at any point of time.
- Optional feature to choose the business hours. Any monitoring can be dynamically turned off.
- Generic alarm interface which can be used by the end users to use SYSMON to notify the incidents from their own scripts.

## Working Theory

SYSMON is a subset of OpenVMS command procedures that use the native OpenVMS DCL commands to monitor a specific entity of an OpenVMS system. Examples of these entities could be free space available on the disks, the availability of print/batch queues, and so on. SYSMON consists of the following components:

a) Client

© Copyright 2010 Hewlett-Packard Development Company, L.P

b) Primary Server
c) Monitor Utility
d) Secondary Server

## Client

The client component comprises of the monitoring routines and a scheduler that triggers them at a predefined interval. Each monitoring script has its own data file, which contains the specification of the entities to be monitored. The monitoring scripts looks after their intended OpenVMS entities and reports the anomalies to the server, if any. The incidents are notified by transferring an alarm file to a unique location on the server. Similarly, when the incident is resolved at the client's end, the client signals the server that the specific incident is resolved and the same is closed at the server end.

## Primary Server

The server component periodically polls each of the client locations and notifies the reported incidents to the HP support team via an SMTP email. In addition, the status of each of the incidents is tracked in a local database by the server. When the server finds an issue to have been resolved (as signaled by the client), it marks the corresponding incident in the master database as closed.

## Monitor Utility

The monitor component is a menu based utility which lets the HP support person to track the status of open incidents and to close them manually, when needed. Developments are underway to include new scripts on the entities such as performance monitoring, security, and so on. Presently, SYSMON can monitor the following entities:

- Node being Unreachable
- System Process Missing
- Disk Status change
- Error count increases on the devices
- Disk Space
- Highest File Version Check
- Memory Page File Utilization
- Monitor OPCOM messages
- Queue Status Monitoring
- Batch job Monitoring
- Shadow set members Decrease/ Increase
- SCS Paths between cluster systems
- Queue Managers' status

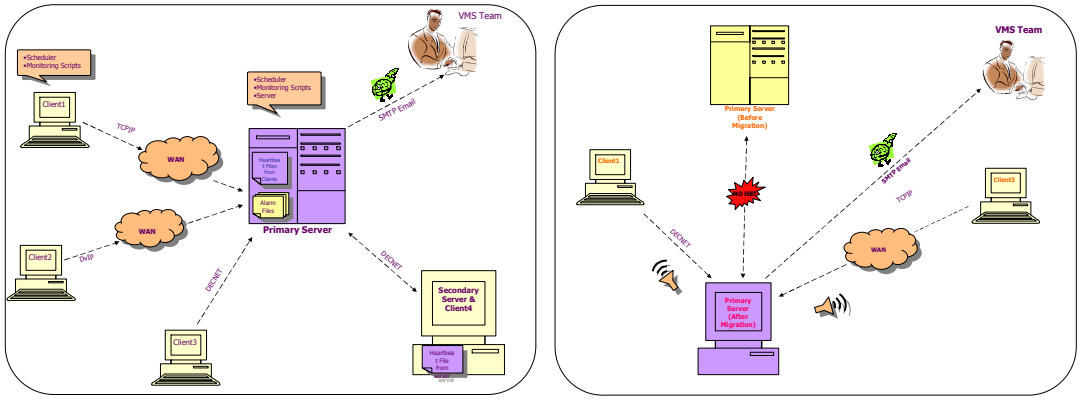
## Secondary Server

SYSMON secondary server is basically a client to the primary server. It periodically polls the primary server to see if the server component is running fine. If the server component

© Copyright 2010 Hewlett-Packard Development Company, L.P

is not running properly for a period of time or if the server is down, the secondary server migrates itself (figure b) as primary server and broadcasts the change to the rest of the clients. Subsequently, the clients will continue to transfer the incidents to the new server. Whenever, the original primary server is up, it downgrades itself as a client and also assumes the role a secondary server.

**SYSMON Architecture Overview**




a) SYSMON - During Normal Operation b) SYSMON – After Migration of Secondary to Primary

**View of the open incidents through the MONITOR Interface**

| ALMNO | NODE | TYPE | TIME_STAMP | ERROR_MESSAGE |
|---|---|---|---|---|
| 01803 | VAX1 | 007 | 30-MAY-2008 14:02 | Error Count Has Increased On Device $1$DUA50 From 00000 To 00021 |
| 01806 | VAX2 | 001 | 1-JUN-2008 18:02 | Disk $1$DKB600 ( DISK$DISK_RZ28_06 ) has less than 5% Free Blocks ( %FR |
| 01924 | DEVTES | 001 | 4-JUL-2008 05:59 | Disk DEVTES$DKA100 ( DISK$D_RZ58_01 ) has less than 10000 Free Blocks ( |
| 01944 | VAX1 | 010 | 8-JUL-2008 14:03 | QUEUE TEST_PRT01$PRINT is PAUSED |
| 01950 | TVAX | 017 | 9-JUL-2008 18:54 | OPERATOR REQUEST 44 IS PENDING |
| 01958 | MYTUS | 099 | 12-JUL-2008 00:00 | ***** SM_SCHEDULER has terminated abruptly. Please check !! ***** |
| 01959 | MYTUS | 012 | 12-JUL-2008 00:45 | **** SYSMON SEEMS TO BE NOT RUNNING **** |
| 01960 | FRPRD1 | 017 | 12-JUL-2008 07:11 | OPERATOR REQUEST 262 IS PENDING |

## Evidence that the Solution Works

SYSMON has been successfully deployed on three customer sites successfully.

## Competitive Approaches

The constraint is that the commercial monitoring tools (for e.g. HP OpenView Operations – OVO) cannot be deployed on the older environments as the tools do not support old OpenVMS versions. On the other hand, the tools available in the past (for e.g. Polycenter Watchdog) are no longer developed and supported on these legacy environments. While SYSMON is targeted for these old platforms, it can run on the latest OpenVMS versions without requiring any modifications. This is proved from the fact that it is presently running on three of our customer's systems (on all three hardware architectures (VAX, ALPHA and Itanium) and all OpenVMS versions (starting from VAX 5.5-1H3 to Integrity servers V8.3)) successfully.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Current Status

SYSMON has been running properly on all the customer systems since its deployment. It has also undergone a few enhancements, where we have introduced new monitoring entities such as monitoring the members of the mirror set, monitoring the cluster communications, and so on.

## Next Steps

The client uses either DECnet – COPY that supports Decnet proxies or through TCPIP – FTP, to transfer the alarm files to the server. As FTP transmits passwords in clear text mode, we are currently enhancing the tool to use secure methods for the alarm file transfers, i.e. SFTP. We are working to integrate SYSMON with an OpenVMS based web server so that all management tasks can be performed over the web interface.

## Conclusions

Having seen the performance for more than a year, SYSMON has proved to be an opt solution for any environment. With this innovative solution, ITO GCI RSC VMS team achieved potential cost and time savings as it would have otherwise cost HP if we had to go to a third-party vendor (or internal HP C&I team) to develop a new tool to suit the customer environment.

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# Configuring TCP/IP Services

Bart Zorn

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Overview

A technique will be presented to configure TCP/IP Services for OpenVMS for multiple systems in a consistent way, without having to go through all of TCPIP$CONFIG.COM for every system. With this method, changes in the hardware configuration are also easy to handle. No changes are made to the standard TCP/IP software; only two DCL command procedures are added. These procedures do not use any undocumented feature of TCP/IP Services.

## Introduction

TCP/IP Services for OpenVMS is not very flexible with regards to changes in the hardware configuration. The TCP/IP interfaces such as we0, ie0, and ie1 are based directly on the corresponding physical devices such as EWA0, EIA0, and EIB0. There is no way to change that relationship using logical names in a similar manner as we are accustomed to doing for most other devices in OpenVMS.

In a real life example, I had to add one Gigabit network adapter to each of four ES47 systems. These ES47 (model 4) systems consist of two 2P boxes and an I/O drawer. Logically, this I/O drawer appears to sit in between the two 2P boxes. (I am not a hardware expert, so I do not know if there are ways to change that.) The new network cards had to be placed in the I/O drawers, because there was no room in either of the two 2P boxes. The result was that the new adapters received a device name somewhere in between the existing ones, and some of the existing ones got a new name! The result of these changes was that I would have to reconfigure TCP/IP services quite extensively. I had anticipated that.

## Implementation

Because TCP/IP Services do not allow the use of logical names to designate physical interfaces, a method had to be found to circumvent that.
Two DCL command procedures have been developed and both these procedures contain relevant information for all systems. Identical copies of the procedures can be used on all systems.
The first command procedure, TCPIP_INIT_CONFIG.COM, should be executed before TCPIP$STARTUP.COM runs. It does the following:

- Sets up logical names to identify all LAN adapters (and their TCPIP alias names) by their hardware MAC address. This is done using the DCL lexical functions F$DEVICE and F$GETDVI.
- Clears out the permanent TCPIP interface database and repopulates it with a default interface described below.
- Clears out any permanent default router information and supplies a new default route described below.

© Copyright 2010 Hewlett-Packard Development Company, L.P

The logical names have the following format:

```
"ADAPTER_00-0F-20-2B-A1-38" = "EWA0","WE0"
```

The default interface and default route for each system are defined in DCL symbols like the following:

```
$ base_interface_<nodename> = -
"00-0F-20-2B-A1-38 10.5.50.11/24 10.5.50.3"
```

representing the MAC address, the IP address in Classless Inter Domain Routing (CIDR) format and the default route for this address. This information, combined with the corresponding logical name, is then used to assemble the "TCPIP SET CONFIGURATION INTERFACE" command for this interface. At least one interface needs to be defined because otherwise TCP/IP Services will not start.

Once this DCL procedure has been run, TCP/IP Services can be started and it will operate on one interface.

The second DCL procedure, CREATE_INTERFACES.COM, is called by TCPIP$SYSTARTUP.COM. This procedure does two things:

- Configures all interfaces and alias addresses
- Sets and resets the default route

The site where this configuration was developed makes extensive use of alias addresses. It appears that the ifconfig utility is much more flexible and powerful than the "TCPIP SET INTERFACE" command. Ifconfig does not create pseudo interfaces for alias addresses. The drawback is that the "TCPIP SHOW INTERFACE" command cannot display information about aliases which were created with ifconfig.

Another quirk is that ifconfig is supposed to create interface entities. I could not get it working. On the other hand, "TCPIP SET INTERFACE <ifname>" without further information creates the interface if it does not already exist. Conveniently, in that case, it does not issue an error message.

To sum it up, for every interface, a "TCPIP SET INTERFACE" command is issued, to make sure that it exists. All further configurations are done with ifconfig.

Next, for every interface, first the DCL symbol MAC is defined and then for each IP address, a call to a set_interface routine is made:

```
$ MAC := 00-0F-20-2B-A1-38
$ call set_interface 10.5.50.11/24
$ call set_interface 192.168.35.1/24 alias
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

The set_interface routine will figure out which interface is to be defined. This one gets IP address 10.5.50.11, and 192.168.35.1 as alias address.

Of course, DCL symbols can be used instead of constants. At the beginning of this procedure, symbols are defined for all IP addresses that are being used. Several IP addresses are being used more than once, for IP failover or cluster alias purposes.

A side effect of setting an address for an interface is that the default route may be erased. Therefore, once all the interfaces have been defined, the default route is set again. This default route is not necessarily the same as the one defined in the first DCL procedure described above.

## The DCL command procedures

The first one is called TCPIP_INIT_CONFIG.COM. It must be called before TCPIP$STARTUP.COM is invoked. I added this procedure to the CONFIG phase of SYSMAN STARTUP, but it can be done in other ways.

```
TCPIP_INIT_CONFIG.COM
$ set noon
$ nodename = f$getsyi("nodename")
$ if f$trnlnm("sys$pipe") .nes. "" then goto 'p1'
$!
$ call say_msg "-I- Executing CLUSTER_COMMON:TCPIP_INIT_CONFIG.COM"
$ nodename = f$getsyi("nodename")
$ debug = p1 .nes. ""
$ icalc := $sys$tools:icalc ! freeware tool, used for mask calculation
$ saved_parse_style = f$getjpi("","parse_style_perm")
$ set process/parse=traditional ! Needed for icalc, a ^ is being used
$!
$! TCPIP_INIT_CONFIG.COM
$!
$! 31-Aug-2006, Bart Zorn
$!
$! This procedure is called before TCPIP$STARTUP.COM and does three things:
$!
$! 1. It sets up logical names to identify all lan adapters (and their TCPIP
$!    alias names) by their hardware MAC address. In addition, logical names
$!    that are common to all systems are defined.
$!
$! 2. It clears out the permanent tcpip interface database and repopulates
$!    it with the default interface defined below.
$!
$! 3. It clears out any permanent default router information and supplies
$!    a new default route defined below.
$!
$ base_interface_node01 := 00-0F-20-2B-A1-38 10.5.50.11/24   10.5.50.3
$ base_interface_node02 := 00-0B-CD-F4-E4-A8 10.5.50.12/24   10.5.50.3
$!
$! 1. Lookup all hardware MAC addresses
$!
$hw_loop:
$ device = f$device("_E*","scom") - "_" – ":"
$ if device .nes. ""
$   then
$     if f$getdvi(device,"unit") .eq. 0
$       then
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
$         address = f$getdvi(device,"lan_default_mac_address")
$         interface = f$extract(1,1,device) + f$extract(0,1,device)
$         unit = %X'f$extract(2,1,device)' - 10
$         interface := 'interface''unit'
$!
$         vf = f$verify(1)
$ define/system/exec/nolog adapter_'address' 'device','interface'
$         ! 'f$verify(vf)'
$       endif
$     goto hw_loop
$   endif
$!
$! 2a. Delete current permanent interface configuration
$!
$ pipe tcpip show configuration interface/full –
> sys$manager:tcpip_saved_configuration.txt 2> nl:
$ pipe tcpip set configuration nointerface */noconfirm > nl: 2> nl:
$!
$! 2b. Repopulate the configuration database
$!
$ lognam = "adapter_" + f$element(0," ",base_interface_'nodename')
$ interface = f$trnlnm(lognam,,1)
$ if interface .eqs. ""
$   then
$     call say_msg –
"-F- TCP/IP default interface is not defined. TCP/IP will not startup."
$     goto exit
$   endif
$ ip_address = f$element(1," ",base_interface_'nodename')
$ mask_length = f$element(1,"/",ip_address)
$ ip_address = f$element(0,"/",ip_address)
$ call generate_mask mask_length mask_longword
$ call convert_longword_to_address mask_longword mask_address
$ vf = f$verify(1)
$ tcpip set configuration interface 'interface' –
/host='ip_address' /network_mask='mask_address'
$! 'f$verify(vf)'
$!
$! 3a. Delete current permanent routing configuration, ignoring errors.
$!
$ pipe tcpip show route/permanent > sys$manager:tcpip_saved_routing.txt 2> nl:
$ pipe tcpip set noroute/permanent/noconfirm/gate=* > nl: 2> nl:
$!
$! 3b. Define permanent default route information
$!
$ def_route = f$element(2,"
",f$edit(base_interface_'nodename',"compress,trim"))
$ vf = f$verify(1)
$ tcpip set route/gateway='def_route'/default/permanent
$
! 'f$verify(vf)'
$!
$exit:
$ set process/parse='saved_parse_style'
$ exit
$!
$say_msg: subroutine
$ msg = f$fao("!8%T !AS",0,P1)
$ write sys$output msg
$ if f$trnlnm("sys$output") .nes. f$trnlnm("sys$error") then write sys$error
msg
$ endsubroutine
$!
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
$generate_mask: subroutine
$ i = 32 - 'p1'
$ pipe icalc 2^'i' > nl:
$ 'p2' == .not. ('ICALC_OUT' - 1)
$ endsubroutine
$!
$convert_longword_to_address: subroutine
$ hex_longword = f$fao("!XL",'p1')
$ a1 = %x'f$extract(0,2,hex_longword)'
$ a2 = %x'f$extract(2,2,hex_longword)'
$ a3 = %x'f$extract(4,2,hex_longword)'
$ a4 = %x'f$extract(6,2,hex_longword)'
$ 'p2' :== 'a1'.'a2'.'a3'.'a4'
$ endsubroutine
```

The second procedure is called CREATE_INTERFACES.COM. It is called from TCPIP$SYSTARTUP.COM.

```
$!
$! CREATE_INTERFACES.COM
$!
$! 24-Mar-2003, Bart Zorn
$!
$ SET NOON
$ CALL SAY_MSG "-I- Executing CLUSTER_COMMON:CREATE_INTERFACES.COM"
$ ifconfig := $tcpip$ifconfig
$ nodename = f$getsyi("nodename")
$!
$! Define symbols to be used here
$!
$ NODE01 := 10.5.50.11/24
$!
$ GOTO SYSTEM_'NODENAME'
$ EXIT              ! Do not fall through
$!
$SYSTEM_NODE01:
$!
$ MAC := 00-0F-20-2B-A1-38
$ call set_interface ‘NODE01’
$ call set_interface 192.168.35.1/24 alias
$!
$ MAC := 00-0F-20-2B-A1-37
$ call set_interface ‘NODE01’
$ call set_interface 192.168.35.1/24 alias
$!
$ goto exit
$!
$SYSTEM_NODE02:
$!
$ MAC := 00-0B-CD-F4-E4-A8
$ call set_interface 10.5.50.12/24
$ call set_interface 10.5.50.18/24 alias             ! Cluster alias
$!
$ MAC := 00-0B-CD-F4-E4-A9
$ call set_interface 10.5.50.12/24
$ call set_interface 10.5.50.18/24 alias             ! Cluster alias
$!
$ MAC := 00-08-02-91-88-CA
$ call set_interface 10.5.50.13/24
$ call set_interface 10.5.50.33/24 alias
$!
$ call set_interface 10.5.52.1/24 home         ! Application alias
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
$!
$ goto exit
$!
$EXIT:
$ gosub reset_default_route
$ EXIT
$!
$say_msg: subroutine
$ msg = f$fao("!8%T !AS",0,p1)
$ write sys$output msg
$ if f$trnlnm("sys$output") .nes. f$trnlnm("sys$error") then –
write sys$error msg
$ endsubroutine
$!
$set_interface: subroutine
$!
$! p1 - address/mask
$! p2 - optional parameters
$! p3 - additional parameters for ifconfig
$!
$ lnm = "ADAPTER_" + MAC
$ interface = f$trnlnm(lnm,,1)
$ if interface .eqs. ""
$   then
$     call say_msg "-E- ''lnm' logical name is missing"
$     exit
$   endif
$!
$! Create interface if it does not already exist
$!
$ tcpip set interface 'interface'
$!
$ if f$edit(p2,"lowercase") .eqs. "home" then p2 := home alias
$ params = f$edit(f$fao("!AS !AS
!AS",p3,interface,p2),"trim,compress,lowercase")
$ sv = f$verify(1)
$ ifconfig 'params' 'p1'
$ ! 'f$verify(sv)'
$ endsubroutine
$!
$reset_default_route:
$ if "''new_default_route'" .eqs. "" then return
$ if f$mode() .eqs. "INTERACTIVE"
$   then
$     if f$trnlnm("tt") .nes. "OPA0:" then return
$   else
$     if f$mode() .nes. "OTHER" then then return
$   endif
$!
$ pipe tcpip netstat -rn | search sys$pipe default | -
       ( read sys$pipe line ; define/job/nolog line &line )
$ line = f$edit(f$trnlnm("line"),"trim,compress")
$ deassign/job line
$ default_present = f$element(0," ",line) .eqs. "default"
$ if default_present
$   then
$     current_default_route = f$element(1," ",line)
$     if current_default_route .eqs. new_default_route then return
$   endif
$ vf = f$verify(1)
$ tcpip set route /gate='new_default_route' /default
$ ! 'f$verify(vf)'
$ if default_present
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
$   then
$     if current_default_route .nes. new_default_route
$       then
$         vf = f$verify(1)
$ tcpip set noroute /gate='current_default_route' /noconfirm
$         ! 'f$verify(vf)'
$       endif
$   endif
$ return
```

## Things yet to be done

When a new system needs to be configured, it is still necessary to run TCPIP$CONFIG.COM once, but there is no need to configure all interfaces. I have not yet reverse engineered what steps TCPIP$CONFIG.COM takes with regard to the host name and domain name settings. Also, the client and server configuration needs to be done with TCPIP$CONFIG.COM.

## Summary

The techniques described here allow for a complete interface configuration for TCP/IP Services for OpenVMS with two DCL command procedures. These procedures are organized in such a way that they contain all relevant information for all systems to be configured. This makes it a lot easier to configure many systems and prevent duplicate IP addresses and other errors.

## For more information

The author can be contacted at Bart.Zorn@Yahoo.com

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# HP OpenVMS CIFS File Security and Management

Shilpa K, HP OpenVMS CIFS File Security and Management

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Intended Audience

This article is intended for OpenVMS administrators who are responsible for managing HP OpenVMS CIFS product. Within the scope of HP OpenVMS CIFS as Member Server, this article provides information about HP OpenVMS CIFS File Security. HP OpenVMS CIFS File Security is used for managing permissions for users and groups that are accessing files on an OpenVMS system through HP OpenVMS CIFS. This article also explains various aspects of HP OpenVMS CIFS File Security such as user, group and permission mapping.

## Introduction to CIFS

HP OpenVMS CIFS is based on Open Source Samba. According to Samba.org, the definition of samba is "*Samba is an Open Source/Free Software suite that provides seamless file and print services to SMB/CIFS clients. Samba is freely available, unlike other SMB/CIFS implementations, and allows for interoperability between Linux/Unix servers and Windows-based clients. Samba is software that can be run on a platform other than Microsoft Windows, for example, UNIX, Linux, IBM System 390, OpenVMS, and other operating systems. Samba uses the TCP/IP protocol that is installed on the host server. When correctly configured, it allows that host to interact with a Microsoft Windows client or server as if it is a Windows file and print server*."

Samba is based on Microsoft Common Internet File System (CIFS) protocol. CIFS protocol mainly uses Server Message Block (SMB) commands for communicating with different systems over network.

Samba being an Open Source product with a world-wide community of developers, it keeps pace with new Microsoft Operating System releases and thus provides seamless integration with Windows systems. To take advantage of this and thus to provide OpenVMS customers with file and print services that keep pace with new Windows releases, Open Source Samba has been ported onto OpenVMS. This is intended as an alternative to the existing file and printer services product, Advanced Server for OpenVMS. The ported version of Open Source Samba on OpenVMS is referred by the name, HP OpenVMS CIFS. HP OpenVMS CIFS is supported on OpenVMS version 8.2 and later on Alpha platform, and OpenVMS version 8.2-1 and later on Integrity servers.

The main features of HP OpenVMS CIFS henceforth referred simply as CIFS (not to be confused with CIFS protocol) are:

1. Domain Support
2. Authentication
3. Cluster Services
4. Browsing
5. File and Print Services
6. File and Print Security

© Copyright 2010 Hewlett-Packard Development Company, L.P

A brief description about each of the main features and a detailed description of File Security are provided in the following sections.

**Domain Support:**
CIFS can act as a NT4-style Member Server in any domain. From CIFS version 1.2 onwards, it can participate as a member in native mode Active Directory Windows domain that uses Kerberos authentication.
It can act as a NT4-style Primary Domain Controller (PDC), but such a domain may only contain Backup Domain Controllers (BDCs) that run CIFS. Similarly, it can function as a NT4-style BDC only if the PDC is also running CIFS. However, unlike HP Advanced Server for OpenVMS and Windows domain controllers, automatic replication of the user accounts database is not possible between CIFS PDC and BDCs. To accomplish the automatic replication of account databases, CIFS requires the assistance of LDAP servers. By configuring the CIFS PDC and BDCs to use the LDAP backend, replication of the accounts database is achieved by virtue of the synchronization between LDAP servers. CIFS can use the LDAP backend to store and obtain user and group account information in the LDAP directory (such as HP Enterprise Directory or an OpenLDAP server).

Though a single LDAP server can be used for both the CIFS PDC and BDCs, it is highly recommended that separate LDAP servers be used for high availability and better performance.

**Authentication:**
CIFS supports the basic and less secure share level security wherein the password is supplied when accessing each share, and a more secure user level security where the username and password must be supplied to successfully establish connection to CIFS Server before accessing shares.

In user level security, CIFS supports the following authentication mechanisms:

1. LM - used by old Windows systems
2. NTLM - used by Windows NT and later systems
3. NTLMv2 - used by Windows NT and later systems
4. Kerberos authentication from CIFS version 1.2 onwards (When CIFS acts as a Member Server in Windows Active Directory)

Additionally, CIFS provides:

1. NT LAN Manager Security Support Provider (NTLMSSP ) support for securing NTLM and NTLMv2 authentication
2. Session security by signing and sealing secure channel data between a domain member and a domain controller. The CIFS session security can use 64-bit or128-bit encryption key for encrypting the secure channel data.
3. SMB signing or security signatures, which is used for securing SMB protocol.

© Copyright 2010 Hewlett-Packard Development Company, L.P

**Cluster Services:**
CIFS can be installed either on a single node in an OpenVMS Cluster or, as a CIFS Member Server, on multiple nodes that share the same CIFS installation directory.

As a single node, CIFS can be installed as a distinct entity (for any CIFS server role) on each cluster node with separate installation and configuration and thus each node acts as if on non-clustered OpenVMS system. In this case, each node where CIFS is installed must not share the same installation directory and must not allow access to the same directory or files through multiple cluster members.

As a Member server, a common cluster configuration is possible where multiple cluster members can share the same CIFS installation and configuration directory and data files. In such an environment, CIFS functions as though the cluster is a single domain entity. In a cluster, the nodes that share the same CIFS installation and configuration directory must also share the same SYSUAF and RIGHTSLIST databases.

**Browsing:**
CIFS supports traditional Windows Browser service functionality. Browser service functionality is responsible for the “Network neighborhood” view provided by Windows.

**File and Print Services:**
CIFS allows users to share files and printers present on OpenVMS system to Windows, Linux, and UNIX clients. These clients see the shared files and printers as being present on a local system. Due to this, users can seamlessly work with shared files and printers using the available interfaces on the client system.

CIFS supports files present on ODS-5 and ODS-2 volumes. It can present files with different OpenVMS file formats and file organizations to Windows clients in a Stream format. Thus the files with different formats are made readable to Windows clients. Additionally, CIFS allows you to create files from clients in Stream, Stream_LF, Fixed, or Undefined format. By default, CIFS supports ASCII character set. Additionally, it supports Extended ASCII character set (CP850/ISO-8859-1) for some European characters and VTF-7 support for Japanese characters.

Using CIFS printing services, you can share printers that are connected directly to an OpenVMS system or any printers present on the network. For sharing printers using CIFS, print queues must be setup on the OpenVMS system. These print queues can be setup using DCPS, TELNETSYM, LAT, or LPD. CIFS supports NT-style printing functionalities such as:

- Printer driver files can be downloaded locally on Windows clients
- Printer driver files can be uploaded onto CIFS Server from the Windows clients using Add Printer wizard

© Copyright 2010 Hewlett-Packard Development Company, L.P

**File and Print Security:**
An important requirement for any File and Print Services is file security. Unlike Advanced Server for OpenVMS, which provides NT ACL based file security as well as OpenVMS based file security, CIFS provides file security to Windows clients using OpenVMS file security alone. That is, it maps Windows security applied on the files and directories to OpenVMS file security. File security can be set for any Windows/CIFS user or group. File and directory access auditing is not provided by CIFS, but standard OpenVMS auditing can be used for this purpose.

CIFS supports Access Control Lists (ACLs) on printer objects as well.

**CIFS Components**
The main components of CIFS that provide the above features are:

1. SMBD process
2. NMBD process
3. WINBIND

SMBD process:
SMBD process is responsible for providing domain support, cluster services, authentication, File and Printer Services, and CIFS File Security mapping functionality. Each client session creates a new SMBD process. Each SMBD process provides domain support, cluster coordination services, authentication, file and print services, and File Security mapping.

NMBD process:
NMBD process provides traditional Windows Browser service functionality apart from handling NETBIOS name registration and resolution.

WINBIND
WINBIND is a special feature of CIFS that supports automatic mapping of Windows domain users and groups to OpenVMS UICs and resource identifiers, nested groups (a group with-in-a group) and trust functionality. In this article, winbind functionality and WINBIND are used interchangeably and they mean the same. The first 2 features of WINBIND will be covered in greater detail under the section “The Role of WINBIND in CIFS File Security”.

On other platforms like Linux, Samba provides winbind functionality through a process named, WINBINDD. On OpenVMS, winbind functionality is integrated into SMBD process.

## The Scope

Now that a brief summary of CIFS features and components have been provided, the scope of this article is limited to explaining File security when CIFS is configured as Member Server.

© Copyright 2010 Hewlett-Packard Development Company, L.P

This article does not explain the steps for setting share ACLs on CIFS shares using Windows 'Computer Management' applet. The steps provided in this article are applicable for setting File Security on CIFS shares apart from directories and files under the CIFS shares.

## CIFS as Member Server

As a member server, CIFS allows the following users to access CIFS shares:

1. Users belonging to the Windows domain, where CIFS is a member
2. Users belonging to the domains trusted by the Windows domain, where CIFS is a member
3. CIFS local users

A Windows domain user can be a member of multiple domain global groups in the Windows domain. The domain global groups can be added as members to CIFS local groups and this is referred as nested grouping. Users and domain global groups belonging to the domains trusted by the Windows domain, where CIFS is a member and CIFS local users and groups can also be a part of CIFS local groups.

CIFS allows you to set permissions on files and directories for the above mentioned users and groups.
You can also set permissions based on the mapped OpenVMS usernames and resource identifiers. This is supported because of the user and group mapping functionality provided by CIFS and due to the additional support it provides for OpenVMS File Security apart from Windows File security.

As mentioned under File and Print Security feature, CIFS maps Windows File Security to OpenVMS File Security. This raises the basic question: Why is CIFS providing this security mapping instead of providing Advanced Server style NT ACL support?

## Why is CIFS Server dependent upon OpenVMS File Security?

For those customers who are planning to use CIFS, the likely concern about File Security is probably well expressed by the words in Samba HOW-TO collection on File Access Controls: "*Advanced MS Windows users are frequently perplexed when file, directory, and share manipulation of resources shared via Samba do not behave in the manner they might expect. MS Windows network administrators are often confused regarding network access controls and how to provide users with the access they need while protecting resources from unauthorized access.*

*Many UNIX administrators are unfamiliar with the MS Windows environment and in particular have difficulty in visualizing what the MS Windows user wishes to achieve in attempts to set file and directory access permissions.*

© Copyright 2010 Hewlett-Packard Development Company, L.P

*The problem lies in the differences in how file and directory permissions and controls work between the two environments. This difference is one that Samba cannot completely hide, even though it does try to bridge the chasm to a degree."*

It continues to say *"This is an opportune point to mention that Samba was created to provide a means of interoperability and interchange of data between differing operating environments. Samba has no intent to change UNIX/Linux into a platform like MS Windows. Instead the purpose was and is to provide a sufficient level of exchange of data between the two environments. What is available today extends well beyond early plans and expectations, yet the gap continues to shrink*."

As part of porting Open Source Samba on to OpenVMS, due to the way File Security is implemented in Samba, CIFS had to map the Windows File Security to the nearest possible or an equivalent File Security provided by OpenVMS. To understand the File Security mapping provided by CIFS, it is necessary to know the following:

1. Windows File Security
2. OpenVMS File Security
3. The need for CIFS to map Windows domain users and groups to OpenVMS usernames and resource identifiers
4. User and group mapping
5. The role of WINBIND in CIFS File Security
6. Windows to OpenVMS File Security mapping provided by CIFS
7. Limitations due to mapping Windows File Security to OpenVMS File Security

Once these aspects are covered, this article further explains the steps required for setting up File Security – from a Windows system and from an OpenVMS system.

Note that the goal of this article is not to cover in detail about Windows and OpenVMS File Security. Only a brief introduction about these topics has been provided, which might help understand CIFS File Security better.

## A brief introduction to Windows File Security

Windows File Security is mainly based on Security Principals and Discretionary Access Control Lists. A security principal can be a user or a group. On a Windows system, each user and group is identified by a unique Security Identifier (SID) in the domain. Windows creates an Access Token structure for a user after successfully authenticating the user. An Access Token structure consists of the logged in user's SID and the SIDs for each of the groups that the user belongs to. In Windows, a local group can contain another group within it and this is referred as nested-grouping. Suppose if a user ANITA belongs to group ACCOUNTS and if the group ACCOUNTS is a member of another group FINANCE, then by virtue of nested grouping, the user ANITA is a member of both ACCOUNTS and FINANCE groups.

© Copyright 2010 Hewlett-Packard Development Company, L.P

Access to each file and directory (otherwise referred to as objects) in a Windows system can be restricted by applying permissions on them. These permissions on an object are applied at the discretion of an owner of the object and are also called Discretionary Access Control Lists (DACLs). The object's security descriptor is made up of these DACLs. The DACL is a list of Access Control Entries (ACE) on the object and each ACE contains permission for a user or group. Within the ACE, a user or group is represented by its SID. An ACE can be granted for users and groups in the same domain, for users and groups in the trusted domain and for users and groups in the local Windows system. Windows allows an explicit deny access to an object apart from allow access. Additionally, a user or a group can be granted explicit access rights (special privileges) by administrators that allow access to the object.

When a user tries to access an object, Windows decides access to the object based on the SIDs in the user's Access Token and the object's security descriptor consisting of ACEs. Windows grants or denies access rights by finding a match for the SIDs in the Access Token of the user with that of the object's ACE list. If a match is not found even after traversing the entire ACE list on the object, then access to the object is denied. In case the object's DACL contains more than one ACE that matches the SIDs in the user's Access Token, then Windows accumulates access rights for each ACE until the accumulated access rights exceeds the requested access rights. Thus, for a user ANITA, if the group ACCOUNTS allowed READ access to an object, and group FINANCE allowed WRITE access to the same object, then Windows grants READ and WRITE access to the object for the user ANITA. From this, we can conclude that the order of ACEs in DACL of the object is not important on a Windows system.

## A brief introduction to OpenVMS File Security

The OpenVMS operating system controls access to any object that contains shareable information. These objects are known as protected objects. Files and directories fall under the category of protected objects. An accessing process or thread carries credentials in the form of rights identifiers, and all protected objects list a set of access requirements specifying who has a right to access the object in a given manner. These are respectively known as User's Security Profile and Object's Security Profile.

User's Security Profile:
The types of identification the system assigns to processes to define their access rights to objects is referred to as the User's Security Profile. When a user tries to access an object, it is the process or thread executing on behalf of the user that uses the user's security profile to access the object. The User's Security Profile is also referred to as Subject's Security Profile. A User's Security Profile includes the following elements:

1. User identification code (UIC) identifying the user
2. Rights identifiers held by the process
3. Privileges, if any

© Copyright 2010 Hewlett-Packard Development Company, L.P

User Identification Code (UIC)
The first element of a subject's security profile is the user identification code (UIC). Your UIC tells what system group you belong to and what your unique identification is within that group.

Rights Identifiers
The second element of a subject's security profile is a set of rights identifiers. A rights identifier represents an individual user or a group of users. While accessing files and directories, the Rights Identifiers of interest are:

1. General identifiers - Defined by the security administrator.
2. UIC identifiers - Based on a user's identification code (UIC), which uniquely identifies a user on the system and defines the group to which the user belongs.

Privileges
A third (optional) element of a subject's security profile is a set of privileges. Privileges let you use or perform system functions that ordinarily would be denied to you. The privileges held by the user can affect access to an object.

How Privileges Affect Protection Mechanisms
Security administrators can assign privileges to users when they create or modify user accounts. The system privileges READALL and BYPASS affect user access, regardless of the access dictated by an ACL for the object or by other elements in its security profile. The privileges SYSPRV and GRPPRV are controlled through the system category of the protection code. The privileges have the following meanings:

| | |
|---|---|
| BYPASS | A user with BYPASS privilege receives all types of access to the object, regardless of its protection. |
| GRPPRV | A user with GRPPRV privilege whose UIC group matches the group of the owner of the object receives the same access accorded to users in the system category. Thus, the user with GRPPRV privilege is able to manage any of the group's objects. |
| READALL | A user with READALL privilege receives read access to the object, even if that access is denied by the ACL and the protection code. In addition, the user can receive any other access granted through the protection code. |
| SYSPRV | A user with SYSPRV privilege receives the access accorded to users in the system category. |

© Copyright 2010 Hewlett-Packard Development Company, L.P

When you define ACLs or protection codes for your objects, remember that users with amplified privileges are entitled to special access to objects throughout the system. For example, there is no way to stop a user with the BYPASS privilege from accessing your files. Users with GRPPRV privilege have the power to perform many system management functions for other members of their UIC group.

Object's Security Profile:
The previous section described User's Security Profile that is required for a user to access the object. The security elements of any object comprise its security profile and the object's security profile defines a list of requirements for accessing the object. An object's security profile contains the following types of information:

1. The owner of the object. The system uses this element in interpreting the protection code.
2. The protection code defining access to objects based on the categories of system, owner, group, and world. This protection code controls broad categories of users.
3. The access control list (ACL) controlling access to objects by individual users or groups of users.

A brief overview of security elements in Object Security Profile:

Owner
The first element of an object's security profile is the UIC of its owner. In most cases, if you create an object, you are its owner. As the owner, you can modify its security profile. The system automatically assigns your UIC to the object and uses it in making access decisions. There are some exceptions to the ownership rule. Files owned by resource identifiers do not have a UIC. When a user creates a file in the directory of a resource identifier, the file may be owned by the resource identifier and not the user who created the file.

Protection Code
The second element of an object's security profile is the object's protection code. The protection code associated with an object determines the type of access allowed to a user, based on the relationship between the user UIC and the owner UIC. A protection code defines the access rights for four categories of users: (a) the owner, (b) the users who share the same group UIC as the owner (the group category), (c) all users on the system (the world category), and (d) those with system privileges or rights (the system category). OpenVMS code lists access rights in a fixed order: the
system category (S), then owner (O), then group (G), and then world (W). It has the following syntax:
[user category: access allowed (,user category: access allowed,...)]

Access Control List (ACL)
The third (optional) element of an object's security profile is the object's access control list. An access control list (ACL) is a collection of Access Control Entries (ACEs) that

© Copyright 2010 Hewlett-Packard Development Company, L.P

define the access rights a user or group of users has to a particular protected object, such as a file, directory, or a device.

With this background information on User's Security Profile and Object's Security Profiles, the next section describes how the OpenVMS system determines a user's access to a protected object.

How the System Determines If a User Can Access a Protected Object

When a user tries to access a protected object, the operating system calls the Check Protection ($CHKPRO) system service to compare the security profile of the user process or thread with the security profile of the object. In the protection check, $CHKPRO compares the user's security profile against the protected object's profile using the following sequence:

1. Evaluate the access control list (ACL).
If the object has an ACL, the system scans it, looking for an entry that matches any of the user's rights identifiers. If a matching access control entry (ACE) is found, the system either grants or denies access, and further checking of the ACL stops.

When the matching ACE denies access, a user can still gain access either through the system and owner fields of the protection code or through privilege. When an ACL has no matching ACE, the system checks all fields of the protection code.

2. Evaluate the protection code.
If the ACL did not grant access and the object's owner UIC is not zero (refer NOTE), the operating system evaluates the protection code. The operating system grants or denies access based on the relationship between the user's identification code (UIC) and the object's protection code.
For cases where an ACL has denied access, the system examines two fields in the protection code---the system and owner fields---to determine if the user is allowed access. The user can still acquire access by being a member of the system or owner categories or by possessing privileges. A user holding GRPPRV (with a matching group UIC) or SYSPRV is granted the access specified for the system category of the protection code.

NOTE: When an object has an owner UIC of zero, the protection code is not checked. Users have all but control access to the object, *provided* the ACL has no Identifier ACEs. If Identifier ACEs are present, then access has to be granted explicitly through the ACL or through privilege.

3. Look for special privileges.
If access was not granted by the ACL or the protection code, privileges are evaluated. Users with certain system privileges may be entitled to access regardless of the protection offered by the ACLs or the protection code. The bypass privilege (BYPASS), group

© Copyright 2010 Hewlett-Packard Development Company, L.P

privilege (GRPPRV), read all privilege (READALL), or system privilege (SYSPRV) amplifies the holder's access to objects.

The above three steps explains the fact that unless the user's security profile has special privileges, and if the object has ACLs set on it, the order of the ACLs on the object's security profile is very important in granting access to a user.

NOTE: For providing a brief description about OpenVMS File Security, almost all the information mentioned above has been directly obtained from the HP OpenVMS Guide to System Security. Minor modifications have been done to keep the security description to files and directories.

## The need for user, group and permission mapping by CIFS

With the above information about Windows and OpenVMS File Security, it can be observed that there is no one-to-one mapping of Windows users and groups with OpenVMS usernames (UICs) and resource identifiers. As a File Server that uses CIFS or SMB protocol, CIFS is required to provide Windows-like File Security based on Windows users and groups. As CIFS is dependent upon host system File Security, it has to use OpenVMS File Security that is based on UICs and resource identifiers. The only way for CIFS to bridge the File Security on these two different Operating Systems is by mapping:

1. Windows users and groups to OpenVMS usernames (UICs) and resource identifiers
2. Windows permissions to OpenVMS permissions.

The following sections describe the Windows to OpenVMS user and group mapping and the permission mapping provided by CIFS.

## User and Group mapping

Using the user and group mapping mechanism, CIFS maps domain users to OpenVMS usernames and domain groups to OpenVMS resource identifiers. Internally, the mapping is done between domain user SIDs and OpenVMS UICs and, domain group SIDs and OpenVMS resource identifier values. For easy readability, it is referred to as mapping of domain users and groups to OpenVMS usernames (UICs) and resource identifiers. The following topics in this section briefly describe the user and group mapping mechanism.

**User Mapping**

User mapping involves mapping users belonging to the Windows domain where CIFS is a member, the domains trusted by it and the CIFS server, to OpenVMS usernames (UICs). There are three ways using which CIFS maps these users to OpenVMS usernames:

1. Automatic user mapping provided by WINBIND.

© Copyright 2010 Hewlett-Packard Development Company, L.P

2. Implicit user mapping – A domain user with the same username in SYSUAF on OpenVMS is implicitly mapped. CIFS local users are also mapped using this mechanism.
3. Explicit user mapping - Using username map file, domain users can be explicitly mapped to any OpenVMS username.

***Automatic User Mapping***

Users belonging to the Windows domain where CIFS is a member or the domains trusted by it can be automatically mapped to OpenVMS usernames provided that WINBIND is enabled and a valid idmap UID range is available in the Samba Configuration File. This will be explained in detailed under the topic 'The role of WINBIND in CIFS File Security'.

***Implicit User Mapping***

A user belonging to the Windows domain where CIFS is a member or the domains trusted by it can be implicitly mapped to an OpenVMS username provided the names are same. For example, when a user ANITA belonging to the Windows domain connects to CIFS Server, the user ANITA is implicitly mapped an OpenVMS username ANITA, if it exists.

By default, CIFS local users are implicitly mapped to OpenVMS usernames. This is because, a CIFS local user can be created using pdbedit utility, only if a matching OpenVMS username exists in SYSUAF on OpenVMS. For example, to create a CIFS local user STEFFI, you must first create or ensure that an OpenVMS username STEFFI exists in SYSUAF database. The CIFS user accounts database file, SAMBA$ROOT:[PRIVATE]PASSDB.TDB maintains records related to CIFS local users.

***Explicit User Mapping - Mapping domain users using username map file***

Explicit user mapping allows you to map users in the Windows domain where CIFS is a member and the domains trusted by it to any OpenVMS username using a username map file.

CIFS supports mapping of multiple domain users to a single OpenVMS username. In this case, the domain users that have been mapped to a single OpenVMS username share the File Security that has been specified for that mapped OpenVMS username. For example, if a directory and the sub directories and files under it are owned by a mapped OpenVMS username ASVUSER, then all the domain users that are mapped to OpenVMS username ASVUSER have access to this directory and the sub directories and files under it. The same is applicable if the mapped OpenVMS username ASVUSER has been explicitly granted resource identifiers in the SYSUAF database, and File Security on any object in a CIFS share contains security for these resource identifiers. In this case, all the domain users that have been mapped to the OpenVMS username ASVUSER will also be able to access objects based on the resource identifiers that have been granted to the user ASVUSER in SYSUAF.

© Copyright 2010 Hewlett-Packard Development Company, L.P

The following examples provide information on how to setup explicit user mapping:

Edit the username map file, SAMBA$ROOT:[LIB]USERNAME.MAP and add the user mappings in any of the following ways based on your CIFS setup requirements:

# The text following a hash (#) or a semi-colon (;) indicate comment lines.
# You can add your own comment lines by prefixing the text with a hash or a semi-colon

; Example for mapping a user in Windows domain (CIFSDOM) to OpenVMS username
SYSTEM=CIFSDOM\administrator

; Example for mapping multiple Windows domain users to a single OpenVMS username
ASVUSER=CIFSDOM\tunga CIFSDOM\kaveri

; Example for mapping a user in trusted domain (TRUSTDOM) to an OpenVMS username
CIFSUSER=TRUSTEDOM\neela

; An example of username map search stopping after encountering the required mapping
!GANGA=CIFSDOM\GANGES

; An example for mapping all the users connecting to CIFS to a single OpenVMS username
cifs$default=*

**NOTE:**

1. By default, the Samba configuration file parameter “username map” points to the CIFS supplied username map file, SAMBA$ROOT:[LIB]USERNAME.MAP
2. If you create your own “username map” file, ensure that it is in Stream or Stream_LF record format. Then you must update the Samba Configuration file parameter “username map” to point to your own username map file.
3. Do not use “cifs$default=*” unless you want to grant same File Security permissions to all the users connecting to CIFS.

**Group Mapping**

Group mapping in CIFS involves mapping domain global groups belonging to the domain where CIFS is a member, the domains trusted by the domain and the CIFS server, to OpenVMS resource identifiers.

The only supported mechanism for mapping global groups belonging to the domain where CIFS is a member and the domains trusted by it, to OpenVMS resource identifiers is the automatic mapping provided by WINBIND. This will be explained in detail under the section ‘The role of WINBIND in CIFS File Security’.

© Copyright 2010 Hewlett-Packard Development Company, L.P

A CIFS local group can be explicitly mapped to an OpenVMS resource identifier using the command “NET GROUPMAP ADD”. This command automatically creates the CIFS local group as well as the required mapping with the OpenVMS resource identifier. The CIFS group mapping database file, SAMBA$ROOT:[VAR.LOCKS]GROUP_MAPPING.TDB stores the information about CIFS local groups, members of CIFS local groups and the mapping between CIFS local groups and OpenVMS resource identifiers.

## Summary

To summarize, CIFS local users and groups cannot be automatically mapped by WINBIND. Automatic mapping of users and groups by WINBIND is applicable only to users and groups in the Windows domain (where CIFS is a member) and the domains trusted by it.

## The role of WINBIND in CIFS File Security

As mentioned earlier, WINBIND is a special feature of CIFS that provides the following functionalities:

1. Automatic Mapping - For domain users and groups, WINBIND automatically creates the corresponding OpenVMS users and groups (resource identifier) if a mapping for the same does not exist.
2. Nested Group Support – Using nested groups, domain global groups can be added to CIFS local groups (thus, a group-within-a-group, or "nested" groups). Nested groups are defined locally on any machine and can contain users and global groups from the domain (where CIFS is a member) and the domains trusted by it.
3. Trusts - WINBIND is required for all Trust functionality when CIFS is a PDC.

The winbind functionality provided by CIFS is controlled though the logical, WINBINDD_DONT_ENV. If it is disabled or not defined on a CIFS node, CIFS provides WINBIND support and if it is enabled by defining it to 1, CIFS turns off the WINBIND support. As this logical is not defined by default, CIFS provides WINBIND support by default. When CIFS is a Member Server, automatic mapping and nested group support provided by WINBIND play an important role in CIFS File Security. Due to this, it is recommended not to disable WINBIND support on a CIFS Member Server by defining the logical WINBINDD_DONT_ENV.

Though CIFS provides WINBIND support by default, the automatic mapping feature provided by WINBIND is enabled, only if you have specified a valid “idmap UID” and “idmap GID” range in the Samba Configuration File. The next topic ‘Automatic Mapping’ describes this in detail and the importance of automatic mapping in CIFS File Security.

© Copyright 2010 Hewlett-Packard Development Company, L.P

**Automatic mapping:**

One of the purposes of WINBIND is to automate the creation of OpenVMS UICs and resource identifiers (in POSIX GID format) and maintain their correspondence to the appropriate Windows SIDs to minimize identity management efforts. The WINBIND identity mapping database file, SAMBA$ROOT:[VAR.LOCKS]WINBINDD_IDMAP.TDB maintains the mapping between Windows SIDs and OpenVMS UICs and resource identifiers. The mapping between a Windows SID and an OpenVMS username or a resource identifier is created only if there is no existing mapping entry in this database file. If the required mapping is missing, the automatic creation and mapping of OpenVMS usernames and groups (resource identifiers) occurs under the following two circumstances:

1. After the user is successfully authenticated, CIFS tries to map the authenticated user and all the groups that the user belongs, to an OpenVMS username (UIC) and groups (resource identifiers). The domain or CIFS groups can be either nested groups or the groups that the authenticated user directly belongs. If a mapping is found missing for an authenticated user and if this user is also a domain user, WINBIND can automatically create the required OpenVMS username and then map this OpenVMS username to the authenticated user. Similarly, if the mapping is found missing for any of the domain global groups that the user belongs; WINBIND can create the required OpenVMS group (resource identifier) and map it to the domain group.
2. When setting security on files and directories in CIFS shares, if the security principal (subject) to whom you are granting the permission is a user or global group in the Windows domain (where CIFS is a member) or the domains trusted by it, WINBIND can create and map the required OpenVMS username (UIC) or resource identifier to a domain user or group SID.

When an OpenVMS username is created by WINBIND, it specifies a UIC for that username. Similarly, when it creates an OpenVMS resource identifier, it creates the identifier in POSIX group identifier format by supplying a GID value. In order for WINBIND to use the UIC and GID values specified by you while creating OpenVMS usernames and resource identifiers, WINBIND provides a mechanism, which allows you to explicitly specify a set of values for UICs and GIDs that are allocated solely to WINBIND. WINBIND obtains the user IDs (UIDs) used to assign a UIC value to an OpenVMS username, and group IDs (GIDs) used to assign a value to the POSIX group identifier (resource identifier) from the Samba configuration file global parameters "idmap UID" and "idmap GID". These parameters must be set to a range of values allocated solely to WINBIND. The next two sections explain the steps used by CIFS to create OpenVMS usernames and resource identifiers using “idmap UID” and “idmap GID” range of values.

© Copyright 2010 Hewlett-Packard Development Company, L.P

**How does WINBIND map domain users to OpenVMS users?**

WINBIND uses the chosen integer "idmap UID" value, to derive both the OpenVMS username and the UIC. The UID value is converted to a hexadecimal value and appended to the string "CIFS$" to derive the OpenVMS username. The UID value is converted to octal and the octal value is used as the UIC group and member number.

NOTE: Since UIC group numbers are limited to a maximum value of Octal 37776 (decimal 16382), the upper range limit on the "idmap UID value is 16382. Similarly, because UIC group numbers below Octal 376 are reserved for use by HP, it is recommended not to specify a value below 255 as the lower range of "idmap UID".

For example, if the Samba configuration file contains:
idmap uid = 1000-2000

WINBIND will initially allocate UID 1000 and create an OpenVMS user named CIFS$3E8 with a UIC of [1750,1750]. The username CIFS$3E8 is created with interactive login disabled, nodisuser flag enabled and with NETMBX and TMPMBX privileges only. After the user CIFS$3E8 is successfully created in SYSUAF, the mapping of UID 1000 (for user CIFS$3E8) to a corresponding domain user SID is then stored in the file, SAMBA$ROOT:[VAR.LOCKS]WINBINDD_IDMAP.TDB. This file must be backed up regularly to avoid the loss of the required mappings necessary to maintain file security.

**How does WINBIND map domain groups to OpenVMS resource identifiers?**

WINBIND uses the chosen integer "idmap GID" value, to derive both the OpenVMS resource identifier (group) name and group identifier (GID) value. The GID value is converted to a hexadecimal value and appended to the string "CIFS$GRP" to derive the OpenVMS resource identifier name.

NOTE: Because WINBIND creates POSIX Group Resource Identifiers (POSIX GID), the maximum value is limited to %xFFFFFF or %d16777215. The lower limit is 1. OpenVMS automatically adds %xA4000000 to the value chosen. When CIFS is installed on an OpenVMS system, by default it automatically uses the GID values %xFFFFF0 thru %xFFFFFF and thus the highest GID value that can be specified is limited to %xFFFFFEF or %d16777199.

For example, if the SMB.CONF file contains:
idmap gid = 5000-10000

WINBIND will initially allocate GID 5000 and create an OpenVMS resource identifier named CIFS$GRP1388. After the resource identifier CIFS$GRP1388 is successfully created in the RIGHTSLIST database on OpenVMS, the mapping of GID 5000 (for group CIFS$GRP1388) to a corresponding domain group SID is then stored in the file, SAMBA$ROOT:[VAR.LOCKS]WINBINDD_IDMAP.TDB. It is critical that this file is

© Copyright 2010 Hewlett-Packard Development Company, L.P

backed up regularly as its loss will result in loss of the required mappings necessary to maintain security.

NOTE:

1. In an existing CIFS configuration, if you have to increase the “idmap uid” or “idmap gid” range values, retain the lower value of the range as it is while increasing only the higher value in the range. For example, in the “idmap uid” range specified above, to increase “idmap uid” range, specify the new range as “1000-3000”. WINBIND will automatically adjust the existing “idmap uid” range while retaining the current mapping entries in the WINBINDD identity mapping database file. This guarantees that the existing security on files and directories that might have been set based on the existing mapping entries are still valid.
2. HP OpenVMS CIFS Administrator’s Guide contains the flow charts on how the user and group mapping occurs in CIFS.
3. WINBIND automatic mapping feature is disabled by default as CIFS does not provide default values for “idmap uid” and “idmap gid” parameters. If WINBIND is enabled, automatic mapping feature is enabled once the valid “idmap uid” and “idmap gid” range values are specified in the Samba configuration file.
4. From CIFS V1.2 onwards, “idmap uid” and “idmap gid” ranges can be specified through Samba configuration utility.

**Domain users and groups mapped by WINBIND:**

WINBIND does not map all the users and global groups in the domain (where CIFS is a member) or in the domains trusted by it, even if automatic mapping is enabled. Instead it maps only the following domain users and groups:

1. Users belonging to the Windows domain (where CIFS is a member) or the domains trusted by it, who successfully established connection to CIFS Server at least once.
2. The domain global groups in the Windows domain where CIFS is a member or the domains trusted by it that contain the successfully authenticated users as members.
3. Users and groups belonging to the Windows domain where CIFS is a member and the domains trusted by it, to whom the CIFS administrator explicitly granted permissions for any Share, File or directory on CIFS.

**Tracking and managing users and groups created by WINBIND**

As explained above, WINBIND uses the range of values specified for “idmap uid” and “idmap gid” parameters in the Samba configuration file while creating OpenVMS usernames and resource identifiers. When WINBIND runs out of either of these idmap ranges that are specified in the Samba configuration file, it will report errors in the CIFS client log files. The client’s log files will be created in the directory SAMBA$ROOT:[VAR].

© Copyright 2010 Hewlett-Packard Development Company, L.P

CIFS provides a utility called WBINFO that can be used for viewing the OpenVMS usernames and resource identifiers created by WINBIND and the corresponding mapping to domain users and groups. You can execute the WBINFO utility as:

```
$ @SAMBA$ROOT:[BIN]SAMBA$DEFINE_COMMANDS.COM
$ WBINFO --hostusers-to-domainusers
$ WBINFO --hostgroups-to-domaingroups
```

The option "--hostusers-to-domainusers", displays the domain/CIFS users along with the mapped OpenVMS usernames. The option "--hostgroups-to-domaingroups", displays the domain/CIFS groups alongside the mapped OpenVMS resource identifiers. These two options are meant only for viewing the mapped users and groups and it cannot be used for enumerating the users and groups either in SYSUAF database on OpenVMS or in the domain/CIFS databases.

To check for the mapping between a single OpenVMS username or resource identifier (group) and a domain user or group, execute the command:

```
$ WBINFO --hostname-to-domainname=<OpenVMS username or resource identifier name>
```

**When is automatic mapping provided by WINBIND required?**

CIFS supports domain group to OpenVMS resource identifier mapping only through WINBIND. If you plan to set permissions on files and directories based on domain groups, automatic mapping provided by WINBIND is a must. Additionally, because WINBIND automates the OpenVMS username and resource identifier creation and the corresponding mapping with Windows SIDs, it can be used to avoid the manual administrative work related to identity management.

**When is automatic mapping provided by WINBIND not required?**

Automatic mapping provided by WINBIND need not be used if the following two conditions are satisfied:

1. All the domain users connecting to CIFS Server are either explicitly or implicitly mapped to the OpenVMS usernames.
2. Security Permissions on CIFS shares/files/directories are not based on domain global groups.

In this case, if you want to set permissions for domain global groups, then you must add them as members to CIFS local groups. Then, security on Files/Folders/Shares in CIFS can be set based on CIFS local groups. By virtue of nested group support provided by WINBIND, the security specified on an object for CIFS local groups that contain the domain global groups is automatically applicable to these domain global groups and the users belonging to them. The topic 'Managing CIFS local groups' provides examples on how to add domain users and domain global groups as members to CIFS local groups.

© Copyright 2010 Hewlett-Packard Development Company, L.P

**Nested group support:**

As already mentioned, group within a group is referred to as nested grouping. Using the nested group support provided by WINBIND, you can add the following users and groups as members to CIFS local groups:

1. Users and domain global groups that belong to the Windows domain where CIFS is a member.
2. Users and domain global groups that belong to the domains trusted by the Windows domain where CIFS is a member.
3. Users and local groups in CIFS server database

As explained under "When is automatic mapping provided by WINBIND not required?" topic, Nested group functionality allows you to setup File Security based on CIFS local groups. WINBIND supports nested groups even when automatic mapping feature provided by it is disabled.

## User Persona Creation

The mapping of domain/CIFS user and group names to OpenVMS UICs and resource identifiers occurs after the user connecting from a client system to CIFS server is successfully authenticated. After the user is authenticated, CIFS creates a persona for the user (User's Security Profile) that will be used by the SMBD process to access any object in CIFS server on behalf of the user. The persona for the authenticated user is internally created for the mapped OpenVMS username by the SMBD process.

The persona for the mapped OpenVMS user is made up of the following:

1. Mapped OpenVMS user's UIC and resource identifiers. These resource identifiers are the successfully mapped identifiers for the domain/CIFS groups that the authenticated user belongs.
2. Default privileges of the mapped OpenVMS username in SYSUAF.
3. Any general identifiers that were explicitly granted by the OpenVMS administrator for the mapped OpenVMS username in SYSUAF.

When an authenticated domain/CIFS user tries to access an object in CIFS share, OpenVMS grants access to the object based on the persona of the mapped OpenVMS user mentioned earlier. The only exception to this is, when the authenticated user is part of "admin users" in CIFS. When a user belongs to "admin users" in CIFS, the authenticated user's persona is same as that of the fully privileged user's persona that was originally created by the SMBD process for authenticating the domain/CIFS user. By default, the SMBD process authenticates the domain/CIFS users using the persona of SAMBA$SMBD account.

NOTE:

1. CIFS mandatorily requires that an identifier with the same UIC value as the UIC of the mapped OpenVMS username is present in the RIGHTSLIST database. For example, for a mapped OpenVMS user named, STEFFI with a UIC of [600,600], there must be a corresponding identifier with a UIC of [600,600].

© Copyright 2010 Hewlett-Packard Development Company, L.P

2. Even though CIFS provides user mapping, a user connecting to the CIFS Server cannot be authenticated based on the credentials of the mapped OpenVMS username in SYSUAF.
3. SAMBA$SMBD account is created by CIFS as part of CIFS installation.

## Windows to OpenVMS Permission Mapping provided by CIFS

The earlier sections explained the first part of CIFS File Security "User and Group Mapping". The following sections explain the Windows permissions to OpenVMS permissions mapping provided by CIFS.

**Windows to OpenVMS Permission Mapping**

OpenVMS allows you to specify READ (R), WRITE (W), EXECUTE (E), DELETE (D) or CONTROL (C) access permission on an object or a combination of RWEDC permissions (Refer note). On Windows, you can specify 'Full Control, Modify, Read and Execute, List Folders Contents (for directories only), Read, Write' standard permissions on an object using the 'Security' dialog box. Using the 'Advanced Security Setting' dialog box, Windows provides special permission setting. "Table 1 - Windows Permissions to OpenVMS permissions mapping" gives the list of Windows permissions that are mapped to corresponding OpenVMS permissions by CIFS. The "Advanced" in brackets indicates that this permission is available on Windows, only under the 'Advanced Security Setting' dialog box. 'Full Control' and 'Read' permissions are part of standard as well as special permissions.

Table 1 – Windows Permissions to OpenVMS Permission Mapping

| **Windows Permissions** | **OpenVMS Permissions** |
|---|---|
| Full Control | RWEDC |
| Modify | RWED |
| Read and Execute | RE |
| Read | R |
| Write | W |
| Full Control (*Advanced*) | RWEDC |
| Traverse Folder / Execute File (*Advanced*) | E |
| List Folder / Read Data (*Advanced*) | R |
| Read Attributes (*Advanced*) | R |
| Read Extended Attributes (*Advanced*) | R |
| Create Files / Write Data (*Advanced*) | W |
| Create Folder / Append Data *(Advanced)* | W |
| Write Attributes *(Advanced)* | W |
| Write Extended Attributes *(Advanced)* | W |
| Delete Subfolders and Files (*Advanced*) | Not supported |
| Delete (*Advanced*) | D |

© Copyright 2010 Hewlett-Packard Development Company, L.P

| | |
|---|---|
| Read Permissions (*Advanced*) | R |
| Change Permissions (*Advanced*) | C |
| Take Ownership (*Advanced*) | C |

NOTE: Though the access permission, ACCESS=NONE specified on an object on OpenVMS is honoured by CIFS as it depends on OpenVMS to grant access, CIFS does not support setting this access permission on an object from a Windows system.

**Windows Inheritance Value to OpenVMS inheritance mapping**

On a Windows system, when setting permissions on a directory, you can specify if the access is applicable only to that directory or to the subfolders and files that will be created under it or to a combination of these.

Similarly, OpenVMS provides DEFAULT ACLs for the directories that are applicable only to the files and directories created under it while the access to the directory is controlled by the access ACE on the directory. “Table 2 – Windows inheritance value to OpenVMS inheritance mapping” provides information about Windows to OpenVMS inheritance mapping provided by CIFS.

Table 2 - Windows inheritance value to OpenVMS inheritance mapping

| **Windows Inheritance Value** | **VMS Inheritance mapping** |
|---|---|
| This Folder only | Maps to access ACE |
| This Folder, Subfolders and Files | An ACE of this type is mapped to both access and default ACE. |
| This Folder and Subfolders | Maps *only* to access ACE for this directory. |
| This Folder and Files | Maps *only* to access ACE for this directory. |
| Subfolders and Files only | Maps to default ACE for this directory. |
| Subfolders only | This type is not supported and any ACE with this type is ignored by the CIFS. |
| Files only | This type is not supported and any ACE with this type is ignored by the CIFS. |

**Mapping OpenVMS RMS protection code to Windows Permissions**

By default, OpenVMS automatically sets the protection code (RMS protection code) on each object when it is created. Though the ACLs on an object are optional, the protection code is a must. The syntax of the protection code consists of permissions for SYSTEM,

© Copyright 2010 Hewlett-Packard Development Company, L.P

OWNER, GROUP, and WORLD categories. Optionally, OpenVMS lets you specify inheritable DEFAULT_PROTECTION ACE on the directories that consists of protection code for these categories. The protection code specified in this ACE will be applied to the newly created files within the directory and the newly created directories within the parent directory will inherit this ACE.

On a Windows system, it has OWNER category to indicate the owner of the object and Everyone group that contains all the users on the Windows system. CREATOR OWNER and CREATOR GROUP present on the parent directory specify the permissions that will be inherited by the child objects for OWNER and the primary group of the owner respectively.

"Table 3 – OpenVMS RMS protection code category mapping to Windows mapping" shows the mapping provided by CIFS for OpenVMS protection code category to corresponding Windows mapping.

Table 3 - OpenVMS RMS protection code category mapping to Windows mapping

| **RMS Protection code category** | **Windows mapping** |
|---|---|
| Owner/SYSTEM | Owner |
| Group | (displayed as) Unix Group |
| World | Everyone |
| Owner of default protection ACE | CREATOR OWNER on the directory |
| Group of default protection ACE | CREATOR GROUP on the directory |
| World of default protection ACE | Everyone for Subfolder and Files |

## Controlling OpenVMS Protection code using SMB.CONF parameters

CIFS provides various Samba Configuration File (SMB.CONF) parameters that allow you to control OpenVMS RMS protection code setting when:

1. The directories and files are newly created
2. Security permissions are explicitly set on the directories.

From CIFS patch set PS009 for CIFS V1.1 ECO1 onwards, the way these SMB.CONF parameters control OpenVMS RMS protection code have been simplified. Due to this, the implementation of many SMB.CONF parameters related to File Security is quite different on CIFS for OpenVMS when compared to Open Source versions of Samba (which are primarily based on the UNIX security model).

With this simplification, CIFS now allows OpenVMS to determine security using standard security rules when new files and directories are created. CIFS then adjusts the RMS protection code and owner based on various SMB.CONF parameters; however, the

© Copyright 2010 Hewlett-Packard Development Company, L.P

defaults for these parameters are such that they will not modify the security that OpenVMS would apply (with exceptions noted).

The following SMB.CONF parameters can be used to modify security applied by OpenVMS:

1. inherit owner - The default value is "no". If set to "yes", causes CIFS to set the RMS owner of new objects to that of the parent directory.

NOTE: If "inherit owner = no" and a parent directory is owned by a Resource Identifier, when a non-privileged user who has WRITE access to this directory, creates a new file, CIFS sets the Owner to the UIC of the user creating the file, rather than the Resource Identifier. Thus, in order to retain the OpenVMS behavior (i.e., to set the resource identifier as owner), add "inherit owner = yes" to the applicable [share] sections of the SMB.CONF file. This will be addressed in a future release.

2. inherit vms rms protections - This is a new parameter with a default value of "no". If set to "yes", causes CIFS to:
    a. Set the RMS protection code to that of parent directory.
    b. Ignore a DEFAULT_PROTECTION ACE, if present.
    c. Ignore the RMS protection mask specified by the SYSGEN parameter RMS_FILEPROT.
    d. Ignore the mask and mode parameter values specified in the SMB.CONF file.

3. “Table 4 – mask and mode parameters” provides the list of mask and mode parameters supported by CIFS. The value of the appropriate "mask" parameter is AND'd with the result of the RMS protection code that OpenVMS would apply. This result is then OR'd with the value of the appropriate mode parameter:

Table 4 – mask and mode parameters

| **Parameter Name** | **Affects RMS protection code when** |
|---|---|
| create mask | Creating new files |
| force create mode | Creating new files |
| directory mask | Creating new directories |
| force directory mode | Creating new directories |
| directory security mask | Windows users modify security on the directory |
| force directory security mode | Windows users modify security on the directory |

© Copyright 2010 Hewlett-Packard Development Company, L.P

NOTE:

i. CIFS for OpenVMS does not use the SMB.CONF parameters "security mask" and "force security mode" when a Windows user modifies the security of a file.
ii. As "create mode" parameter is same as "create mask" parameter, if required, use "create mask" parameter instead of "create mode".
iii. The following mask and mode parameter are related for doing AND & OR operation to generate a resultant RMS protection code:
    a. "create mask" and "force create mode"
    b. "directory mask" and "force directory mode"
    c. "directory security mask" and "force directory security mode"
iv. By default, the value of the mask parameters is 07777 and that of mode parameters is 00000 with the only exception of “force directory mode”. From CIFS version 1.2 onwards, the default value for “force directory mode” is 04000. This is to allow DELETE permission for OWNER category of the protection code when a new directory is created.

**SMB.CONF parameters that cannot be modified**

The following SMB.CONF parameters must not be modified from their default values and if you modify the value to a non-default value, it is not supported by CIFS:

1. inherit acls - Default is "yes"; you cannot disable inheritance of ACLs
2. inherit permissions - Replaced by the "inherit vms rms protections" parameter
3. security mask - Not supported
4. force security mode - Not supported

**Delete access support for RMS protection code when using mask and mode parameters**

One of the significant changes introduced with the release of patch set PS006 for CIFS V1.1 ECO1 concerns granting DELETE access in the RMS protection code on new objects. Previously, the various mask and mode parameters tied DELETE access to the WRITE bit; i.e. if you enabled WRITE access you also enabled DELETE access. However, with the release of PS006, DELETE and WRITE protections have been separated as documented below.

In addition, with the release of patch set PS009 for CIFS V1.1 ECO1, the default values for the mask and mode parameters have been changed such that they will not adjust the security that OpenVMS would itself apply (except where noted).

© Copyright 2010 Hewlett-Packard Development Company, L.P

The values for the mask and mode parameters now have this significance:

<mask or mode parameter name> = 0dogw

Where:

'0' = Indicates the value is Octal
'd' = Controls granting DELETE access across all categories of the RMS protection code (see below)
'o' = Controls granting READ, WRITE, and EXECUTE access for the Owner category of the RMS protection code
'g' = Controls granting READ, WRITE, and EXECUTE access for the Group category of the RMS protection code
'w' = Controls granting READ, WRITE, and EXECUTE access for the World category of the RMS protection code

NOTE: The System category of the RMS protection code receives the same permission as the Owner category; there is no option to modify this behavior.

DELETE access is signified using a bitmask with the following values:

4 = Grant DELETE access to Owner category of RMS protection code
2 = Grant DELETE access to Group category of RMS protection code
1 = Grant DELETE access to World category of RMS protection code

The Owner, Group, and World access values are also bitmasks which signify the following access:

4 = Grant READ access
2 = Grant WRITE access
1 = Grant EXECUTE access

## ACL order while applying CIFS File Security

In OpenVMS File Security section, it showed that the order in which the ACLs are applied on the files and directories is very important on an OpenVMS system while the same does not hold for Windows systems.

Due to the contrasting nature of Windows and OpenVMS systems' ACL processing, it is important to understand the order in which the ACLs are applied by CIFS when setting security on an object. When a CIFS administrator sets security permission on an object present in the CIFS share from a Windows system, CIFS first converts these Windows permissions to OpenVMS permissions. If the security was applied for users and groups, the converted OpenVMS permissions will contain OpenVMS ACLs for the mapped OpenVMS usernames and resource identifiers. After this conversion, CIFS applies the OpenVMS ACLs on the object. While applying OpenVMS ACLs on an object, CIFS must preserve the existing OpenVMS specific ACLs that would have been added by an

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS administrator and these ACLs should ideally be retained in their original order (Refer note for information about OpenVMS specific ACEs). For example, on an OpenVMS system, users who fail to have access to a file based on any of the top order ACEs could be granted READ access by using an ACE, (IDENTIFIER=*,ACCESS=READ). Typically, an ACE similar to this would be present at the bottom of ACLs. CIFS, when setting permission, needs to ensure that this ACE is almost always present at the bottom of the ACLs on an object. Due to this reason, following design is implemented in CIFS when applying security on an object from a Windows system:

1. When a new ACE for a user or group is applied on an object in a CIFS share from a Windows system, after converting the Windows ACE to OpenVMS ACE format, CIFS applies this ACE at the top of the OpenVMS ACL order on an object.
2. While modifying an existing ACE for a user or group on an object in a CIFS share from a Windows system, after converting the Windows ACE to OpenVMS ACE format, CIFS finds out the location of the existing ACE that is getting modified. Then, it replaces the existing ACE with a modified ACE entry at the same location.
3. All the OpenVMS specific ACEs like AUDIT, ALARM, IDENTIFIER=*, PROTECTED and HIDDEN are retained in their existing locations.

NOTE: OpenVMS specific ACEs like AUDIT, ALARM, IDENTIFIER=*, PROTECTED, and HIDDEN that exist on an object in a CIFS share cannot be viewed or modified from a Windows system.

## Limitations due to File Security Mapping

Until now, the article explained how the Windows File Security is mapped to OpenVMS File Security. This mapping mechanism is not without its limitations as Windows and OpenVMS systems vastly differ in the way they process ACLs on an object to grant access to the object. The limitations due to File Security mapping provided by CIFS are in the following areas:

1. Object access
2. File permission setting
3. Built-in Administrators group
4. Windows inheritance value mapping
5. Windows permission mapping
6. Viewing permissions on a directory from Windows

**Object access limitation:**

As mentioned in Windows File Security, Windows systems allow access to an object based on the accumulated access permissions for an object unless the user has special rights. Refer section ‘Windows File Security’ to understand how Windows derives accumulated access permissions for an object. Thus the order in which the ACLs appear on an object has no importance on Windows systems.

© Copyright 2010 Hewlett-Packard Development Company, L.P

On OpenVMS systems, the order in which the ACLs appear on an object is very important. This is because OpenVMS grants access to an object based on the first matching ACE that it encounters from the top of ACE list unless the user has special privileges or is an owner of the file.

When a user tries to access an object in a CIFS share from a client system, CIFS lets the OpenVMS system to decide access to the user based on the persona of the mapped OpenVMS user that was created by CIFS. According to the OpenVMS security rules, if the user is not an owner of the object and has no special privileges, user connecting to CIFS would be granted access to an object based on the first matching ACE encountered by OpenVMS system on that object. This can lead to access permission limitation in the following scenario:
A user ANITA belongs to two domain global groups FINANCE and ACCOUNTS. On an object, an ACE corresponding to one of the domain global groups FINANCE is granted READ access and is above the ACE corresponding to another domain global group ACCOUNTS which has full permissions (RWEDC). When a user ANITA that belongs to these 2 domain global groups access this object on a CIFS share, OpenVMS grants read only access to this user for the object based on the first matching ACE. On this object, the first matching ACE for the user ANITA is for the domain global group FINANCE. Thus, the ACE corresponding to the domain global group ACCOUNTS which has higher access permissions is not exercised for the user ANITA when the object is accessed by this user. This is an architectural limitation.

The workaround to this problem is to set security on an object in a CIFS share path from an OpenVMS system. While doing so, it must be ensured that the ACEs with highest access permissions are at the top of ACL order and the ACEs with lowest access permissions are at the bottom of ACL order.

**File Permission setting limitation:**

CIFS does not support exclusive permission setting on a file, though the same is supported for directories/folders. For example, in a directory with 100 files and folders under it, one user may have READ access to this directory and the files and folders under it. Just for a single file in this directory, you may want to grant modify (RWED) access for this user. This is not supported by CIFS with one exception. The exception to this problem is, either you make this user as OWNER of the file or grant ‘Change Permission’ (CONTROL) access to the file for this user. You may want to use this workaround only if you want this user to have control access to this file and not otherwise.

CIFS does not support exclusively specified OpenVMS specific ACEs (like PROTECTED, HIDDEN etc) on a file though the same is supported for directories. For example, in a directory with 100 files and folders, you might have exclusively set an AUDIT ACE on a particular file in this directory. A user with modify (RWED) access to this file, opens it, then saves the modified file and closes it. The AUDIT ACE that was exclusively set on this file might have been lost after the file was closed. This is

© Copyright 2010 Hewlett-Packard Development Company, L.P

particularly applicable for Microsoft office application files. The only workaround to this problem is to apply an inheritable default ACEs on the parent directory. This will lead to all the files and sub directories under the parent directory inheriting the ACE.

**Built-in Administrators group limitation:**

Prior to CIFS patch set PS005 for CIFS V1.1 ECO1, all the members of CIFS built-in (local) Administrators group were granted the two special OpenVMS privileges, BYPASS and SYSPRV. This allowed full administrative privileges to users belonging to built-in 'Administrators' group. By virtue of nested grouping, if the built-in 'Administrators' contained any other CIFS local groups or domain global groups (like 'Domain Admins'), the users belonging to these nested groups were also granted the BYPASS and SYSPRV privileges. Due to this, these users were automatically entitled to perform CIFS Administrative work like File Security setting, user and group management and so on. From CIFS patch set PS005 for CIFS V1.1 ECO1 onwards, members of built-in Administrators group are no longer granted BYPASS and SYSPRV privilege. As a result, though the members of built-in 'Administrators' group can perform rest of the administrative work, they cannot set security on files and folders. Next section describes how the members belonging to built-in 'Administrators' group can be granted permission on files and directories for managing security on the same.

**Windows inheritance value mapping limitation**

When setting permissions on a directory in CIFS share from a Windows system, Windows provides multiple options on how the permissions can be applied on a directory. CIFS does not support 'Subfolder only' and 'Files only' Windows inheritance value options. Refer to "Table 2 - Windows inheritance value to OpenVMS inheritance mapping" for interpretation of rest of the Windows inheritance value options and their mapping to OpenVMS ACEs.

**Windows permissions mapping limitation**

CIFS does not support the special permission 'Delete Subfolder and Files' options that is available from the permissions list in the 'Advanced Security Setting' dialog box. If you try to set permission for this option, you are likely to encounter "Access denied" error.

**Limitation in viewing permissions on a directory from Windows**

From Windows, when viewing permissions on a CIFS directory or share using the 'Security' dialog box, you will see empty permissions for users and groups. This does not correctly reflect the permissions existing for the users and groups on that directory or share. In order to correctly view permissions on a CIFS directory or share, always use 'Advanced Security Setting' dialog box. In the 'Security' dialog box, click on 'Advanced' tab to go to 'Advanced Security Setting' dialog box.

© Copyright 2010 Hewlett-Packard Development Company, L.P

On the other hand, you can correctly view permissions on files in a CIFS folder using the 'Security' dialog box itself.

## Permissions and Privileges required by a user for setting up CIFS File Security

A user who tries to setup File Security on an object in a CIFS share must have certain privileges or permissions to successfully set up File Security. This section describes the permissions and privileges that allow a user to set up CIFS File Security.

**Permissions for users belonging to built-in Administrators group:**

The topic 'Built-in Administrators group limitation' mentioned the reasons why a user who belongs to built-in Administrators either directly or through nested grouping, can no longer set File Security or access files and directories in CIFS shares. CIFS provides an alternative option such that all the users belonging to the built-in Administrators group either directly or through nested grouping, can still manage File Security on files and directories in a CIFS share. To use this option, you should grant READ access to the files and directories in CIFS shares for the OpenVMS resource identifier, CIFS$ADMINISTRATORS. You can apply this identifier either for all the files and directories under all the CIFS share paths or only to a selected files and directories under certain share paths. Once the resource identifier CIFS$ADMINISTRATORS is applied on an object with READ access, while setting permissions from a Windows system, a user belonging to built-in Administrators group (either directly or through nested grouping) can take ownership of the object. After the user becomes an owner of the object, the user is allowed to set required permissions for other users and groups on that object.

The READ access to the OpenVMS resource identifier, CIFS$ADMINISTRATORS can be granted by executing the commands similar to the following on an OpenVMS system:

```
$ SET DEFAULT [CIFSSHARE]
```

To apply inheritable default ACE to directories under CIFSSHARE.DIR, execute:

```
$ SET SECURITY/ACL=
(IDENTIFIER=CIFS$ADMINISTRATORS,OPTIONS=DEFAULT,ACCESS=READ) –
_$  [...]*.DIR
```

To apply access ACE for files and directories under CIFSSHARE.DIR, execute:

```
$ SET SECURITY/ACL= (IDENTIFIER=CIFS$ADMINISTRATORS,ACCESS=READ) [...]*.*;*
```

If you would like to grant CIFS$ADMINISTRATORS resource identifier to the parent directory CIFSSHARE.DIR, execute:

© Copyright 2010 Hewlett-Packard Development Company, L.P

Applying default ACE for inheritance:

```
$ SET SECURITY/ACL=
(IDENTIFIER=CIFS$ADMINISTRATORS,OPTIONS=DEFAULT,ACCESS=READ) –
_$ [-]CIFSSHARE.DIR
```

Applying access ACE:

```
$ SET SECURITY/ACL= (IDENTIFIER=CIFS$ADMINISTRATORS,ACCESS=READ) [-
]CIFSSHARE.DIR
```

NOTE:

1. CIFS, by default creates CIFS$ADMINISTRATORS OpenVMS resource identifier after CIFS has been successfully configured and connection was established to it at least once.
2. You can grant any other access permission to CIFS$ADMINISTRATORS in addition to READ depending upon the security setup in your CIFS environment.

**Full privileges to selected domain users and CIFS local users:**

For granting full administrative privileges to a selected list of domain users and CIFS local users, you can use the SMB.CONF parameter, “admin users”. By administrative privilege, it is meant full privileges that can be granted to a user on an OpenVMS system. You can follow the steps mentioned below for granting administrative privilege to a selected list of users.

On a CIFS member server, to grant an administrative privilege to a user belonging to the Windows domain (where CIFS is a member) or the domains trusted by it, the format of the “admin users” parameter in the Samba configuration file is:

admin users = <DOMAINNAME\domain-user>

On CIFS a member server, to grant an administrative privilege to a CIFS local user, the format of the “admin users” parameter in the Samba configuration file is:

admin users = <CIFS local username>

NOTE:

1. You can specify multiple usernames in the same line. When specifying multiple usernames, separate each username by a comma.
2. Domain usernames and CIFS local usernames can be specified in the same line. For example, domain users and CIFS local users can be specified in the same line as:

   admin users = <DOMAINNAME\domain-user>, <CIFS local username>

3. Prior to CIFS version 1.2, the “admin users” parameter can be added in the [global] section of the Samba configuration file SAMBA$ROOT:[LIB]SMB.CONF. From CIFS version 1.2 onwards, you can

© Copyright 2010 Hewlett-Packard Development Company, L.P

update the “admin users” parameter in the Samba configuration include file, SAMBA$ROOT:[LIB]ADMIN_USERS_SMB.CONF.

4. “admin users” parameter, if specified under the [global] section of Samba configuration file grants full privileges to all the CIFS shares and files and folders under these shares in addition to allowing these admin users the right to do other administrative operations
5. “admin users” parameter, if specified under the share section of the Samba configuration file, grants administrative privileges to the specified admin users only for that share.

**Granting privileges to a mapped OpenVMS username**

A domain or a CIFS local user whose account is mapped to an OpenVMS username will derive the same privileges as the mapped OpenVMS username. Due to this, a domain or a CIFS local user can become a privileged administrative user, provided you have granted the necessary privileges (like BYPASS, SYSPRV, GRPPRV) for the mapped OpenVMS username. The privileges mentioned here are the default privileges of a mapped OpenVMS username in SYSUAF database. CIFS does not honor authorized privileges of a mapped OpenVMS username.

You can use the following DCL command to grant privileges for a mapped OpenVMS user:

> $ MCR AUTHORIZE MODIFY <OpenVMS-username>/DEFPRIVILEGES = (<privileges>)

Privileges can be BYPASS, SYSPRV, and GRPPRV etc. Multiple privileges can be specified by separating them using a comma.

NOTE: Some of the utilities provided by CIFS honor current privileges of the terminal process that is executing the utilities on behalf of the user.

**Special permission – ‘Change Permission’ or CONTROL access**

A user, who has ‘Change Permission’ or CONTROL access to an object, can modify permissions on that object for other users and groups. The same is possible, if any of the groups that the user belongs to (either directly or through nested grouping), has ‘Change Permission’ or CONTROL access to an object. Again, this is subject to OpenVMS security rules and depends on the ACE location in the ACLs on the object unless the user is an owner of the object or has special privileges.

By granting a ‘Change Permission’ (CONTROL access) on an object for a user or to any of the groups that the user belongs to, you allow the user to set permissions on this object for any other users and groups.

© Copyright 2010 Hewlett-Packard Development Company, L.P

**Privileges/Permissions for a user when setting CIFS File Security from an OpenVMS system**

A user who tries to set up CIFS File Security on an object in a CIFS share path from an OpenVMS system must have full privileges (BYPASS privilege is a must) to set security.

## Steps for setting up CIFS File Security

This section describes the steps that are required for setting up File Security on CIFS shares and directories. For setting File Security on a share using the steps provided below, the share is treated as a directory. This section includes the following topics:

1. Managing CIFS local groups – Describes how to add users and groups to CIFS local groups
2. Setting up File Security from a Windows system – Describes how to set up File Security from a Windows system.
3. Setting up File Security from an OpenVMS system - Describes how to set up File Security from an OpenVMS system.

NOTE: Before you follow rest of the topics, you must ensure that you have successfully added CIFS as member server to a Windows domain as rest of the topics assume that a working CIFS configuration is already in place.

As the above mentioned topics will be based on practical examples in most cases, sample configuration information used in these examples is as follows:

Windows Domain name (where CIFS is a Member Server): CIFSDOM
CIFS Member Server name (OpenVMS system name): PIANO
Windows PDC emulator name for the domain CIFSDOM: ROX3

**Managing CIFS local groups**

This section describes the steps and commands that can be executed for managing CIFS local groups. Following users and groups can be added as members to a CIFS local group:

1. Users and global groups of the domain where CIFS is a member
2. Users and global groups of the domains trusted by the domain where CIFS is a member
3. CIFS local users and groups

**NOTE:** In an OpenVMS cluster, if multiple nodes in a cluster share the same samba$root installation directory where CIFS is configured as Member Server, use the CIFS cluster alias name for the “-W” option of NET RPC GROUP command instead of OpenVMS system name.

The steps for managing CIFS local groups are as follows:

© Copyright 2010 Hewlett-Packard Development Company, L.P

Step 1: Login to an OpenVMS system and define Samba commands

1. Login to an OpenVMS system (PIANO) where CIFS is configured, using a fully privileged OpenVMS user account (say, SYSTEM).

2. Define Samba commands by executing:

   $ @SAMBA$ROOT:[BIN]SAMBA$DEFINE_COMMANDS.COM

Step 2: Creating a privileged CIFS user

To execute the commands mentioned in 'Step 3' that let you manage CIFS local groups, you must first create a CIFS local user with an administrative privilege if no such user exists. To do this, execute the following commands:

1. Create an OpenVMS username say, CIFSADMIN:

   $ SHOW LOGICAL SYSUAF

   If it is not defined, execute:
   $ DEFINE SYSUAF SYS$SYSTEM:SYSUAF

   $ MCR AUTHORIZE COPY SAMBA$TMPLT CIFSADMIN/UIC=[600,600]/FLAG=NODISUSER

   NOTE: From CIFS version 1.2 onwards, CIFS by default creates CIFSADMIN account in SYSUAF.

2. Create a CIFS user account with the same name as the OpenVMS username created in previous step:

   $ PDBEDIT "-a" CIFSADMIN
   new password: any1willdo
   retype new password: any1willdo

3. Update "admin users" parameter
   a. For versions prior to CIFS V1.2, edit the Samba configuration file, SAMBA$ROOT:[LIB]SMB.CONF and add or update (append) the following parameter in the [global] section:
      admin users = cifsadmin
   b. From CIFS version 1.2 onwards, edit the Samba configuration include file, SAMBA$ROOT:[LIB]ADMIN_USERS_SMB.CONF and update (append) the following parameter:
      admin users = cifsadmin

© Copyright 2010 Hewlett-Packard Development Company, L.P

Step 3: Executing commands to add/modify/delete CIFS local groups

1. Adding or Creating a CIFS local group:

   a. Add a resource identifier say, CIFSUSERS in the RIGHTSLIST database by executing the following command:

```
$ MCR AUTHORIZE ADD/IDENTIFIER/ATTRIBUTE=RESOURCE CIFSUSERS
```

   b. To add/create a CIFS local group by mapping it to the resource identifier, CIFSUSER, execute:

```
$ NET GROUPMAP ADD NTGROUP=CIFSUSERS UNIXGROUP=CIFSUSERS TYPE="L"
```

2. To list the groups in CIFS local database, enter one of the following commands:

```
$ NET GROUPMAP LIST
                                    OR
$ NET RPC GROUP LIST "-W" PIANO "-S" PIANO "-U" CIFSADMIN%"Pwd of CIFSADMIN"
```

3. To add an already existing CIFS local user (STEFFI) or a local group (PLAYERS) to a CIFS local group (CIFSUSERS), enter the following commands:

```
$ NET RPC GROUP ADDMEM CIFSUSERS STEFFI "-W" PIANO "-S" PIANO -
_$ "-U" CIFSADMIN%"Pwd of CIFSADMIN"

$ NET RPC GROUP ADDMEM CIFSUSERS PLAYERS "-W" PIANO "-S" PIANO -
_$ "-U" CIFSADMIN%"Pwd of CIFSADMIN"
```

4. To a CIFS local group (CIFSUSERS), to add a domain user (ANITA) or a domain global group (ACCOUNTS) that belong to the domain CIFSDOM, enter the following commands:

```
$ NET RPC GROUP ADDMEM CIFSUSERS CIFSDOM\ANITA "-W" PIANO –
_$ "-S" PIANO "-U" CIFSADMIN%"Pwd of CIFSADMIN"

$ NET RPC GROUP ADDMEM CIFSUSERS CIFSDOM\ACCOUNTS "-W" PIANO –
_$ "-S" PIANO "-U" CIFSADMIN%"Pwd of CIFSADMIN"
```

5. To list members of a CIFS local group (CIFSUSERS), enter the following command:

```
$ NET RPC GROUP MEMBERS CIFSUSERS "-W" PIANO "-S" PIANO –
_$ "-U" CIFSADMIN%"Pwd of CIFSADMIN"
```

6. To delete a group from a CIFS local group (CIFSUSERS), enter the following command:

```
$ NET RPC GROUP DELMEM CIFSUSERS CIFSDOM\ACCOUNTS "-W" PIANO –
_$ "-S" PIANO "-U" CIFSADMIN%"Pwd of CIFSADMIN"
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

NOTE: The same command can be used to delete a CIFS local user or a group OR a domain user or a domain global group. To delete a CIFS local user or a group, specify only the name, i.e., without the 'DOMAINNAME\'

7. To delete a CIFS local group (CIFSUSERS), enter the following command:

```
$ NET RPC GROUP DELETE CIFSUSERS "-W" PIANO "-S" PIANO –
_$ "-U" CIFSADMIN%"Pwd of CIFSADMIN"
```

NOTE: You must first delete all the members from CIFS local group (CIFSUSERS) before deleting the group itself.

Additional Note:

In the above examples, to add/remove users and groups that belong to domains trusted by a Windows domain (CIFSDOM) to a CIFS local group, use the member name as 'TRUSTEDDOMAIN\\<USERNAME or GROUPNAME>

**Setting up File Security from a Windows system**

This section describes the steps that should be followed for setting up File Security on shares/folders from a Windows system. It is assumed that you will be using a user account that has the required permissions or privileges (described in earlier sections) to execute the following steps.

Step 1: Login to CIFS Server

From the Windows system, connect to CIFS server (PIANO) by specifying \\<CIFS server name or IP address> at the prompt, "Start->Run" or map the required CIFS share by right-clicking on the 'My Computer' icon on the desktop and then select 'Map Network drive' in the menu. This will take you to 'Map Network drive' dialog box. In the 'Map Network Drive' dialog box, enter shared folder name under 'Folder' drop–down and then click 'Finish'. If prompted for credentials, connect using a user account (say, ANITA) that has the required privileges/permissions to manage the security on the shared folder.

Step 2: Setting security

a. Select the shared folder or any folder within the shared folder that you would like to manage and right-click on it and select Properties->security->Advanced.
b. If the user account (ANITA) that you used for connecting to CIFS Server in step 1 (under this topic) is a member of built-in Administrators group (either directly or because of nested grouping) and you are managing the share/folder only because of the permissions that have been granted for the OpenVMS resource identifier CIFS$ADMINISTRATORS, then follow rest of the steps under 'b', otherwise go to step 'c'.
    i. Select 'Owner' tab in the 'Advanced Security Setting' dialog box.

© Copyright 2010 Hewlett-Packard Development Company, L.P

ii. When the ‘Owner’ dialog box is displayed, check if you are the owner of this object. If so, proceed to step ‘c’.
iii. When the ‘Owner’ dialog box is displayed, if the connected user account name is not displayed in ‘Change owner to’ list, click on ‘Other Users or Groups’. This will take you to ‘Select User, Computer, or Group’ dialog box.
iv. In the ‘Select User, Computer, or Group’ dialog box, if connected user (ANITA) does not belong to the location (domain) displayed, click on ‘Locations’ tab. In the ‘Locations’ list box, select the appropriate location where the connected user account (ANITA) is present and click ‘OK’.
v. In ‘Select User, Computer, or Group’ dialog box, enter the connected user name (ANITA) under the ‘Enter the object name to select (examples):’ and click on ‘Check Names’. If Windows finds multiple names for the name that you have entered, it will display ‘Multiple Names Found’ list box. In this box, select the appropriate name (RDN) and click ‘OK’. If ‘Multiple Names Found’ box is not displayed or once you return from the ‘Multiple Names Found’ box, click ‘OK’ in the ‘Select User, Computer, or Group’ dialog box.
vi. After returning to ‘Owner’ dialog box, select the connected user name (ANITA) under ‘Change owner to’ list box and click ‘Apply’ button. Once the owner name is set to connect user name (ANITA), select ‘Permissions’ tab. As you are now an owner of this object, proceed to step ‘c’ for setting permissions to other users and groups.

c. In the ‘Advanced Security Setting’ box, choose permissions tab (by default permissions tab is chosen). Click on “Add” tab if you would like add a new permission entry for a user or group, or to modify/remove an existing permission entry, select the user or group under the ‘Permission entries’.
d. If you want to remove the selected ‘Permission entry’, click on ‘Remove’ tab and then click on ‘Apply’ button. Now, go to step ‘i’.
e. If you want to modify the selected ‘Permission entry’, skip this step and go to step ‘f’. If you want add a new ‘Permission entry’, follow rest of the steps under this step.
    i. If you want to add a new permission entry, click on the ‘Add’ tab. Windows now displays ‘Select User, Computer, or Group’ dialog box.
    ii. In the ‘Select User, Computer, or Group’ dialog box, if a user or group name to whom you want to grant permissions belongs to the displayed location (domain), proceed to next step ‘iii’. If a user or group belongs to any other location (domain), click on ‘Locations’ tab; Select the appropriate ‘Location’ in the ‘Locations’ list box and click ‘OK’.
    iii. In ‘Select User, Computer, or Group’ dialog box, under ‘Enter the object name to select (examples)’, type the user or group name to whom you want to grant permission, Then, click on ‘Check Names’ tab. If Windows finds multiple names for the name that you have entered, it will display ‘Multiple Names Found’ list box. In this box, select the appropriate name (RDN) and click ‘OK’. If ‘Multiple Names

© Copyright 2010 Hewlett-Packard Development Company, L.P

Found' box is not displayed or once you return from 'Multiple Names Found' box, click 'OK' in the 'Select User, Computer, or Group' dialog box. From this step onwards, follow instructions from step 'f'.

f. Windows will now display 'Object' dialog box. Select the permissions that you would like to grant for the selected 'Name' from the 'Permissions' list. If you are setting permissions for a directory or a shared folder object, select the appropriate inheritance value from the 'Apply onto' drop-down. Next, click 'OK'.
g. Once you return to 'Permissions' dialog box in 'Advanced Security Setting', click on 'Apply' button.
h. Next, you can either repeat steps 'b' to 'g' or click on 'OK' to exit the 'Advanced Security Setting' dialog box. Once you return to 'Security' dialog box, click 'OK' to exit the 'Security' dialog box.

**Setting up File Security from an OpenVMS system:**

This section describes the steps/commands that should be followed for setting File Security from an OpenVMS system for users and groups belonging to the Windows domain, where CIFS is a member (CIFSDOM), the domains trusted by it and the CIFS Server. This method is limited to a user who has a fully privileged user account on the OpenVMS system. The steps to be followed are:

1. Login to an OpenVMS system using a fully privileged OpenVMS account, say "SYSTEM".

2. Define Samba commands by executing:

   $ @SAMBA$ROOT:[BIN]SAMBA$DEFINE_COMMANDS.COM

3. Find out the mapped OpenVMS identifier of a domain/CIFS user or group to whom you are trying to grant permissions by executing the following command and note down the identifier name (VMSIDENTIFIER-NAME) displayed:

   $ WBINFO --domainname-to-hostname=<DOMAINNAME\USERNAME OR GROUPNAME>
   VMSIDENTIFIER-NAME

   NOTE:
   a. If it is a user or a group belonging to Windows domain (CIFSDOM), where CIFS is a member, replace DOMAINNAME with the Windows domain name (CIFSDOM).
   b. If it is a user or a group belonging to a domain trusted by Windows domain (CIFSDOM), replace DOMAINNAME with the trusted domain name.
   c. If it is a user or a group that exists on the CIFS local database, replace DOMAINNAME with CIFS server name (PIANO).

© Copyright 2010 Hewlett-Packard Development Company, L.P

4. Execute the following command to set required permission for the identifier (VMSIDENTIFIER-NAME) noted down in step ‘3’. For example,
    a. if you want to grant READ and EXECUTE access permissions to a directory DIRECTORY_NAME.DIR that is under a CIFS share path, you can execute:

```
$ SET SECURITY/ACL=(IDENTIFIER=VMSIDENTIFIER-NAME,ACCESS=READ+EXECUTE) –
_$ DEVICENAME:[PARENT_DIR_PATH]DIRECTORY_NAME.DIR
```

    b. If you would like to apply the inheritable DEFAULT ACE on this directory, you can set DEFAULT ACE by executing:

```
$ SET SECURITY/ACL=(IDENTIFIER=VMSIDENTIFIER-NAME,OPTIONS=DEFAULT, -
_$ ACCESS=READ+EXECUTE) DEVICENAME:[PARENT_DIR_PATH]DIRECTORY_NAME.DIR
```

5. To modify permissions for an existing ACE and to replace the modified ACE at the existing ACE location, you can execute:

```
$ SET SECURITY/ACL=(IDENTIFIER=VMSIDENTIFIER-NAME,ACCESS=READ+EXECUTE) –
_$ /REPLACE= (IDENTIFIER=VMSIDENTIFIER-NAME,ACCESS=READ+WRITE+EXECUTE) -
_$ DEVICENAME:[PARENT_DIR_PATH]DIRECTORY_NAME.DIR
```

6. To insert an ACE after another ACE, you can execute:

```
$ SET SECURITY/ACL=(IDENTIFIER=VMSIDENTIFIER-
NAME,ACCESS=READ+WRITE+EXECUTE) –
_$ /AFTER=(IDENTIFIER=EXISTING-VMSID-NAME,ACCESS=READ+WRITE+EXECUTE+DELETE)
-
_$ DEVICENAME:[PARENT_DIR_PATH]DIRECTORY_NAME.DIR
```

    NOTE: The ACE specified for /AFTER must exactly match the existing ACE on the directory.

    NOTE: This step can be used as a workaround for the “File access limitation”. The ACE with higher permission can be place before the ACE with lower permission.

7. To remove an ACE for a user or a group whose identifier was obtained in ‘3’, execute the following command:

```
$ SET SECURITY/ACL=(IDENTIFIER=VMSIDENTIFIER-NAME, -
_$ ACCESS=READ+WRITE+EXECUTE)/DELETE –
_$ DEVICENAME:[PARENT_DIR_PATH]DIRECTORY_NAME.DIR
```

NOTE: Execute $ HELP SHOW SECURITY to obtain more help.

## Backing up CIFS DATABASE files

The previous sections covered user and group mapping, permission mapping, and the steps that can be used for setting permissions. It can be noted that the permission on the objects in a CIFS share is applied based on the mapped OpenVMS usernames and

© Copyright 2010 Hewlett-Packard Development Company, L.P

resource identifiers (groups) and not based on the domain/CIFS users and groups that they are mapped to. From this, it should be noted as to how important it is to maintain this user and group mapping information for retaining security on the objects. As already mentioned in previous sections, few of the CIFS databases maintain this mapping information. The following databases in CIFS are crucial and they must be backed up regularly in order to avoid any loss of data due to avoidable or unavoidable reasons:

| | |
|---|---|
| winbindd_idmap.tdb | - Stores records for WINBIND created mapping between domain SIDs and idmap UID/GID values (or OpenVMS UICs and resource identifiers). |
| group_mapping.tdb | - Stores records for CIFS local groups and the members of these groups |
| passdb.tdb | - Stores records for CIFS local users when passdb backend = tdbsam |
| secrets.tdb | - Stores private information like workstation passwords, the ldap admin dn and trust account information |
| account_policy.tdb | - Stores NT account policy settings such as pw expiration |
| share_info.tdb | - Stores information about share ACLs |
| ntdrivers.tdb | - Stores information about installed Printer drivers |
| ntforms.tdb | - Stores information about installed Printer forms |
| ntprinters.tdb | - Stores information about installed printers |

The passdb.tdb and secrets.tdb database files will be present in the directory, SAMBA$ROOT:[PRIVATE]. Rest of the database files will be present in the directory, SAMBA$ROOT:[VAR.LOCKS].

## For more information

The information present in this article has the following sources:

- SAMBA Official HOW-TO
- HP OpenVMS CIFS Administrator’s Guide
- HP OpenVMS Guide to System Security
- Microsoft Authentication and Access Control Technologies
- Release notes for CIFS Patch set for CIFS V1.1 ECO1

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# Making the Business Case for Migrating VMS Oracle Applications to the Web

Jennifer McNeill
Vice President, Oracle Practice
Unify/CipherSoft Corporation

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Introduction

Many companies have millions of dollars invested in existing Oracle VMS application systems, with most of these mission critical systems that run their business. Application software and its associated databases are valuable corporate assets. Companies now want to leverage their current investment in data, products and applications with new Web-enabled applications. It is generally not practical to spend countless dollars replacing existing, functional systems with new technology or a new programming paradigm. It makes far more business sense to implement new technology (such as Oracle 10g, 11g and Java) where the impact will be the greatest (such as in the front-end of the application) and the relative cost will be the most reasonable. The goal is to leverage existing application systems and data while providing new Web based functionality.

Many organizations are contemplating utilizing the new capabilities that are being provided with Web technology. While past versions of Oracle Forms have enabled companies to access the Web, 10g and 11g are providing one of the most robust environments thus far. However, the move to the Web is not always as easy as it seems. It is important that organizations understand the business reasons to move their applications and determine if the effort and dollars spent will justify the migration costs. Understanding the risks of migration, the options available and the benefits of Java is imperative to ensure an organization is not spending precious IT dollars on an alternative that does not make business sense.

While migrating VMS Oracle applications to the Web seems challenging, the ability to maintain the years of application development with proven automated tools provides tremendous benefits to organizations. By utilizing an automated migration tool, organizations can save costs as well as time over rewriting an application.

As companies review the differences between migrating to Java and performing development in 10g and 11g there are some key components that assist in the decision for the future development environment.

## Migration to Forms 10g and 11g is beneficial if:

- The organization has very few Java resources and retains a skill set in Forms and PL/SQL development
- The application doesn't require changes to the look and feel of the application
- Skill sets in Forms and PL/SQL are readily available to the organization long-term (stability in resource retention)
- The Forms being used presently are character based versions of Forms (this is due to the learning curve involved in migrating from a client-server technology to multi-tiered Java)
- Java plug-ins (downloads) are acceptable to clients using the application

## Migration to Java is beneficial if:

- Java has been chosen as the future development environment for the organization
- Application development costs require reduction (Java development provides much lower development costs)
- The legacy Forms application requires integration with other applications developed in other environments

© Copyright 2010 Hewlett-Packard Development Company, L.P

- Other applications within the organization are utilizing Java technology
- The use of open source technology is beneficial to the organization
- Provision of choices within the client's environment is required (such as HTML, DHTML, use of browers, etc.)
- The use of SOA architecture is beneficial to the organization
- The organization has determined that the use of proprietary technology is no longer beneficial

## The Benefits of Java

Java offers clients a robust, interactive environment. It is one of the most powerful Object Oriented Programming languages available and one of the most "open" technologies available. Java conforms both to its own standardized (and published) specifications as well as to other industry standards such as CORBA. JDBC (Java Database Connectivity) provides a standardized interface for relational databases that can interface, providing a greater level of database independence and portability. It also provides platform independence allowing an organization to utilize the most efficient and effective hardware available. Having choices for hardware and database software can be very liberating for an organization, deterring vendors from holding companies hostage to their proprietary environments.

## Why is Java Superior?

Java is currently the only technology that provides a fully interactive GUI interface for the Web. The Java architecture was designed with security in mind and not as an afterthought. This provides a simplified and consistent means of protecting IT assets. Java also provides features that allow programming to become easier and more powerful. These include: Multi-threading capabilities:

1. Automatic "garbage collection" (for efficient use of memory)
2. Standardized error trapping and detection
3. Distributed processing capabilities

## Java's Capabilities for Web Development

One of the benefits of utilizing Java as a development environment is the ability to create a user interface in several different ways:

1. HTML using JSF or JSP
2. Java Plug-in running in a browser
3. Java plug-in running outside of a browser

When utilizing Oracle Forms, the User Interface is provided as part of Form and is an applet running in a browser window. While this may be a good choice for the present, in order to move forward with development technology such as Oracle's WebCenter, ADF, struts, spring and the Web development capabilities Java provides many more choices and capabilities.
Java provides well-known standards that provide developers with full control of their application environment and integration. This enables Web development of applications to be much more flexible.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Migration Process

Exodus - The Steps of the Automated Conversion process

1. Oracle Forms
The Oracle Forms .fmb module, library and PL/SQL procedures are used as source files. All of this source programming is utilized by Exodus which creates a readable version of the Forms application in Java.

2. Exodus Migration
Once the key structures and functions have been organized, Exodus translates the original application's components into the corresponding Java plug-in or JSF (ADF) components as required. In doing so, the migrated application maintains the same business logic and can optionally either show the same look and feel as the original application, or convert to a specific user defined screen.

3. Compilation
The compilation process can be performed in any IDE, such as JDeveloper, VisualAge and NetBeans to compile, maintain, and upgrade the migrated application. The Application is now 100% J2EE compliant Java code and as such can be integrated into ANY Java supporting environment or development tool.

4. Deployment
The generated Java Code is object oriented, readable and fully maintainable, ready to be deployed on any J2EE compliant application server and environment. It can now be free of and dependency on proprietary environments and tools. The entire application can now be maintained by any Java proficient developer.

5. Maintaining the Converted Application
Exodus-VE is a visual editor that provides drag and drop capabilities to the maintenance and enhancement of the converted application. This enables clients to start maintaining and enhancing their application immediately after conversion with relative ease.

## Oracle's Strategic Direction?

Oracle has been straightforward with their approach as they move towards new technology. Their goals include:

1. Pooling server-side Java virtual machines to reduce the memory footprint of applications that call middle-tier Java
2. Reduced application pre-starting
3. Performance and scalability on the Web
4. Expanding the scope and depth of the Forms management tuning and problem diagnosis facilities within Enterprise Manager
5. Extensible client and middle-tier Java integration (Java Importer and Pluggable Java Component Interface)
6. Development of their own Enterprise applications with technology such as ADF and JDeveloper

© Copyright 2010 Hewlett-Packard Development Company, L.P

Oracle cites research from IDC to make the case that the enterprise market is headed in the J2EE direction:

The Options for Migration

There are several options available for migration of Oracle Forms or PL/SQL into the new Java environment. Conversions and migrations have been performed over the past 25 years as organizations move from older technology to the newest. Many lessons have been learned through these conversions. With the options available, an organization can determine the methodology that suits their requirements based on the time available before the migrated application is required to be in production. This review should include their financial restraints as well as their internal capabilities. The following outlines options for migrating the applications.

Rewrite the Application

There's a subtle reason that programmers always want to throw away code and start over. The reason is that they think the old code is a mess. And here is the interesting observation: they are probably wrong. The reason that they think the old code has problems is because of a cardinal, fundamental law of programming: “It's harder to read code than to write it”. This is why code reuse is so hard. Programmers tend to write their own function because it's easier and more fun than figuring out how the old function works.

In other words, rewriting applications from scratch is the most expensive and time-consuming option. Many organizations believe that rewriting will solve their problems. They choose to rewrite because they need additional functionality, or because they believe it will save them time and money. One of the obstacles with rewriting applications is the enormous amount of testing that has to be performed to ensure business logic is handled properly, users understand the application, and the application is applicable for the business.

Web-enable Forms

This option seems to be simple. Utilize the capabilities of Forms and become Web-enabled. However, as an organization moves towards 10g, this option becomes less feasible. Some sort of migration effort is required to fully take advantage of the capabilities of 10g. There are companies who are just not ready to migrate their development to Java. This may be due to resource restraints or knowledge level, or simply because they prefer to continue to develop their applications in Oracle Forms. In this case, utilizing migration services or tools to migrate to 10g makes sense. This will provide an organization the benefits of the 10g technology as well as prepare them for a future move to Java.

Manual Conversion

This alternative is by far one of the highest risks. It is not only time consuming, but also shares the negatives that rewriting the application presents. This includes human error issues, lack of resources or skill set and disruptions to the business operations. In addition, escalating costs usually coincide with manual conversions. Many organizations have started their Forms to Java conversions manually only to determine that the time difference between manual and automatic is significant.

Automatic Conversion

Automatic conversion is one of the best alternatives to migrating applications. With an effective migration tool, the conversion can be performed quickly and effectively with

© Copyright 2010 Hewlett-Packard Development Company, L.P

tremendous cost savings. Some conversion tools have benchmarked to be 90% faster than manual migration and 80% less expensive. In addition, automatic conversion reduces the risks for the organization. Functionality is maintained and users do not require retraining. There are several options for automatic conversion that enable an organization to move quickly into the new environment. It is important to note, however, that all automatic conversion alternatives are not the same.

There are presently very few tools available in the market with some venders offering a service to perform a Forms to Java conversion. In some cases, these vendors require that the client send their code "offshore" to have the conversion performed. In addition, there are questions that should be asked of the vendor to ensure that the migration is being converted to a true Java or J2EE environment. These include:

1. Is my application being converted to truly compliant J2EE code?
2. Are we able to purchase the tools, or is this a service offering only?
3. Where is my conversion to be performed (on-site or at the client's site)?
4. Are we able to discontinue licensing of Oracle Forms and PL/SQL or do I still have to license these products?
5. Is the vendor available to assist with any issues and training once we migrate to Java?
6. What percentage of conversion is automatic or how much manual work is involved once it is converted?
7. Is the J2EE code "clean", i.e., is it easily maintainable once I get into the Java environment?
8. Does the converted code integrate with JDeveloper and utilize the ADF and BC4J environment from Oracle?

It is important that as a company asks these questions, they actually work with the vendor to provide a "sample" of the converted Java code. An organization should have the option to send in a small sample of an application to see the resulting converted code. This will ensure that the code is easily maintainable and functional as soon as it is migrated. Functionality can be added or changed during an automatic migration, as it can be more closely controlled than with other migration alternatives.

## Summary

It is sometimes difficult to determine the best solution for migrating applications to new technology. The conversion of applications should not be a painful exercise, but one that provides efficient alternatives to the organization enabling the most cost efficient option.

Making the business decision to determine how to move into new environments is not always easy. Weighing the benefits to the business of becoming Web-enabled against the time and effort required for a migration process is an important part of this decision. Keeping costs and risks to a minimum, as well as causing little business interruption will ensure that the organization continues to benefit from the project.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## For more information

Jennifer McNeill
Vice President, Oracle Practice
CipherSoft (A wholly owned subsidiary of Unify Corporation)

Jennifer.mcneill@ciphersoftinc.com

www.ciphersoftinc.com

© Copyright 2010 Hewlett-Packard Development Company, L.P

OpenVMS Technical Journal V14

# SimH Supports Availability Manager Development Environment at HP

Barry Kierstein, Software Engineer

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Introduction

During my tenure at Hewlett-Packard, I served as the project leader for ***Availability Manager*** on OpenVMS. As such, I was responsible for its development and testing on the VAX, Alpha, and Integrity platforms. This article describes the deployment of the open source SimH VAX emulator to replace aged VAX hardware while maintaining the *Availability Manager* test environment. This article was sponsored by Migration Specialties.

The article is in two main sections. The first section tells the story of why SimH VAX was selected and how it was utilized to solve legacy hardware and testing issues. The second section, Configuration and Implementation, provides details on how the SimH VAX environment was created.

## The Problem: Legacy VAX Test Hardware

Over time, the primary development and test cluster for *Availability Manager* evolved to consist of a number of VAX, Alpha, and Integrity systems. These machines were good solid machines, but some were getting to be pretty old, especially the VAX systems. By 2007, I was concerned that the VAX systems would eventually fail. Since I had a limited hardware budget, this would leave me in the difficult position of supporting VAX systems in the field without any VAX hardware in the lab. Much of the *Availability Manager* development work involved device drivers, with the associated potential of crashing systems during development and testing. Hence, using machines outside the *Availability Manager* development cluster was not an option.

## The Virtual Option

In the summer of 2007, VAX hardware reliability concerns prompted me to research replacement options. I needed four VAX systems to replace the VAX workstations that I possessed at that time – one for each version of OpenVMS supported by *Availability Manager* (OpenVMS V6.3, V7.1, V7.2, and V7.3). I was looking for something that would fit my budget and work on the available Windows PC systems.

Virtual machines were becoming common place and the idea of a virtual VAX was intriguing. Virtual machines are on ongoing interest for me. I had studied the Virtual Machine (VM/CMS) operating system by IBM in college, seen commercial VAX emulators demonstrated at various forums, attended VMware sessions at the HP Technology Forum, and used the Microsoft Virtual PC software.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Virtual VAX Advantages

A virtual VAX offered the following advantages for *Availability Manager* development and testing:

- Easy to backup and restore a virtual machine
    - Allowed for quicker recovery from development bugs
    - Allowed a quick system restore to a known state for repeated tests
- Easy system reconfiguration of memory, disk drives, and network adapters
    - Allowed quick move of a disk drive from one system to another
    - Allowed development and test system configurations that I could not previously afford using physical hardware, such as 512MB of memory, many disk drives, and multiple network adapters

## Evaluating SimH

After investigating alternatives, the free, open source SimH VAX emulator looked promising. The SimH website, http://simh.trailing-edge.com/, provided the then current installation kit, version 3.7-0. I downloaded the software and started reading the documentation. I also subscribed to the SimH mailing list (the address is under "Help With SimH" on the SimH homepage). Between the documentation and some postings from the SimH website, I was able to get my first virtual VAX up and running.

Working with this simulation, I learned a number of things:

- The *Availability Manager* successfully installed and ran in the virtual environment.
- The ability to backup and restore virtual machines was as easy as expected, opening up possibilities of repeated testing of a system starting from a known state.
- The ability to reconfigure the virtual machines by changing the amount of memory, changing the number of disk devices, etc. was as easy as expected, enhancing configuration testing capabilities.
- The ability to transfer a disk from one virtual machine to another was as easy as expected, which improved operationally efficiency and flexibility.
- Each virtual machine needed its own network adapter and MAC address.

## Handling Multiple Network Adapters

The need for separate network adapters was based on a subtle, but important behavior. This involves how applications sharing an Ethernet-based network adapter are able to view traffic from each other. The adapter, by default, does not send the outbound traffic from one application to the other applications on a shared adapter. In my case, if two virtual machines shared the same adapter, outbound traffic from one of them was not seen by the other. So, if the two virtual machines were part of the same cluster, they wouldn't see the cluster traffic from each other. As a result, the two virtual machines cannot form a cluster connection between them.

This network adapter behavior is well known and can be addressed in a couple of ways. One solution is using one network adapter per virtual machine. This is the simplest setup and the one with better performance. The other is to install additional software that allows

© Copyright 2010 Hewlett-Packard Development Company, L.P

applications sharing a network adapter to see the outbound traffic from each other. This solution allows one to use fewer physical network adapters.

In my case, I wanted to keep the software setup as simple as possible so I obtained enough network adapters to have one per virtual machine.

## SimH Implementation

Evaluation of the initial SimH virtual VAX established that the emulator would support the OpenVMS VAX environment required for *Availability Manager* development and testing. I had a single dual-CPU PC available to host four instances of the VAX emulator. The limited host hardware impacted performance of each virtual VAX, but since I didn't need great performance, this was acceptable. I used the following steps to generate the initial virtual VAX configurations:

- Installed the additional network adapters into the host system.
- Built the configuration files for each virtual VAX system.
- Assigned a unique MAC addresses to each system.
- Created virtual disk drives for each VAX system. The virtual drives appear as large files on the host system and are referred to as container files. Virtual drives can be created using the SimH ATTACH command.
- Test booted each virtual VAX to the boot prompt and verified the configuration via console commands.

The next step was to migrate the drive contents from the four legacy VAX systems to the virtual VAXen. I accomplished this by installing OpenVMS on one virtual VAX. I then configured this virtual machine to use all of the virtual machine disk drives, booted the system into the cluster, and used $ BACKUP/IMAGE to move the drive contents from the physical VAX systems to the virtual VAX disk drives. Using image backups on live disks does entail the risk of not picking up all the contents of open files. However, the open files on my systems were mostly log files so this was not an issue.

Once I had migrated the drive contents, I set the four virtual VAX to their final configuration and booted them. They joined the *Availability Manager* development cluster and acted like their VAX hardware counterparts!

After finishing the initial conversion of the four VAX systems, I created four more virtual systems. These had the same configuration as the first four systems, but with a fresh install of OpenVMS. These virtual machines were for testing initial installations of *Availability Manager* kits. I had arranged for some disk space from the main OpenVMS cluster for the virtual machine backups. I copied the virtual machine container files to the OpenVMS cluster to establish a base copy of the virtual machines before installing *Availability Manager*. With this backup, I could install on these systems and then simply copy the container files from the OpenVMS cluster to restore the systems to their initial state. Much faster than using tapes, etc. on physical systems!

Configuration and implementation details are in the section "Configuration and Implementation" at the end of this article.

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Putting SimH to Work

Development and testing of each version of *Availability Manager* requires a fair number of test installations. A great bonus of the virtual VAX setup was the ease in which a virtual machine could be restored to a known state after a test. By copying the disk container files from the OpenVMS cluster to the PC, I didn't have to fiddle with standalone backups, tapes, etc. I was able to do a thorough job of testing than before. Part of this expanded testing involved changing the amount of memory each virtual machine had, the numbers of disk drives available, etc. This type of testing had been limited in the past by physical hardware availability.

Another virtualization benefit was that I could do more radical testing since I didn't have to be as careful with the virtual machines as I did with the original VAX hardware. Unlike physical hardware, recovery from a bad software install or test could be achieved by simply restoring copies of the base system disk container files from the main OpenVMS cluster.

While performance of each virtual VAX was a bit slow due to the limitations of the host system, it was acceptable. Idle mode detection was available in the version of SimH I deployed which helped alleviate performance issues.

As it turns out, my premonition about the physical VAX hardware came to pass. In the winter of 2008, there were a number of power interruptions and outages that did in the VAX hardware. It was sad to see the old hardware go, but fortunately, the applications they supported lived on in their new virtual home.

## Conclusion

The time taken to evaluation and implement the SimH VAX emulators proved to be a good investment. System reliability, test efficiency, test flexibility, and overall productivity improved after installing the SimH VAX systems. This in turn allowed me to do additional development and testing. *Availability Manager* is a better product as a result.

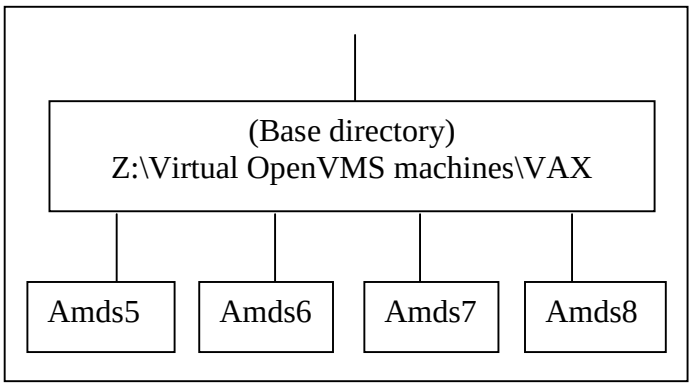
## Configuration and Implementation

This section covers the configuration and implementation details of the *Availability Manager* SimH test environment I created.

Configuration for SimH is not complex. The main areas of focus are disk volumes, network adapters and the console for the virtual machine.

A SimH virtual machine consists of the disk volume container files, a file to contain the non-volatile RAM and an editable text file with the configuration details. I created the following directory structure to contain the files for all the virtual machines.

Directories Amds5 through Amds8 were named after the OpenVMS VAX systems being emulated. These directories housed the configuration and virtual disk files unique to each emulated VAX system. The configuration file for each system specified the non-volatile RAM and disk container files with relative path names from the base directory. This made moving the virtual machines from one place to another easier.




© Copyright 2010 Hewlett-Packard Development Company, L.P

Specifying a configuration file for SimH VAX is fairly simple – simply specify the path and file name as the first parameter. Running SimH on Windows, I created a shortcut and set the Target field to the following:

```
"Z:\Virtual OpenVMS machines\VAX\vax.exe" amds8\simh_amds8_config.txt
```

**Figure 5: SimH shortcut.**

I also set the Start in field to "Z:\Virtual OpenVMS machines\VAX", the base directory, for the relative path references in the configuration file.

This is the configuration file for one of the virtual machines – AMDS8. The configuration defines two disk volumes and one network adapter. It also has a number of commented-out commands as a reminder of how to configure various items. Note that this configuration file was used with SimH V3.7-0.

```
; Load CPU microcode
;
load -r ka655x.bin
;
; Attach non-volatile RAM to a file (default boot device settings,
etc.)
;
attach NVR amds8\ka655.nvr
;
; Set the memory to 128MB
;
set cpu 128m
;
; Set the CPU to allow the emulator to idle when OpenVMS is idle.
;
set cpu idle
d idle_ipl 8
d idle_wait 1000
;
; Set the halt to return to the ROM console
;
set cpu conhalt
;
; Set the localhost TELNET port for the console
;
set TELNET 5308
;
; Enable and disable various devices
;
set RL disable
set LPT disable
set TQ disable
set TS disable
set rq0 ra92
attach rq0 amds8\amds8_sys.dsk
set rq1 ra92
attach rq1 amds8\amds8_user.dsk
set rq2 disable
set rq3 disable
;
; Set MAC address and adapter
```

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
;
set xq mac=00-08-C8-08-CE-DE
attach xq eth4
;b cpu
```

**Figure 6: Sample SimH VAX Configuration File.**

I put a number of comments in this file as well as various commands that are commented out. I left these in as a reminder of what possible. Here are some highlighted sections:

```
; Set the memory to 128MB
;
set cpu 128m
```

**Figure 7: Setting emulator memory parameter.**

This command sets the physical memory for the virtual machine. For the MicroVAX 3900 emulator, this has been extended so that it can emulate up to 512MB from the usual maximum of 64MB.

```
; Set the localhost TELNET port for the console
;
set TELNET 5308
```

**Figure 8: Setting the console port.**

This command defines the console input/output to port 5308. Note that the equivalent command for SimH 3.8-1 is "set CONSOLE TELNET=5308". Since I was running a number of the emulators at the same time, I had a system of starting the port numbers at 5300 and then appending the node ID number (8 for Amds8, etc.).

Using the telnet port, the virtual machine console can be connected to the terminal emulator of your choice. I decided to use the freeware PuTTY emulator[1] since I was familiar with it and satisfied with its VT emulation.

I created a PuTTY session – AMDS8 Console – that specified the Telnet port. Then, I created a shortcut for the Amds8 console that started PuTTY and specified the PuTTY session. The Target field of the shortcut is as follows:

```
"R:\Program Files\PuTTY\putty.exe" -load "AMDS8 console"
```

**Figure 9: PuTTY shortcut.**

To start Amds8, I would first double-click on the Amds8 SimH shortcut to start the emulator. The emulator would start to execute in a Windows console window and then prompt for input with a sim> prompt. At this point I would double-click on the Amds8 console shortcut and PuTTY would connect to the emulator console. I would then enter the B CPU command at the sim> prompt to start the virtual machine. The console displayed the normal output for a MicroVAX 3900, and I interacted with this console as if it were the real thing!

```
set rq0 ra92
attach rq0 amds8\amds8_sys.dsk
set rq1 ra92
attach rq1 amds8\amds8_user.dsk
```

[1] PuTTY is available at http://www.chiark.greenend.org.uk/~sgtatham/putty/.

© Copyright 2010 Hewlett-Packard Development Company, L.P

```
set rq2 disable
set rq3 disable
```

**Figure 10: Virtual disk setup.**

These are the commands used to set up the two virtual disk drives for the virtual VAX. By default, there are four RQ devices per controller. I needed only two RQ devices so, I explicitly disabled the other two. If these devices aren't explicitly disabled, they show up under OpenVMS as two online disk drives. However, when you try to mount them, OPCOM complains that the medium is offline and that you must mount a device for the drive – confusion that is easily avoided by taking advantage of the disable feature.

If a virtual disk needs to be moved from another machine to this one, all that is needed is to set RQ2 or RQ3 to specify the location of the container file, and the disk is available to the virtual machine!

There are a number of other disk controllers that can be enabled, allowing a virtual machine to emulate a fairly large number of disk drives. In SimH version 3.8-1, the VAX 11/780 emulator can emulate up to eight RP04/05/06/07 or RM02/03/05/08 disks, four RL11/RL01/02 disks, and 16 UDA50 MSCP disks on four controllers, as well as TS11, TUK50, and TM03 tape drives and two DEUNA/DELUA Ethernet controllers. The MicroVAX 3900 can emulate up to 512MB of memory and has a similar number of disk, tape, and Ethernet controllers as the VAX 780 emulator.

```
set xq mac=00-08-C8-08-CE-DE
attach xq eth4
```

**Figure 11: Setting up a network adapter.**

These are the commands used to set up the network adapter. I set the MAC address to what I found was a unique address in the network segment. Setting the MAC address is an option; it is not required. Since I was hosting four SimH VAX emulators on one physical system, I wanted to ensure each emulated VAX had a unique MAC. Note that eth4 is the mnemonic for the fourth adapter on the host machine (Amds5 took the first one adapter on my system, Amds6 the second, etc.).

Eth1 is bound to the first adapter, Eth2 is bound to the second adapter (if it exists), etc. If you type in the "attach" command interactively at the SimH prompt, SimH displays which network adapter is bound to Eth1, Eth2, etc. If a binding doesn't exist for a particular mnemonic, an error message is displayed.

Once this configuration was done, the virtual machines were ready to run.

## Installing OpenVMS

Installing OpenVMS onto a newly created virtual machine can be done through a CDROM device on the virtual machine. An example of configuration for the CDROM drive is shown below:

```
set rq2 cdrom
attach –r rq2 VAXVMS073.ISO
```

**Figure 8: Setting up a CDROM drive.**

Note that the contents of the OpenVMS installation CD have been transferred to an ISO file on the host system. There are a number of Windows applications that can create an ISO file from a CD, and many Windows systems have an application that does this already installed.

© Copyright 2010 Hewlett-Packard Development Company, L.P

Once the ISO file has been created, then it can be used by the virtual machine for the OpenVMS installation.

## About the Author

Barry Kierstein has worked with OpenVMS since 1981, and worked for Hewlett-Packard for almost fourteen years. His work at Hewlett-Packard was with the System Management group within the OpenVMS organization and is most known for his project leadership work and advocacy of the *Availability Manager* product. He is also known for various presentations and hands-on workshops at the OpenVMS Bootcamp, DECUS, CETS, HP Tech Forum, European Technical Update Days, and various webinars dealing with OpenVMS performance, *Availability Manager*, HP Virtualization strategy around HP SIM/VSE, Blades, and virtual machines. Barry is under exclusive contract to Migration Specialties for emulator and *Availability Manager* services.
The SimH VAX development and test environment was transferred to the Indian development and support group and continues to be utilized.
This article was sponsored by Migration Specialties. Information about Migration Specialties emulator products and OpenVMS services can be found at www.MigrationSpecialties.com

© Copyright 2010 Hewlett-Packard Development Company, L.P

# Virtual Alpha Systems: Quality Control & Testing

Dr. Robert Boers, CEO & Founder Stromasys SA.
(For bio see http://www.stromasys.ch/about-us/board-of-directors)

© Copyright 2010 Hewlett-Packard Development Company, L.P

## Introduction

Since 1994, Stromasys has been developing system emulators, starting with PDP-11 emulation. Now known as “cross-platform virtualization,” these system emulators provide a complete software model of a legacy hardware system, running on a modern server. This technology allows the users to keep their original operating system and applications running in an unmodified form, while gaining performance improvements, increased storage capacity, and increased reliability.

When we built the first system emulators, we had no idea that the market would demand for these solutions. Nor, did we realize the efforts required to develop commercial quality emulator products, particularly the importance of a comprehensive product testing strategy. This article gives an insight to the development of testing technologies and procedures that we created to ensure reliable replacement products with superior performance.

## Designing and Testing Virtual Systems

In principle, creating a software abstraction layer of a hardware computer component is straightforward. A subsystem, for instance a CPU, is an implementation of sequential logic. The functionality is well defined and can be described in a high-level programming language. In modern CPU design, creating a software model before creating the desired hardware is a common practice, and we repeat this process for all the relevant subsystems.

If the emulator architecture follows the hardware architecture closely, the result is a library of emulated subsystem components, each representing a hardware subsystem in the form of a code module. In the case of the first generation PDP-11 emulator, we designed approximately 130 modules written in C++, representing the various CPU modes, memory management, disk and tape controllers, various serial and parallel I/O options, Ethernet controllers, and so on.

Given the resemblance between the original hardware and the emulated components, the obvious choice for testing the emulator was using the original hardware diagnostics. Having few other tools at our disposal at that time, we pushed the component emulation to a level where the original hardware diagnostics would even show the revision numbers of the hardware components that were emulated. This unnecessary complexity resulted in a slow implementation compounded by running with the virtual system modules in synchronous mode to avoid timing problems. However, in dealing with PDP-11 and slow VAX models, this got us a correctly working emulator with the existing diagnostics.

However, we cannot always use the existing hardware diagnostics. An emulated tape drive, for instance, has a tape positioning time that is close to zero. The hardware diagnostics find that hard to accept, and even on a correct implementation, software might malfunction. Putting NOOP’s in drivers to make them work correctly will create havoc in a faster virtual interface, as it would with faster hardware.

In the development cycle of a virtual system, booting an operating system on it is a milestone that indicates that a minimum of virtual hardware is working correctly. Interestingly, most operating systems are not very useful by themselves to debug a virtual system. When we could boot VMS on our MicroVAX emulator, later tests revealed that about 85% of the emulated VAX instructions had errors in some modes. A surprise was that without floating point instructions VMS would not boot. Unfortunately, floating point

© Copyright 2010 Hewlett-Packard Development Company, L.P

instructions for VAX are complex to implement correctly on an x86 architecture. Nevertheless, the ability to run the original OS is an important step, since it greatly facilitates the development and execution of custom diagnostic tools.

Before the complexity of our virtual system products drove us to the development of custom diagnostic tools, and from there to a formal test process. We found that implementation of an emulator on more than one host platform helps error discovery. From the same source code, errors typically appear in different places, or only in one implementation, and are easier to track down. For example, in an industrial application on a virtual PDP-11 hosted on Windows NT, using Alpha hardware the menu items were correctly displayed in white text. On an x86 platform, some menu items appeared in red, because the x 86 C++ compilers generated an error that propagated in a rarely used FORTRAN call.

## Introduction of Formal Tools

In the development of our CHARON-VAX emulators, we used a tool called AXE, which we obtained from Compaq. AXE was the verification tool for the VAX hardware development groups in Digital. AXE had two purposes at Digital:

- Locate bugs in the hardware during development
- Verify that we meet the requirements of DEC STD 032, the functional requirements for a VAX hardware system.

AXE creates test cases composed of a VAX CPU instruction with operand specifiers appropriate for that instruction. After execution, AXE compares the relevant state with a "known good" result and reports the differences to the user. The individual test cases are generated on the fly, creating billions of unique sequences. These variations cover essentially all aspects of a VAX CPU and memory management unit: opcode, opcode or operand placement with respect to page boundaries, (stack) page protection, translation buffer validity, overlap of string operands, page faulting, etc. AXE also covers testing of exception handling like reserved opcodes, arithmetic traps addressing modes, and invalid translations.

The final AXE test before Digital released VAX hardware was a comparison against a reference database. The green light was given if a test of ten million cases of each user mode instruction (in all access modes), and 5000 cases of each privileged instruction mode were passed without a single error. AXE has been valuable in the development of our virtual VAX systems, and we base the first phase of our virtual Alpha product readiness tests on a similar concept, using proprietary tools. Such testing is very time-consuming. We passed our virtual VAX 3100 AXE readiness test in the year 2002 in a certification sequence that lasted more than three weeks on a 500 MHz Celeron processor.

Both for "hardware" and "virtual hardware" developers, a good working CPU is only one of the many components to integrate and test. An advantage is that many hardware components have standard functions such as SCSI controllers, UARTS, disk drives, Ethernet controllers, and so on. These components are either fully tested by the supplier or can be tested individually while connected to a good system. Virtual system developers do not have that luxury. To create a complete hardware abstraction layer of a new system, we must emulate such components as well, in order to create a portable product. As we learned over the years, this makes the reuse of known good virtual components extremely important. If we make the elements of a virtual component library usable on different

© Copyright 2010 Hewlett-Packard Development Company, L.P

virtual systems, saving development time is far from the most important benefit. The repeated integration testing of such components in multiple designs leads to a significant increase in reliability. Provided, that strict code management, re-utilization, and maintenance is enforced. After 10 years, and more than 40 PDP-11, VAX, and Alpha models, the investment pays off in the form of a rich set of reliable virtual components. As an example, a virtual Sparc Workstation prototype reuses 70% of the existing virtual VAX components, allowing fast development and less testing effort.

## The Current Virtual Alpha System Test Process

As we pushed the technology further, in particular to develop fast, virtual SMP systems (for instance a virtual 16-CPU CHARON-AXP system), the processes we used to ensure the reliability of a single CPU virtual VAX were insufficient. We had to change the virtualization technology in several fundamental ways:

- High speed virtual CPUs require complex virtualization algorithms to obtain the desired performance, typically involving more than one host CPU per virtual CPU. Just the drive for performance required years of research, as we wanted to reach two goals at the same time: achieve portability of the acceleration code (for reuse with other architectures) and match the clock rate with more modern CPUs.
- It is no longer feasible to run the components of a virtual system in synchronous mode when virtual systems get more complex. The interrupt latency suffers badly and interferes with proper operation. It also prevents effective use of multi-core host systems.
- An important constraint for virtual SMP systems is that their CPUs must clock at exactly the same frequency. This is easy to implement in a hardware system, but requires a very sophisticated synchronization mechanism in the virtual environment. Testing this is implicit, as an operating system like Tru64 shuts down any CPU that has a perceived clock rate that is even slightly different from the primary one.

An advanced virtual SMP system design requires three kinds of tests:

- Functionality tests to ensure that the binary compatibility with its hardware CPU equivalent is uncompromised. In particular, we must test mathematical operations extensively to assure correct overflow and underflow in floating point formats. For virtual Alpha systems, we developed a suite of proprietary tests (similar to AXE) that we calibrated on a hardware Alpha of the same generation as the virtual implementation. These tests include verification that self-modifying code works correctly (there is a myth that emulators would not be able to handle such code). We, typically, run such tests every night when we are working on code optimization of the virtual CPU implementation.
- Performance tests to ensure that the ongoing CPU optimizations have the desired effect. The optimization algorithms are dependent on binary code structures; an automated test procedure submits 100’s of binary Alpha code segments representing many different types of user applications. A weighted average is calculated with a check that no type of use is significantly degraded. We aim at a small but overall improvement with every release.
- System integration and Quality control tests, which have become the predominant tests in terms of calendar time before every product release. As we extend our virtual products with new components, for instance, FDDI or Fibre channel support, we must scrutinize the impact on the rest of the system in even more depth than for

© Copyright 2010 Hewlett-Packard Development Company, L.P

the original hardware, where the influence between the individual components was limited. Furthermore, our products can run on many different host systems. Proper operation on a representative set of host systems is an essential factor in quality verification.

## Organizational Impact

During the development of larger VAX systems in 2004-2005, the time needed to test our virtual systems started to limit our product release throughput, in spite of having a dedicated full-time test engineer in addition to the usual testing process in engineering. Splitting the testing formally in subsets (used during development) and creating a formal "product readiness test" (PRT) improved the situation somewhat. Products that pass a PRT are initially coded in their license as Field test capable. The field test program aims at collecting 20 user months of field test experience, for instance, five users during 4 months, but commercial pressure can reduce this. Field tests are a great way to verify the quality of the documentation. During the field test period, minor product problems are fixed and documentation is updated. This process is iterative, and a typical product release goes through a PRT three times before its official release as a product. The PRT for the Alpha platform contains more than 200 individual tests, and takes about 2 to 3 weeks. The number of tests is rising as we add functionality to our products.

Engineering groups tend not to be the best judges of the readiness of their products. In light of this, the solution was to create a separate Quality Control group in our organization, also responsible for measuring and publishing formal performance numbers and reviewing product documentation.

## Conclusion

We aimed at giving some insight into the process of creating and testing a reliable virtual computer system. Successful hardware emulation depends not only on good design, but also on rigorous testing and quality control procedures. The technologies we have developed and the understanding we have built of the testing processes to create a reliable virtual system are universal, making development of other virtual legacy systems easier. Our newest development center in Shenzhen, China, is now expanding to take these testing technologies outside of our historic focus on DECs PDP-11, VAX, and Alpha systems.

## For More Information

To find out more about Stromasys and our CHARON-VAX and CHARON-AXP virtualization software products, please visit www.stromasys.com.

© Copyright 2010 Hewlett-Packard Development Company, L.P