OpenVMS Technical Journal V15, September 2010

Table of Contents

OpenVMS Technical Journal V15

vKVM Functionality and Usage on OpenVMS

Abhishek KR

Introduction

The challenges of system administrator positions require that they be on-site and physically present to do their job. However, remote management can help reduce the need for the physical presence of system administrators by providing virtual support, saving time and human resources. Certain remote management features, such as vKVM, also provide virtual devices eliminating the need for procurement of physical devices. vKVM is an Integrated Lights Out (iLO) integrated console feature that is provided by Lights Out Advanced, and the vKVM acronym is short for virtual keyboard, video, and mouse. The vKVM, also known as the IRC (Integrated Remote Console) feature on the Lights-Out Advanced (LOA) card, enables system administrators to view video output from the managed host where the LOA card is installed, providing a seamless view from the server boot to OS desktop. It also provides keyboard and mouse input to the console from the remote system to the host.

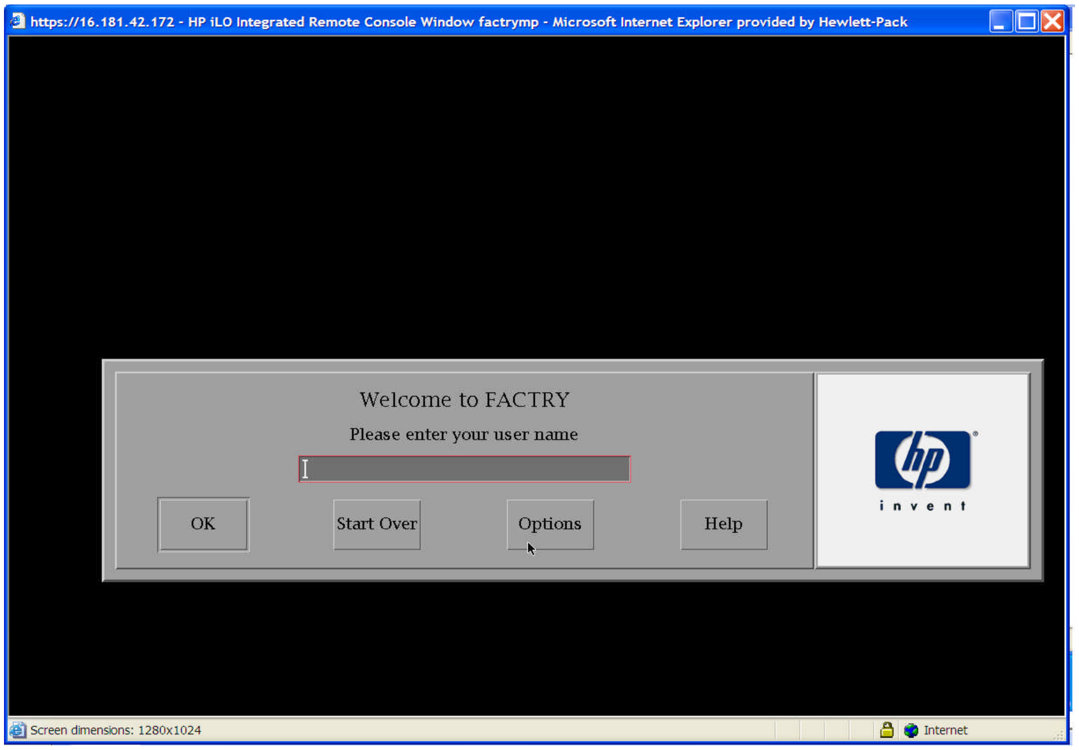
This paper describes the configuration and usage of vKVM on OpenVMS.

What is vKVM?

vKVM is a USB keyboard and mouse implemented in firmware by the management processor, which is designed to allow a remote user to control the system as if they were using VGA monitor and keyboard directly attached to the local system.

vKVM allows a remote PC to display the contents of a built-in graphics card on both the X11 and VGA consoles. Integrated console functionality allows pre-boot (EFI), OpenVMS boot, and runtime (both text console and DECwindows) to be displayed and interacted with from the iLO browser window.

DECwindows and the graphics console can be accessed and controlled from a remote Windows PC, which has a browser and can connect to the iLO webpage of the server. The output of the DECwindows screen will appear on a Windows server as shown:


https://16.181.42.172 - HP iLO Integrated Remote Console Window factrymp - Microsoft Internet Explorer provided by Hewlett-Pack
Welcome to FACTRY
Please enter your user name
OK
Start Over
Options
Help
invent
Screen dimensions: 1280x1024
Internet

The keyboard and mouse attached to the Windows server can be used to control the DECwindows screen and perform operations as and when required.

Why is vKVM required?

Physical access to the server has always been an issue due to security reasons. When performing GUI-based testing, an administrator's physical presence is required in front of the monitor to see the output and control the tests. As everything else can be controlled remotely, why not include graphics as well?

If there are multiple blades in an enclosure, there will be no reduction in the number of monitors, keyboards, and mice required to control and visualize the graphics. Even though the space required by the servers has been reduced, the space needed for accessories is not decreased.

vKVM helps to solve these issues.

Advantages of vKVM

- The server graphics console can be viewed and controlled just as if you were standing in front of the remote server. It is not necessary to be physically present in front of the monitor connected to the server to control the mouse or the keyboard. The mouse and keyboard connected to the remote Windows server can be used to control the input.
- The server can be accessed from any location on the same network.

- A controlled reset of the server, regardless of the state of the host operating system, should remain connected to monitor the reboot process. The EFI shell, VGA console and DECwindows screen can be controlled from the remote server.
- Multiple keyboards and mice connected to the server work on DECwindows and can be used to control the inputs to the server.
- vKVM works with a standard browser and no additional software is required on the remote server or client system.
- Multiple browsers can be opened to monitor multiple servers, which eliminates the need for multiple monitors.
- The data stream is encrypted, enabling secure monitoring and management of the server.

Prerequisites for vKVM

For Integrity entry-class servers, the advanced features are enabled with a license key. For Integrity cell-based servers, the advanced features are enabled with a PCI-X accessory card instead of a key.

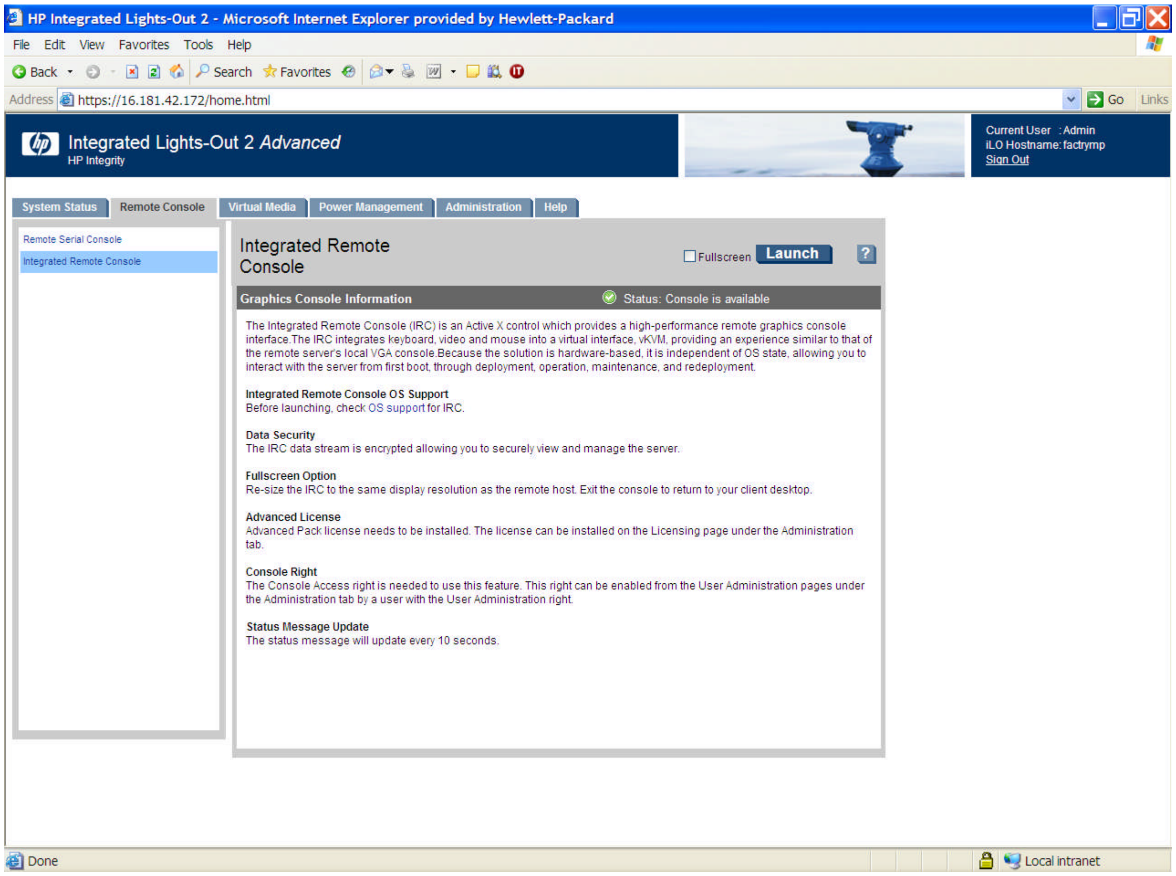
The Integrity Lights-Out Advanced (LOA) or KVM card is an optional accessory which combines a physical graphics or USB card with additional logic to enable the Lights-Out Advanced features of Virtual Media and Integrated Remote Console.

vKVM is supported on Windows clients running Internet Explorer 6.0 SP1 (minimum configuration).

Download and usage of signed ActiveX controls must be enabled. The IRC runs as an ActiveX control that is downloaded to Windows clients. The ActiveX control automatically downloads from iLO 2 on the initial client connection.

Steps to use vKVM

1. Open the Internet browser and enter the console IP address in the address bar.
2. Enter the login credentials for the console.
3. Click **Remote Console**.
4. Click on the **Integrated Remote Console** link on the left side of the web page.


HP Integrated Lights-Out 2 - Microsoft Internet Explorer provided by Hewlett-Packard
File Edit View Favorites Tools Help
Back Search Favorites
Address https://16.181.42.172/home.html
Go Links
Integrated Lights-Out 2 Advanced
HP Integrity
Current User : Admin
iLO Hostname: factrymp
Sign Out
System Status
Remote Console
Virtual Media
Power Management
Administration
Help
Remote Serial Console
Integrated Remote Console
Integrated Remote Console
Fullscreen
Launch
?
Done
Local intranet

Graphics Console Information Status: Console is available

The Integrated Remote Console (IRC) is an Active X control which provides a high-performance remote graphics console interface.The IRC integrates keyboard, video and mouse into a virtual interface, vKVM, providing an experience similar to that of the remote server's local VGA console.Because the solution is hardware-based, it is independent of OS state, allowing you to interact with the server from first boot, through deployment, operation, maintenance, and redeployment.

Integrated Remote Console OS Support
Before launching, check OS support for IRC.

Data Security
The IRC data stream is encrypted allowing you to securely view and manage the server.

Fullscreen Option
Re-size the IRC to the same display resolution as the remote host. Exit the console to return to your client desktop.

Advanced License
Advanced Pack license needs to be installed. The license can be installed on the Licensing page under the Administration tab.

Console Right
The Console Access right is needed to use this feature. This right can be enabled from the User Administration pages under the Administration tab by a user with the User Administration right.

Status Message Update
The status message will update every 10 seconds.

5. Click **Launch**.
6. To view the display in full-screen mode, select **Full Screen.**
   A new pop-up window appears with the DECwindows login screen.

Support in OpenVMS for vKVM

If vKVM is supported on the server, DECwindows will start without any keyboard and mouse attached to the system. (Earlier versions of OpenVMS had to wait for the keyboard and mouse to be connected to the server before DECwindows was started.) From OpenVMS Version 8.4 onwards, DECwindows starts automatically without any keyboard or mouse needing to be attached to the server.

As a result of changes for vKVM, additional devices will be displayed (e.g., KBX0, MOX0, KBD0, MOU0) on OpenVMS.

- The KBX0 and MOX0 are dummy devices that are used by OpenVMS.
- KBD0 and MOU0 are virtual keyboard and mouse, respectively, provided by the firmware used for vKVM.

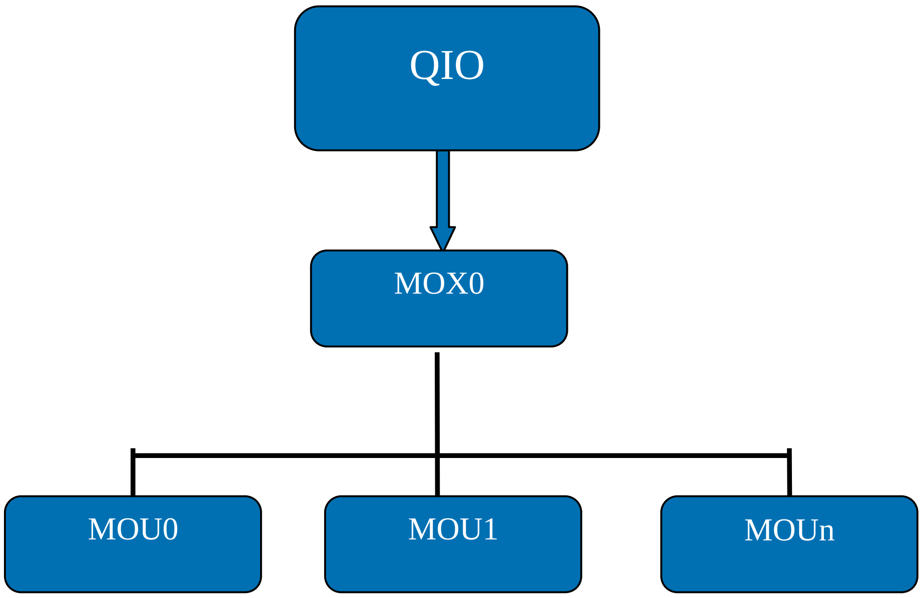
The input from the Windows server will be routed through these devices and the appropriate changes will be reflected on the screen.

This will also change the way QIO's are performed on keyboards and mouse devices.

For example, the following command will switch the middle and right buttons of all the mouse devices attached to the server:

```
DECW$CONFIG>SET BUTTON_MAP MOX0 0 1 4 5 2 3 6 7
```

This can be done because MOX and KBX devices are dummy devices that are used to direct the output to all the input devices attached to the server. If there are multiple mouse devices attached to the server, any operation done on MOX device will be fanned out to all the related devices (MOU1, MOU2, and so on), as shown below.



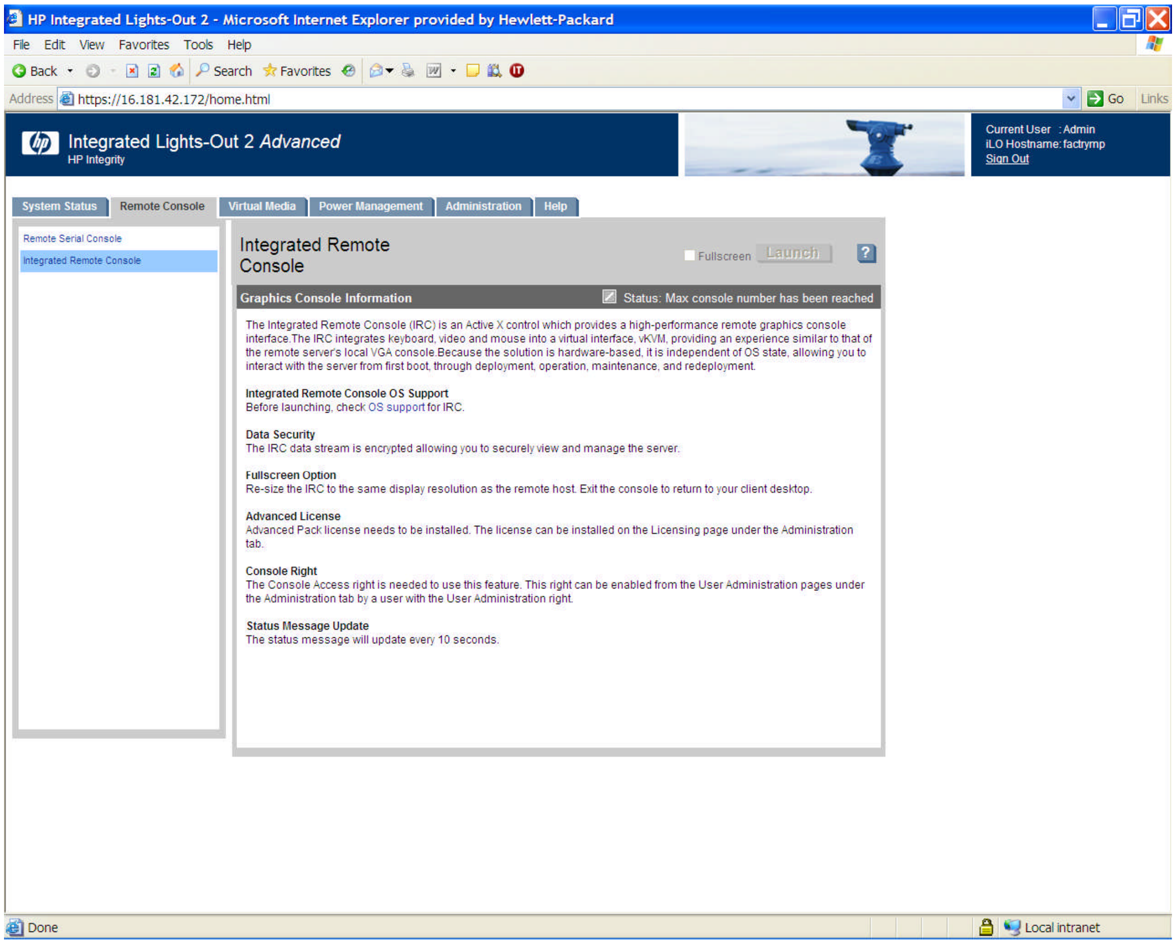
If an operation has to be performed on a specific device, the device name has to be mentioned in the command as shown in the command below:

```
DECW$CONFIG>SET BUTTON_MAP MOU2 0 1 4 5 2 3 6 7
```

Troubleshooting

Before starting vKVM, verify the following:

- Verify whether vKVM is available. Only one user can control the IRC at a time. If a remote console session already exists on the system, you are notified that IRC is unavailable. To determine if the vKVM or IRC is available, click the **Integrated Remote Console** link. If **Launch** is grayed out and the maximum console number has been reached, the following status message appears: “The remote console/IRC is in use by another client.”



Integrated Remote Console

Graphics Console Information

Status: Max console number has been reached

The Integrated Remote Console (IRC) is an Active X control which provides a high-performance remote graphics console interface.The IRC integrates keyboard, video and mouse into a virtual interface, vKVM, providing an experience similar to that of the remote server's local VGA console.Because the solution is hardware-based, it is independent of OS state, allowing you to interact with the server from first boot, through deployment, operation, maintenance, and redeployment.

Integrated Remote Console OS Support
Before launching, check OS support for IRC.

Data Security
The IRC data stream is encrypted allowing you to securely view and manage the server.

Fullscreen Option
Re-size the IRC to the same display resolution as the remote host. Exit the console to return to your client desktop.

Advanced License
Advanced Pack license needs to be installed. The license can be installed on the Licensing page under the Administration tab.

Console Right
The Console Access right is needed to use this feature. This right can be enabled from the User Administration pages under the Administration tab by a user with the User Administration right.

Status Message Update
The status message will update every 10 seconds.

- Verify that you have console access permission on the **User Administration** page, or if that right must be granted.
- Verify that the system is licensed for IRC use. To view this information, go to **Administration Licensing** tab.
- Disable any pop-up-blocking applications, which prevent IRC from running.
- Verify that the resolution of DECwindows is set to 1024 X 768 or below.
- Some key combinations might not work as expected. For example, the auto-repeat key functionality cannot be disabled or enabled on the remote keyboard. All the limitations with the key combinations are limited to the remote mouse and keyboard and not to locally attached devices.
- Multihead display will not be transported over the network. The display from the onboard graphics card will be transported over the network and will be available at the remote Windows server.
- The list of supported browsers are available at http://www.hp.com/go/integrityilo.

Conclusion

The vKVM solution assists DECwindows customers by enabling remote access to OpenVMS desktop from any location, thereby eliminating the requirement to be physically present at the servers.

BL8x0c i2: Overview, Setup, Troubleshooting, and Various Methods to Install OpenVMS

Aditya BS and Srinivas Pinnika

Intended Audience

This paper is intended for OpenVMS system administrators who are planning to install and run OpenVMS on the BL8x0c i2 hardware. It provides quick reference information, as well as details on how to install and configure various hardware components, troubleshoot, and configure the Integrity i2 server blades.

Abbreviations

CLI	Command Line Interface
CM	Command Menu
DIMM	Dual-in Line Memory Module
FC	Fiber Channel
GUI	Graphical User Interface
iLO	Integrated Lights Out
ICH	I/O Controller Hub
LOM	LAN On Motherboard
LUN	Logical Unit Number
OA	Onboard Administrator
ORCA	Option ROM Configuration for Arrays
SAS	Serial Attached SCSI
SCSI	Small Computer System Interface
SUV	Serial, USB and VGA
USB	Universal Serial Bus
VC	Virtual Connect
VCM	Virtual Connect Manager

Introduction to HP Integrity BL8x0c i2 Server Blade

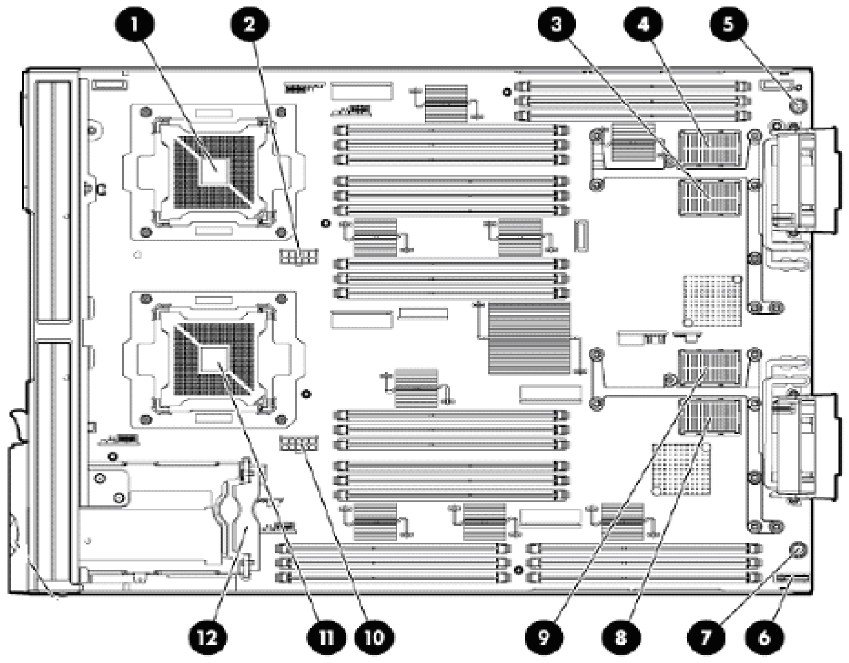
We know that blades have advantages with respect to more processing power in less space, simplified cabling, storage, maintenance and shared resources. A single enclosure can house multiple servers that share power source and other components, and also consolidate related resources, such as storage and networking equipment, into a smaller architecture than would be the case with a farm of regular servers.

BL8x0c i2 is a full-height Integrity server blade with the Intel Itanium 9300 processor series. The family of BL8x0c i2 products includes single blade 2S (BL860c i2), dual blade 4S (BL870c i2), and quad blade 8S (BL890c i2) variants (where S represents the number of CPU sockets), each capable of running a single operating system image across all blades/sockets. The Intel Itanium 9300 processor is available both as quad-core and also as dual-core.

BL860c i2
BL860c i2
BL860c i2 in c3000 Enclosure
BL860c i2 in c7000 Enclosure
BL870c i2
BL870c i2
BL870c i2 in c3000 Enclosure
BL870c i2 in c7000 Enclosure
BL890c i2
BL890c i2
BL890c i2 in c3000 Enclosure
BL800c i2 in c7000 Enclosure

BLADE	SOCKETS (Cores)	DIMM Slots	Max. Memory Support (with 8GB DIMM's)	No of PCIe Mezz Cards	No of SAS Hard Disks
BL860c i2	2S (8 Cores)	24	192 GB	3	2
BL870c i2	4S (16 Cores)	48	384 GB	6	4
BL890c i2	8S (32 Cores)	96	768 GB	12	8

BL860c i2 Blade Components



1	CPU 0
2	CPU 0 POWER CONNECTOR
3	TYPE 1 MEZZANINE CONNECTOR SLOT 1
4	TYPE 1 or 2 MEZZANINE CONNECTOR SLOT 2
5	THUMBSCREW 1
6	BATTERY
7	THUMBSCREW 2
8	ICH MEZZANINE CONNECTOR SLOT
9	TYPE 1 or 2 MEZZANINE CONNECTOR SLOT 3
10	CPU 1 POWER CONNECTOR
11	CPU 1
12	SAS BACK PLANE

For complete preparation and installation guidelines for the BL8x0c i2 server in the c-class enclosure, see the *HP Integrity BL860c i2, BL870c i2 & BL890c i2 Server Blade User Service Guide*:
http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c02110937/c02110937.pdf

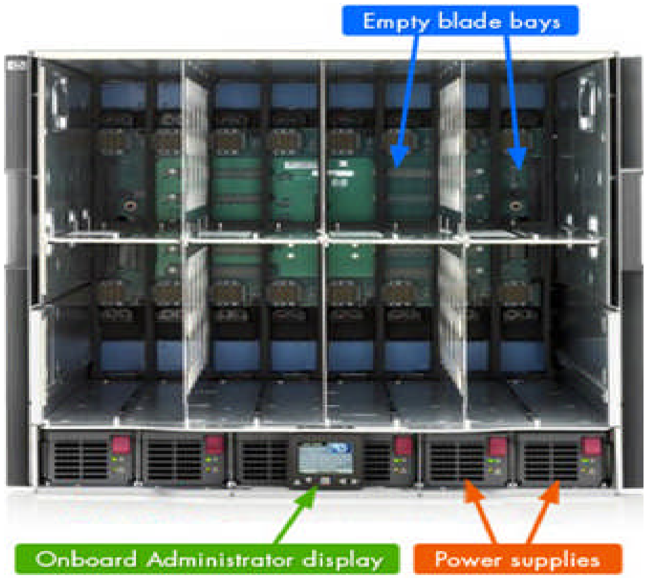
c-CLASS ENCLOSURES

HP offers two versatile c-Class enclosure models:

- HP Blade System c7000 Enclosure
- HP Blade System c3000 Enclosure

The c7000 provides 16 device bays and eight interconnect module bays in a 10U rack-mount configuration.

C7000 Enclosure Front View



C7000 Enclosure Rear View

C3000 Enclosure Front View

The c3000 provides eight device bays and four interconnect module bays in a 6U rack-mount or tower configuration.

C3000 Enclosure Rear View

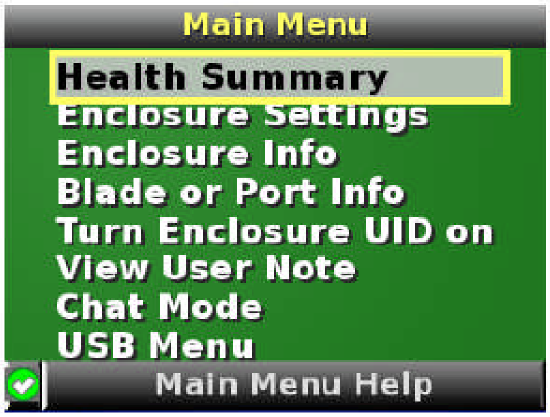
Both enclosure models also include the Onboard Administrator and the Insight Display diagnostic panel. They use the same hardware, software, and processes for management.

Onboard Administrator

The core of c-class enclosure management is the Blade System Onboard Administrator module. It performs four management functions for the entire enclosure:

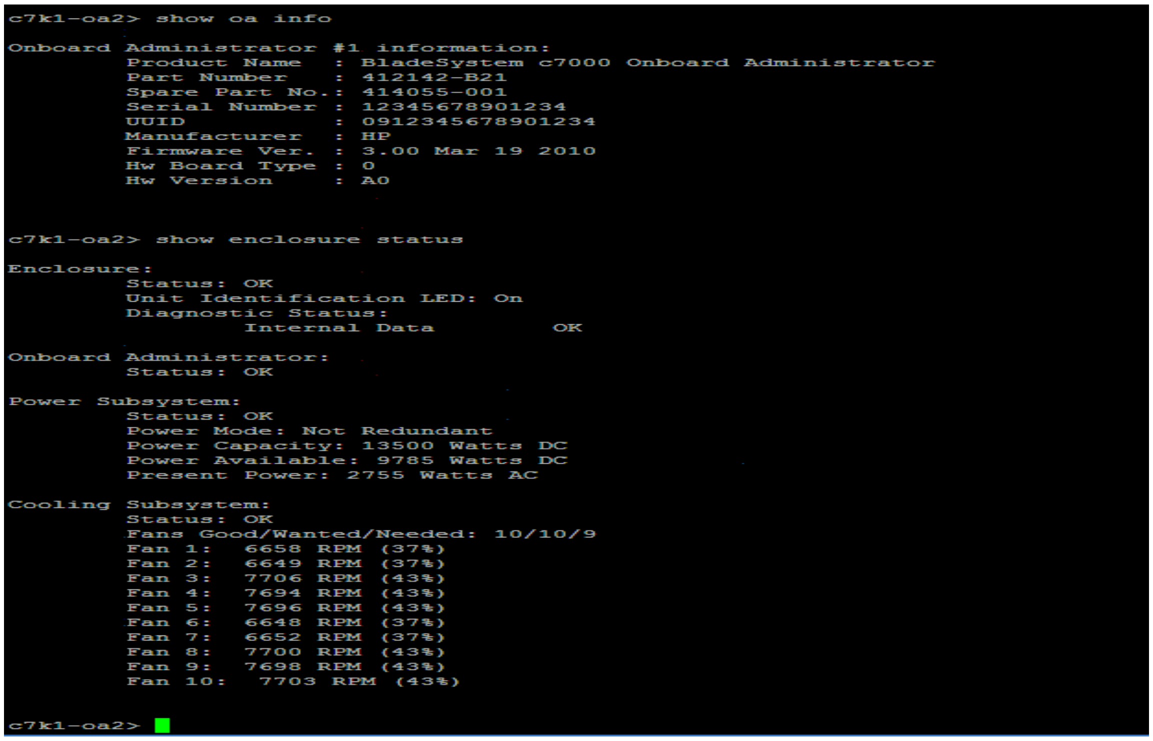
- Detect component insertion and removal
- Identify components and required connectivity
- Manage power and cooling
- Control components

The following methods are different ways for technicians and administrators to access the Onboard Administrator:
An Insight Display screen on each HP Blade System c-class enclosure provides ready access for quick setup and daily maintenance.



Onboard Administrator Command Line Interface (CLI)

The Onboard Administrator CLI allows administrators to use serial, telnet, or SSH connections to control enclosure and device operation, including the use of scripts for automation. CLI commands include commands to connect to the iLO on each compute blade and to any supported interconnect module management processors such as Virtual Connect.

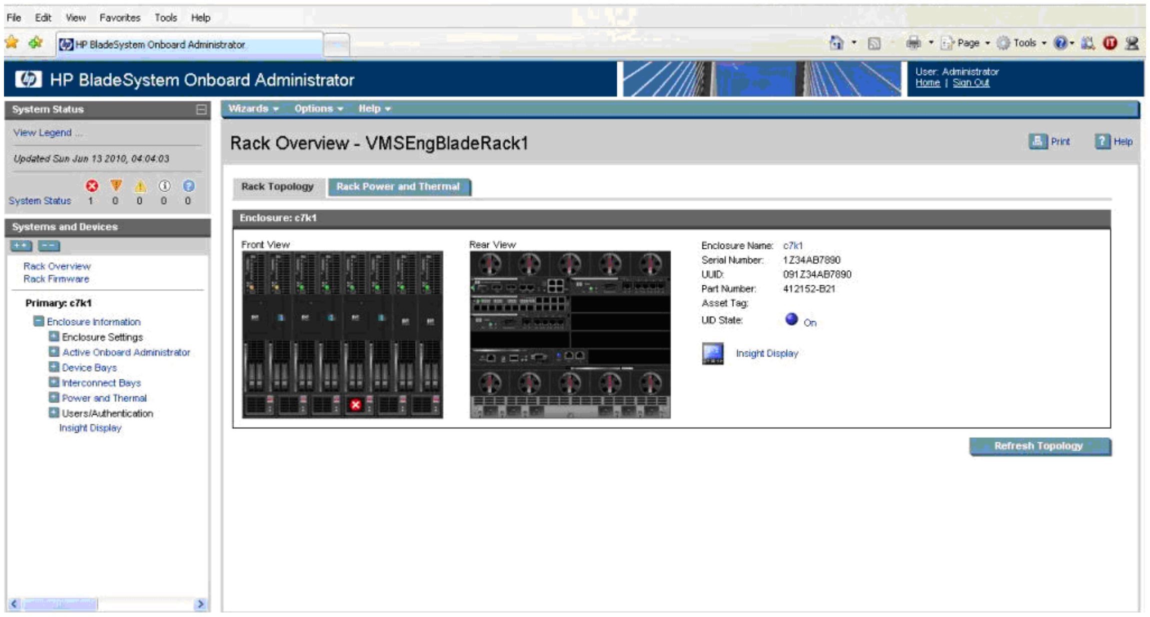


```
c7k1-oa2> show oa info

Onboard Administrator #1 information:
        Product Name   : BladeSystem c7000 Onboard Administrator
        Part Number    : 412142-B21
        Spare Part No. : 414055-001
        Serial Number  : 12345678901234
        UUID           : 0912345678901234
        Manufacturer   : HP
        Firmware Ver.  : 3.00 Mar 19 2010
        Hw Board Type  : 0
        Hw Version     : A0


c7k1-oa2> show enclosure status

Enclosure:
        Status: OK
        Unit Identification LED: On
        Diagnostic Status:
                Internal Data        OK

Onboard Administrator:
        Status: OK

Power Subsystem:
        Status: OK
        Power Mode: Not Redundant
        Power Capacity: 13500 Watts DC
        Power Available: 9785 Watts DC
        Present Power: 2755 Watts AC

Cooling Subsystem:
        Status: OK
        Fans Good/Wanted/Needed: 10/10/9
        Fan 1:   6658 RPM (37%)
        Fan 2:   6649 RPM (37%)
        Fan 3:   7706 RPM (43%)
        Fan 4:   7694 RPM (43%)
        Fan 5:   7696 RPM (43%)
        Fan 6:   6648 RPM (37%)
        Fan 7:   6652 RPM (37%)
        Fan 8:   7700 RPM (43%)
        Fan 9:   7698 RPM (43%)
        Fan 10:   7703 RPM (43%)

c7k1-oa2>
```

Onboard Administrator Graphical User Interface

The GUI provides remote administration capabilities from a desktop web browser. The GUI allows administrators to simplify tasks such as managing users and network settings, virtual power control, boot order control, and enclosure DVD attachment to one or more blades. The GUI can also simplify administrative tasks when identical operations are performed on multiple compute blades.

The GUI contains graphical views for server-to-interconnect port mapping, zone cooling measurements, and power use history. At a glance, the administrator can tell if any devices in the enclosure need attention. If multiple enclosures in a rack are connected using the enclosure links, the administrator can view and control one or more enclosures from a single GUI.



Updated Sun Jun 13 2010, 04:04:03

System Status 1 0 0 0 0

Primary: c7k1

Rack Overview - VMSEngBladeRack1

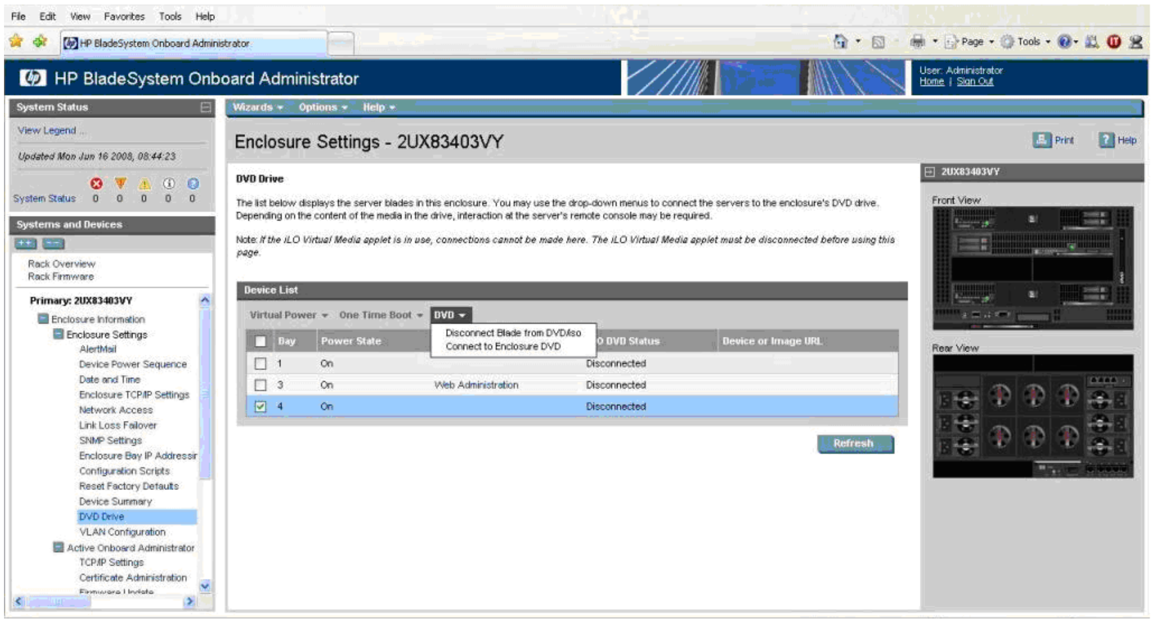
Enclosure: c7k1

Enclosure Name:	c7k1
Serial Number:	1Z34AB7890
UUID:	091Z34AB7890
Part Number:	412152-B21
Asset Tag:	
UID State:	On

Enclosure DVD

The Onboard Administrator (OA) can provide USB CD/DVD drive connectivity to one or more servers in an enclosure with the enclosure DVD feature. In addition, with a USB key plugged into the Onboard Administrator, ISO files can be connected to one or more servers, the OA firmware can be updated from a file, and the enclosure configuration can be saved or restored from a file on the USB key. This feature can dramatically simplify the firmware update of all servers, the Onboard Administrator modules, or initial setup of an enclosure from a custom configuration file.

DVD window access from OA Web Interface



Enclosure Settings - 2UX83403VY

DVD Drive

The list below displays the server blades in this enclosure. You may use the drop-down menus to connect the servers to the enclosure's DVD drive. Depending on the content of the media in the drive, interaction at the server's remote console may be required.

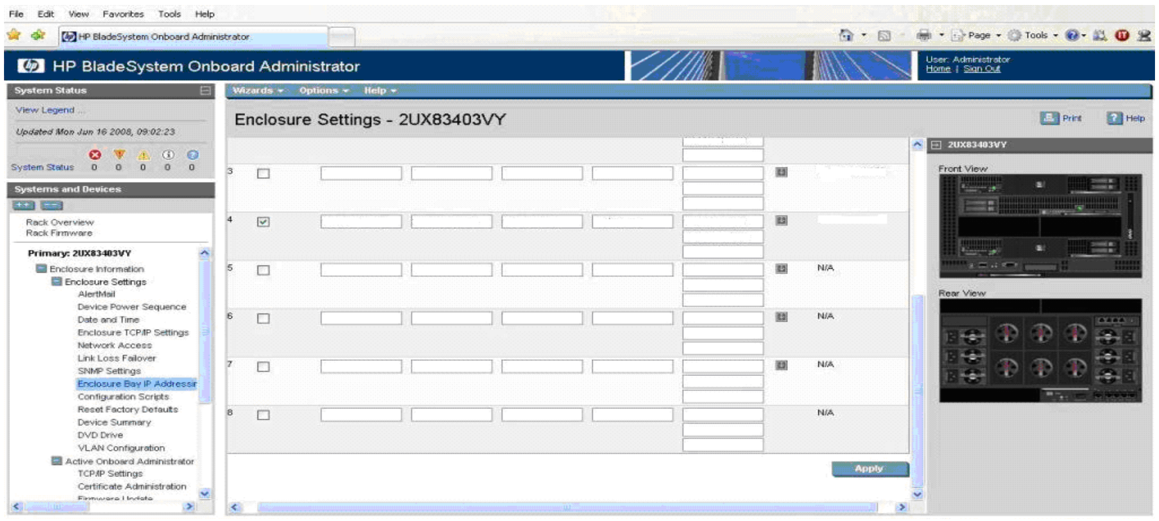
Note: *If the iLO Virtual Media applet is in use, connections cannot be made here. The iLO Virtual Media applet must be disconnected before using this page.*

Device List

	Bay	Power State		DVD Status	Device or Image URL
☐	1	On		Disconnected	
☐	3	On	Web Administration	Disconnected	
✓	4	On		Disconnected	

Enclosure Bay IP Addressing

The Onboard Administrator significantly enhances the management network infrastructure by offering a single point to assign IP addresses to the compute blade iLO management ports and the interconnect module management ports.



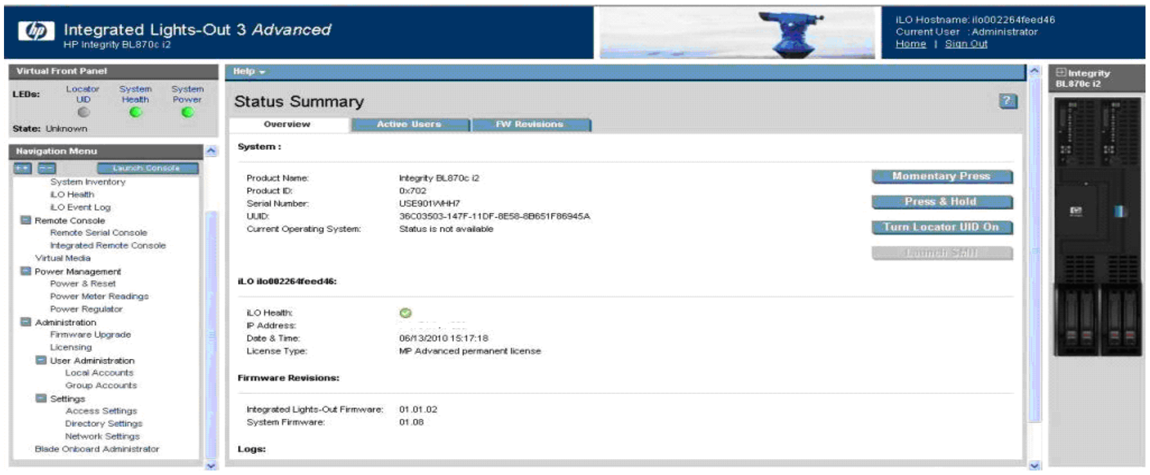
Enclosure Settings - 2UX83403VY

3	☐	
4	✓	
5	☐	N/A
6	☐	N/A
7	☐	N/A
8	☐	N/A

Integrated Lights-Out (iLO) Web Interface

HP Integrity Integrated Lights-Out (iLO) simplifies remote management of your Integrity servers from anywhere in the world.

Shown below is the iLO web interface:



iLO Hostname: ilo002264feed46
Current User : Administrator

State: Unknown

System :

Product Name:	Integrity BL870c i2
Product ID:	0x702
Serial Number:	USE901WHH7
UUID:	36C03503-147F-11DF-8E58-8B651F86945A
Current Operating System:	Status is not available

iLO ilo002264feed46:

iLO Health:	✓
IP Address:	[illegible]
Date & Time:	06/13/2010 15:17:18
License Type:	MP Advanced permanent license

Firmware Revisions:

Integrated Lights-Out Firmware:	01.01.02
System Firmware:	01.08

Logs:

iLO can be used to connect to system serial console using “remote serial console” option and also install the operating system using vMedia option. iLO also supports Integrated Remote Console which provides a high-performance graphical remote console.

Prerequisites

To setup or to configure the HP Integrity BL8x0c i2 server blades, the following prerequisites for hardware, firmware and software must be met:

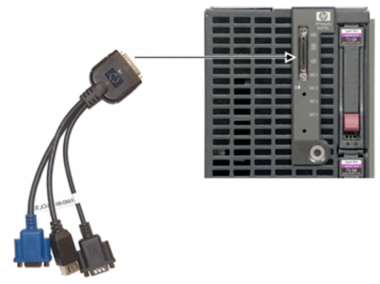
- C7000 and/or C3000 c-class enclosures.
- Onboard Administrator firmware – recommended version
- BL8x0c i2 Blade and required/necessary IO cards have to be updated with the latest supported firmware version.
- SUV Cable for Serial, USB and VGA connectivity.
- Respective IP’s (OA, MP LAN) have to be configured.
- OpenVMS V8.4 for Integrity server kit in ISO or BCK format.
- HP DVD burnt with OpenVMS V8.4 ISO.
- Update kits for OpenVMS V8.4 for Integrity servers.

Overview of BL8x0c i2 server blades Setup

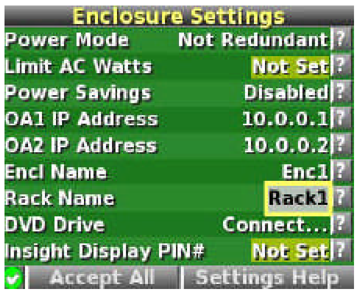
To setup the BL8x0c i2 server blades, follow these steps regarding logging into OA and upgrading OA firmware:

1. Plug the BL8x0c i2 blade to the C7000 or C3000 enclosure.

2. Connect the SUV cable to the SUV port on the Server Blade. Access the blade using the serial port (RS232 port) on the SUV cable, which is the cable with 3 ports or serial, USB and VGA.



3. Assign the Onboard Administrator (OA) IP address on the OA console present on c-Class enclosure.
4. Enter the username and password from the tag supplied with the OA module to access the remote OA web interface and complete the OA first-time installation wizard.



Enclosure Settings

Power Mode	Not Redundant
Limit AC Watts	Not Set
Power Savings	Disabled
OA1 IP Address	10.0.0.1
OA2 IP Address	10.0.0.2
Encl Name	Encl
Rack Name	Rack1
DVD Drive	Connect...
Insight Display PIN#	Not Set

5. The enclosure information screen displays information about the enclosure, including the following details:
    - Active OA IP address
    - Active OA service IP address
    - Current health status of the enclosure
    - Current enclosure ambient temperature
    - Current AC input power to the enclosure
    - Enclosure name
    - Rack name

For detailed procedure, see the HP BladeSystem c7000/c3000 Enclosure Setup and Installation Guide:
C7000:
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00698286/c00698286.pdf
C3000:
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01167165/c01167165.pdf

6. OA firmware version can be obtained either from the web interface of the OA or from the OA's Command Line Interface (TELNET session) using the SHOW OA INFO command.
7. Based on the information displayed, it can be upgraded to the recommended firmware version. For more information about OA firmware and installation instructions, see the following website: http://h20000.www2.hp.com/bizsupport/TechSupport/SoftwareIndex.jsp?lang=en&cc=us&prodNameId=4204753&prodTypeId=3709945&prodSeriesId=4186428&swLang=13&taskId=135&swEnvOID=4094

Logging into MP and configuring MP LAN:

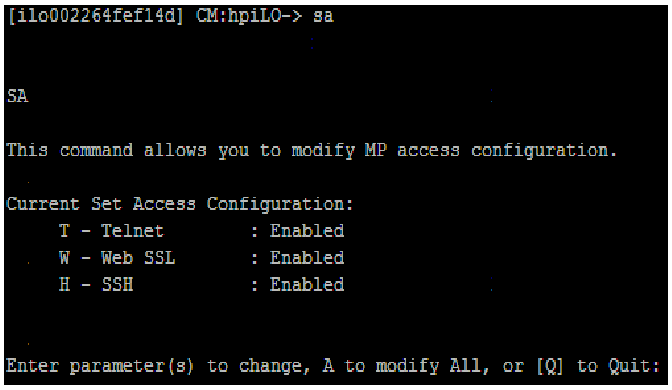
1. Login to the blade MP using the username and password present on the blade server tag. Type CM to reach the MP command menu and create a new user with the necessary access setting.
2. From the MP command menu issue the command LC (LAN Configuration) to configure the MP LAN. If you do not have a DHCP server on your local network, and are assigning a static IP address to the server blade, you must disable DHCP on the server blade. To add static IP address, subnet masks, and gateway address, follow these steps:
   NOTE: Obtain the required addresses from a system administrator

   a. From the MP Main Menu, enter the "CM" command, which brings up the MP command menu screen is displayed.
   b. Enter the "LC" command, LC Menu screen is displayed. At each prompt you may type DEFAULT to set the default configuration or Q to Quit.

   Default LAN Configuration:
```
     - - MAC Address                         : 0x00110aa50058
     D - DHCP Status                         : Disabled
     I - IP Address                          : 0.0.0.0
     M - MP Host Name                        : ilob5122
     S - Subnet Mask                         : 0.0.0.0
     G - Gateway Address                     : 0.0.0.0
     L - Link State                          : Auto Negotiate
     O - Duplex Option                       : n/a
     R - Remote Serial Console Port          : 2023
     H - SSH Access Port                     : 22
Enter parameter(s) to change. A to modify All. Or (Q) to
Quit:
```
   c. From the LC Menu, you can select any of the options as follows:
      - Enter "d" to access the DHCP Status screen and follow the onscreen instructions to change the DHCP status from Enabled to Disabled.
      - Enter "Y" to confirm the DHCP status change.
      - Enter "I" to access the IP Address screen and follow the onscreen instructions to add static IP address obtained from the system administrator.

- Enter “S” to access the subnet mask screen and follow the onscreen instructions to add subnet mask address obtained from the system administrator.
- Enter “G” to access the gateway address screen and follow the onscreen instructions to add gateway address obtained from system administrator.
- Verify the configuration and confirm the changes.
- The server blade is now set up for remote access with static IP address.

3. Enable TELNET access using the command “SA” command.



```
[ilo002264fef14d] CM:hpiLO-> sa


SA

This command allows you to modify MP access configuration.

Current Set Access Configuration:
     T - Telnet         : Enabled
     W - Web SSL        : Enabled
     H - SSH            : Enabled


Enter parameter(s) to change, A to modify All, or [Q] to Quit:
```

4. Reset the MP using the XD -R –NC command. You can access MP/iLO by using the web interface and also by using the TELNET session on the Remote console.

Upgrading MP firmware:

1. From the MP Command Menu (CM), check the blade firmware version using the command “SR” or “SYSREV” command. Upgrade the firmware to the recommended version and also ensure that the firmware for the respective IO Cards (LOM, SAS and so on) are the latest versions.
2. For the steps to obtain and update the firmware, see: http://h20000.www2.hp.com/bizsupport/TechSupport/SoftwareIndex.jsp?lang=en&cc=us&prodNameId=4204753&prodTypeId=3709945&prodSeriesId=4186428&swLang=13&taskId=135&swEnvOID=54

Getting to EFI and configuring LUN:

1. POWER ON the system and get to the SHELL prompt.
2. MAP -R command on the Shell> provides the list of disks (BLKx device). If no disks are detected, the LUNs are not created.
3. To create LUNs use the DRVCFG command provided by ORCA utility. For a detailed procedure on how to use DRVCFG command, see section *“BL8x0c i2 Troubleshooting Methods”*.

4. After the LUN's are created, and the controller is restarted, the shell "MAP -R" command must display the BLKx devices.
5. Install OpenVMS using one of the following methods as described in Section *"Methods to Install OpenVMS"*.

Methods to Install OpenVMS

OpenVMS can be installed on an Integrity server system using any one of the method as explained below.

vMedia

For a detailed description on how to install OpenVMS using vMedia, see the *HP OpenVMS Version 8.4 Upgrade and Installation Manual* at:
http://h71000.www7.hp.com/doc/84final/ba322_90087/ba322_90087.pdf

External or Internal USB DVD Drive

Note: C3000 enclosure has an internal USB DVD Drive which can be connected to a particular blade using the Onboard Administrator.

The C7000 enclosure does not have an internal DVD drive, but an external USB DVD drive can be connected to a USB port via a SUV cable. This method can be used on C3000 enclosure as well.

Insert the OpenVMS V8.4 DVD into the respective DVD drive and boot the system with the DVD, which provides the options to install or upgrade OpenVMS.

InfoServer

For detailed description on how to setup the InfoServer and install OpenVMS using InfoServer, see the *HP OpenVMS Version 8.4 Upgrade and Installation Manual* at:
http://h71000.www7.hp.com/doc/84final/ba322_90087/ba322_90087.pdf

The following examples below explain how to perform the InfoServer boot.

Example 1: InfoServer boot with memory disk and directed lanboot and without DBPROFILE:

```
SHELL> LANBOOT SELECT -SIP <SERVER IP> -CIP <CLIENT IP> -GIP <GATEWAY
IP> -M <SUBNET_MASK> -B "<FULL PATH TO THE BOOT FILE ON THE BOOT
SERVER> "-OD "-FL 0,200400 -SERVICE <SERVICE NAME>"
```

Example 2: InfoServer boot without memory disk and with directed lanboot (by choosing services)

```
SHELL> LANBOOT SELECT -SIP <SERVER IP> -CIP <CLIENT IP> -GIP <GATEWAY
IP> -M <SUBNET MASK> -B "<FULL PATH TO THE BOOT FILE ON THE BOOT
SERVER>"
```

VMS Kit Disk on SAS Disk

Copy the OpenVMS saveset on to BL8x0c i2 server blade running OpenVMS and perform the following steps. For example, let us assume DKA0: as a target device (VMS kit disk).
NOTE: This method can be used on a system that already has OpenVMS installed and running.

1. Foreign mount the target disk on which the kit disk has to be created using the following command:
   `$ MOUNT /FOREIGN DKA0:`
2. Take aback up of the OpenVMS V8.4 saveset onto the target disk using the following command:
   `$ BACKUP /IMAGE I64V84.BCK /SAVE DKA0:`
3. After the backup is complete, dismount the target disk using:
   `$ DISMOUNT DKA0`

Now the target disk is ready to be used as a kit disk. Boot the system with the newly created kit disk, which provides the options to install or upgrade OpenVMS.

VMS Kit Disk on a USB Flash Drive/Disk

Note: USB flash drives/disks are not supported on OpenVMS. However, some makes of flash drives are found to work with OpenVMS. The minimum size of the USB flash drive/disk required for this procedure must be 5 GB. HP recommends using this method if none of the above options are feasible.

To create a VMS kit disk on a USB flash drive/disk, you can use one of the following methods:

Method 1

Copy the OpenVMS V8.4 saveset onto system running OpenVMS and perform the following steps. Plug the USB flash drive/disk to the USB port. This device will be configured as DNxx device.

NOTE: This is only possible on a system that already has OpenVMS installed and running. If this method needs to be used on a fresh system, then the kit disk needs to be created on another system running OpenVMS.

For example, let us assume DNA0: as a target device (VMS kit disk).

1. Mount the USB flash drive/disk (DNA0:) which can be made bootable (kit disk) as foreign, using the following command:
   `$ MOUNT /FOREIGN DNA0:`

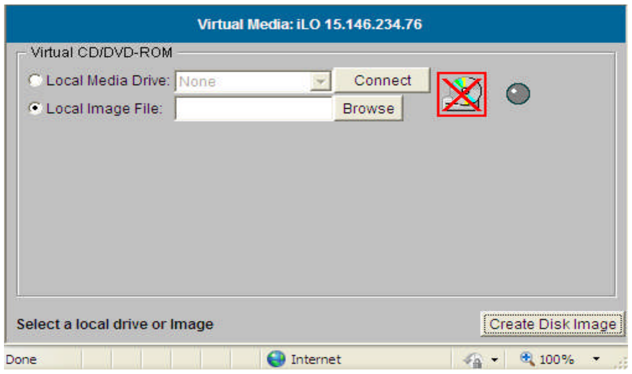
2. Take the back up onto the target disk by using the following command:
```
$ BACKUP /IMAGE I64V84.BCK /SAVE DNA0:
```
3. After the backup is complete, dismount the target disk using:
```
 $ DISMOUNT DNA0
```

Now USB kit disk is ready. Plug the USB flash drive/disk (kit disk) to the USB port on the SUV of the target BL8x0c i2 system. Boot the system using the USB flash drive/disk (kit disk), which provides the options to install or upgrade OpenVMS.

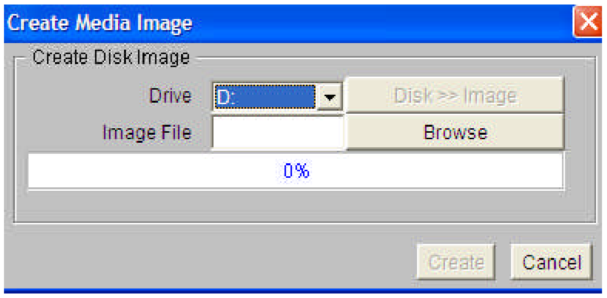
Method 2

Place the OpenVMS V8.4 ISO on the Windows desktop and then plug in the USB flash drive/disk and perform the following steps.

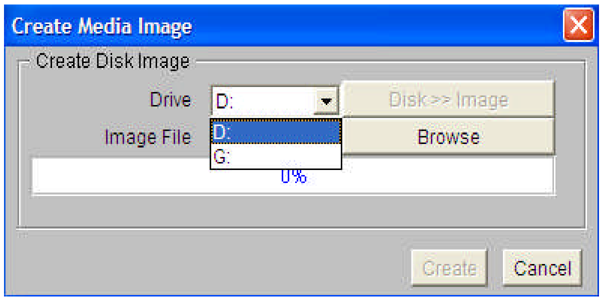
1. Login to the iLO web interface and launch vMedia.
2. Click Create Disk Image on the vMedia window.



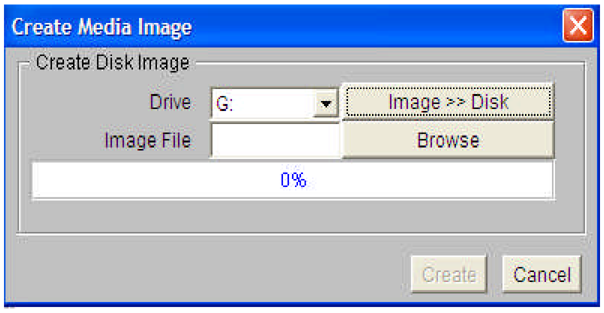
The Create Media Image window appears:



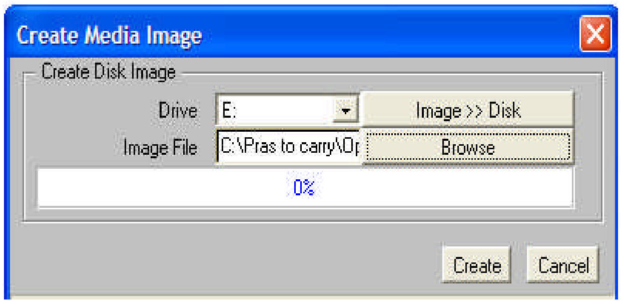
3. Click Drive menu and select the USB Drive as shown below, where, G: is the USB flash drive/disk.



4. To create VMS kit disk on a USB flash drive/disk from an OpenVMS V8.4 ISO image file, click the Disk>>Image to change it to Image>>Disk.
5. Click Browse and specify the location of the OpenVMS V8.4 ISO image file.



6. Click Create to start the preparation of VMS kit disk on a USB flash drive/disk.



7. Plug the USB (kit disk) to the BL8x0c i2 and boot the system, which provides options to install or upgrade OpenVMS.

NOTE: Booting the system either with VMS kit disk or DVD or vMedia or InfoServer -- as explained above -- provides the following options to install or upgrade the OpenVMS.

You can install or upgrade the OpenVMS I64 operating system or you can install or upgrade layered products that are included on the OpenVMS I64 distribution media (CD/DVD).

You can also execute DCL commands and procedures to perform "stand-alone" tasks, such as backing up the system disk.

Choose one of the following:

1. Upgrade, install, or reconfigure OpenVMS I64 Version V8.4
2. Display layered products that this procedure can install
3. Install or upgrade layered products
4. Show installed products
5. Reconfigure installed products
6. Remove installed products
7. Find, install, or undo patches, as well asshow or delete recovery data
8. Execute DCL commands and procedures
9. Shut down this system

Enter CHOICE or? for help: (1/2/3/4/5/6/7/8/9/?)

For detailed installation or upgrade procedure information, see the *HP OpenVMS Version 8.4 Upgrade and Installation Manual* at:
http://h71000.www7.hp.com/doc/84final/ba322_90087/ba322_90087.pdf

Virtual Connect Technology

Introduction to Virtual Connect

Virtual Connect is the next step in virtualization that extends the benefits of virtualization beyond the server to other infrastructure.

Virtual Connect accomplishes this by defining a server connection profile for each server in the enclosure before the server is installed. These defined profiles establish MAC addresses for all the network adapters, world wide numbers for all the host bus adapters and SAN boot parameters.

Previously, network administrators only had two options to connect each of the servers to the outside world, either through pass-through or switches. With HP Virtual Connect Architecture, they will be able to manage the entire blade architecture, thereby reducing the burden of managing hundreds of switches and cables and thus reducing the cost.

HP Virtual Connect Enterprise Manager

HP Virtual Connect Enterprise Manager (VCEM) is a management option for multiple BladeSystem enclosures configured with Virtual Connect. VCEM provides a central console that aggregates Virtual Connect resources, improves productivity, and enables faster response to changing data center workload demands. It also provides a single console to manage up to 800 BladeSystem enclosures and runs on Virtual Connect Enet Module.

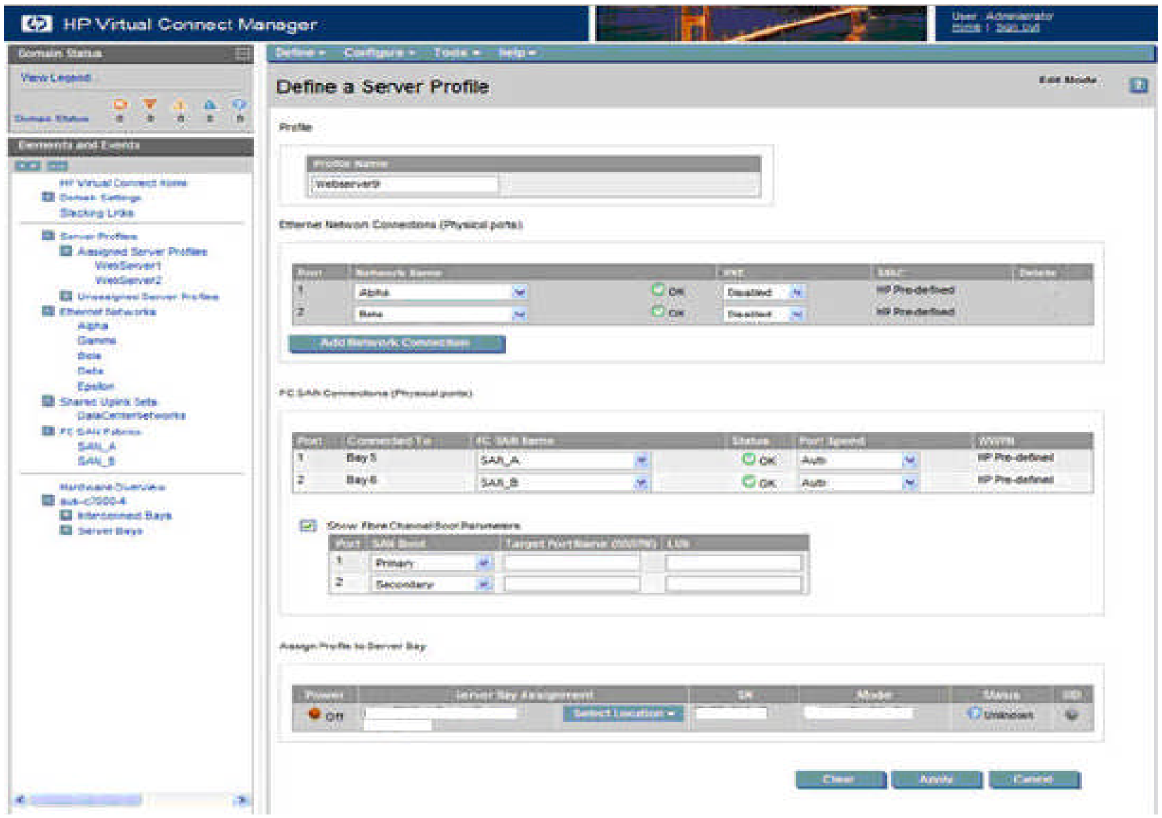
HP Virtual Connect Flex-10 Technology

Virtual Connect Flex-10 technology builds even more flexibility into each server blade to add up to 4 times as many NIC connections, fine tuning the bandwidth for each connection. It helps remove up to 75 percent of server edge infrastructure, lower purchase costs by up to 66 percent, and reduce power consumption up to 50 percent. It also enables you to transform your infrastructure and uncover its full potential.

Virtual Connect Manager

The VC Manager contains utilities and profile wizards that allows the system administrators to create and allocate server connection profiles to servers. The server profiles include the Ethernet MAC addresses and Fiber Channel HBA world wide names or WWNs and SAN boot configurations.

Here is the sample screen shot of the VCM:



Define a Server Profile

Profile

Profile Name
Webserver9

Ethernet Network Connections (Physical ports)

Port	Network Name		PXE	MAC	Delete
1	Alpha	OK	Disabled	HP Pre-defined	
2	Beta	OK	Disabled	HP Pre-defined	

FC SAN Connections (Physical ports)

Port	Connected To	FC SAN Name	Status	Port Speed	WWPN
1	Bay 5	SAN_A	OK	Auto	HP Pre-defined
2	Bay 6	SAN_B	OK	Auto	HP Pre-defined

Show Fibre Channel Boot Parameters

Port	SAN Boot	Target Port Name (WWPN)	LUN
1	Primary		
2	Secondary		

Assign Profile to Server Bay

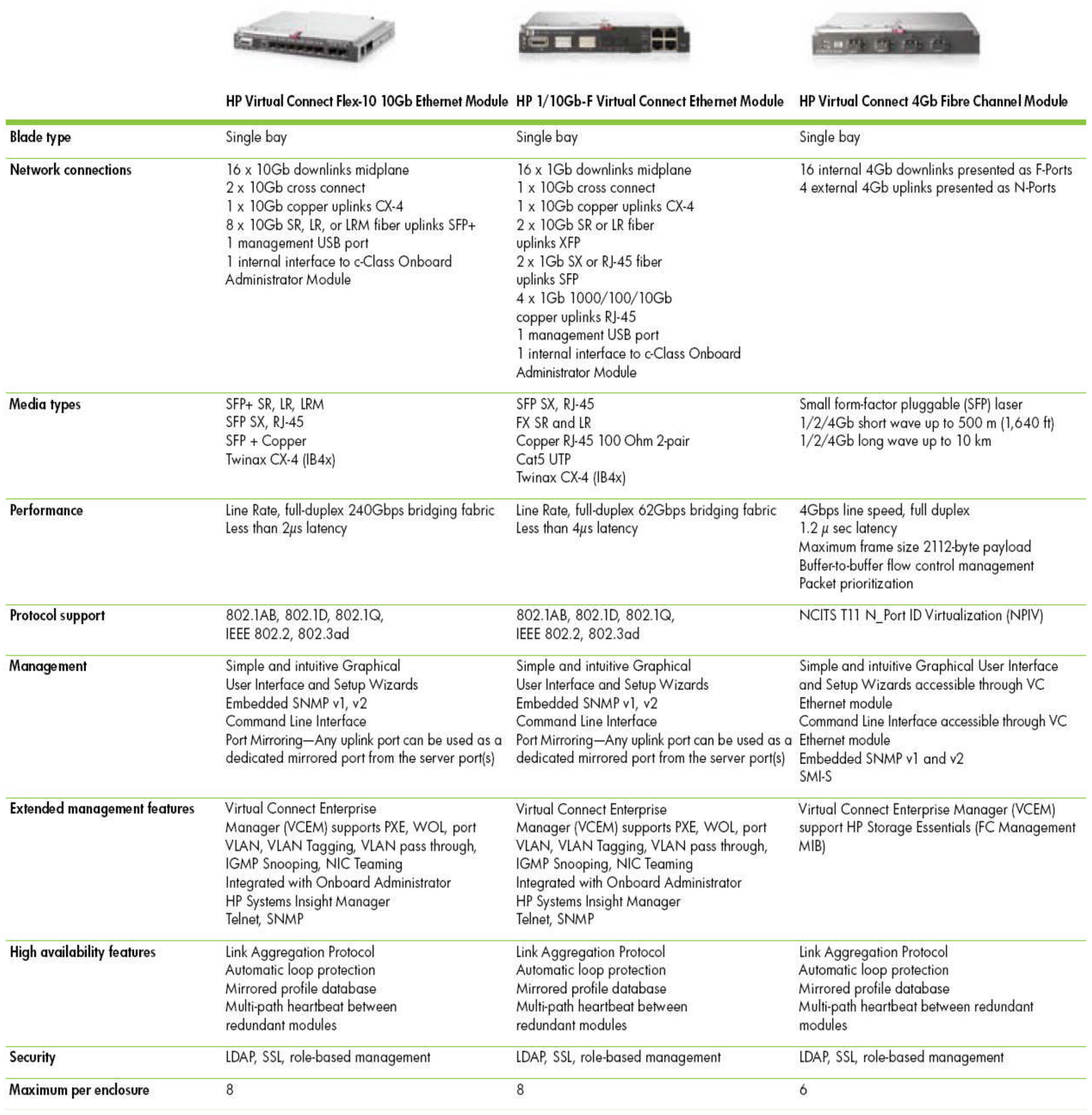
Power	Server Bay Assignment	SN	Model	Status	UID
Off				Unknown	

Virtual Connect Firmware

For more information about Virtual Connect firmware and upgrade procedures, see the following website:
http://h20000.www2.hp.com/bizsupport/TechSupport/SoftwareIndex.jsp?lang=en&cc=us&prodNameId=4204762&prodTypeId=3709945&prodSeriesId=4186432&swLang=13&taskId=135&swEnvOID=4001#29154

Virtual Connect Ethernet Modules

This section provides a list of various VC Ethernet modules and their specifications.



	HP Virtual Connect Flex-10 10Gb Ethernet Module	HP 1/10Gb-F Virtual Connect Ethernet Module	HP Virtual Connect 4Gb Fibre Channel Module
Blade type	Single bay	Single bay	Single bay
Network connections	16 x 10Gb downlinks midplane 2 x 10Gb cross connect 1 x 10Gb copper uplinks CX-4 8 x 10Gb SR, LR, or LRM fiber uplinks SFP+ 1 management USB port 1 internal interface to c-Class Onboard Administrator Module	16 x 1Gb downlinks midplane 1 x 10Gb cross connect 1 x 10Gb copper uplinks CX-4 2 x 10Gb SR or LR fiber uplinks XFP 2 x 1Gb SX or RJ-45 fiber uplinks SFP 4 x 1Gb 1000/100/10Gb copper uplinks RJ-45 1 management USB port 1 internal interface to c-Class Onboard Administrator Module	16 internal 4Gb downlinks presented as F-Ports 4 external 4Gb uplinks presented as N-Ports
Media types	SFP+ SR, LR, LRM SFP SX, RJ-45 SFP + Copper Twinax CX-4 (IB4x)	SFP SX, RJ-45 FX SR and LR Copper RJ-45 100 Ohm 2-pair Cat5 UTP Twinax CX-4 (IB4x)	Small form-factor pluggable (SFP) laser 1/2/4Gb short wave up to 500 m (1,640 ft) 1/2/4Gb long wave up to 10 km
Performance	Line Rate, full-duplex 240Gbps bridging fabric Less than 2μs latency	Line Rate, full-duplex 62Gbps bridging fabric Less than 4μs latency	4Gbps line speed, full duplex 1.2 μ sec latency Maximum frame size 2112-byte payload Buffer-to-buffer flow control management Packet prioritization
Protocol support	802.1AB, 802.1D, 802.1Q, IEEE 802.2, 802.3ad	802.1AB, 802.1D, 802.1Q, IEEE 802.2, 802.3ad	NCITS T11 N_Port ID Virtualization (NPIV)
Management	Simple and intuitive Graphical User Interface and Setup Wizards Embedded SNMP v1, v2 Command Line Interface Port Mirroring—Any uplink port can be used as a dedicated mirrored port from the server port(s)	Simple and intuitive Graphical User Interface and Setup Wizards Embedded SNMP v1, v2 Command Line Interface Port Mirroring—Any uplink port can be used as a dedicated mirrored port from the server port(s)	Simple and intuitive Graphical User Interface and Setup Wizards accessible through VC Ethernet module Command Line Interface accessible through VC Ethernet module Embedded SNMP v1 and v2 SMI-S
Extended management features	Virtual Connect Enterprise Manager (VCEM) supports PXE, WOL, port VLAN, VLAN Tagging, VLAN pass through, IGMP Snooping, NIC Teaming Integrated with Onboard Administrator HP Systems Insight Manager Telnet, SNMP	Virtual Connect Enterprise Manager (VCEM) supports PXE, WOL, port VLAN, VLAN Tagging, VLAN pass through, IGMP Snooping, NIC Teaming Integrated with Onboard Administrator HP Systems Insight Manager Telnet, SNMP	Virtual Connect Enterprise Manager (VCEM) support HP Storage Essentials (FC Management MIB)
High availability features	Link Aggregation Protocol Automatic loop protection Mirrored profile database Multi-path heartbeat between redundant modules	Link Aggregation Protocol Automatic loop protection Mirrored profile database Multi-path heartbeat between redundant modules	Link Aggregation Protocol Automatic loop protection Mirrored profile database Multi-path heartbeat between redundant modules
Security	LDAP, SSL, role-based management	LDAP, SSL, role-based management	LDAP, SSL, role-based management
Maximum per enclosure	8	8	6

Virtual Connect Fiber Channel Modules

This section provides a list of various VC Fiber Channel modules and their specifications.

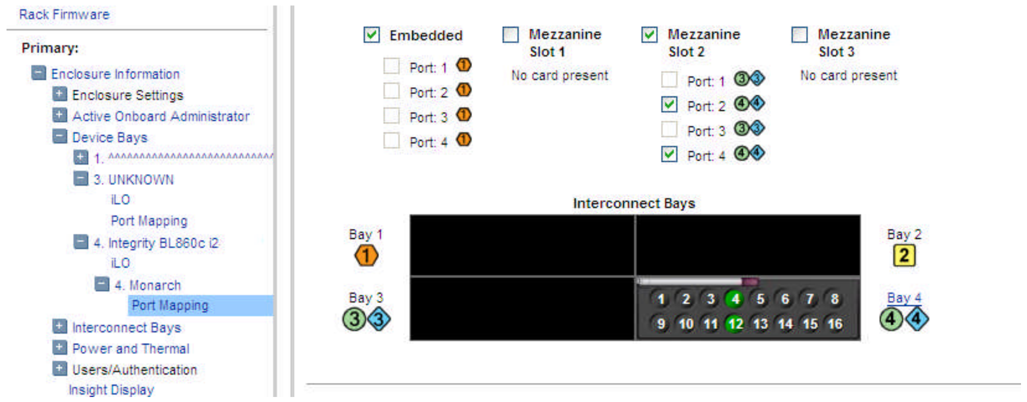
	HP 8Gb Virtual Connect 20-Port Fibre Channel Module for BladeSystem	HP Virtual Connect 8Gb 24-Port Fibre Channel Module
Blade type	Single bay	Single bay
Network connections	16 internal 4Gb downlinks presented as F-Ports 4 external 4Gb uplinks presented as N-Ports	16 internal 8Gb downlinks presented as F-Ports 8 external 8Gb uplinks presented as N-Ports
Media types	Small form-factor pluggable (SFP) laser 2/4/8Gb short wave up to 500 m (1,640 ft) 1/2/4Gb long wave up to 10 km	Small form-factor pluggable (SFP) laser 1/2/4Gb short wave, long wave SFP+ 8/4/2Gb short wave, long wave
Performance	8Gbps line speed, full duplex 1.2 μ sec latency Maximum frame size 2112-byte payload Buffer-to-buffer flow control management Packet prioritization	8Gbps line speed, full duplex .74 μ sec latency Maximum frame size 2148 bytes (2112 byte payload)
Protocol support	NCITS T11 N_Port ID Virtualization (NPIV)	NCITS T11 N_Port ID Virtualization (NPIV)
Management	Simple and intuitive Graphical User Interface and Setup Wizards accessible through VC Ethernet module Command Line Interface accessible through VC Ethernet module Embedded SNMP v1 and v2 SMI-S	Simple and intuitive Graphical User Interface and Setup Wizards accessible through VC Ethernet module Command Line Interface accessible through VC Ethernet module Embedded SNMP v1 and v2 SMI-S
Extended management features	Virtual Connect Enterprise Manager (VCEM) supports HP Storage Essentials (FC Management MIB)	Virtual Connect Enterprise Manager (VCEM) supports HP Storage Essentials (FC Management MIB)
High availability features	Link Aggregation Protocol Automatic loop protection Mirrored profile database Multi-path heartbeat between redundant modules	Link Aggregation Protocol Automatic loop protection Mirrored profile database Multi-path heartbeat between redundant modules
Security	LDAP, SSL, role-based management	LDAP, SSL, role-based management
Maximum per enclosure	6	6

BL8x0c i2 Troubleshooting Tips

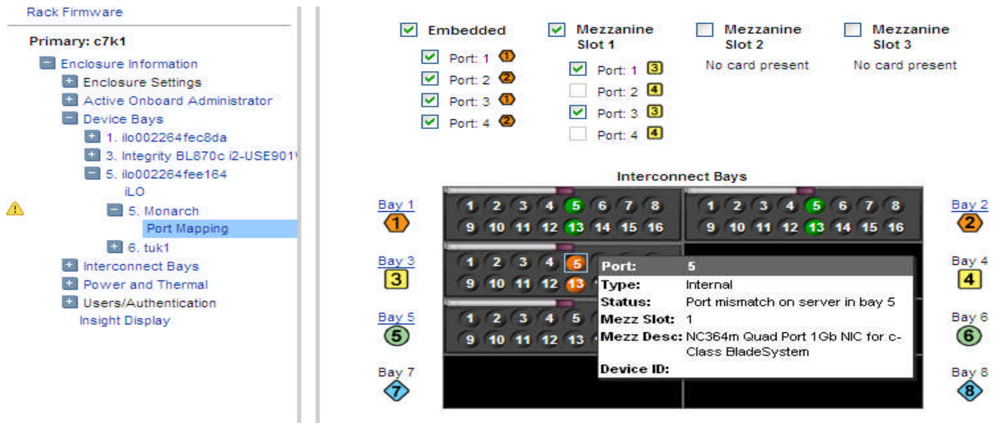
This section provides the BL8x0c i2 troubleshooting tips.

1. Connecting power cables to C7000 enclosure
   The order in which the power cables are connected to the C7000 enclosure is important. If the order is not followed properly, it results in powering up only a few power modules.
   There are 6 power ports and the power cables must be connected in the following order:
   Power SLOT 1 and Power SLOT 4
   Power SLOT 2 and Power SLOT 5
   Power SLOT 3 and Power SLOT 6
2. Port mapping of server blade to interconnect module
   Ensure correct port mapping for proper operation of blades. If there is any mismatch in port mapping, it results in server blade degradation.

Port mapping of each server blade to interconnect module in the back plane can be viewed from the Onboard Administrator (OA) web interface. The figure below captured from the OA depicts the port mapping of the mezzanine cards in the server blade to the interconnect module in the interconnect bay with appropriate status.

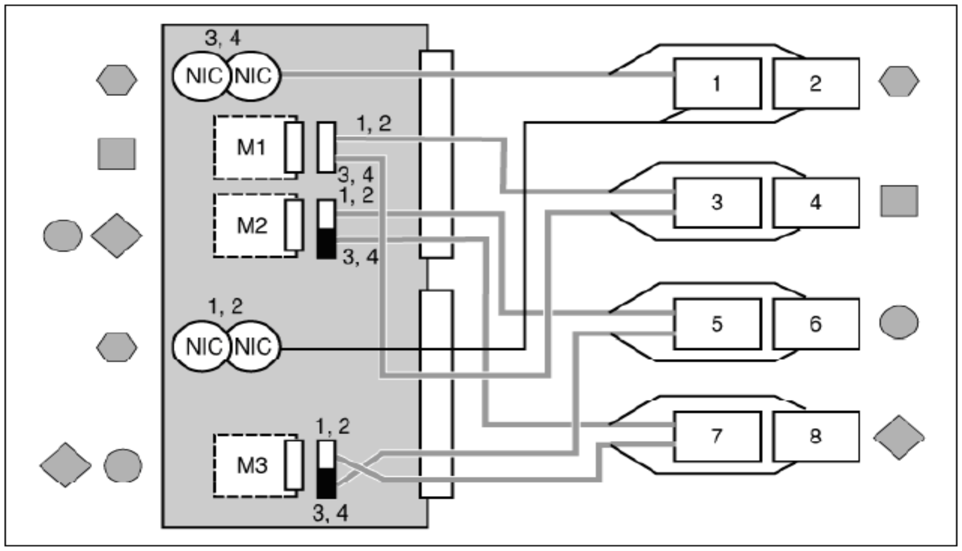


If there is any mismatch in the port mapping, it will be highlighted in the OA web interface as shown below:

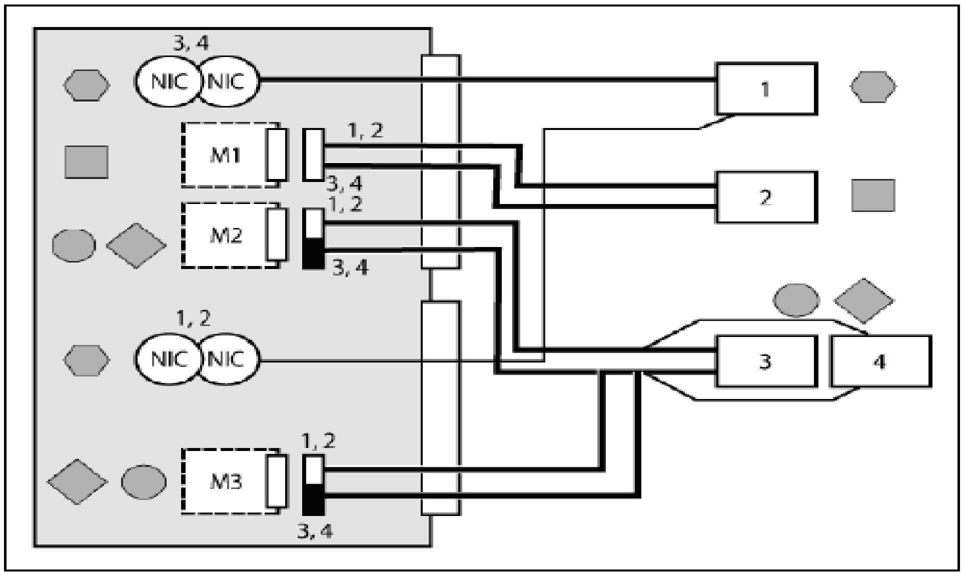


Port:	5
Type:	Internal
Status:	Port mismatch on server in bay 5
Mezz Slot:	1
Mezz Desc:	NC364m Quad Port 1Gb NIC for c-Class BladeSystem
Device ID:	

Proper port mappings on C7000 enclosure is as shown:



For detailed procedure, see the *HP BladeSystem c7000 Enclosure Setup and Installation Guide* at
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00698286/c00698286.pdf.
Proper port mappings on C3000 enclosure is as shown:



For detailed procedure, see the *HP BladeSystem c3000 Enclosure Setup and Installation Guide* at
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01167165/c01167165.pdf

3. 4S (BL870c i2) and 8S (BL890c i2) setup
   If you want to configure your blades as required to scale up/down needs, the following tip is useful:

4-Socket configuration (BL870c i2): Connect the two 2S blades (BL860c i2) using scalable blade link. The left most blade will always be the monarch and the other blade will be the auxiliary. Reset the monarch from the Onboard Administrator, and this will setup the 4S configuration.

8-Socket configuration (BL890c i2): Connect the four 2S blades (BL860c i2) using scalable blade link. The left most blade will always be the monarch and the other blades will be the auxiliaries. Reset the monarch from the Onboard Administrator, and this will setup the 8S configuration.

NOTE: Upgrading the firmware on the monarch will automatically upgrade the firmware on the auxiliary as well.

4. Creating LUNs using ORCA utility.
   If LUNs are not created, then the respective SAS disk devices cannot be detected.

   a. Create LUNs using the DRVCFG command from EFI (SHELL>)
   b. Find the "SAS driver id" using the following command:

```
SHELL> drivers
```

‘

‘

‘

A4 00000318 B X X 1 2 Smart Array SAS Driver v3.18 MemoryMapped(0xB,0x

‘

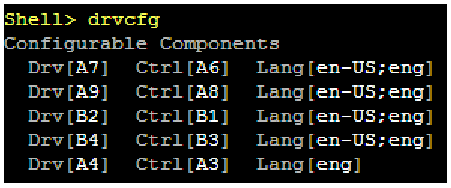
‘

Where:
‘A4’ is the SAS driver id.

   c. Obtain the respective "controller id" using the following command:

```
SHELL> DRVCFG
```

This command lists a series of SAS driver IDs and their respective controller IDs as shown in the figure:

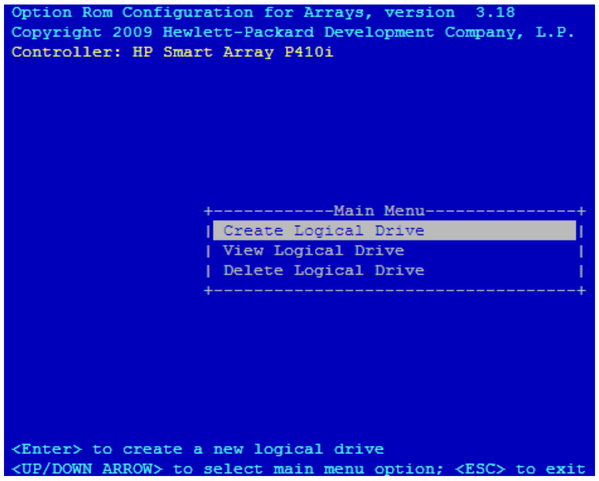


```
Shell> drvcfg
Configurable Components
  Drv[A7]  Ctrl[A6]  Lang[en-US;eng]
  Drv[A9]  Ctrl[A8]  Lang[en-US;eng]
  Drv[B2]  Ctrl[B1]  Lang[en-US;eng]
  Drv[B4]  Ctrl[B3]  Lang[en-US;eng]
  Drv[A4]  Ctrl[A3]  Lang[eng]
```

In this example, ‘A4’ is the "SAS Driver id" and its "controller id" is ‘A3’.

   d. Launch the **ORCA** utility using the respective driver ID and controller ID obtained from above steps:

```
SHELL> DRVCFG A4 A3 -s
```



```
Option Rom Configuration for Arrays, version  3.18
Copyright 2009 Hewlett-Packard Development Company, L.P.
Controller: HP Smart Array P410i



                         +------------Main Menu---------------+
                         | Create Logical Drive               |
                         | View Logical Drive                 |
                         | Delete Logical Drive               |
                         +------------------------------------+



<Enter> to create a new logical drive
<UP/DOWN ARROW> to select main menu option; <ESC> to exit
```

e. Select Create Logical Drive option and then create the LUNs.
f. Click **Yes** to restart the controller.

From the SHELL> execute the MAP –R command to list the devices.

5. AUTOGEN unable to create page, swap, and dump files of appropriate sizes.
   On systems with large physical memory, sometimes AUTOGEN does not create page, swap and dump files of appropriate size. HP recommends that the system manager review the AUTOGEN feedback report named SYS$SYSTEM:AGEN$PARAMS.REPORT and create the page, swap, and dump files of required sizes manually once the system boots up. See the AGEN$FEEDBACK.REPORT for any AUTOGEN error messages and take appropriate actions.
6. Ensure to assign MP IP to monarch and its auxiliaries in 4S/8S configuration.
   The firmware upgrade fails if the MP IP is assigned only to monarch but not to auxiliaries. To ensure proper firmware upgrade, it is necessary to assign the MP IP's to the auxiliaries as well.

References

- HP BladeSystem c7000 Enclosure Quick Setup Instructions
  http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00698534/c00698534.pdf

- HP BladeSystem c3000 Enclosure Quick Setup Instructions
  http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01167169/c01167169.pdf

- BL8x0c i2 White Paper
  http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA1-1295ENW.pdf

- HP Integrity BL860c i2, BL870c i2 & BL890c i2 Server Blade User Service Guide
  http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c02110937/c02110937.pdf

5 – A Powerful Array Language

Dr. Bernd Ulmann
Hochschule fuer Oekonomie und Management, Frankfurt

VAX APL and then?

Do you remember VAX APL? Wasn't it great to have something as powerful as APL at your fingertips on a VMS system (back then without the "Open")? Many years ago I used VAX APL extensively while studying mathematics. What I loved most about APL was its power and expressiveness -- which came at a price, though. APL programs, having been written in an interpreted language, were never as fast as programs written in FORTRAN or C. However, by using APL, my time-to-market was faster by at least one or two orders of magnitude compared with other, more traditional languages.

VAX APL is long since gone and with it much of the fun and productivity when it comes to exploring the behavior of some mathematical function or the quick implementation of a newly discovered algorithm.

Although there are quite a few APL interpreters available for various operating systems, most are commercial by nature, apart from A+[1]. Additionally, not one of them is available on OpenVMS systems, which is unfortunate, especially for me since I still prefer to do most of my daily work on an OpenVMS system (namely, FAFNER, a VAX-7000/820).

Expressiveness and complexity

What makes APL so powerful is its approach to handle deeply nested data in a transparent and elegant way. Adding two vectors in APL is as easy as writing `A+B`. The binary function '`+`' is extended on an item-by-item basis if it is applied to non-scalar values, so it will be applied to every two corresponding elements of the two vectors `A` and `B`. But APL goes further with complex functions and operators, which all apply naturally on nested data structures. In many cases, these can eliminate the need for explicit loops or conditionals in APL programs.

This expressiveness of APL comes at a price, too. APL makes use of a rather strange character set, which required terminals with loadable fonts or X-terminals in the times of VAX APL (as well as a keyboard with additional labels -- or the very good memory of the programmer).

What makes things even more arcane for the non-initiated programer is the fact that APL evaluates arithmetic expressions from right to left. This is mathematically elegant since it makes things like the evaluation of polynomials easy without cluttering the expression with lots of parentheses. But from a practical point of view, with the occasional APL programmer in mind, this was and still is a constant source of errors until one really gets adapted to APL.

Combining APL and Forth

Allowing for a short digression from the main theme of "array languages", let us explore a related topic. Many of you have worked with HP pocket calculators featuring RPL(Reverse Polish LISP) as their programming language, as I have. RPL is an interesting concept, as it combines the main features of two well-known languages, namely Forth and LISP. RPL extends the rather simple Forth stack with a stack that can hold nearly arbitrary objects, and allows the execution of LISP-like functions operating on these structures found on the stack. This works since nested data structures are

[1] Cf. `http://www.aplusdev.org`.

represented as nested lists, which are readily accessible to a LISP-style approach of programming.

Some time ago I began thinking about how a language would look and feel if it combined the features of Forth and APL, two of my favorite programming languages. After some experimentation, with paper and pencil at first, it became clear that such an approach could very well work. Namely, by combining the powerful functions and operators of APL with the bottom-up programming style of Forth, the result would encompass the expressiveness of both languages and the simplicity of the resulting parser in a very powerful blend.

Realizing this, the development of such a language, which has been named **5**, was started, using Perl as the basis for the 5 interpreter. A first implementation was done in a couple of weeks during spare evenings and weekends, and served as a test bed to explore the possibilities of such a stack-based array language. The findings of this first iteration were then used as the foundation for a second implementation, which was done from scratch[2]. It is this second implementation that will be described in the following sections.

It was decided at the very beginning of the development of 5 to abandon the complex character set of APL in favor of easily remembered mnemonics for all functions and operators to be implemented.

Since the 5 interpreter is written in pure Perl and does not require any non-standard packages, it can be run on any system which has a Perl interpreter installed. The 5 interpreter has been written explicitly with the goal of easy portability – and specifically with OpenVMS in mind as a target system. It runs out-of-the-box on any OpenVMS system that has a Perl interpreter installed.

The complete 5 project is hosted by sourceforge. The homepage for this project can be found at `http://lang5.sourceforge.net/`. To get started with 5 on any reasonable platform, including OpenVMS, you will need the distribution package, which can be downloaded from `https://sourceforge.net/projects/lang5/files/` as a ZIP-file (about 270 KB). This file contains the 5-interpreter, some examples, and some PDF documents containing an introduction to programming in 5. To install the interpreter on an OpenVMS system, extract the ZIP-file into a suitable subdirectory. To use 5, all you need to do is to define a foreign command, such as[3]:

```
$ FIVE :== "PERL DISK$SOFTWARE:[5]5"
```

To start the interpreter in its interactive mode, just type `five` on the DCL prompt:

```
ULMANN:FAFNER$ five
----> loading mathlib.5
----> loading stdlib.5
5> "Hello world!\n" .
Hello world!
5>
```

[2] The author would like to thank Mr. Thomas Kratz who did most of this second implementation.

[3] This, of course, assumes that the interpreter resides in a directory called `[5]` located on the disk `DISK$SOFTWARE` – adapt this definition to suit your local situation.

Programming 5

Basics

At the beginning of the VAX APL Reference Manual, it says:

> "This manual is not a tutorial and is inappropriate for novice users. Programmers experienced with other languages such as FORTRAN or BASIC can learn VAX APL from this manual, but are advised to study it in conjunction with an APL language primer."

The same holds true for the following sections. 5 is far too complex to be described in a few pages, but it is possible to give some examples illustrating the expressiveness of 5 and the ease by which even complex algorithms may be implemented. A much more complete description of the language, together with introductory examples, can be found in the PDF files of the distribution kit, which has already been mentioned.

The look and feel of 5 is quite like Forth at first sight, since there is a central stack holding all data to which functions and operators will be applied. The parser is thus rather simple since all operators and functions only act on the stack so that no precedence rules -- parentheses, etc., must be taken care of. In fact, the parser just splits the input read from `STDIN` or from a file on whitespace characters and either executes the tokens found if these are valid operators, functions, or words, or pushes raw data onto the stack. Thus, to compute the result of (1+2)*3 using 5, one just has to enter this expression in postfix notation at the prompt of the interpreter:

```
5> 3 2 1 + * .
9
5>
```

The single dot at the end of the command line prints the element found on the top of stack, which will be referred to as *TOS* in the following sections. Now, how does this example work? First, the interpreter has pushed the three scalar values `3`, `2,` and `1` onto the stack. Following this, the binary operator '`+`' has been applied, which in effect has removed the two top-most elements from the stack (or `1` and `2` respectively), and placed the resulting sum on the TOS, which now contains the scalar value `3`. The next step then calculates the product of `3` and this sum and places the result, `9`, onto the stack, which is then printed by using the dot function.

As in APL, the unary and binary operators as they are referred to in 5, normally act on simple scalar values. If such operators are applied to nested structures, they will, in effect, be applied in an element-wise fashion to all elements of the affected structure(s), as the following example shows:

```
5> [1 2 3] 2 * .
[    2     4     6  ]
5> [1 2 3] [4 5 6] + .
[    5     7     9  ]
5>
```

This is where 5 departs from a traditional Forth interpreter, as its stack can hold arbitrarily structured data instead of only simple scalars. In the first example shown above, every element of a three-element vector is multiplied by 2, while in the second example, two three-element vectors are added element by element.

User-defined words and more complex array operations

Another feature that 5 borrows from Forth is the possibility to extend the language itself by introducing user-defined words, which can be used in exactly the same way as the built-in functions and operators. The following example shows the definition of a word that returns the square of a value:

```
5> : square dup * ;
5> 5 square .
25
5>
```

As one can easily see, any word definition that starts with a colon followed by the name of the word can be defined. The end of such a definition, which in effect is just a list of 5 operators, functions, other words or operands, is denoted by a semicolon. The word `square` defined above will duplicate the element found on TOS and multiply the resulting two elements. Thus, applying this newly defined word `square` to the value `5` on the TOS yields the result `25` on the TOS.

Let us have a look at this slightly more complex example[4] that simulates throwing a six-sided dice 100 times and returns the arithmetic mean of the results:

```
: throw_dice
  6 over reshape
  ? int 1 +
  '+ reduce swap /
;

100 throw_dice .
```

What does this example illustrate? First of all a new word, `throw_dice`, is defined. This word places the value `6` onto the TOS and copies the number of runs, which is now on the element just below the TOS, using the `over` function to the TOS again. Assuming that `100` runs are to be performed, the stack now contains the values `100`, `6,` and `100`. The two last values are then used as arguments for the `reshape` function, which yields a vector containing `100` elements, each having the value `6` on the TOS: `[6 6 … 6].`

This vector is then used as argument for the `?` operator, which generates a pseudo-random number between `0` (inclusively) and the element to which it is applied (exclusively). Since the `?` operator is a unary operator, it will be applied to all the elements of a nested data structure automatically by the 5 interpreter. This will result in a vector containing pseudo-random numbers between `0` and `6` being placed on the TOS. Applying the `int-`operator will remove the fractional part of the elements of this vector; adding `1` will yield a vector with elements ranging randomly between `1` and `6`.

In the following step, something tricky happens. A multiplication operator is pushed onto the stack, which is done by preceding it with a single quote to prevent the interpreter from executing it immediately. This operator and the vector described above are then used as arguments to the `reduce-`function, which acts exactly like the reduce operator in APL. It applies the operator found on TOS between every two elements of the vector found on the element just below TOS. In effect, this calculates the sum of all elements of the vector.

Since we duplicated the value found on the TOS in the very first step of the user-defined word, which corresponded to the number of times we should throw the dice, there is still

[4] Note that word definitions which are made in interactive mode have to fit in a single line! This example is taken from a file which was executed by the 5 interpreter.

a copy of this value found in the element just below TOS. Using the `swap`-function, this value and the value of the sum calculated above are interchanged, so that applying the `/`-operator will yield the desired arithmetic mean.

A more complex example – calculating primes

The next example is a bit more complex than the dice simulation shown above. The goal is to generate a list of prime numbers between 2 and a number given on the TOS. Instead of applying test divisions by odd integers or the like, a more APL-like approach has been chosen.

Imagine that a list of primes between 2 and 100 is to be calculated. This information will be used to form a vector `[2 3 4… 100]`. Two of these vectors (copies are made using the `dup`-function) are then used to create a matrix by computing an outer vector product. This matrix contains 99 times 99 elements, none of which is a prime since all the elements in this matrix are the result of at least one multiplication. In the next step, the original vector and this matrix are used as arguments to the set operation `in,` which yields a vector containing 99 elements being `0` or `1`. A `0` denotes an element of the original vector that was not found in the matrix, while `1` represents the opposite case. Clearly, this vector contains a `0` at every location, where a prime number was found in the original vector. Inverting this vector will yield a selection vector containing the value `1` at every location, corresponding to a prime element in the original vector. This selection vector is then applied to the original vector using the `select`-function, which yields all prime numbers between 2 and 100.

The corresponding user-defined word in 5 looks like this:

```
: prime_list
  1 – iota 2 +
  dup dup dup
  '* outer
  swap in not
  select
;
```

Entering `100 prime_list .` will yield:

```
[    2     3     5     7    11    13    17    19    23    29    31
37    41    43    47    53    59    61    67    71    73    79    83
89    97  ]
```

as a result.

At first glance, this program does not look very intuitive to someone who is used to programming in traditional imperative or OO languages like C, Fortran, etc. However, one can quickly become accustomed to the idea of array languages in general, and 5 in particular, since the interactive nature of the 5-interpreter makes experiments easy and ensures that it can be learned quickly.

It is noteworthy that things like `if-else` constructions, recursion or loops, which are all supported by 5, are used only rarely when an array-oriented approach to programming is employed!

Even more complex – multiplying a matrix by a vector

To illustrate the basic idea of this more or less loop-free programming style, let us have a look at another example program, which implements a matrix-vector multiplication word in 5:

```
5> : inner+{u} '+ reduce ;
5> : mv* 1 compress '* apply 'inner+ apply ;
5> [[1 2 3][4 5 6][7 8 9]] [10 11 12]
5> mv* .
[   68   167   266   ]
```

The main word is mv* (short for matrix-vector-multiplication), which expects a vector on TOS and a matrix in the element below TOS. As an initial step, the vector found in TOS is enclosed in another vector yielding a vector of the form `[[ … ]]`. Following this, the operator * is pushed onto the stack and then applied in an element-wise fashion along the first axis of the matrix and vector found in the two top-most stack elements by means of the `apply`-function. This yields a vector with all partial products of the desired matrix-vector product. Applying a special user-defined word `inner+` will then reduce these partial products by adding them together. Note that this word definition differs from a simple user-defined word as described in the preceding sections by specifying `{u}` after the name of the word to be defined. This tells the interpreter that the word to be defined will be a unary word, which will be handled in exactly the same way as a built-in unary operator. Otherwise, it would not have been possible to apply this word to all elements of the resulting structure along its first axis using the `apply` function.

Conclusion

5 is a rather powerful array language and supports a highly interactive style of programming due to the features inherited from Forth. Since the basic data structures that 5 operates on are nested arrays, many problems can be solved without the need for explicit loops or complex conditionals, as the preceding examples have shown. Since I have 5 running on my OpenVMS system, I do not miss VAX APL any longer since 5 brings nearly all of the power of APL to my system without the need for costly third-party software, etc.

Nevertheless, 5 is still a work in progress. Although the set of operators and functions which are implemented at the time of writing this journal is rather large, there may be problems that may require extensions to the language. In many cases, these extensions can be incorporated in the standard library `stdlib.5` or the mathematical library `mathlib.5,` which are both part of the 5 distribution kit. If this is not feasible for a particular problem, it is rather easy to extend the interpreter itself.

If you like what you have been introduced to in this paper and are interested in learning more, please feel cordially invited to participate in the development efforts of 5. The next steps of the development team will involve the addition of new words in the libraries, enhancing the documentation (a quick reference guide is currently being written, in addition to the rather complete introductory documents), and the development of a stable regression test suite to facilitate further developments, etc.

For more information

Contact the author at ulmann@vaxman.de.

New Time Features on OpenVMS and Read-Write Consistency Check Data Algorithm

Clarete Riana, Burns Fisher, and Sandeep Ramavana

Introduction

With the increasing processing speed of newer processors and the growing transaction rates of certain applications, it becomes necessary for time to be tracked at a finer granularity than is currently available. To this end, we have added new time features on OpenVMS. They are as follows:

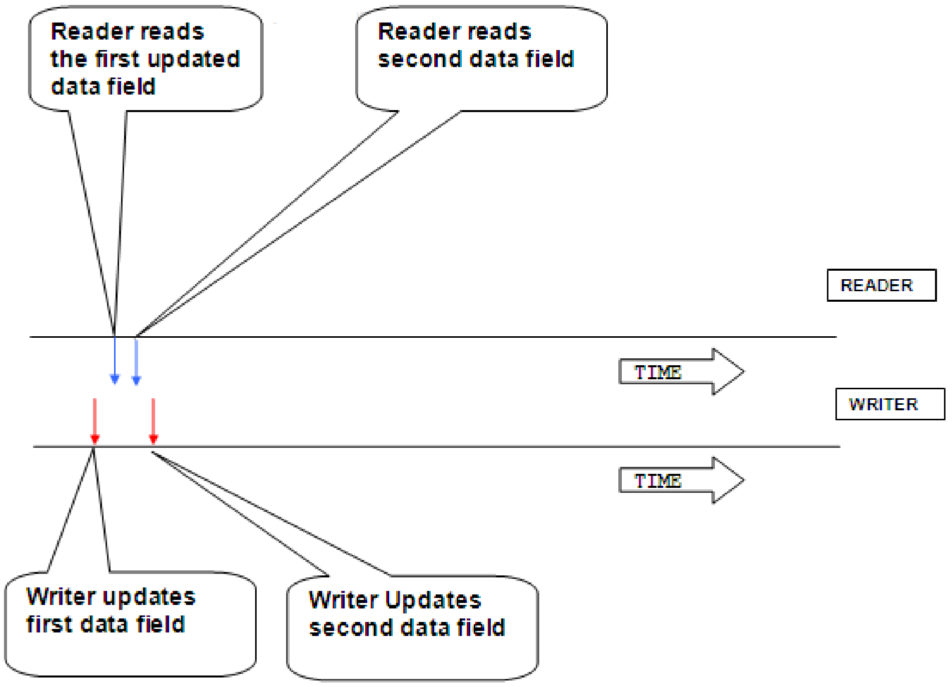
- A mechanism to obtain high-precision time using a new system service
- A mechanism to obtain a medium resolution time-since-boot of the system, by making use of a new flag with the existing $GETTIM system service.
- A method of ensuring the consistency of UTC time across a time-zone or summer/winter time change. This can be used by the applications to determine the right pair of TDF and system time, which otherwise can be a problem when seasonal time changes occur.
  This feature is also used by the $GETUTC service to ensure that $GETUTC never receives a glitch. The high precision time and consistent UTC change make use of a lightweight read-write consistency algorithm.

This paper describes each of the new features in detail.

Read-write consistency check data algorithm

When there are lockless concurrent accesses to shared resources, there will be scenarios where the reads of these fields will result in reading inappropriate values, as a write operation on these fields might be in progress.

The problem is captured in the figure below, where the writer updates the values of data fields one after the other and the reader reads the data fields. There is a possibility that the reader reads the data fields only when one of them is updated by the writer, and not the other, thereby reading the updated value of one field and old value of the other field, when it is required to read the updated values of both the fields.


Reader reads the first updated data field
Reader reads second data field
READER
TIME
WRITER
TIME
Writer updates first data field
Writer Updates second data field

There are existing locking mechanisms through which read/write consistency can be achieved to solve the above stated problem. Achieving read/write consistency of concurrent reads and writes across these data fields using the traditional locking mechanisms may prove very expensive in certain scenarios where write or read access must be fast. In such scenarios, we propose to make use of the unique lightweight checksum algorithm to achieve consistency of read/write of data fields.

When multiple data fields are to be written and read concurrently, in order to ensure read/write consistency, we propose to use another data field, which will contain appropriate checksum of the data fields. This checksum will be calculated and stored in the new data field during the write operation of the data fields whose read/write consistency is to be ensured. When there is an attempt to read the data fields, their checksum is again calculated using the same algorithm that was used during the write operation and matched with the checksum that was saved when the data fields were written. The reads are considered valid only if the checksum calculated after read matches the checksum stored during the write operation of the data fields. The field of checksum identification is well researched and various checksum algorithms are available. A checksum algorithm uniquely representing the data field is to be chosen here, which should detect if any of the data fields are changed.

It is to be noted here that this algorithm does not ensure any synchronization, but only ensures read/write consistency across multiple data fields. In other words, the algorithm does not ensure that the reader reads the updated values; it only ensures that it reads consistent values (perhaps the old values, perhaps the new ones).

High-precision time

Time services are provided by the operating system (OS) to give delta and absolute time services, including obtaining the time as well as alarm services at specified times. Modern day processors have timer clock hardware built into the chip itself. The timer clock hardware generates hardware interrupts based on a minimum time period, in clock ticks, specified by the OS. The timing generated from the timer interrupt is used to update system time in a granularity defined by the OS. There are some applications that may require higher resolution timer services. It is non-optimal to reduce the time period of the timer interrupts, as this would create more interrupt load on the system.

OpenVMS provides a system service $GETTIM, which returns a 64-bit number present in EXE$GQ_SYSTIME, which represents the current time in 100ns intervals beyond the Smithsonian base date of 17-Nov-1858. However, this time is only updated approximately every millisecond as a part of the hardware interrupt servicing that happens every millisecond. Thus, although the granularity of the OpenVMS system time is 100ns, the user can only have the accuracy of one millisecond.

Itanium processors have two registers that help generate the timer interrupt. One is writable by the OS and holds the value at which the next timer interrupt is generated, usually called the match register. The second register ticks in the order of a few megahertz depending on the type and speed of the processor, also called the cycle counter/time counter. When the value of this register matches that of the first register, a timer interrupt is generated, which updates the system time. The cycle counter and the match registers are utilized to obtain high resolution time.

A new system service $GETTIM_PREC is implemented on OpenVMS. The usage of $GETTIM_PREC system service is the same as that of the existing $GETTIM system service. The user has to pass a buffer in which high-precision time is returned. When the $GETTIM_PREC system service is issued--the system time updated at the last 1ms interrupt -- the value of match register at the last 1ms interrupt and the current value of the cycle counter register are read.

While using these values to calculate high-resolution time, there should be effective consistency of the reads and writes of system time and match register values. One of the problem scenarios is when one of the processors issues a system call to get the high precision time, which in turn reads the global data cells, when another processor is in the interrupt service routine (ISR) updating the global data cells. This might result in reading one of the updated data cells but not the other. To solve this issue, we use the proposed method of ensuring read/write consistency. We save the checksum of the match register and system time in the ISR. So when the system call is issued, the saved system time, match register, and checksum are read and then the checksum of system time and match register is calculated locally. If the locally calculated checksum does not match the checksum that is read, it means that the system call was issued when the other processor was updating the data fields. So, the system call discards the values read, and re-reads them. Therefore, the data fields are re-read until the checksums are equal before calculating the high-precision time, thus making sure that the right pair of system time and match register are read. The checksum algorithm we used was based on selecting appropriate bit fields from system time and the time at which the interrupt occurred. The bit fields were selected such that the wrap time of checksum was a significant value.

This provides a higher resolution of system time than what is already available with $GETTIM system service. Further, since the high resolution time is based on the normal system time, it will be updated with $SET TIME, automatic seasonal time changes, or time synchronization services such as NTP.

This feature is an Itanium-only feature. The $GETTIM_PREC system service, when executed on ALPHA is equivalent to $GETTIM and just returns low-precision time in the user provided buffer, with a return status of SS$_LOWPREC, to indicate that only low-precision time is returned. On rare occasions, if the precision time calculation is inconsistent, you may also get a status of SS$_LOWPREC on Integrity systems.

Following is the output of executing the two system services in a test program, “High prec time.” In the output is the time returned by the $GETTIM_PREC system service, and “low prec time” in the output is the time returned by the $GETTIM system service. It is observed that $GETTIM_PREC returns a higher precision time value.

```
$ run diff_services
High prec time is A797CAAC6C88F6,  low prec time is A797CAAC6C7A08
High prec time is A797CAAC6C9C77,  low prec time is A797CAAC6C7A08
$
$ run diff_services
High prec time is A797CAAE2A6CB2,  low prec time is A797CAAE2A56A8
High prec time is A797CAAE2A8BFD,  low prec time is A797CAAE2A56A8
$
$ run diff_services
High prec time is A797CAB04E87BA,  low prec time is A797CAB04E6F28
High prec time is A797CAB04EA6FE,  low prec time is A797CAB04E6F28
$
```

```
$ run diff_services
High prec time is A797CAB11CCBAE,  low prec time is A797CAB11CBBA8
High prec time is A797CAB11CED45,  low prec time is A797CAB11CBBA8
```

As of today, there is no service that converts the system time returned to a high-precision ASCII time format. Conversion has to be done manually.

This feature is available on OpenVMS V8.3-1H1 with a patch kit and with all later OpenVMS releases.

High-resolution time-since-boot

Currently, OpenVMS has two data cells that are updated with the time-since-boot for the system. The first one, EXE$GQ_ABSTIM, is updated every second and therefore has an accuracy of a second. The second data cell, EXE$GQ_ABSTIM_TICS, is updated every 10 milliseconds, and therefore the accuracy of EXE$GQ_ABSTIM_TICS is only 10 milliseconds.

A new feature is added to provide a high-resolution time_since_boot. A new global data cell EXE$GQ_ABSTIM_100NS has been added to provide a high-resolution time_since_boot. This data cell is updated approximately every millisecond as a part of the hardware-interrupt servicing that happens every millisecond. Therefore, the time-since-boot of the system in this cell will have the accuracy of one millisecond, and a granularity of 100ns, equivalent to that of the system time in EXE$GQ_SYSTIME.

The user can read the high resolution time-since-boot directly from the global symbol EXE$GQ_ABSTIM_100NS, or obtain the high resolution time-since-boot using the existing $GETTIM system service. An optional new parameter flag has been added to the $GETTIM system service. When the system service is issued with this flag value as one, the $GETTIM system service reads and returns to the user the value in EXE$GQ_ABSTIM_100NS.

This feature is available on supported OpenVMS versions beginning with V7.3-2 (with a patch kit) and with all later OpenVMS releases on Alpha and Itanium.

Method of ensuring consistency of UTC time across time zone changes

Along with the global cell EXE$GQ_SYSTIME, OpenVMS has another data cell, EXE$GQ_TDF, which maintains the time differential factor. During seasonal time change, both data cells are updated to reflect the seasonal time change. There is a timing window present in updating the time data cells, EXE$GQ_SYSTIME and EXE$GQ_TDF. Reading the data cells in this window results in wrong values being read. We have seen this problem manifest itself in two cases. The first case is a glitch seen when $GETUTC was called around the seasonal time change. The second case is a Pthread library hang.

As a fix for this issue, a new data cell, EXE$GQ_UTC, is added, which is updated with the difference of EXE$GQ_SYSTIME and EXE$GQ_TDF as a part of the hardware-clock interrupt servicing that happens every millisecond.

$GETUTC simply reads EXE$GQ_UTC rather than calculating UTC from SYSTIME and TDF. EXE$GQ_UTC is always consistent because it is only updated while holding the HWCLOCK spinlock in the interrupt routine, as well as when the TDF is changed.

Pthread library code now reads the values of EXEGQ_UTC, EXEGQ_SYSTIME, and EXE$GQ_TDF, and compares them for consistency. The read-write consistency algorithm is used here again in this case to ensure that the right pair of system time and TDF is read. Library code locally calculates the UTC as a difference of EXE$GQ_SYSTIME and EXE$GQ_TDF, and reads the values in EXEGQ_UTC, EXEGQ_SYSTIME, and EXE$GQ_TDF in a loop until the locally calculated UTC matches the EXE$GQ_ UTC.

Using XFC to get System/Volume/File/IO Statistics

P Muralidhar Kini

XFC introduction

The OpenVMS operating system has two types of caches -- the data cache and the metadata cache. The data cache is referred to as eXtended File Cache (XFC) and the metadata cache is called the eXtended Qio Processor (XQP). XFC caches contents of all files present on the disk, excluding directory files. XQP caches file headers and directory files.

Statistics provided by XFC

XFC provides various statistics of the system, which can be used for the overall system performance tuning. XFC provides the statistics at the system, volume, file and I/O level. The DCL commands or the XFC SDA extension can be used to get the XFC statistics. In most scenarios, DCL as well as XFC SDA extensions provide a similar type of statistics. The XFC SDA extension generally provides little more information than the corresponding DCL commands.

Some of the important statistics include read hit rate, XFC cache memory, I/O size distribution, and so on. This paper explores the various statistics that are provided by XFC.

HotFile statistics

Hotfile is a file in the system that records many activities in terms of read and write operations. In many environments, it is observed that 80 percent of the I/Os that are issued are on 20 percent of the files present in the system. This being the case, it is very important to know which files that are present in the system fall into this 20 percent category.

Hotfile – TOPQIO

On a given volume, the TOPQIO keyword instructs XFC to display the first ‘N’ entries in descending order of the number of IOs done to the file.

DCL command

```
$ SHOW MEM/CACHE=(VOLUME=<VOL-NAME>,OPEN,CLOSED,TOPQIO:N)
<VOL-NAME>: Volume name
N: Number of entries to be displayed
```

Sample output

```
$SHOW MEMORY/CACHE=(VOLUME=BRAMHA$DKB200:,OPEN,CLOSED,TOPQIO:2)
System Memory Resources on 30-MAY-2010 03:13:44.48

Extended File Cache Top QIO File Statistics:

_BRAMHA$DKB200: (DISK$BR_V83R), Caching mode is VIOC Compatible

_BRAMHA$DKB200:[MURALI]PERFORM_TEST.COM;68 (389451, 1057, 0) (closed)

Caching is enabled, active caching mode is Write Through
 Allocated pages             1     Total QIOs          1177
 Read hits                1176     Virtual reads       1176
 Virtual writes              1     Hit rate             100%
 Read aheads                 0     Read throughs       1176
 Write throughs              1     Read arounds           0
```

```
                                  Write arounds            0

_BRAMHA$DKB200:[MURALI]TEST_CONFIG.COM;47 (389469, 13875, 0) (closed)

Caching is enabled, active caching mode is Write Through
 Allocated pages              1     Total QIOs           393
 Read hits                  392     Virtual reads        392
 Virtual writes               1     Hit rate             100%
 Read aheads                  0     Read throughs        392
 Write throughs               1     Read arounds           0
                                  Write arounds            0
Total of 2 files for this volume

$
```

Interpreting the output

On the disk, BRAMHA$DKB200, the maximum number of IOs are done to the file PERFORM_TEST.COM (389451, 1057, 0).

- The total number of IOs done to the file is 1177. Among these IOs, 1176 are read IOs and 1 write IO.
- The read hits of the file are 100 percent. This is excellent because all attempts to read the file were satisfied from the cache itself.
- Allocated pages of the file is 1 page. This is a small temporary file, and hence consumes a very small portion of XFC cache memory.

Hotfile – TOPHITRATE

On a given volume, the TOPHITRATE keyword instructs XFC to display the first 'N' entries in descending order of read hit rates of the file.

DCL command

```
$ SHOW MEMORY/CACHE=(VOLUME=<VOL-NAME>,OPEN,CLOSED,TOPHITRATE:N)
<VOL-NAME>: Volume name
N: Number of entries to be displayed
```

Sample output

```
$SHOW MEMORY/CACHE=(VOLUME=BRAMHA$DKB200:,OPEN,CLOSED,TOPHITRATE:2)
System Memory Resources on 30-MAY-2010 03:27:08.67

Extended File Cache Top Hitrate File Statistics:

_BRAMHA$DKB200: (DISK$BR_V83R), Caching mode is VIOC Compatible

_BRAMHA$DKB200:[MURALI]TRACE_SETUP.COM;5 (389474, 1, 0) (closed)
Caching is enabled, active caching mode is Write Through
Allocated pages             1     Total QIOs             5
Read hits                   4     Virtual reads          4
Virtual writes              1     Hit rate             100%
Read aheads                 0     Read throughs          4
Write throughs              1     Read arounds           0
                                  Write arounds          0

_BRAMHA$DKB200:[MURALI]PERFORM_TEST.COM;68 (389451, 1057, 0) (closed)
Caching is enabled, active caching mode is Write Through
Allocated pages             1     Total QIOs          1177
Read hits                1176     Virtual reads       1176
Virtual writes              1     Hit rate             100%
```

```
Read aheads                 0     Read throughs         1176
Write throughs              1     Read arounds             0
                                  Write arounds            0
Total of 2 files for this volume
$
```

Interpreting the output
On the disk, BRAMHA$DKB200, the file with maximum read hit rate is TRACE_SETUP.COM (389474, 1, 0).

- The total number of IOs done to the file is 5. Among these IOs, 4 are read IOs and 1 write IO.
- The read hits of the file are 100 percent. This is excellent because all attempts to read the file were satisfied from the cache itself.
- Allocated page of the file is one page. This is a small temporary file, and hence consumes a very small portion of the XFC cache memory.

Memory statistics

The "SHOW MEM/CACHE" DCL command provides the XFC memory utilization statistics at the system level. It also provides other important statistics, such as read hit rate, and number of read/write IOs at the system level.

DCL command

```
$SHOW MEMORY/CACHE
```

Sample output
```
$SHOW MEMORY/CACHE
              System Memory Resources on 30-MAY-2010 04:37:45.74

Extended File Cache  (Time of last reset: 29-MAY-2010 00:25:38.06)
 Allocated (MBytes)   80.90    Maximum size (MBytes)   2047.81
 Free (MBytes)         0.00    Minimum size (MBytes)      3.12
 In use (MBytes)      80.90    Percentage Read I/Os         77%
 Read hit rate          90%    Write hit rate                0%
 Read I/O count       70914    Write I/O count         20452
 Read hit count       63875    Write hit count             0
 Reads bypassing cache  204    Writes bypassing cache     24
 Files cached op        398    Files cached closed       312
 Vols in Full XFC mode    0    Vols in VIOC Compatible mode  1
 Vols in No Caching mode  0    Vols in Perm. No Caching mode 0
$
```

Interpreting the output

- XFC memory can grow to a maximum of 2GB, that is 2047.81 MB
- Currently XFC has allocated itself a memory of 80.9 MB
- The free memory with XFC is 0MB. Any subsequent request to cache a file would cause XFC to expand in size.
- The overall read hit rate of the system is 90 percent. This is a good hit rate.
- Out of the total IOs issued in the system, 77 percent of the IOs are read IOs.

SDA command
```
SDA> SHOW MEM/CACHE
```

Sample output

```
SDA> SHOW MEM/CACHE
System Memory Resources from Running System on 30-MAY-2010 04:44:48.61
----------------------------------------------------------------------
Extended File Cache  (Time of last reset: 29-MAY-2010 00:25:38.06)
 Allocated (MBytes)  80.90    Maximum size (MBytes)       2047.81
  Free (MBytes)       0.00    Minimum size (MBytes)          3.12
  In use (MBytes)     80.90    Percentage Read I/Os            77%
  Read hit rate         90%    Write hit rate                   0%
  Read I/O count      71049    Write I/O count              20453
  Read hit count      64006    Write hit count                  0
  Reads bypassing cache 204    Writes bypassing cache          24
  Files cached open     399    Files cached closed            311
  Vols in Full XFC mode   0    Vols in VIOC Compatible mode     1
  Vols in No Caching Mode 0    Vols in Perm. No Caching mode    0
SDA>
```

Interpreting the output

The output is interpreted similar to that of the corresponding “SHOW MEM/CACHE” DCL command.

System statistics

The "SHOW MEM/CACHE" DCL command or the “XFC SHOW SUMMARY” SDA command provides the XFC statistic at the system level. Some of the important statistics that can be obtained are read hit rate and the number of read/write IOs.

DCL command

```
$ SHOW MEM/CACHE
```

Sample output

```
$SHOW MEMORY/CACHE
           System Memory Resources on 30-MAY-2010 04:37:45.74

Extended File Cache  (Time of last reset: 29-MAY-2010 00:25:38.06)
 Allocated (MBytes)    80.90    Maximum size (MBytes)        2047.81
 Free (MBytes)          0.00    Minimum size (MBytes)           3.12
 In use (MBytes)       80.90    Percentage Read I/Os             77%
 Read hit rate           90%    Write hit rate                    0%
 Read I/O count        70914    Write I/O count                20452
 Read hit co           63875    Write hit count                    0
 Reads bypassing cache   204    Writes bypassing cache            24
 Files cached open       398    Files cached closed              312
 Vols in Full XFC mode     0    Vols in VIOC Compatible mode       1
 Vols in No Caching mode   0    Vols in Perm. No Caching mode      0
$
```

Interpreting the system output

- The overall read hit rate of the system is 90 percent. This is a very good hit rate.
- Out of the total IOs issued in the system 77 percent of the IOs are read IOs.
- In this system, the number of open files cached by XFC is 398, and the number of closed files cached by XFC is 312.

SDA command

```
SDA> XFC SHOW SUMMARY
```

Sample output
```
SDA> XFC SHOW SUMMARY
XFC Summary (current time 30-MAY-2010 04:59:24.36)
------------------------------------------------
...
Virtual Reads:                     71090
Read Hits:                         64019
Read Cache Hit Percentage:         90.05 %
Virtual Writes:                    20770
Open Files:                          400
Closed Files in the Cache:           311
Total Files in Cache:                711
...
SDA>
```

Interpreting the system output
- 71090 virtual read IOs and 20770 virtual write IOs are issued
- Read hit rate is 90.05 percent. This is a very good hit rate.
- The total number of files in the cache is 711. Out of these, 400 files are open files and 311 files are closed files.

Periodic system activity statistics

The "SHOW MEM/CACHE" DCL command provides the XFC memory utilization statistic at the system level. It also provides other important statistics such as read hit rate and the number of read/write IOs at the system level. XFC records the activity of the system on a periodic basis, which is at a regular interval every 10 minutes. XFC records various information about the system, which would be useful to track the behavior of the system. Some of the statistics captured are:
- Number of read/write IOs
- Read hit rate
- Open/closed files in cache
- XFC cache memory

SDA command
```
SDA>XFC SHOW HISTORY
```

Sample output
```
SDA>XFC SHOW HISTORY
XFC Performance History (current time 30-MAY-2010 05:20:43.64)
--------------------------------------------------------------
    Time      Peak.....Cache   I/O   Counts.......     Allocated
               IO     Reads  Hits   Rate   Writes     Memory
...
29-MAY 01:05     1       10      7   70%      316 ...  73.5 MB
29-MAY 00:55     3    49364  46923   95%     2231 ...  71.0 MB
29-MAY 00:45     1      315    265   84%       25 ...  66.9 MB
29-MAY 00:35     8    17234  13279   77%      167 ...  65.9 MB
...
SDA>
```

Interpreting the output
Between the time intervals of 29-MAY 00:35 and 29-MAY 00:45, the following data was gathered:
- Number of read IOs was 315, number of write IOs was 25
- Read hit rate was 84 percent

- The XFC cache memory increased from 65.9 MB to 66.9MB

Systemwide IO size distribution statistic

XFC also keeps track of all the IOs issued in the system based on the IO size, i.e., 1 block or 2 blocks, and so on.

For IOs of particular size, XFC keeps track of statistics such as:

- Number of read operations
- Number of write operations
- Number of read hits

This statistic provides information on the read hit of IOs based upon the IO size.

DCL command
```
$ SHOW MEM/CACHE/FULL
```

Sample output
```
$SHOW MEMORY/CACHE/FULL
              System Memory Resources on 31-MAY-2010 02:36:47.05

Extended File Cache  (Time of last reset: 29-MAY-2010 00:25:38.06)
...
I/O Statistics - Distributions (MAX_IO_SIZE: 127)
-------------------------------------------------
Transfer Size:      Reads  Read Hits  Writes
   1 Block IO:      28160  27428       2384
   2 Block IO:       6948   6345         24
   3 Block IO:        402    307         34
...
 126 Block IO           2      0          0

>127<256 Block IO:     27                 3
>255     Block IO:     53                 0
-------------------------------------------------
Totals:              73403  65848     34574
$
```

Interpreting the system output

- 2384 write IOs were of size 1 block. 28160 IOs were of size 1 block and among these 27428 IOs were a read hit.
- 27 read IOs and 3 write IOs that were issued were of size in the range of 127-256 blocks. None of them was a read hit.
- 53 read IOs and no write IOs issued were of size greater than 255 blocks. None of them was a read hit.

SDA command
```
SDA> XFC SHOW SUMMARY/STATS
```

Sample output
```
SDA> XFC SHOW SUMMARY/STATS
XFC Summary (current time 31-MAY-2010 02:51:16.69)
--------------------------------------------------
...
I/O Statistics - Distributions (MAX_IO_SIZE: 127)
---------------------------------------------------
Transfer Size     Reads     Hit Rate   Writes ...
 (blocks)
```

```
  1               28196        97%     2384 ...
  2                6962        91%       24 ...
  3                 403        76%       34 ...
...
Reads between 128 and 255 blocks since last reset:      27
Read hits between 128 and 255 blocks since last reset:   0 (0%)
Reads larger than 256 blocks since last reset:          53
Read hits larger than 256 blocks since last reset:       0 (0%)
Writes between 128 and 255 blocks since last reset:      3
Writes larger than 256 blocks since last reset:          0
...
SDA>
```

Interpreting the system output

- 28196 read IOs and 2384 write IOs were of size 1 block. The read hit rate for 1 block IOs is 97 percent.
- 27 read IOs and 3 write IOs issued were of size in the range of 127-256 block. None of them was a read hit.
- 53 read IOs and no write IOs that were issued were of size greater than 255 blocks. None of them was a read hit.

Volume statistics

The "SHOW MEM/CACHE=VOL=<VOL-NAME>" DCL command or the “SDA>XFC SHOW VOLUME/NAME=<VOL-NAME>” command provides the XFC statistics at the volume level. Some of the important statistics are read hit rate, and the number of read/write IOs.

DCL command

```
$ SHOW MEMORY/CACHE=VOLUME=<VOL-NAME>
<VOL-NAME>: Name of the volume
```

Sample output

```
$SHOW MEMORY/CACHE=VOLUME=BRAMHA$DKA100:
              System Memory Resources on 30-MAY-2010 05:38:56.87

Extended File Cache Volume Statistics:

_BRAMHA$DKA100: (DISK$BR_V84R), Caching mode is VIOC Compatible
    Open files           398     Closed files           314
    Files ever opened   1519     Files ever deposed     807
    Allocated pages     9715     Locks acquired           0
    Total QIOs         85299     Read hits            61370
    Virtual reads      66304     Virtual writes       18995
    Read hit rate         92 %   Read aheads            344
    Read throughs      66132     Write throughs       18992
    Read arounds         172     Write arounds            3
    Ave Disk I/O Resp Time incl cache hits (microseconds) 1126

Total of 1 volume in cache
$
```

Interpreting the output

On the volume BRAMHA$DKA100:

- The number of open files in cache is 398, and the number of closed files in the cache is 314.
- This volume has consumed 9715 pages of the total XFC cache memory

- The read hit rate of the volume is 92 percent. This is a very good read hit rate.
- Total IOs done on this volume is 85299. Out of this amount, 66304 IOs are virtual reads and 18995 IOs are virtual writes.

SDA command
```
SDA> XFC SHOW VOLUME/NAME=<VOL-NAME>
<VOL-NAME> = DISK$<LBL>
             <LBL> is the label of the volume.
```

Sample output
```
SDA> XFC SHOW VOLUME/NAME=DISK$BR_V84R

Full List of XFC Cached Volumes (CVBs)
--------------------------------------
Cache Volume Block (CVB)
------------------------
Statistics Valid From:   29-MAY-2010 00:25:40.06
...
Cached Open Files:       400
Cached Closed Files:     312
...
Name:                    DISK$BR_V84R
QIO count:               85447
Virtual Read Count:      66452
Virtual Write Count:     18995
Read Percentage:         77 %
Hit Rate:                71 %
...
I/O Response Times (This Volume)
  Overall Average:   1.1242 milliseconds
  Cache Hit:         0.0070 milliseconds
  Disk:              3.9962 milliseconds
  Accuracy:              99 %
...
SDA>
```

Interpreting the output

On the DISK$BR_V84R volume:

- The number of open files in cache is 400, and the number of closed files in the cache is 312.
- The hit rate of the volume is 71 percent.
  Note: This is its hit rate and NOT the read hit rate, where
  Hit rate = read hits / (virtual reads + virtual writes)
- Total IOs done on this volume is 85447. Out of this amount, 66452 IOs are virtual reads and 18995 IOs are virtual writes.
- XFC computes the IO response time for various IOs that are issued to this volume.
  I.e., on any given volume, for each IO, XFC keeps track of the following:
  o IOs that start and end on the same CPU and are cache hits
  o IOs that start and end on the same CPU and are cache misses

File statistics

The "SHOW MEM/CACHE=FILE=<FILE-NAME>" DCL command or the “SDA>XFC SHOW FILE/ID=<FILD-IN-HEX>” command provides the XFC statistics at the file level. Some of the important statistics are read hit rate, number of read/write IO's, and XFC cache memory consumed by the file.

DCL command
```
$SHOW MEMORY/CACHE=FILE=<FILE-NAME>
<FILE-NAME>: Full path of the file
```

Sample output
```
$SHOW MEMORY/CACHE=FILE=BRAMHA$DKB200:[MURALI]PERFORM_TEST.COM
              System Memory Resources on 30-MAY-2010 06:01:04.46
Extended File Cache File Statistics:

_BRAMHA$DKB200:[MURALI]PERFORM_TEST.COM;68 (389451, 1057, 0) (closed)
 Caching is enabled, active caching mode is Write Through
    Allocated pages      1     Total QIOs           1177
    Read hits         1176     Virtual reads        1176
    Virtual writes       1     Hit rate              100 %
    Read aheads          0     Read throughs        1176
    Write throughs       1     Read arounds            0
                               Write arounds           0
$
```

Interpreting the output
For the file PERFORM_TEST.COM (389451, 1057, 0):

- This file has consumed one page of the total XFC cache memory.
- The read hit rate of the file is 100 percent. This means that all attempts to access the file were satisfied from the cache itself.
- Total IOs done on this file is 1177. Out of these, 1176 IOs are virtual reads and 1 IO is virtual write.

SDA command
```
SDA> XFC SHOW FILE/ID=<FID-IN-HEX>/STATS
<FID-IN-HEX> = File number component of the file's FID in Hex
```

Sample output
```
SDA> XFC SHOW FILE/ID=2368/STATS

Full XFC Cache File Block (CFB) Details
---------------------------------------
File: DISK$TEST:[SYS0.SYSERR]ERRLOG.SYS;1
...
External FID:          (9064,1,0)
...
File IO Statistics (current time 29-MAY-2010 09:19:13.85)
---------------------------------------------------------
Statistics Valid From:   24-MAY-2010 03:13:24.80
Total QIOs to this file:                2297
Read IOs to this file:                  1530
Read Hits:                              1220
Hit Rate:                              53.11 %
Write IOs to this file:                  767
Truncates:                                 1
Accesses:                                765
Deaccesses:                              765

I/O Response Times (This File)
  Overall Average:                     1.9058 milliseconds
  Cache Hit:                           0.0266 milliseconds
  Disk:                                4.0346 milliseconds
  Accuracy:                              100 %
...
SDA>
```

Interpreting the output
For the file ERRLOG.SYS (9064, 1, 0):

- Total IOs done on this file is 2297. Out of these, 1530 IOs are virtual reads and 767 IOs are virtual writes.
- The hit rate of the file is 53.11 percent.
  (Note: This is the hit rate and NOT the read hit rate).
  Hit rate = read hits / (virtual reads + virtual writes)
- There has been 1 truncate, 765 access, and 765 deaccess operations on the file
- XFC computes the IO response time for various IOs that are issued to this file
  I.e., on the file, for each IO, XFC keeps track of:
  o IOs that are cache hits which start and end on the same CPU
  o IOs that are cache misses which start and end on the same CPU

IO statistics

XFC creates a context block for every IO that it handles. The context block stores various information about the IO request such as IRP, operation start time, current IO phase and so on. At any given point of time, the context block indicates the current state of the IO.

IO statistics are generally used for debugging purpose. While analyzing the crash dump, you may be interested to know the current state of an IO request. In such cases, if the IO request has gone through XFC, then XFC would have created a context block for that IO request. The context block can then be examined to know the current state of the IO request.

If XFC is handling multiple IO requests, then it would have created multiple context blocks, one for each IO request. In such cases, you must examine the context block that corresponds to the IO request that you are looking for. This is done by examining the list of all the context blocks, and then selecting a particular context block based on whether its IRP field corresponds to the IRP we are looking for.

SDA command
```
SDA> XFC SHOW CONTEXT
```

Sample output
```
SDA> XFC SHOW CONTEXT

List of All XFC Active Contexts (CTX) (current time 31-MAY-2010
03:04:47.33)
---------------------------------------------------------------
Context (CTX) Address: FFFFFFFF89379910
I/O Phase:             eiopDiskIO
I/O Type:              eiotWriteThrough
Operation started:     31-MAY-2010 03:04:47.32
Stall Reason:          estrDiskIO
...
Stall Op (IRP):        FFFFFFFF8916DBC0
...
  Cache Hit:           False
...
Volume (CVB):          0000000000000000
Volume Id:              FFFFFFFFEE0189120
File Id:                000000000000398F
...
Cache File Block:       FFFFFFFF7D715940
Process (PCB):          FFFFFFFF896B0CC0
...
SDA>
```

Interpreting the output

For the context block FFFFFFFF89379910:

- XFC received the IO request on "31-MAY-2010 03:04:47.32".
- The IO request is for a write operation.
- The file id of the file to which this IO is done is "398F"
- This IO request is not a cache hit, and hence the data has to be fetched from the disk
- Other structures associated with this IO are:
    - IRP - FFFFFFFF8916DBC0 --> IO request packet
    - PCB - FFFFFFFF896B0CC0 --> Process control block

Summary

The techniques described here allow a user to get the XFC statistics at the system, volume, file, or IO level. These statistics should then be analyzed to determine the hot files in the system at the volume level, and also if the overall system performance is acceptable or not.

One of the important parameters that impacts the system performance is the read hit rate of the system/volume/file, which depends on a number of factors, such as:

- Number of local read/write IOs to the file
- Number of cluster-wide read/write IOs to the file

Once we collect the XFC statistics, for example,

- Read hit rate of the system is 50 percent

This means the overall cache read hit rates on the system is 50 percent. This is very low and hence there is a need to tune the system to increase the overall cache read hit rates. This would involve analyzing the IOs issued on the system, i.e., on different volumes by various applications.

- Read hit rate of the volume V1 is 50 percent

This means that the cache hit rate on the volume V1 is 50 percent. This is very low and hence there is a need to tune the system with respect to the IO operations on the volume to increase the read hit rate of the volume. This would involve analyzing the IOs issued on the volume V1 by various applications.

Implementing Triple DES (TCBC) on OpenVMS

Rupesh Shantamurty

Introduction

Data Encryption Standard (DES) has been supported by OpenVMS since Version 7.3 and support for Advanced Encryption Standard (AES) was added in OpenVMS Version 8.3. Both these encryption modes are now available along with the operating system. Triple Data Encryption Standard, also known as 3DES or TDEA was introduced as a temporary replacement for DES till AES was made available.
Using DES three times consecutively is 3DES. Following are the four modes of DES operation:

- Electronic Codebook (ECB) mode
- Cipher Block Chaining (CBC) mode
- Cipher Feedback (CFB) mode
- Output Feedback (OFB) mode

For more information about DES, see the NIST document[5]. 3DES has seven modes of operation, four of which have been derived from DES.

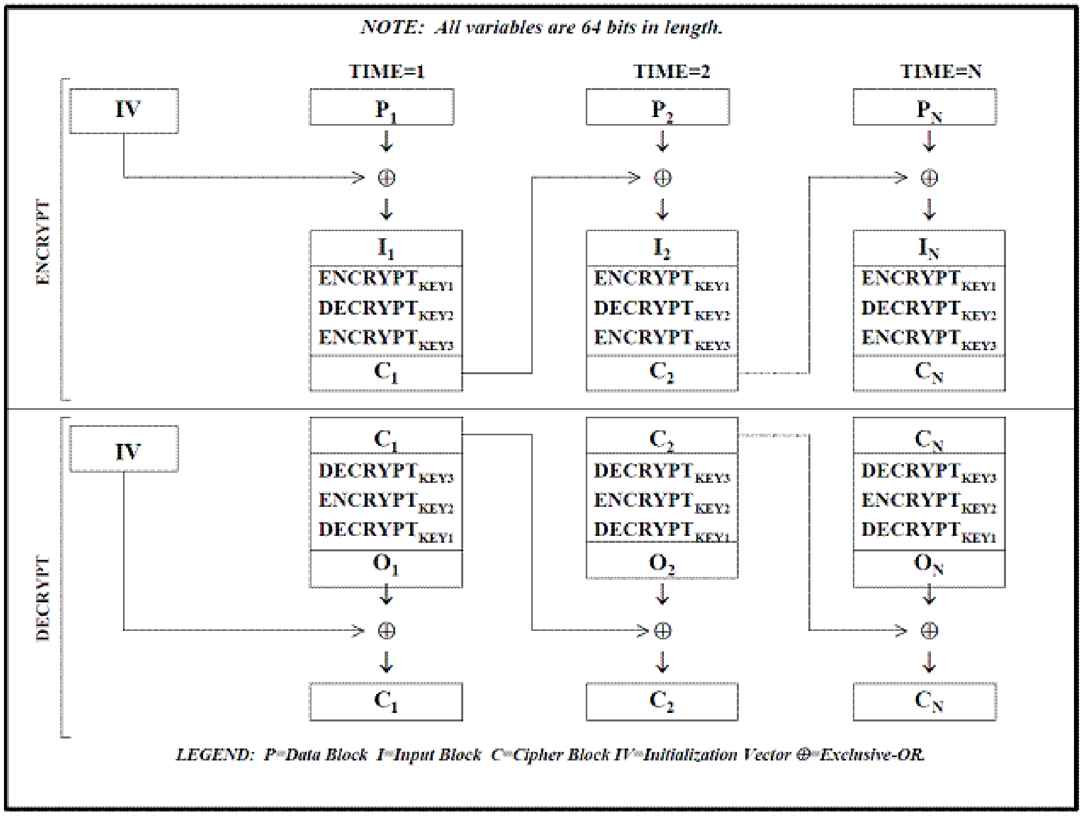
OpenVMS does not provide direct Application Programming Interface (API) to encrypt or decrypt using 3DES. But some OpenVMS users have used the available DES APIs to encrypt and decrypt. There are a few subtle points which should be considered when implementing 3DES using the DES APIs on OpenVMS.

This article demonstrates how to encrypt and decrypt programmatically in C, using the example of Cipher Block Chaining (CBC) mode of operation. Coding has been done based on the schema presented in the NIST document explaining TDEA[6] as follows:

5 DES http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

6 TDEA http://csrc.nist.gov/publications/nistpubs/800-20/800-20.pdf

TDEA Cipher Block Chaining (TCBC) Mode



NOTE: All variables are 64 bits in length.
TIME=1
TIME=2
TIME=N
IV
P1
P2
PN
I1
I2
IN
ENCRYPT KEY1
DECRYPT KEY2
ENCRYPT KEY3
C1
C2
CN
ENCRYPT
DECRYPT
DECRYPT KEY3
ENCRYPT KEY2
DECRYPT KEY1
O1
O2
ON
LEGEND: P=Data Block I=Input Block C=Cipher Block IV=Initialization Vector ⊕=Exclusive-OR.

The most important thing to remember in this implementation is to take just 64 bits of data at a time. For the encryption phase, the cipher output of the first set of 64 bits is used as the IV (Initialization Vector) for the second set of 64 bits, and so on. For decryption, the IV will have to be used only in the last decrypt operation using key 1 and the IV for the next set of 64 bits is the first set of 64 bits of the cipher text.

Below is the program in C programming language for encryption using TDEA CBC mode

tdea_vms.c

```
----------------
/* The cipher text generated can be verified using the Cryptosys API
 * ( www.cryptosys.net ,the Personal Edition is free for personal
 *   use on a stand-alone computer)
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <descrip>
#include <ssdef>
#include <time>
#include <libdtdef>
#include <lib$routines.h>
#include "ENCRYPT$EXAMPLES:encrypt_def.h"
#include <string.h>

#define KEY_VALUE_FLAG 1
char cbc_out[8];
char *des_cbc_once(char *data,char *key, char *iv,int option);
```

```
void main ()
{
  unsigned char input_data[32], my_data[32], decryp_data[32];

  /* Note: The key has been hardcoded to avoid any endian conflicts,
   * Refer to OpenVMS FAQ for more information on endian
considerations.
   */

  unsigned char mykey[24]= { 0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef,
                             0xfe,0xdc,0xba,0x98,0x76,0x54,0x32,0x10,
                             0xab,0xcd,0xef,0x01,0x23,0x45,0x67,0x89
                            };

  /* The initialization vector has been hardcoded */

  unsigned char iv[8]={0x12,0x34,0x56,0x78,
                       0x90,0xab,0xcd,0xef};

  static unsigned char cbc_iv[8];

  unsigned char * output;
  unsigned char cipher[32],des_data[8],key1[8],key2[8],key3[8];

  int i,input_length,noof64bit;

  memset(my_data,0,32);
  printf("\nType the text that needs to be encrypted(max 32 chars) :");
  gets((char *)input_data);
  input_length = strlen((char *)input_data);

  noof64bit = ceil((float)input_length/8);
  memcpy(my_data,input_data,noof64bit*8);

  printf("\n The Input text in hex is : \n");
  for (i=0;i<noof64bit*8;i++)
     printf("%.2X",(unsigned char *)my_data[i]);

  /* Set the IV to it’s initial value */
  memcpy(cbc_iv,iv,8);

  printf("\n The IV in hex is : ");
  for (i=0;i<8;i++)
     printf("%.2X",(unsigned char *)cbc_iv[i]);

  printf("\n The 192 bit Key in hex is : \n");
  for (i=0;i<24;i++)
     printf("%.2X",(unsigned char *)mykey[i]);

  memcpy(key1,mykey,8);
  memcpy(key2,mykey+8,8);
  memcpy(key3,mykey+16,8);

  /* Encryption Phase */

  for (i=1;i<=noof64bit;i++)
  {
     memset(des_data,0,8);
     memcpy( des_data,my_data+((i-1)*8),8);

     /* The Core part of encryption in TCBC mode */
```

```
      /* First encrypt using key1 , use IV also */
      output = (unsigned char *)des_cbc_once((char *)des_data,
                                             (char *)key1,
                                             (char *)cbc_iv,1);

      /* Decrypt the output of previous step using key2 */
      output = (unsigned char *)des_cbc_once((char *)output,
                                             (char *)key2,NULL,0);

      /* Finally encrypt the output of previous step using key3 */
      output = (unsigned char *)des_cbc_once((char *)output,
                                             (char *)key3,NULL,1);

      memcpy(cipher+((i-1)*8),output,8);

      memcpy(cbc_iv,output,8);
   }
   printf("\n The Cipher text in hex is : \n");
   for (i=0;i<noof64bit*8;i++)
      printf("%.2X",(unsigned char *)cipher[i]);

   /* Decryption Phase */

   /* Set the IV to it's initial value */
   memcpy(cbc_iv,iv,8);

   for (i=1;i<=noof64bit;i++)
   {
      memset(des_data,0,8);

      memcpy( des_data, cipher+((i-1)*8),8);

      /* The Core part of deryption in TCBC mode */

      /* First decrypt using key3 , with no IV */
      output = (unsigned char *)des_cbc_once((char *)des_data,
                                             (char *)key3,NULL,0);

      /* Encrypt the output of previous step using key2 */
      output = (unsigned char *)des_cbc_once((char *)output,
                                             (char *)key2,NULL,1);

      /* Finally decrypt the output of previous step using key1,
         use IV also */
      output = (unsigned char *)des_cbc_once((char *)output,
                                             (char *)key1,
                                             (char *)cbc_iv,0);
      memcpy(decryp_data+((i-1)*8),output,8);

      memcpy(cbc_iv, des_data,8);
   }
   printf("\n The Deciphered text in hex is : \n");
   for (i=0;i<noof64bit*8;i++)
      printf("%.2X",(unsigned char *)decryp_data[i]);

   printf("\n The Deciphered text is : \n");
   puts((char*) decryp_data);

} /* End of main */

/* Function to do CBC, using DES API, once */
```

```
char * des_cbc_once( char *data,char *key, char *iv,int option)
{
    char int_data[8],int_key[8],int_iv[8];
    unsigned long  context     = 0;
    unsigned long  key_type    = KEY_VALUE_FLAG ;
    int            encr_out_len = 0;
    long           status      = 0;

    memcpy(int_data,data,8);
    memcpy(int_key,key,8);
    if(iv!=NULL)
    memcpy(int_iv,iv,8);

    /* Define input, output, algo and key descriptors */
    $DESCRIPTOR(input_desc,     "");
    $DESCRIPTOR(output_desc,    "");
    $DESCRIPTOR(algo_desc,"DESCBC");
    $DESCRIPTOR(key_buf_desc, "");

    input_desc.dsc$b_dtype = DSC$K_DTYPE_NU ;
                                /* Numeric String Unsigned    */
    input_desc.dsc$a_pointer = (char *)&int_data;
    input_desc.dsc$w_length  = 8;

    output_desc.dsc$b_dtype = DSC$K_DTYPE_NU ;
    output_desc.dsc$a_pointer = (char *)&cbc_out;
    output_desc.dsc$w_length = 8;

    key_buf_desc.dsc$b_dtype = DSC$K_DTYPE_NU ;
    key_buf_desc.dsc$a_pointer = (char *)&int_key;
    key_buf_desc.dsc$w_length  = 8 ;

    status  = encrypt$init ( &context , &algo_desc , &key_type,
                         &key_buf_desc , int_iv );

    if (!(status & 1))
    {
        printf ("\nencrypt$init() function failed\n");
        lib$signal (status);
        return (0);
    }
    memset(cbc_out,0,8);

    /* If option is 1 then do encryption, else do decryption */

    if(option==1)
    {
       status = encrypt$encrypt( &context, &input_desc,
                                 &output_desc, &encr_out_len );
       if (!(status & 1))
       {
         printf ("\n encrypt$encrypt() function failed.\n");
         lib$signal (status);
         return (0);
       }
    }
    else
    {
       status = encrypt$decrypt( &context, &input_desc ,
                                 &output_desc, &encr_out_len );
       if (!(status & 1))
```

```
        {
          printf ("\n encrypt$decrypt() function failed.\n");
          lib$signal (status);
          return (0);
        }
    }

    status = encrypt$fini (&context);
    if (!(status & 1))
    {
        printf ("\n encrypt$fini() function failed.\n");
        lib$signal (status);
        return (0);
    }
    return cbc_out;
}
```

Sample output

```
$ run TDEA_VMS

Type the text that needs to be encrypted(max 32 chars) : Now is the
time for all
 The Input text in hex is :
4E6F77206973207468652074696D6520666F7220616C6C20
 The IV in hex is : 1234567890ABCDEF
 The 192 bit Key in hex is :
0123456789ABCDEFFEDCBA9876543210ABCDEF0123456789
 The Cipher text in hex is :
80B31486E9FE855A033837A54A4DDF1A97E7D083F1FD8269
 The Deciphered text in hex is :
4E6F77206973207468652074696D6520666F7220616C6C20
 The Deciphered text is :
Now is the time for all
$
```

Summary

The example program has demonstrated how easy it is to implement TCBC mode of 3DES on OpenVMS. You can also implement other modes of 3DES using a similar method. For more information, please refer to the NIST documentation.

Class Scheduling on OpenVMS

Shriniketan Bhagwat

Executive Summary

This article outlines the OpenVMS class scheduling, its usage, its advantages, and its internal data structures.

The following topics are covered:

- Introduction to class scheduling
- Class scheduling advantages
- Class scheduler database file
- In-memory data structures

Introduction to OpenVMS class scheduling

The class scheduler provides the ability to limit the amount of CPU time that a system's users may receive by placing the users into scheduling classes. Each class is assigned a percentage of the overall system's CPU time. As the system runs, the combined sets of users in a class are limited to the percentage of CPU execution time allocated to their class. To invoke the class scheduler, use the SYSMAN interface.

Class scheduling is implemented in the SYSMAN utility, which allows users to define classes based on username, UIC, or account. SYSMAN allows users to create, delete, modify, suspend, resume, and display scheduling classes.

Creating a scheduling class

The CLASS_SCHEDULE ADD command creates a scheduling class. The highlights of the CLASS_SCHEDULE ADD command include:

- Identifies users in the class, by account name, user name, or UIC
- Specifies the percent of CPU time allotted to processes that are run by users in this scheduling class on primary days and secondary days, and specifies the hourly ranges during which the CPU time restriction applies
- Allows a scheduling class to receive additional CPU time when the CPU is idle.

Note:
Creating the scheduling class does not affect a process already running. Adding the scheduling class through SYSMAN requires OPER privilege.

Example:
```
SYSMAN> CLASS_SCHEDULE ADD MAINCLASS -
_SYSMAN> /ACCOUNT = (ACCTNAME1, ACCTNAME2) -
_SYSMAN> /USERNAME = HOTSHOT -
_SYSMAN> /CPULIMIT = (PRIMARY, 08-17=15, SECONDARY, 00-23=30) -
_SYSMAN> /WINDFALL
```

Deleting a scheduling class

The `CLASS_SCHEDULE DELETE` command deletes the scheduling class from the class scheduler database file. The highlights of the CLASS_SCHEDULE DELETE command include:

- All processes that are members of this scheduling class are no longer class scheduled.

- A scheduling class cannot be deleted if the scheduling class is active, where active is represented by any of the users who belong to the class having a logged-in status.

Note: Deleting the scheduling class through SYSMAN requires OPER privilege.

Example:
```
 SYSMAN> CLASS_SCHEDULE DELETE MAINCLASS
```

Modifying a scheduling class

The CLASS_SCHEDULE MODIFY command changes the characteristics of a scheduling class.

Note:
Modifying the scheduling class through SYSMAN requires OPER privilege.

Example:
```
SYSMAN> CLASS_SCHEDULE MODIFY MAINCLASS -
_SYSMAN> /ACCOUNT = (ACCTNAME1, ACCTNAME2) -
_SYSMAN> /USERNAME = HOTSHOT -
_SYSMAN> /CPU_LIMIT = (PRIMARY, 08-13=20, SECONDARY, 00-20=40) -
_SYSMAN> /NOWINDFALL
```

Displaying a scheduling class

The CLASS_SCHEDULE SHOW command displays information about a scheduling class.

Note:
Displaying the scheduling class through SYSMAN requires OPER privilege.

Example:
```
SYSMAN> CLASS_SCHEDULE SHOW MAINCLASS /FULL
```

Suspending a scheduling class

The CLASS_SCHEDULE SUSPEND command suspends the specified scheduling class. All processes that are part of the scheduling class remain as part of the scheduling class but are granted unlimited CPU time.

Note:
Suspending the scheduling class does not affect a process already running. Suspending the scheduling class through SYSMAN requires OPER privilege.

Example:
```
SYSMAN> CLASS_SCHEDULE SUSPEND MAINCLASS
```

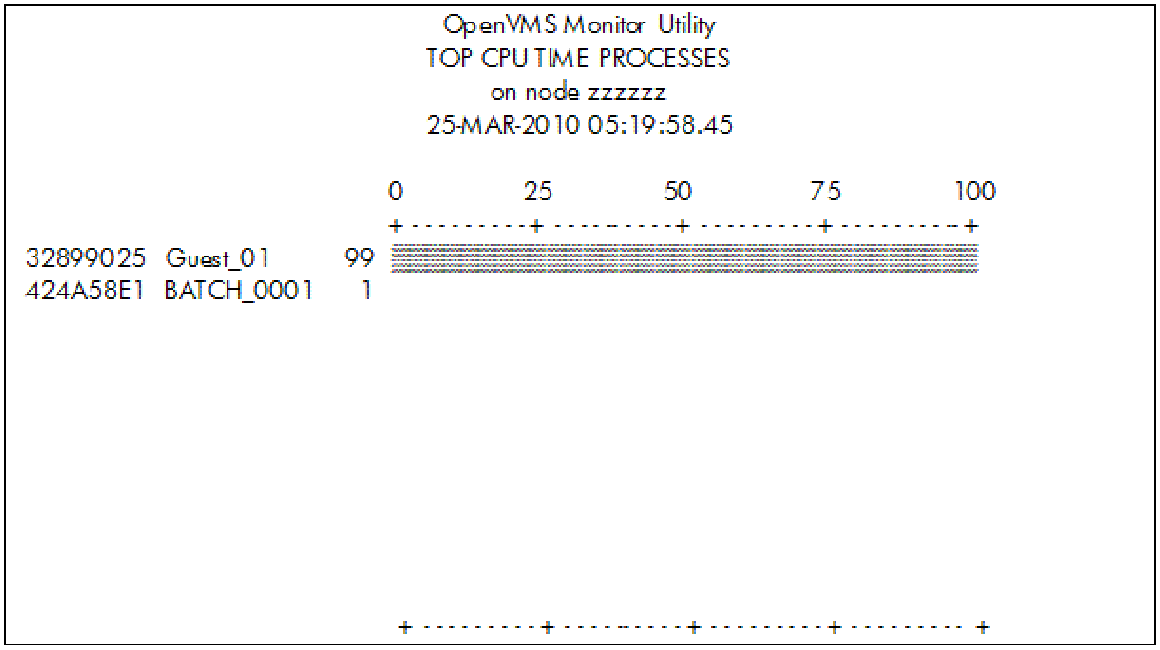
Resuming a scheduling class

The CLASS_SCHEDULE RESUME command resumes a scheduling class that is currently suspended.

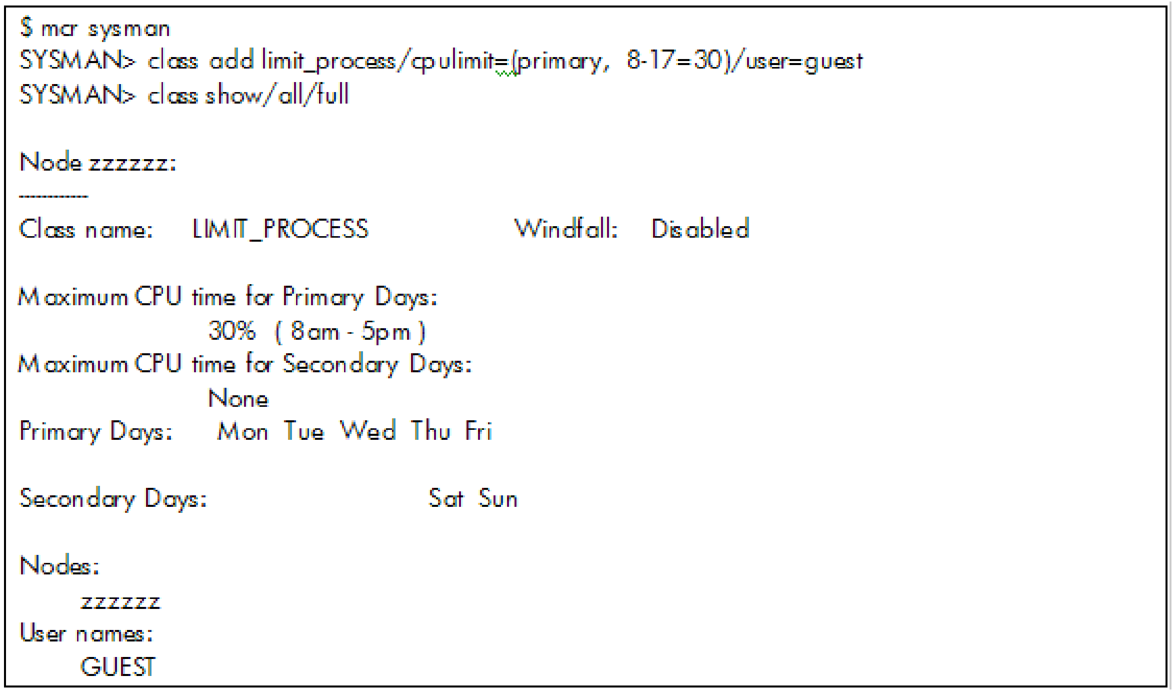
Note:
Resuming the scheduling class does not affect a process already running. Resuming the scheduling class through SYSMAN requires OPER privilege.

Example: Harsha Swaroop
```
SYSMAN> CLASS_SCHEDULE RESUME MAINCLASS
```
Advantages of class scheduling

The following example explains the use of the class scheduling. The user has installed a third-party application, for example, ABC, over which the user does not have any control. When ABC is started, assume that it consumes an entire CPU's processing power and impacts other applications on the system. The output of $ MONITOR PROCESS/TOPCPU is similar to the following:

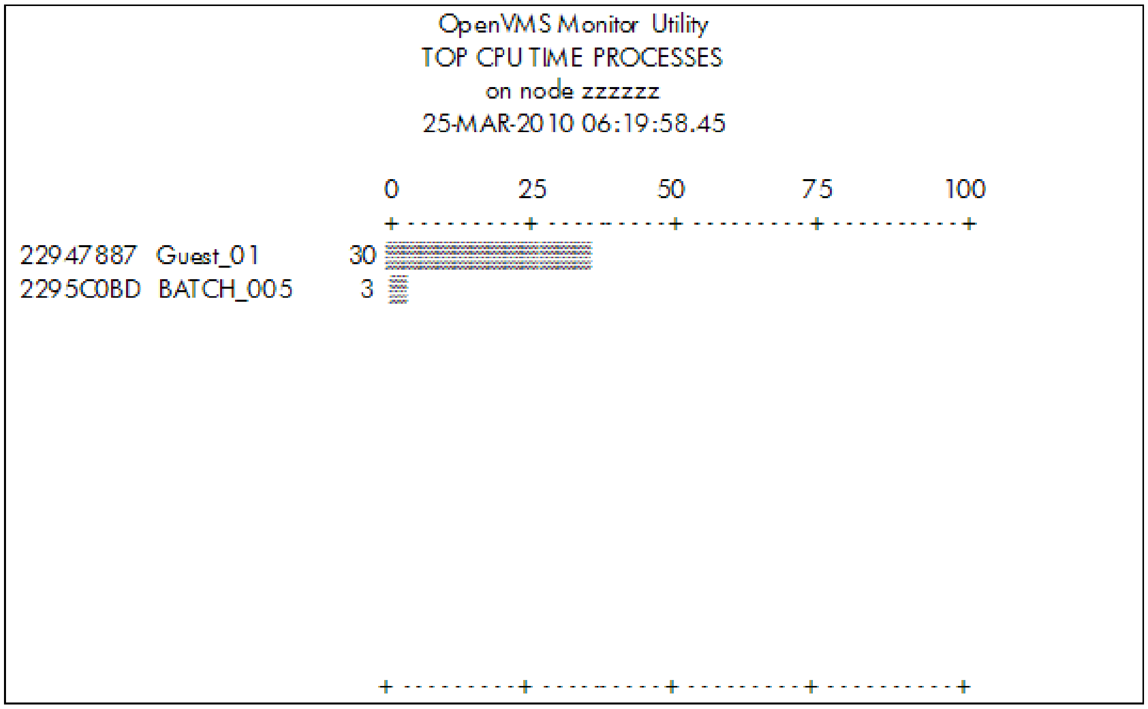

```
                         OpenVMS Monitor Utility
                        TOP CPU TIME PROCESSES
                            on node zzzzzz
                        25-MAR-2010 05:19:58.45

                         0        25        50        75       100
                         + - - - - -+- - - - - + - - - - -+- - - - - +
32899025 Guest_01    99
424A58E1 BATCH_0001   1



                         + - - - - -+- - - - - + - - - - -+- - - - - +
```

To overcome this problem, create a class to limit the amount of CPU the processes can consume. The process is running under the username "GUEST". You can create a class to limit the CPU for this username to 30 percent during working hours, Monday to Friday, and allow the user free reign on the weekends and any time outside the primary hours specified:



```
$ mcr sysman
SYSMAN> class add limit_process/cpulimit=(primary,  8-17=30)/user=guest
SYSMAN> class show/all/full

Node zzzzzz:
------------
Class name:    LIMIT_PROCESS              Windfall:   Disabled

Maximum CPU time for Primary Days:
                30%  ( 8am - 5pm )
Maximum CPU time for Secondary Days:
                None
Primary Days:    Mon  Tue  Wed  Thu  Fri

Secondary Days:                    Sat  Sun

Nodes:
      zzzzzz
User names:
      GUEST
```

Now, when the application ABC is run from the GUEST account during the day, you can see the following from the MONITOR output. If the application is invoked by different users from a different user account, then you can put all the users into a class to limit the amount of CPU the processes can consume.



```
                                  OpenVMS Monitor Utility
                                  TOP CPU TIME PROCESSES
                                     on node zzzzzz
                                  25-MAR-2010 06:19:58.45

                              0         25        50        75        100
                              + - - - - - - - - -+ - - - - - - - - -+ - - - - - - - - -+ - - - - - - - - - -+
22947887  Guest_01         30 ▒▒▒▒▒▒▒▒▒▒▒▒
2295C0BD  BATCH_005         3 ▒

                              + - - - - - - - - -+ - - - - - - - - -+ - - - - - - - - -+ - - - - - - - - - -+
```

Class scheduler database file

SYSMAN creates a class scheduler database file called
VMS$CLASS_SCHEDULE.DATA
when the first scheduling class is created by the CLASS_SCHEDULE ADD SYSMAN command. This database resides on the system disk in
SYS$SYSTEM:VMS$CLASS_SCHEDULE.DATA and is an RMS-indexed file. This

database file is a permanent database file, which allows OpenVMS to class schedule processes automatically after a system has been booted and rebooted.

In case you want to have a common class scheduler database on a cluster with different system disks, or if you want to have a separate database for each node on a cluster with a common system disk, then on each node you must define the system logical name:
VMS$CLASS_SCHEDULE
to point to the location of the database that you want that node to use.

For example:

```
$ DEFINE/SYSTEM VMS$CLASS_SCHEDULE
disk:[directory]VMS$CLASS_SCHEDULE.DATA
```

Process creation

Once a scheduling class is added, the SYSUAF file is updated. By using the class scheduler, a process is placed into a scheduling class at process creation time. Once a new process is created, it needs to be determined whether this process belongs to a scheduling class based upon the data present in the SYSUAF file. The Loginout image obtains the process information from SYSUAF file. Loginout class schedules the process if it determines that the process belongs to a scheduling class. Adding the scheduling class does not affect a process already running.

There are two types of processes: subprocess and detached process. A subprocess becomes part of the same scheduling class as the parent process, even though it may not match the class's criteria. That is, its user and account name and/or UIC may not be part of the class's record. A detached process only joins a scheduling class if it executes the Loginout image during process creation.

Though a process can join a scheduling class at process creation time, a user can change or modify its scheduling class during runtime with the $ SET PROCESS/SCHEDULING_CLASS command.

How to determine if a process is class scheduled?

To determine whether a process is class scheduled, use one of the following methods:

SHOW command

You can find out if a process is class scheduled by issuing the $ SHOW PROCESS/SCHEDULING_CLASS command. You can also use $ SHOW SYSTEM /SCHEDULING_CLASS[=class_name], which displays processes that belong to a specific scheduling class (class_name). If the class name is not specified, all class scheduled processes are displayed along with the name of their scheduling class.

SYS$GETJPI system service

The JPI$_CLASS_NAME item code in the system service $GETJPI can be used to get the name of the scheduling class the process belongs to. If the process is not class scheduled, then this system service returns zero (0) to the caller.

Authorize utility

When a new user is added to the SYSUAF file, or when a user's record is modified, the Authorize utility searches the class scheduler database file to determine if this user is a member of a scheduling class. If the user is a member, then Authorize displays "UAF-I-SCHEDCLASS", which indicates that the user is a member of a scheduling class.

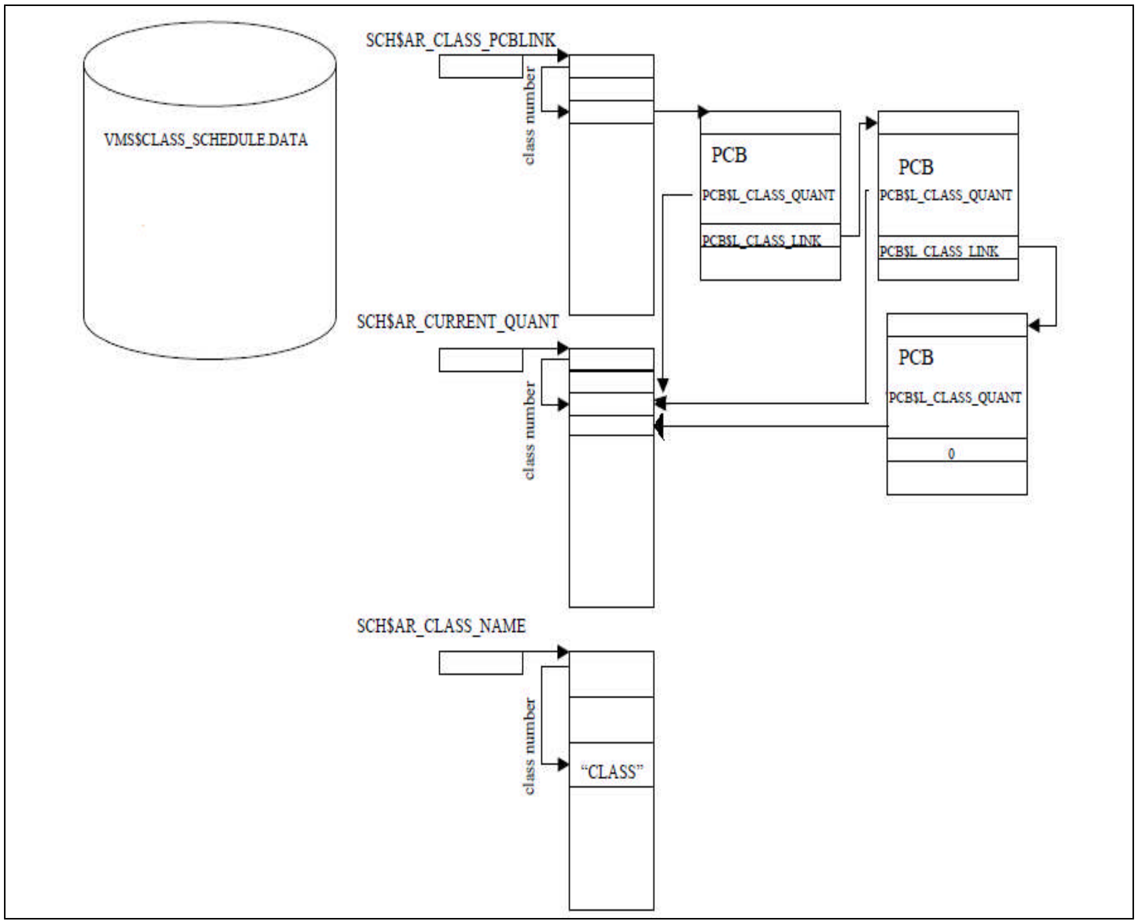
SYS$SCHED system service

The SYS$SCHED system service allows you to create a temporary class scheduling database. The processes are class-scheduled by PID, after the process has been created. The SYSMAN interface creates a separate and permanent class scheduling database that schedules you at process creation time. A process cannot belong to both databases, the SYS$SCHED and SYSMAN database. Therefore, the SYS$SCHED system service checks to see if the process to be inserted into a scheduling class is already class scheduled before it attempts to place the specified process into a scheduling class. If it is already class scheduled, then the SS$_INSCHEDCLASS error message, is returned from SYS$SCHED. There is an example of this in SYS$EXAMPLES:CLASS.C on a running OpenVMS system.

In-memory data structure

The CLASS_SCHEDULE ADD command adds the class to the scheduler database file and creates the in-memory database array or SCH$AR_ORIGINAL_QUANT. When you log in, if this in-memory database array exists then, Loginout image calls the SYSMAN routine to find the scheduling class associated with that account. If the logged in user's username or UIC or account belongs to the scheduling class, then SYSMAN routine puts the process which belongs to the account into the scheduling class by updating the below PCB fields of the process:

- Bit PCB$V_CLASS_SCHEDULED indicates that this process is subject to class scheduling. This bit is part of PCB$L_STS2.
- Bit PCB$V_CLASS_SUPPLIED indicates that a class scheduler has specified a class for this process. This bit is also part of PCB$L_STS2.
- Process PCB is placed in a linked list of PCBs for all members of the class. The list is located by indexing, using the class number, into an array pointed to by SCH$AR_CLASS_PCBLINK. The field PCB$L_CLASS_LINK points to the next PCB in this class, if any. Bit PCB$V_WINDFALL indicates that the process is eligible for scheduling, even though its class is out of quantum, if there are no other eligible kernel threads to schedule.
- PCB$L_CLASS_QUANT contains the address of a class quantum data structure (that is SCH$AR_CURRENT_QUANT) created by the $SCHED system service in non-paged pool when a new class is defined.
- Class name is stored in an array of 16 byte entries pointed to by SCH$AR_CLASS_NAME.
- Primary/secondary day/hour restrictions are described in the array SCH$AR_TIME_RESTRICT.



Class Scheduling Database

SDA example

```
$ ANALYZE/SYSTEM

OpenVMS system analyzer

SDA>
SDA> show class      ! Displays information about active
                    ! scheduling class in the system or dump being
                     ! analyzed.

Scheduling Classes
------------------

                     Original  Current     Time    Process
   Class Name        Quantum   Quantum   Restrict   Count
----------------    --------  --------  --------  --------
MAINCLASS           0000001E  0000001E  0003F800  00000001
SDA>
SDA> show process
Process index: 002E   Name: KETAN             Extended PID: 0000042E
--------------------------------------------------------------------
Process status:          02040001  RES,PHDRES,INTER
        status2:         01608000

PCB address              88362EC0    JIB address               88363B40
PHD address              8E8BC000    Swapfile disk address     00000000
KTB vector address       883631E8    HWPCB address    FFFFFFFF.8E8BC080
Callback vector address  00000000    Termination mailbox           0000
Master internal PID      0001002E    Subprocess count                 0
```

```
Creator extended PID     00000000    Creator internal PID     00000000
Previous CPU Id          00000001    Current CPU Id           00000001
Previous ASNSEQ  0000000000000001    Previous ASN     0000000000FFDFA2
Initial process priority        4    # open files remaining
126/128
Delete pending count            0    Direct I/O count/limit
150/150
UIC                 [00200,000006]    Buffered I/O count/limit
150/150
Abs time of last event   001D1A08    BUFIO byte count/limit
127552/127552
# of threads                    1    ASTs remaining
297/300
Swapped copy of LEFC0    00000000    Timer entries remaining
100/100
Swapped copy of LEFC1    00000000    Active page table count         0
Global cluster 2 pointer 00000000    Process WS page count         478

    Press RETURN for more.
SDA>

Process index: 002E   Name: KETAN            Extended PID: 0000042E
--------------------------------------------------------------------
Global cluster 3 pointer 00000000    Global WS page count           57
PCB Specific Spinlock    88363800    Subprocesses in job             0

! scheduling class associated with the process
Scheduling class         "MAINCLASS"
Original Quantum         0000001E    Current Quantum          0000001E

    Press RETURN for more.
SDA>
```

```
Process index: 002E   Name: KETAN            Extended PID: 0000042E
--------------------------------------------------------------------

Thread index: 0000
------------------
Current capabilities:    System:     000C               QUORUM,RUN
                         User:       0000.00000000
Permanent capabilities:  System:     000C               QUORUM,RUN
                         User:       0000.00000000

Current affinities:                  00000000.00000000
Permanent affinities:                00000000.00000000

Thread status:                       02040001

! scheduling class associated with the process
       status2:                      01608000
CLASS_SCHED_PERM,CLASS_SCHEDULED,CLASS_SUPPLIED,WINDFALL

KTB address              88362EC0    HWPCB address   FFFFFFFF.8E8BC080
PKTA address             7FFEFF98    Callback vector address  00000000
Internal PID             0001002E    Callback error           00000000
Extended PID             0000042E    Current CPU id           00000001
State                     CUR 001    Flags                    00000000
    Press RETURN for more.
SDA>
```

Reference documentation

For more information, please refer to the following documents:

- OpenVMS System Manager's Manual, Volume 1: Essentials
- OpenVMS Programming Concepts Manual, Volume I
- OpenVMS System Services Reference Manual: GETUTC–Z