



VSI OpenVMS x86-64 E9.2-3

Release Notes

Publication Date: August 2024

VSI OpenVMS x86-64 E9.2-3 Release Notes



Copyright © 2024 VMS Software, Inc. (VSI), Boston, Massachusetts, USA

Legal Notice

Confidential computer software. Valid license from VSI required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for VSI products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. VSI shall not be liable for technical or editorial errors or omissions contained herein.

HPE, HPE Integrity, and HPE Alpha are trademarks or registered trademarks of Hewlett Packard Enterprise.

Intel and x86 are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Apple and macOS are registered trademarks of Apple Computer Inc.

VirtualBox is a registered trademark of Oracle Corporation.

VMware is a registered trademark or trademark of VMware, Inc.

PuTTY is copyrighted by Simon Tatham.

Apache and the Apache feather logo are trademarks of The Apache Software Foundation.

Motif is a registered trademark of The Open Group.

POSIX is a trademark of The IEEE.

Kerberos is a trademark of the Massachusetts Institute of Technology.

OpenSSL is a registered trademark owned by OpenSSL Software Foundation.

Table of Contents

1. Introduction	6
2. Read These Before You Start	6
2.1. Supported Disk Types	6
2.1.1. VSI OpenVMS Does Not Support Thin-Provisioned Volumes on Any Architecture	7
2.2. Tested Platforms	7
2.3. MD5 Checksum for the X86E923OE.ZIP File	8
2.4. CPU Compatibility Checks for Virtual Machines	8
2.5. Terminal Emulator Settings	9
2.6. Memory Disk and the Command Procedure SYSS\$MD.COM	10
2.7. x86-64 Licensing	10
2.8. Networking Options	10
2.8.1. VSI DECnet	11
2.8.2. Empty File for DECnet-Plus	11
2.8.3. Bridged Networking	11
2.8.4. VSI FTP Service Might Not Connect Correctly In Virtual Environments	11
2.8.5. Re-configuring Applications After Network Configuration Change	12
2.9. Layered Products Updates	13
2.9.1. Default Layered Products	13
2.9.2. OpenSSL Update	13
2.9.3. VSI OpenSSH V8.9-1H01 for OpenVMS	13
2.9.4. VSI Kerberos V3.3-3 for OpenVMS	14
2.9.5. VSI DECwindows Motif V1.8-1 for OpenVMS	14
2.9.6. VSI OSAK V3.1-2 for OpenVMS	14
2.9.7. ZIP/UNZIP Tools	14
3. New in This Release	14
3.1. New Features	14
3.1.1. E1000E Network Adapter Support	15
3.1.2. VMware vMotion Support	15
3.1.3. VMware VMDirectPath I/O Support for Data Disks	15
3.1.4. Guest Console	15
3.1.5. Supported VMware License Types	16
3.1.6. Images are Installed with /OPEN by Default	16
3.1.7. Updated Global Page Tables	16
3.1.8. PCSI Kits Can Be Set to Ignore Non-Default Installation Paths	16
3.1.9. VMS ACME Agent	17
3.1.10. MOUNT Command Changes	18
3.2. Features Not Available in VSI OpenVMS x86-64 E9.2-3	18
3.3. Issues Discovered in This Release	18
3.3.1. Error Message From TRACEBACK Utility	18
3.3.2. Clusters Over IP Configuration Template Has Incorrect Record Format	19
4. Known Issues and Limitations	19
4.1. Certain Files Might Update Incorrectly	19
4.2. Privileged Images Linked Using /SYSEXEC Should Be Relinked	19
4.3. CLUE\$STARTUP Incorrectly Processes Lists of Possible DOSD Devices	19
4.4. DECnet Over Wi-Fi	19
4.5. BACKUP/INITIALIZE to a Disk Mounted /FOREIGN Does Not Work	20
4.6. ENCRYPT Utility Does Not Work as Expected	20
4.7. Running x86-64 Images on Integrity Systems Causes an Access Violation	20
4.8. Connecting to a Shared LD Container in a Mixed-Architecture Cluster	20

4.9. Cluster Nodes Running VSI OpenVMS V9.2 May Cause All x86-64 Cluster Members to Crash	21
4.10. Booting With the DEVELOPER Boot Flag	21
5. Features Specific to VSI OpenVMS x86-64	21
5.1. Additional Prompt During OpenVMS x86-64 Installation	21
5.2. Extended File Cache (XFC)	22
5.3. HYPERSORT Utility Available	22
5.4. LIB\$INITIALIZE Handling in the Linker	22
5.5. Linker: Informational Messages	22
5.6. Handling of Threaded Applications in Linker	23
5.7. Different Image Layout on x86-64 and Itanium	23
5.8. Memory Disks	23
5.9. VSI OpenVMS x86-64 Will Not Support Swap Files	24
5.10. Process Dumps	24
5.11. SYSGEN Parameter Changes	24
5.12. System Crash Dumps	27
5.13. Traceback Support	30
5.14. Viewing Call Stack in Pthread Debugger	30
5.15. VSI DECram for OpenVMS	31
5.16. Symbolic Links and POSIX Pathname Support	31
5.16.1. Device Names in the POSIX Root	31
5.16.2. /SYMLINK Qualifier in DCL Commands	32
5.16.3. Symlink Support in COPY and CREATE	32
5.16.4. Symlink Support in RENAME	32
5.16.5. Symlink Support in BACKUP	32
5.16.6. Symlinks and File Versions	33
5.16.7. Symlinks Pointing to Multiple File Versions	33
5.17. Symbolic Debugger	33
5.17.1. Supported Registers	33
5.17.2. Older Versions of Compilers Always Set Language to C	34
5.17.3. Language Support Limitations	34
5.17.4. Source Line Correlation	34
5.17.5. Floating-Point Support	34
5.18. User-Written x86-Assembler Modules	35
5.19. CD Audio Functionality Not Supported on x86-64	35
5.20. STABACKIT.COM Deprecated	35
5.21. SMP Timeout Parameters Increased	36
5.22. Improvements to System Memory Allocation	36
5.23. Contiguous Best Try Qualifier for SET FILE/ATTRIBUTES	36
5.24. /EXTENTS Qualifier for ANALYZE/DISK_STRUCTURE	36
5.25. /OPTIONS qualifier for PRODUCT SHOW PRODUCT	37
5.26. CHECKSUM Utility Supports SHA1 and SHA256 Algorithms	38
5.27. VSI C Run-Time Library (C RTL) Update	38
5.28. SCS Jumbo Packets Not Enabled on Boot	38
5.29. Large Hardware Page Usage	38
5.30. Entropy	39
5.30.1. SYSGEN Parameter RANDOM_SOURCES	39
5.30.2. SYS\$GET_ENTROPY System Service	39
5.30.3. SHOW ENTROPY	40
5.30.4. Symmetric Multiprocessing (SMP)	40
6. Virtualization Notes	41
6.1. Changing Settings of a Running Virtual Machine May Cause a Hang	41

6.2. Time of Day May Not Be Maintained Correctly in Virtual Machine Environments	41
6.3. System Time on KVM Virtual Machines	42
6.4. VirtualBox and Hyper-V Compatibility on Windows 10 and 11 Hosts	42
6.5. VirtualBox: TCP Ports May Become Unusable After Guest Is Terminated	43
6.6. VMware Guest May Fail to Boot After Adding Second SATA Controller	43
6.7. Boot Manager Displays Incomplete List of Bootable Devices	43
6.8. Possible Issues with VMware Virtual Machines	44
6.9. One VirtIO-SCSI Adapter Supported on KVM	46
6.10. OpenVMS Clusters on Virtual Machines	46
7. Layered and Open-Source Products Notes	47

1. Introduction

VMS Software, Inc. (VSI) is pleased to introduce VSI OpenVMS x86-64 E9.2-3 Field Test.

This document is intended for all users of VSI OpenVMS x86-64 E9.2-3. Read this document before you install or use VSI OpenVMS x86-64 E9.2-3.

The following documents provide additional information in support of this release:

- [VSI OpenVMS x86-64 V9.2-3 Installation Guide \[https://docs.vmssoftware.com/vsi-openvms-x86-64-v923-installation-guide/\]](https://docs.vmssoftware.com/vsi-openvms-x86-64-v923-installation-guide/)
- [VSI OpenVMS x86-64 Boot Manager User Guide \[https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide-922/\]](https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide-922/)
- [VSI OpenVMS Calling Standard Manual \[https://docs.vmssoftware.com/vsi-openvms-calling-standard/\]](https://docs.vmssoftware.com/vsi-openvms-calling-standard/)
- [VSI OpenVMS Linker Utility Manual \[https://docs.vmssoftware.com/vsi-openvms-linker-utility-manual/\]](https://docs.vmssoftware.com/vsi-openvms-linker-utility-manual/)
- [VSI OpenVMS x86-64 Cross-Tools Kit Installation and Startup Guide \[https://docs.vmssoftware.com/vsi-x86-64-cross-tools-kit-installation-and-startup-guide-v922/\]](https://docs.vmssoftware.com/vsi-x86-64-cross-tools-kit-installation-and-startup-guide-v922/)

2. Read These Before You Start

Before you download the VSI OpenVMS x86-64 E9.2-3 installation kit, VSI strongly recommends that you read the notes in this section. These notes provide information about the hypervisors tested by VSI, CPU feature checks for virtual machines, terminal emulator settings, and licensing on OpenVMS x86-64 systems.

Note that if an entry describing a problem or a change in one of the previous release notes (especially for field test versions) is not present in the current release notes, then it is either no longer applicable or has been fixed.

2.1. Supported Disk Types

VSI OpenVMS x86-64 E9.2-3 does *not* support cluster common system disks.

VSI OpenVMS x86-64 supports SATA disks across all supported hypervisors.

On Oracle VirtualBox, VSI OpenVMS also supports VirtIO-SCSI disks.

On VMware ESXi, VSI OpenVMS also supports the configuration of up to four (4) LSI Logic Parallel SCSI controllers per virtual machine. Note that the LSI Logic SAS controllers are *not supported*.

Starting with the E9.2-3 release, VSI OpenVMS x86-64 supports the VMware VMDirectPath I/O feature with the HPE SN1100Q fibre channel HBAs for data disks. For more details, refer to Section 3.1.3, “VMware VMDirectPath I/O Support for Data Disks”

On KVM/QEMU, VSI OpenVMS also supports SCSI disks on a VirtIO controller. Note that only *one* VirtIO-SCSI adapter can be configured and used on a VSI OpenVMS x86-64 V9.2-x system. This functionality requires QEMU version 6.2 or later.

2.1.1. VSI OpenVMS Does Not Support Thin-Provisioned Volumes on Any Architecture

VSI OpenVMS *does not* support thin-provisioned volumes on any hypervisor. Attempting to use thin-provisioned virtual disks may result in loss or corruption of data on the device. VSI currently has no plans to add thin provisioning support on any hypervisor.

When creating new virtual volumes for VSI OpenVMS, *do not* enable thin provisioning, even if the hypervisor that you are using supports it.

For information on changing the provisioning of a virtual disk, refer to VMware documentation.

If you have any further questions, please contact VSI support via <support@vmssoftware.com>.

2.2. Tested Platforms

VSI OpenVMS x86-64 E9.2-3 can be installed as a guest operating system on KVM, VMware, and VirtualBox virtual machines using the **X86E923OE.ISO** file.

KVM

For KVM, VSI recommends ensuring that your system is up-to-date with KVM kernel modules and the associated packages necessary for your particular Linux distribution.

VSI has tested VSI OpenVMS x86-64 E9.2-3 with KVM on several Linux distributions. The following table includes the Linux distribution, version, and the QEMU version:

Linux Distribution	QEMU Version (package information)
Rocky Linux 9.4	QEMU of 8.2.0 (qemu-kvm-8.2.0-11.e19_4.4)
Ubuntu 22.04 LTS	6.2.0 (Debian 1:6.2+dfsg-2ubuntu6.15)
Oracle Linux 9.4	8.2.0 (qemu-kvm-8.2.0-11.e19_4.3)

VMware

VSI has tested VSI OpenVMS x86-64 E9.2-3 with the following VMware products:

VMware Product	Version Tested by VSI
ESXi	V8.0.x (with compatibility of V8, V7U2, V7U1); V7.0.x
vSphere Client	V8.0
Workstation Pro	V17.0.x
Fusion Pro	V13.5.x

VMware Licenses

Starting with the current release, all VMware license types are supported if you are connecting to your virtual machine using the Guest Console feature (see Section 3.1.4, “Guest Console”).

Note, however, that if you are using a virtual serial port connection, only the following VMware license types are supported for running VSI OpenVMS x86-64 E9.2-3:

VMware License	VSI Tested
Enterprise Plus	ESXi V6.7.x, V7.0.x, and V8.0.x

VMware License	VSI Tested
Enterprise	Not tested by VSI
Essentials Plus	Supported
Essentials	Supported
Standard	Supported
Hypervisor	Supported

The VMware licenses that are marked as "Supported" work with the Console Guest feature but not with the use of virtual serial lines in a virtual machine.

VirtualBox

VSI tests with VirtualBox V7.0.x and regularly installs patches when they are available.

2.3. MD5 Checksum for the X86E923OE.ZIP File

VSI recommends that you verify the MD5 checksum of the **X86E923OE.ZIP** file after it has been downloaded from the VMS Software Service Platform to your target system. The MD5 checksum of **X86E923OE.ZIP** must correspond to the following value:

```
3F330FA890074F3769FD3BC271237A08
```

To calculate the MD5 checksum, you can use any platform-specific utilities or tools that are available for your system.

2.4. CPU Compatibility Checks for Virtual Machines

VSI OpenVMS x86-64 requires that the CPU supports certain features that are not present in all x86-64 processors. When using virtual machines, both the host system and guest virtual machine must have the required features.

Host System Check

Before downloading the VSI OpenVMS x86-64 E9.2-3 installation kit, VSI recommends that you determine whether your host system has the required CPU features to run VSI OpenVMS x86-64. For this purpose, execute a Python script called **vmscheck.py** on your host system. This script, along with the accompanying PDF document entitled *VMS CPUID Feature Check*, can be downloaded from the official [VSI OpenVMS x86-64 web page \[https://vmssoftware.com/openkits/alpopen/opensource/vmscheck.zip\]](https://vmssoftware.com/openkits/alpopen/opensource/vmscheck.zip).

The *VMS CPUID Feature Check* document contains instructions on how to run the script and interpret the results and also describes script limitations.

Guest Virtual Machine Check

The OpenVMS Boot Manager performs the CPU feature check on the guest virtual machine. The CPU feature check is performed automatically every time the Boot Manager is launched. If the check has passed successfully, the following message is displayed:

```
Checking Required Processor Features:      PASSED
```

In addition, before booting VSI OpenVMS x86-64 E9.2-3, you can issue the following Boot Manager command to list the compatibility details:

 BOOTMGR> DEVICE CPU

Important

VSI OpenVMS x86-64 will not boot on the system that fails either of the following CPU feature checks:

- The host system check (via the **vmscheck.py** script)
 - The guest virtual machine check (via the OpenVMS Boot Manager).
-

Note

In case the system has the required CPU features but lacks some of the optional CPU features, the OpenVMS operating system may have noticeably lower performance.

2.5. Terminal Emulator Settings

When you are in the Boot Manager and before proceeding with the installation of VSI OpenVMS x86-64, you are required to access the system either via the new Guest Console feature (see Section 3.1.4, “Guest Console”) or through a serial port connection with a terminal emulator (such as PuTTY). Read the remainder of this section only if you are using a terminal emulator.

You may need to experiment in order to find the appropriate setting for your emulator. Refer to the *VSI OpenVMS x86-64 Boot Manager User Guide* [<https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide-922>] and *VSI OpenVMS x86-64 E9.2-3 Installation Guide* [<https://docs.vmssoftware.com/vsi-openvms-x86-64-e923-installation-guide/>] for more details on the emulator settings.

On Windows, VSI recommends using PuTTY. Some PuTTY users have found success with the following settings:

- If the connection type is **Raw**, the following settings should be used:

Category	Setting	Value
Session	Connection type	Raw
Terminal	Implicit CR in every LF	Unchecked
Terminal	Implicit LF in every CR	Unchecked
Terminal	Local echo	Force off
Terminal	Local line editing	Force off (character mode)

- If the Connection type is **Telnet**, the following settings should be used:

Category	Setting	Value
Session	Connection type	Telnet
Connection > Telnet	Telnet negotiation mode	Switch from Active to Passive . This yields a connection to a PuTTY window.
Connection > Telnet	Telnet negotiation mode	Uncheck Return key sends new line instead of ^M

Note

As there is no Telnet server on the virtual machine host for the console communication, it is not literally a Telnet connection, but it can be used because not all emulators support a Raw connection.

2.6. Memory Disk and the Command Procedure SYS\$MD.COM

VSI OpenVMS x86-64 uses a boot method called Memory Disk that simplifies the boot process by eliminating boot complexity and decoupling the VSI OpenVMS Boot Manager (chain loader) from a specific device or version of VSI OpenVMS x86-64. Memory Disk is created and deployed automatically during the installation process.

The Memory Disk contains all files that are required to boot the minimum OpenVMS kernel and all files needed to write system crash dumps. Changes such as file modifications or PCSI kit/patch installations require the operating system to execute a procedure to update the Memory Disk container, thus assuring that the next boot will use the new images. A command procedure, SYS\$MD.COM, keeps the Memory Disk up-to-date.

Note

Do not invoke SYS\$MD.COM directly unless you are advised to do so by VSI Support, or when required while following documented procedures. For example, if you load a user-written EXECLET by running SYS\$UPDATE:VMS\$SYSTEM_IMAGES.COM, you must then invoke SYS\$UPDATE:SYS\$MD.COM. For more details, see Section 5.8, “Memory Disks” of this document.

Note

VSI does not recommend changing or manipulating SYS\$MD.DSK or SYS\$EFI.SYS (or the underlying EFI partition) in any way. These files are needed to boot the system and are maintained by OpenVMS.

2.7. x86-64 Licensing

During the installation, you will be prompted to register Product Authorization Keys (PAKs) for the base operating environment and any layered products that are not included in the base OS.

For more information about licensing, refer to *VSI OpenVMS x86-64 E9.2-3 Installation Guide* [<https://docs.vmssoftware.com/vsi-openvms-x86-64-v923-installation-guide/#d0e305>].

2.8. Networking Options

VSI OpenVMS x86-64 V9.2-x provides support for the following network stacks:

- VSI TCP/IP Services
- VSI DECnet Phase IV
- VSI DECnet-Plus

VSI TCP/IP Services V6.0-25 is part of the OpenVMS x86-64 E9.2-3 installation and will be installed along with the operating system. For more information, refer to *VSI TCP/IP Services Version V6.0*

Release Notes and other VSI TCP/IP Services documentation available at [docs.vmssoftware.com \[https://docs.vmssoftware.com/\]](https://docs.vmssoftware.com/docs.vmssoftware.com/).

During the OpenVMS x86-64 E9.2-3 installation, you will also be offered the option to install either VSI DECnet Phase IV or VSI DECnet-Plus. Note that if you plan to use DECnet, you *must choose* between VSI DECnet Phase IV and VSI DECnet-Plus. *Only one* of these products can be installed on your system at a time.

2.8.1. VSI DECnet

Install either VSI DECnet Phase IV or VSI DECnet-Plus on VSI OpenVMS x86-64 E9.2-3 and then configure the product you have chosen, just as you would for an OpenVMS Alpha or OpenVMS Integrity release.

If you have DECnet Phase IV installed on your system and you want to use DECnet-Plus, you have to uninstall DECnet Phase IV and then install and configure DECnet-Plus.

Note

If your DECnet installation was *not* part of the main installation procedure for OpenVMS x86-64, you *must* update the Memory Disk after you install DECnet. The Memory Disk update ensures that SYS\$NETWORK_SERVICES.EXE is loaded on boot. Use the following commands:

```
$ @SYS$UPDATE:SYS$MD.COM
```

After the next system reboot, you may want to purge the Memory Disk.

```
$ PURGE SYS$LOADABLE_IMAGES:SYS$MD.DSK
```

If you install DECnet as part of the main OpenVMS x86-64 installation procedure, you do *not* need to update the Memory Disk. The Memory Disk is updated at the end of the OpenVMS x86-64 installation.

After DECnet has been installed and configured, you can set host and copy files to/from other systems running DECnet.

2.8.2. Empty File for DECnet-Plus

The OpenVMS x86-64 installation procedure provides an empty file NET\$CONFIG.DAT before installing the DECnet-Plus kit.

2.8.3. Bridged Networking

To understand how to configure your virtual machine network (devices and network configuration), please consult the VirtualBox, KVM, and VMware documentation. Some configurations may be incompatible with the operation of OpenVMS applications. For example, configuring a bridged adapter with a MacVTap device may inhibit the MAC address change done by DECnet, and configuring DECnet on VirtualBox may require allowing promiscuous mode on the NIC.

2.8.4. VSI FTP Service Might Not Connect Correctly In Virtual Environments

If the FTP service does *not* work after it has been started, switch to passive mode with the following command:

```
FTP> SET PASSIVE ON
Passive is ON
```

In passive mode, the FTP client always initiates a data connection. This is useful in virtual machine environments when there is network address translation (NAT) in your network.

To run this command automatically when you invoke FTP, put it into SYSS\$LOGIN:FTPINIT.INI. For the full description of the SET PASSIVE command, refer to the *VSI TCP/IP Services for OpenVMS User's Guide* [https://docs.vmssoftware.com/vsi-tcpip-services-for-openvms-user-s-guide-60/#FTP_COMMANDS_SEC].

2.8.5. Re-configuring Applications After Network Configuration Change

On "system" startup (in this context, the word "system" refers to the bare metal hardware/firmware or to the VMS Boot Manager/hypervisor), the "system" scans the I/O buses and presents a list of devices to the OpenVMS boot code, which then loads drivers and calls initialization routines for the devices to be configured for VMS operation. System parameters and other configuration mechanisms (such as I/O auto-configuration) affect what devices are configured in what order.

LAN devices get configured as they are found. Device names, such as EIA, EIB, EIC, and so on, are assigned by the LAN driver according to SYSS\$CONFIG.DAT. On Alpha, Itanium, and x86-64 bare metal systems, the device names are encoded with the bus type. For example, the names of XMI Ethernet devices start with EX, names of Ethernet PCI devices start with EW, names of FDDI PCI devices start with FW. On x86-64 virtual machines, there are only EI device names. Pseudo-devices such as LAN Failover (LL) and VLAN (VL) devices are also a consideration.

When you add a new hardware device to an Alpha, Itanium, or x86-64 bare metal system, the device names may change the next time the system is booted. For example, if you have EIA, EIB, and EIC devices and replace EIB with a dual-port card, you will see EIA, EIB, EIC, and EID. In addition, the physical connection of each device may change. When the device names change, you must reconfigure the applications on the system that use those devices. For example, TCP/IP may be using EIC (ie2) which needs to be changed to EID (ie3), or a LAN Failover Set may be defined with EIB and EIC but needs to be changed to EIB and EID.

On x86-64 VMs, a similar reconfiguration effort is needed when the LAN device configuration changes. It might be necessary more often since the system network configuration is much easier to change for VMs compared to bare metal systems.

On x86-64 VMware VMs, you can change the network adapter type to e1000, e1000e, or vmxnet3. Changing the adapter type from vmxnet3 to e1000 or e1000e requires deleting the existing network adapter and then adding a new one of the desired type. Any changes may require a reconfiguration effort. Note that the order of the devices and the MAC addresses may change.

On x86-64 KVM VMs, you can change the network adapter type to e1000, e1000e, or VirtIO. Upon reboot, the adapter type will change but the device names and MAC addresses will stay the same. In such cases, no change to the application configuration is required. Other cases, such as adding and removing NICs, may require reconfiguration.

Reconfiguration can be done as follows:

1. Record the VM NIC configuration before and after adding, removing, or changing NICs.
2. Make the changes and reboot the system.

3. Do **MC LANCP SHOW CONFIG** (or **SHOW CONFIG/USER**) comparing the adapter types and MAC addresses against the VM NIC configuration.
4. Adjust any applications, such as TCP/IP, as well as LAN Failover and VLAN definitions to apply to the correct network adapters.

You can adjust the network configuration above using graphical tools such as VMM (KVM), Cockpit (Linux), and GUI (VMware). You can also use command line utilities or edit the XML VM definition files. You can manually assign addresses using these tools, which might help you correlate the devices seen by these tools with the devices that VMS "sees" after boot.

2.9. Layered Products Updates

2.9.1. Default Layered Products

The following layered products will be installed unconditionally with VSI OpenVMS x86-64 E9.2-3:

- VSI TCP/IP Services
- VSI Kerberos
- VSI SSL111
- VSI SSL3
- VSI OpenSSH
- VMSPORTS x86VMS PERL534 T5.34-0

2.9.2. OpenSSL Update

VSI SSL3 V3.0-13 is the default SSL offering on VSI OpenVMS E9.2-3.

VSI SSL111 V1.1-1W is also available in this release in order to allow any existing SSL-based customer applications to continue to run. VSI SSL3 is designed to co-exist in parallel with VSI SSL111 by means of using different symbols for different versions.

Warning

SSL111 *will not* receive updates in future releases, so VSI strongly recommends that applications still using VSI SSL111 be moved to VSI SSL3.

2.9.3. VSI OpenSSH V8.9-1H01 for OpenVMS

VSI OpenSSH V8.9-1H01 is a required layered product that will be installed unconditionally with VSI OpenVMS x86-64 E9.2-3.

Warning

If you are upgrading to VSI OpenVMS E9.2-3 from an earlier version with OpenSSH V8.9-1G or earlier installed, make sure to uninstall VSI OpenSSH from your system *before* you start the upgrade procedure.

For post-installation and configuration instructions, refer to the [VSI OpenVMS x86-64 E9.2-3 Installation Guide](https://docs.vmssoftware.com/vsi-openvms-x86-64-v922-installation-guide/#networkingChap) [https://docs.vmssoftware.com/vsi-openvms-x86-64-v922-installation-guide/#networkingChap].

For a detailed description of the features and bug fixes included in this release of OpenSSH V8.9, please refer to <https://www.openssh.com/txt/release-8.9>.

2.9.4. VSI Kerberos V3.3-3 for OpenVMS

VSI OpenVMS x86-64 E9.2-3 includes VSI Kerberos V3.3-3 for OpenVMS.

2.9.5. VSI DECwindows Motif V1.8-1 for OpenVMS

VSI OpenVMS x86-64 E9.2-3 includes VSI DECwindows Motif V1.8-1.

Note that VSI DECwindows Motif will *not* be installed unconditionally with VSI OpenVMS x86-64 E9.2-3 and must be installed manually, if necessary.

2.9.6. VSI OSAK V3.1-2 for OpenVMS

VSI OpenVMS x86-64 E9.2-3 includes VSI DECnet-Plus OSI Applications Kernel (OSAK) V3.1-2.

Note that VSI DECnet-Plus OSAK will *not* be installed unconditionally with VSI OpenVMS x86-64 E9.2-3 and must be installed manually, if necessary.

2.9.7. ZIP/UNZIP Tools

VSI provides the Freeware executables for managing ZIP archives on OpenVMS x86-64 systems. In VSI OpenVMS x86-64 E9.2-3, the installation procedure automatically puts these files in the following directories on the system disk:

Folder	Files
SYS\$COMMON:[SYSHLP.UNSUPPORTED.UNZIP]	UNZIP.EXE UNZIPSFX.EXE UNZIPSFX_CLI.EXE UNZIP_CLI.EXE UNZIP_MSG.EXE
SYS\$COMMON:[SYSHLP.UNSUPPORTED.ZIP]	ZIP.EXE ZIPCLOAK.EXE ZIPNOTE.EXE ZIPSPLIT.EXE ZIP_CLI.EXE ZIP_MSG.EXE

3. New in This Release

This section lists the features and bug fixes introduced in the current release, as well as applicable known issues and limitations.

3.1. New Features

This section lists features introduced in the current release.

3.1.1. E1000E Network Adapter Support

Starting with version E9.2-3, VSI OpenVMS x86-64 supports the E1000E network adapter on VMware ESXi 8 and KVM.

3.1.2. VMware vMotion Support

Starting with version E9.2-3, VSI OpenVMS x86-64 supports the VMware vMotion technology, with the following limitations:

- Your VMware ESXi cluster must be configured for vMotion. For details, refer to the official VMware documentation.
- vMotion is *not* compatible with the VMware VMDirectPath I/O (fibre channel passthrough) feature (see Section 3.1.3, “VMware VMDirectPath I/O Support for Data Disks”).
- VSI Guest Console *must* be used to interact with virtual machines.

For information on how to migrate a live VSI OpenVMS Virtual Machine using VMware vMotion, refer to <https://docs.vmssoftware.com/vsi-performing-vmotion/> and [this video \[https://youtu.be/T4P0ASqrIpc\]](https://youtu.be/T4P0ASqrIpc) on the official VSI YouTube channel.

3.1.3. VMware VMDirectPath I/O Support for Data Disks

Starting with version E9.2-3, OpenVMS x86-64 supports the VMware VMDirectPath I/O feature that can significantly improve the performance of the VMware ESXi systems utilizing high-speed I/O devices (such as Fibre Channel) as data disks. VMDirectPath I/O allows a guest operating system on a virtual machine to directly access the physical PCI and PCIe devices connected to the host system, bypassing the virtualization layer.

Warning

Booting from fibre channel disks is not supported. Cluster common system disks are not supported.

Each VMware virtual machine can be connected to up to two PCI devices. For more information, refer to the *VSI OpenVMS x86-64 E9.2-3 Installation Guide* [<https://docs.vmssoftware.com/vsi-openvms-x86-64-v923-installation-guide/#d0e1002>].

3.1.4. Guest Console

In previous versions of OpenVMS x86-64, all console interactions required a legacy serial port device. Starting with version E9.2-3, OpenVMS x86-64 features the Guest Console – a console interface that does *not* depend on old serial hardware. The Guest Console provides the necessary keyboard driver and terminal emulator features to allow users to interact with the system from the initial boot through the operator login.

Note that the current implementation of the Guest Console provides a minimal terminal that lacks certain important features, such as scrolling and copy-pasting. This will be addressed in a future release of OpenVMS x86-64.

To enable the Guest Console, enter the following Boot Manager command:

```
BOOTMGR> OPA0
```

Warning

The Guest Console requires the OpenVMS Boot Manager version that is shipped with the E9.2-3 release.

Note

The new Guest Console feature has not been fully tested with MAC keyboards; therefore, users of VMware Fusion are encouraged to try this feature, while being aware that use of a traditional serial port console connection may be the best solution for this Field Test release.

3.1.4.1. Bad Canvas Error Message

If you are running the Guest Console, you might see the following error message:

```
%GCTERM-E-MEMALOC Bad Canvas
```

If that happens, you must increase the size of the non-paged S2 pool by entering the following command in the following order:

```
$ MC SYSGEN
SYSGEN> SET NPAGEDYN_S2 8
SYSGEN> SET NPAGEXPVIR_S2 10
SYSGEN> WRITE CURRENT
```

Then, either reboot your system, or enter **WRITE ACTIVE**.

3.1.5. Supported VMware License Types

With the release of the Guest Console feature (see Section 3.1.4, “Guest Console”), OpenVMS x86-64 supports all VMware license types.

3.1.6. Images are Installed with /OPEN by Default

Starting with OpenVMS E9.2-3, images are installed as open known images by default. This applies to the **ADD** and other related commands. A user no longer needs to add the **/OPEN** qualifier to get an image installed as an open known image. The existing behavior of **/NOOPEN** is maintained. If a user specifies **/NOOPEN**, the image is not installed as an open known image.

3.1.7. Updated Global Page Tables

Starting with version E9.2-3, the minimum and default values of the **GBLPAGES SYSGEN** parameter in VSI OpenVMS x86-64 have been increased to 8,388,608 pagelets (524,288 pages). This provides a minimum of 4 megabytes of global page table and allows OpenVMS to take advantage of larger physical page sizes available on the x86-64 platform.

3.1.8. PCSI Kits Can Be Set to Ignore Non-Default Installation Paths

The POLYCENTER Software Installation (PCSI) utility now supports the **IGNORE DESTINATION_PATH** statement. Use this statement if you want your PCSI kit to always install to the default location (**SYSSYSDEVICE:[VMS\$COMMON]** and lower-level subdirectories), ignoring any user-defined installation paths.

For more information, refer to the *POLYCENTER Software Installation Utility Manual* [<https://docs.vmssoftware.com/vsi-openvms-polycenter-software-installation-utility-manual/>].

3.1.9. VMS ACME Agent

Starting with the E9.2-3 release, an in-development version of VMS ACME Agent is available on x86-64. The current implementation provides the basic login functionality from TELNET, DECNET, and SSH. Configurations with other agents, such as LDAP, are not supported.

VSI recommends that any testing be done using LOGIN_UAF.

A fully functional production-ready version of VMS ACME Agent will be provided with the upcoming release of VSI OpenVMS x86-64 V9.2-3.

Read This Before Using VMS ACME Agent

In the current implementation of VMS ACME Agent, any username must have an associated account name set in the UAF record before logging in. This limitation will be removed in the production release of the VMS ACME Agent.

In order to enable access to the system via SSH using the VMS ACME Agent, the SSH\$SSH account for the SSH Server must have an account name set in its UAF record.

```
$ set def sys$system
$ mcr authorize
UAF> show ssh$ssh/br
  Owner      Username    UIC          Account    Privs Pri Directory
  SSH$SSH    SSH$SSH    [3656,1]          All      4  SSH$ROOT:[VAR]

UAF> mod ssh$ssh/account="SSH$SSH"
UAF> show ssh$ssh/br
  Owner      Username    UIC          Account    Privs Pri Directory
  SSH$SSH    SSH$SSH    [3656,1]  SSH$SSH    All      4  SSH$ROOT:[VAR]
UAF>^z
$
```

Known Issues in VMS ACME Agent

The following issues have been discovered at the time of the current release:

- The restart functionality may cause a process dump before the server re-establishes itself.
- Each login request will trigger a diagnostic entry written to the ACME\$SERVER.LOG file indicating a fatal condition and the following message to the operator console:

```
%%%%%%%%%% OPCOM 10-JUL-2024 08:26:32.08 %%%%%%%%%%
Message from user SYSTEM on FYFE
VMS ACME failure - check the log file ACME$SERVER

%ACME-I-LOGAGENT, agent initiated log event on 8-JUL-2024 17:18:07.23
-ACME-I-THREAD, thread: id = 4, type = EXECUTION
-ACME-I-REQUEST, request information, id = 1, function = AUTHENTICATE_PRINCIPAL
-ACME-I-CLIENT, client information, PID = 3DC0090C
-ACME-I-AGENT, agent information, ACME id = 1, name = VMS
-ACME-I-CALLOUT, active callout routine = acme$co_principal_name
-ACME-I-CALLBACK, active callback routine = acme$cb_send_logfile
-LOGIN-S-PROCSTEP, Processing Step was PRINCIPAL_NAME_PROCESSING_PROMPT
-LOGIN-F-TEXT, PROCESSING_PROMPT: cb_queue_dialogue status 122322945
```

Such messages may safely be ignored.

- The Welcome notice from VMS ACME Agent might display the wrong hardware platform information.

3.1.10. MOUNT Command Changes

Previously, a volume could either be formatted as a Files-11 volume (with the OpenVMS file structure and files) or as a CDROM with the ISO9660 format. The **MOUNT** command would correctly identify the volume type and mount it accordingly. Starting with the E9.2-3 release, a volume can be dual-formatted with both Files-11 and ISO9660 formats. If no special qualifiers are added to the **MOUNT** command and a dual-formatted volume is detected, it will be mounted as CDROM by default.

The existing **MOUNT** qualifier **/MEDIA** now has two keywords, **CDROM** and **FILES11**, which allows the user to override the default. For example, entering the command **MOUNT /MEDIA=FILES11 DKA300** will mount the DKA300 volume as a Files-11 volume. Note that the **/MEDIA** qualifier is only useful when mounting dual-formatted volumes.

To mount different disk volumes systemwide, they must have unique volume labels. This restriction has not changed, but for the ISO9660 volumes it also applies to the *Volume Set Identifier*. If you want to mount multiple different ISO9660 volumes systemwide, they must have both unique volume labels and unique volume set identifiers, since they are part of the volume lock name, which has to be unique in a system or cluster environment.

3.2. Features Not Available in VSI OpenVMS x86-64 E9.2-3

The following functionalities, products, and commands are *not* available in VSI OpenVMS x86-64 E9.2-3:

- Availability Manager (only the data collector is currently available; the software that uses the data will be available in a future release)
- Process swapping (see Section 5.9, “VSI OpenVMS x86-64 Will Not Support Swap Files” of this document)
- RAD support
- Support for privileged applications, such as:
 - User-written device drivers
 - Code that directly calls internal system routines (such as those that manage page tables)
- TECO Editor

3.3. Issues Discovered in This Release

3.3.1. Error Message From TRACEBACK Utility

Currently, the TRACEBACK facility might output the following error message:

```
Error: last state machine operation was not end_sequence
```

This message does not impact the traceback or the printed values and may safely be ignored. This issue will be corrected in the upcoming VSI OpenVMS V9.2-3 release.

3.3.2. Clusters Over IP Configuration Template Has Incorrect Record Format

The configuration template for Clusters Over IP that is supplied with the operating system has an incorrect record format of VFC and record attributes of "Print file carriage control". This may result in errors at boot. This issue may be corrected with the following commands:

```
$ set file/attribute=(rfm:var,rat:cr) sys$system_md:tcpip$cluster.dat
$ set file/attribute=(rfm:var,rat:cr) sys$system:tcpip$cluster.template
```

4. Known Issues and Limitations

This section lists the issues and limitations that were introduced or discovered in earlier versions of VSI OpenVMS x86-64 and are still applicable to the current release.

For information about bugs fixed in previous releases of VSI OpenVMS x86-64, refer to the corresponding release notes at <https://docs.vmssoftware.com> [<https://docs.vmssoftware.com/>].

4.1. Certain Files Might Update Incorrectly

Before upgrading to VSI OpenVMS E9.2-3, check for multiple versions of SY\$HELP:HELPLIB.HLB and SY\$LIBRARY:STARLET.OLB on your disk. If more than one version of either file exists, there is a chance that the latest version will be lost during the upgrade. To avoid this, purge the files so that only one version remains.

This issue will be fixed in a future release of VSI OpenVMS.

4.2. Privileged Images Linked Using /SYSEXE Should Be Relinked

VSI recommends that any privileged images linked using the **LINK/SYSEXE** command in VSI OpenVMS V9.2 or earlier be relinked for the current release because data structures and interfaces are subject to change between releases.

Note that the images linked using **LINK/SYSEXE** in VSI OpenVMS V9.2-1 do not need to be relinked for the current release.

4.3. CLUE\$STARTUP Incorrectly Processes Lists of Possible DOSD Devices

Currently, CLUE\$STARTUP can only process the first Dump Off System Disk (DOSD) device in a list of DOSD devices.

This issue will be fixed in a future release.

4.4. DECnet Over Wi-Fi

DECnet Phase IV will not work over a Wi-Fi connection. To be able to use DECnet Phase IV on an x86-64 virtual machine and communicate with other systems, you must have the host machine connected via an Ethernet cable.

DECnet Plus will work over a Wi-Fi connection, but it must be tunnelled through TCP/IP.

4.5. BACKUP/INITIALIZE to a Disk Mounted /FOREIGN Does Not Work

A BACKUP command of the form:

```
$ BACKUP/INITIALIZE input-disk:[directories...]*.*; output-disk:save-set.bck/SAVE
```

where the output volume is a disk that has been mounted /FOREIGN, does *not* work in VSI OpenVMS x86-64 E9.2-3 and may yield the following error:

```
%BACKUP-F-OPENOUT, error opening DKA200:[000000]DKA200$504.BCK; as output
-SYSTEM-W-BADIRECTORY, bad directory file format
```

This type of operation was originally developed to allow the backup of a large non-removable disk onto a series of smaller removable disks. The feature, referred to as "sequential disk" operation, is described in the *VSI OpenVMS System Manager's Manual, Volume 1: Essentials* [https://docs.vmssoftware.com/vsi-openvms-system-manager-s-manual-volume-1-essentials/#SAVESETS_SEQDISK_BCK].

As a workaround, initialize and mount an output volume of a size sufficient to hold the entire backup save set before issuing the BACKUP command as in the following example:

```
$ INITIALIZE output-disk: label
$ MOUNT output-disk: label
$ CREATE/DIRECTORY output_disk:[]
$ BACKUP input-disk:[directories...]*.*; output-disk:save-set.bck/SAVE
```

If a sufficiently large output disk is not available, you can instead create and mount a volume set using multiple disks with the INITIALIZE and MOUNT/BIND commands.

4.6. ENCRYPT Utility Does Not Work as Expected

Most operations with the ENCRYPT utility return the following error:

```
%ENCRYPT-F-ILLALGSEL, algorithm selection unknown, unavailable, or unsupported
```

This issue will be addressed in a future release of VSI OpenVMS x86-64.

4.7. Running x86-64 Images on Integrity Systems Causes an Access Violation

When you run a VSI OpenVMS x86-64 image on VSI OpenVMS for Integrity Servers, no message from the image activator appears, but an access violation occurs.

This issue will be corrected in a future patch kit for VSI OpenVMS for Integrity Servers.

4.8. Connecting to a Shared LD Container in a Mixed-Architecture Cluster

In OpenVMS x86-64 E9.2-3, a container file can only be connected as a shared LD device accessible to multiple architectures as follows:

1. Connect to the file on all OpenVMS Alpha and/or OpenVMS Integrity systems using a command such as:

```
$ LD CONNECT/allocclass=1/share DISK$LDTEST:[LD]SHARED_LD.DSK LDA5:
```

2. Once all required connections on OpenVMS for Alpha and/or OpenVMS for Integrity systems are complete, you may then connect to the file on any OpenVMS x86-64 systems.

If you connect to the file on an OpenVMS x86-64 system first, any subsequent attempts to connect on an OpenVMS for Alpha and/or OpenVMS for Integrity system will fail with an error message such as:

```
%LD-F-FILEINUSE, File incompatible connected to other LD disk in cluster  
-LD-F-CYLINDERS, Cylinders mismatch
```

VSI intends to provide an update for OpenVMS for Alpha and OpenVMS for Integrity systems. That update will allow the connections to the shared container file to be performed in any order.

4.9. Cluster Nodes Running VSI OpenVMS V9.2 May Cause All x86-64 Cluster Members to Crash

In a mixed-version cluster where at least one node is running VSI OpenVMS x86-64 V9.2, certain conditions might trigger a crash of all x86-64 systems, regardless of which OpenVMS version they run. This can happen when a V9.2 MSCP node that is serving disks to other cluster members gets rebooted. In this case, other x86-64 cluster members will see the disks served by MSCP go into mount verification. Upon the verification completion, the MSCP client node might crash.

The fix for this problem is to upgrade all x86-64 cluster nodes to VSI OpenVMS x86-64 V9.2-1 or later.

4.10. Booting With the DEVELOPER Boot Flag

The DEVELOPER boot flag 0x08000000 is reserved for use by VSI, and its function is subject to change. Booting with this flag set can result in an unexpected system behavior.

5. Features Specific to VSI OpenVMS x86-64

This section lists features that are specific to VSI OpenVMS x86-64 and as such, are not available in versions of OpenVMS for other architectures.

5.1. Additional Prompt During OpenVMS x86-64 Installation

During VSI OpenVMS x86-64 installation, if you choose to install DECnet Phase IV for OpenVMS x86-64 or TCP/IP Services for OpenVMS x86-64, you will see an output similar to the following:

```
* Product VSI X86VMS TCPIP Vx.x-x requires a system reboot.  
Can the system be REBOOTED after the installation completes? [YES]
```

Note

The product named in the message may be either DECnet Phase IV or TCP/IP Services.

If this happens, you must answer YES (the default response), otherwise the installation will be terminated.

Later in the installation, you will see the following messages:

```
%PCSI-I-SYSTEM_REBOOT, executing reboot procedure ...
```

```
Shutdown/reboot deferred when this product is installed as part of the O/S
installation/upgrade
```

If you are installing both optional products, you will see these messages twice.

These messages may safely be ignored. VSI will address this issue in a future release.

5.2. Extended File Cache (XFC)

VSI OpenVMS x86-64 has extended file caching (XFC) enabled by default.

5.3. HYPERSORT Utility Available

The high-performance Sort/Merge utility (HYPERSORT) is available in VSI OpenVMS x86-64. Enable the utility with the following command:

```
$ DEFINE SORTSHR SYS$LIBRARY:HYPERSORT.EXE
```

5.4. LIB\$INITIALIZE Handling in the Linker

Programs that use the LIB\$INITIALIZE startup mechanism must declare a LIB\$INITIALIZE PSECT and include the LIB\$INITIALIZE module from STARLET.OLB when linking. Traditionally, besides the PSECT, source programs simply declared an external reference to that module, and the linker resolved the reference from STARLET.OLB. However, the LLVM backend used by the cross-compilers removes that external reference from the object file since there were no additional source references to the routine.

The linker was changed to automatically include the required module if it encounters a LIB\$INITIALIZE PSECT. For details, see the *VSI OpenVMS Linker Utility Manual* [<https://docs.vmssoftware.com/vsi-openvms-linker-utility-manual/>].

This change does not affect any source module where external references to the LIB\$INITIALIZE module were declared. This change also does not affect any existing link commands that explicitly include the LIB\$INITIALIZE module from STARLET.OLB.

5.5. Linker: Informational Messages

When the linker encounters writable code sections, with PSECT attributes set to WRT and EXE, it prints the following informational message:

```
%ILINK-I-MULPSC, conflicting attributes for section <PSECT name>
    conflicting attribute(s): EXE,WRT
    module: <module name>
    file: <obj-or-olb-filename>
```

When the linker finds unwind data in a module, but no section with the PSECT attribute set to EXE, it prints the following informational message:

```
%ILINK-I-BADUNWSTRCT, one or more unwind related sections are
missing or corrupted
```

```
section: .eh_frame, there is no non-empty EXE section
module: <module name>
file: <obj-or-olb-filename>
```

These messages are seen mainly with MACRO-32 and BLISS source modules. All code sections must be non-writable. You must have code in sections with the PSECT attribute set to EXE.

5.6. Handling of Threaded Applications in Linker

When linking applications that use the POSIX Threads Library (PTHREAD\$RTL), you can set up the application in such a way that it can receive upcalls from VMS and/or that VMS should map user threads to multiple kernel threads. This behavior can be enabled with the /THREADS_ENABLE qualifier. However, if that qualifier is not specified, the linker automatically enables upcalls and displays an informational message to make the user aware of that. The user can overwrite the default behavior by explicitly specifying /NOTTHREADS_ENABLE.

5.7. Different Image Layout on x86-64 and Itanium

When porting an application from Itanium to x86-64, be aware that the image layout may change in an incompatible way – although the compile and link commands/options did not change. This is an architectural difference.

On Itanium, the compiler may generate **short data** which is accessed in an efficient way. The Itanium linker always collects short data to the DEFAULT CLUSTER, no matter where the object module that defines this short data is collected. That is, in a partially protected shareable image, an object module may be collected into a protected linker cluster, but its short data may be collected into an unprotected cluster, and so it is not protected. User-mode code in the shareable image can write to it.

On x86-64, there is no short data. All data defined in an object module will go where the module goes (except the defining PSECT which is moved with an explicit COLLECT option). That is, on x86-64, for partially protected shareable images, all data defined by an object module which is collected into a protected linker cluster will be protected. User-mode code in the shareable image cannot write to it.

5.8. Memory Disks

If you change anything that affects the boot path or dumping, you must run the command procedure SYS\$MD.COM before rebooting. For instance, if you change any files referenced or loaded during booting (up to and including the activation of STARTUP), or any files used by the dump kernel, then you must run SYS\$MD.COM.

However, in VSI OpenVMS x86-64 V9.2-x there are two exceptions to the above statement. If you make any of the following changes that affect the boot path or dumping, you do not need to run SYS\$MD.COM:

1. Use `SYSGEN WRITE CURRENT` or `SYSMAN PARAM WRITE CURRENT`. These commands will access the parameter file on the memory disk directly.
2. Copy a file directly to the memory disk when specifically advised by VSI support specialists to do so.

Use the following command exactly as specified here:

```
$ @SYS$UPDATE:SYS$MD
```

No parameters are needed, the defaults should apply.

When SYS\$MD.COM completes, you must reboot.

When SYS\$MD.COM is invoked, the system will display something like the following:

```
$ @SYS$UPDATE:SYS$MD
    Created memory disk DKE100:[VMS$COMMON.SYS$LDR]SYS$MD.DSK;1
    - using 184064 blocks in 1 extent with 1010 spare blocks
    - mounted on LDM9323: with volume label MD230120DD6A
    - contains OpenVMS V9.2-1

$
```

After the next system reboot, purge the memory disk with the following command:

```
$ PURGE SYS$LOADABLE_IMAGES:SYS$MD.DSK
```

5.9. VSI OpenVMS x86-64 Will Not Support Swap Files

VSI OpenVMS x86-64 does not support swap files. The system's physical memory should be managed with appropriately sized system memory and page file(s).

The AUTOGEN and SWAPFILES procedures no longer create swap files on the system disk. If a swap file resides on the system disk, it will no longer be installed as part of the system startup.

In the current release, the SYSGEN INSTALL command does not support the /SWAPFILE qualifier. The use of the qualifier will result in a syntax error.

Processes may be seen in the Computable Outswapped (COMO) state. This is a transient state for newly created processes. Processes will never appear in the Local Event Flag Wait Outswapped (LEFO) or Hibernate Outswapped (HIBO) states. All performance counters associated with swapping are still present in the system. Various MONITOR displays will show swapping metrics.

5.10. Process Dumps

VSI OpenVMS x86-64 provides support for process dumps. The only method currently available for analyzing process dumps is using the System Dump Analyzer (SDA). Most SDA commands that display data about a process can be used to examine the state of the process. For example, SHOW PROCESS, SHOW CRASH, SHOW EXCEPTION, SHOW CALL, EXAMINE, MAP. Support for the Symbolic Debugger interface will be added in a future release of VSI OpenVMS x86-64.

5.11. SYSGEN Parameter Changes

The following changes and additions have been made to the SYSGEN Utility for VSI OpenVMS x86-64. For more information about SYSGEN qualifiers and parameters, see *VSI OpenVMS System Management Utilities Reference Manual, Volume II: M-Z* [<https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-ii-m-z/#d0e27590>].

Table 1. SYSGEN Commands Used for VSI OpenVMS x86-64

Command	Parameter	Description
USE	CURRENT	Specifies that source information is to be retrieved from the current system parameter file on disk. On OpenVMS x86-64 systems, the system parameter file is SYS\$SYSTEM:X86_64VMSSYS.PAR.

Command	Parameter	Description
WRITE	CURRENT	<p>Specifies that source information is to be written to the current system parameter file on disk. The new values will take effect the next time the system is booted.</p> <p>On OpenVMS x86-64 systems, command modifies the current system parameter on disk, SYS\$SYSTEM:X86_64VMSSYS.PAR.</p>

Table 2. System Parameters

Parameter	Description								
BOOT_BITMAP1	On x86-64 systems, this parameter defines the required size in megabytes of the first boot-time allocation bitmap used by SYSBOOT during the bootstrap process on x86-64. If this value is too small, the system may be unable to boot.								
BOOT_BITMAP2	On x86-64 systems, this parameter defines the required size in megabytes of the second boot-time allocation bitmap used by SYSBOOT during the bootstrap process on x86-64. If this value is too small, the system may be unable to boot.								
DISABLE_X86_FT	<p>On x86-64 systems, DISABLE_X86_FT is a bit mask used to inhibit the use of certain x86 processor features by the operating system.</p> <p>The following bits are defined:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>If 1, do not use the XSAVEOPT instruction.</td> </tr> <tr> <td>1</td> <td>If 1, do not use the RDFSBASE, WRFSBASE, RDGSBASE, or WRGSBASE instructions.</td> </tr> <tr> <td>2</td> <td>If 1, do not provide software mitigation against the Intel MDS vulnerabilities.</td> </tr> </tbody> </table> <p>DISABLE_X86_FT is a STATIC parameter.</p>	Bit	Definition	0	If 1, do not use the XSAVEOPT instruction.	1	If 1, do not use the RDFSBASE, WRFSBASE, RDGSBASE, or WRGSBASE instructions.	2	If 1, do not provide software mitigation against the Intel MDS vulnerabilities.
Bit	Definition								
0	If 1, do not use the XSAVEOPT instruction.								
1	If 1, do not use the RDFSBASE, WRFSBASE, RDGSBASE, or WRGSBASE instructions.								
2	If 1, do not provide software mitigation against the Intel MDS vulnerabilities.								
GH_EXEC_CODE_S2	<p>On x86-64 systems, GH_EXEC_CODE_S2 specifies the size in pages of the execllet code granularity hint region in S2 space.</p> <p>GH_EXEC_CODE_S2 has the AUTOGEN and FEEDBACK attributes.</p>								
GH_EXEC_DATA_S2	<p>On x86-64 systems, GH_EXEC_DATA_S2 specifies the size in pages of the execllet data granularity hint region in S2 space.</p> <p>GH_EXEC_DATA_S2 has the AUTOGEN and FEEDBACK parameters.</p>								
GH_RES_DATA_S2	<p>On x86-64 systems, GH_RES_DATA_S2 specifies the size in pages of the resident image data granularity hint region in S2 space.</p> <p>GH_RES_DATA_S2 has the AUTOGEN and FEEDBACK attributes.</p>								

Parameter	Description								
GH_RES_CODE	<p>This parameter now applies to x86-64 systems. On x86-64, Integrity, and Alpha systems, GH_RES_CODE specifies the size in pages of the resident image code granularity hint region in S0 space.</p> <p>GH_RES_CODE has the AUTOGEN and FEEDBACK attributes.</p>								
GH_RO_EXEC_S0	<p>On x86-64 systems, GH_RO_EXEC_S0 specifies the size in pages of the read-only execlet data granularity hint region in S0 space.</p> <p>GH_RO_EXEC_S0 has the AUTOGEN and FEEDBACK attributes.</p>								
GH_RO_RES_S0	<p>On x86-64 systems, GH_RO_RES_S0 specifies the size in pages of the read-only resident image data granularity hint region in S0 space.</p> <p>GH_RO_EXEC_S0 has the AUTOGEN and FEEDBACK attributes.</p>								
LOAD_SYS_IMAGES	<p>This special parameter is used by VSI and is subject to change. Do not change this parameter unless VSI recommends that you do so.</p> <p>LOAD_SYS_IMAGES controls the loading of system images described in the system image data file, VMS\$SYSTEM_IMAGES. This parameter is a bit mask.</p> <p>The following bits are defined:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0 (SGN\$V_LOAD_SYS_IMAGES)</td> <td>Enables loading alternate execlets specified in VMS \$SYSTEM_IMAGES.DATA.</td> </tr> <tr> <td>1 (SGN\$V_EXEC_SLICING)</td> <td>Enables executive slicing. Note that executive slicing is always enabled on x86-64 systems.</td> </tr> <tr> <td>2 (SGN\$V_RELEASE_PFNS)</td> <td>Enables releasing unused portions of granularity hint regions on Alpha servers.</td> </tr> </tbody> </table> <p>These bits are on by default. Using conversational bootstrap exec slicing can be disabled.</p> <p>LOAD_SYS_IMAGES is an AUTOGEN parameter.</p>	Bit	Description	0 (SGN\$V_LOAD_SYS_IMAGES)	Enables loading alternate execlets specified in VMS \$SYSTEM_IMAGES.DATA.	1 (SGN\$V_EXEC_SLICING)	Enables executive slicing. Note that executive slicing is always enabled on x86-64 systems.	2 (SGN\$V_RELEASE_PFNS)	Enables releasing unused portions of granularity hint regions on Alpha servers.
Bit	Description								
0 (SGN\$V_LOAD_SYS_IMAGES)	Enables loading alternate execlets specified in VMS \$SYSTEM_IMAGES.DATA.								
1 (SGN\$V_EXEC_SLICING)	Enables executive slicing. Note that executive slicing is always enabled on x86-64 systems.								
2 (SGN\$V_RELEASE_PFNS)	Enables releasing unused portions of granularity hint regions on Alpha servers.								
RAD_SUPPORT	<p>RAD_SUPPORT enables RAD-aware code to be executed on systems that support Resource Affinity Domains (RADs).</p> <p>On x86-64 systems, the default, minimum, and maximum values for RAD_SUPPORT are all zeros because RAD support is not currently available on that platform.</p>								
SCSBUFFCNT	<p>SCSBUFFCNT is reserved for VSI use only.</p> <p>On x86-64, Alpha, and Integrity servers, the system communication services (SCS) buffers are allocated as needed, and SCSBUFFCNT is not used.</p>								

Parameter	Description	
VCC_FLAGS	The static system parameter VCC_FLAGS enables and disables file system data caching. If caching is enabled, VCC_FLAGS controls which file system data cache is loaded during system startup.	
	Value	Description
	0	Disables file system data caching on the local node and throughout the OpenVMS Cluster. In an OpenVMS Cluster, if caching is disabled on any node, none of the other nodes can use the extended file cache or the virtual I/O cache. They cannot cache any file data until that node either leaves the cluster or reboots with VCC_FLAGS set to a non-zero value.
	1	Enables file system data caching and selects the Virtual I/O Cache. This value is relevant only for Alpha systems.
	2	Enables file system data caching and selects the extended file cache.
<p>Note</p> <p>On x86-64 and Integrity servers, the volume caching product [SYS\$LDR]SYS\$VCC.EXE is not available. XFC caching is the default caching mechanism. Setting the VCC_FLAGS parameter to 1 is equivalent to not loading caching at all or to setting VCC_FLAGS to 0.</p> <hr/> <p>VCC_FLAGS is an AUTOGEN parameter.</p>		

All system parameters are exposed on every platform: x86-64, Integrity, and Alpha. In addition, flags can be set or cleared on any platform using the SYSGEN Utility. However, the flag may not have any effect on a platform for which it is not intended.

5.12. System Crash Dumps

VSI OpenVMS x86-64 system crash dumps are written using a minimal VMS environment called the Dump Kernel. All files used by the Dump Kernel are included in the Memory Disk described in Section 5.8, “Memory Disks”.

Interleaved Dumps

Starting with the V9.2-1 release, VSI OpenVMS x86-64 supports two system crash dump types: Compressed Selective Dump and Interleaved Dump.

A Compressed Selective Dump is written using only the primary CPU running in the Dump Kernel, while an Interleaved Dump makes use of the available secondary CPUs. If the dump device is a Solid State Disk (SSD), the dump can be written much faster, thus allowing the system to be rebooted sooner.

The DUMPSTYLE parameter specifies which dump type is being used. The default value for the parameter is 9 (Compressed Selective Dump). However, if the system has more than one CPU and the SYS\$SYSTEM:MODPARAMS.DAT file does not include a value for the DUMPSTYLE parameter, then, when AUTOGEN runs, it will set the value of DUMPSTYLE to 128 (Interleaved Dump).

In OpenVMS x86-64, the only other pertinent bits in the DUMPSTYLE parameter are bit 1 (full console output: registers, stack, and system image layout) and bit 5 (only dump system memory, key processes, and key global pages). Either or both of these bits can be set in addition to the two base values (9 and 128). Bit 2 (dump off system disk) is no longer required.

Dump Off System Disk

Crash dumps can be written to the system disk or to an alternate disk which is specifically designated for this purpose.

Dumps to the system disk are written to SYS\$SYSDEVICE:[SYSn.SYSEXE]SYSDUMP.DMP which can be created or extended using the SYSGEN utility.

Dumps to an alternate device can be set up as described in the example below:

1. Create a dump file on the desired device using the SYSGEN utility. In this example, we will use the DKA100: disk.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE DKA100:[SYS0.SYSEXE]SYSDUMP.DMP /SIZE=200000
SYSGEN> EXIT
```

2. Enter the following command:

```
$ SET DUMP_OPTIONS/DEVICE=DKA100:
```

You have set DKA100: as the dump device.

You can view the setting by using the SHOW DUMP_OPTIONS command. The change is effective immediately, a reboot *is not* required.

Changes in SET DUMP_OPTIONS

A new qualifier, /MAXIMUM=CPUS=n, has been added to the **SET DUMP_OPTIONS** command. This qualifier sets the number of CPUs used by the Dump Kernel when writing an Interleaved Dump. It has no effect when a Compressed Selective Dump is used. By default, the Dump Kernel will use all eligible secondary CPUs that are available in the system, up to a maximum of 10 (including the primary CPU). Eligible CPUs are those that were halted successfully when the system crashed and that did *not* trigger the crash with a MACHINECHK or KRNLSTAKNV Bugcheck.

The maximum number of CPUs that can be specified in the command is also 10.

System Dump Analysis

VSI strongly recommends that the version of SDA.EXE and SDA\$SHARE.EXE used to analyze a system dump be exactly the same as the version of OpenVMS that was in use when the system crash occurred. However, it is often possible to use SDA images from a different version of OpenVMS,

provided there are no major differences between the versions and the warning messages by SDA are ignored (either %SDA-W-LINKTIMEMISM, or %SDA-W-SDALINKMISM, or both).

New SDA command qualifiers in OpenVMS x86-64

The SDA commands below now have new qualifiers that are specific to OpenVMS x86-64. Also, the format of several output messages has been changed to accommodate the architectural differences.

SHOW DUMP command

The qualifier **/PERFORMANCE** has been added to display the performance data collected by the Dump Kernel while writing a dump.

Depending on the additional keywords used, the data displayed by the **SHOW DUMP/PERFORMANCE** command can include system information (**/PERFORMANCE=SYSTEM**), per-CPU information (**/PERFORMANCE=CPU**), and per-process information (**/PERFORMANCE=PROCESS**). If no additional keywords are specified with the command, all information mentioned above will be displayed.

SHOW EXECUTIVE command

The qualifier **/SDA_EXTENSION** has been added to limit the list of displayed executive images to those that are paired with SDA extensions. For example, SWIS\$DEBUG.EXE.

SHOW PAGE_TABLE and SHOW PROCESS /PAGE_TABLE commands

The qualifiers **/BPTE**, **/PDE**, **/PDPTE**, **/PML4E**, and **/PML5E** have been added to allow you to display a specific level of page table entries. The default behavior is to display the base page table entry (**BPTE**).

The qualifier **/MODE** has been added to display the page tables for a specific mode. The default behavior is to display the page tables for all modes. Valid keywords are: **KERNEL**, **EXECUTIVE**, **SUPERVISOR**, and **USER**.

SHOW POOL command

The output for the qualifier **/RING_BUFFER** is now affected by the additional qualifiers **/ALL** and **/S2_NPP**. The default behavior for **SHOW POOL /RING_BUFFER** is to display all entries in the Nonpaged Pool ring buffer. If **/S2_NPP** is specified, then the ring buffer for S2 pool is displayed. If **/ALL** is specified, then the contents of both ring buffers are displayed, interleaved by their timestamps.

The qualifier **/S2_NPP**, when used without **/RING_BUFFER**, displays only the S2 pool. The default behavior is to display all pool types.

The qualifier **/USB** displays only the nonpaged pool that is reserved for use by USB devices. The default behavior is to display all pool types.

VALIDATE POOL command

The qualifier **/S2_NPP** allows validation of only the S2 pool. The default behavior is to validate all pool types.

The qualifier **/USB** allows validation of only the nonpaged pool that is reserved for use by USB devices. The default behavior is to validate all pool types.

SDA commands and qualifiers not available in VSI OpenVMS x86-64

The following commands and qualifiers are not applicable to VSI OpenVMS x86-64 systems:

- SHOW GALAXY
- SHOW GCT
- SHOW GLOCK
- SHOW GMDB
- SHOW SHM_CPP
- SHOW SHM_REG
- SHOW VHPT
- VALIDATE SHM_CPP
- EVALUATE and EXAMINE, qualifiers /FPSR, /IFS, /ISR, /PFS, and /PSR
- SHOW PAGE_TABLE and SHOW PROCESS /PAGE_TABLE, qualifiers /L1, /L2, /L3, and /SPTW
- SHOW POOL, qualifier /BAP
- VALIDATE POOL, qualifier /BAP

Other SDA command changes

The COPY command qualifiers /COMPRESS, /DECOMPRESS, and /PARTIAL cannot be used with an Interleaved Dump.

5.13. Traceback Support

The linker includes sufficient traceback information in the image file for a functional symbolic traceback. As a result, by default, the image file may be larger than in previous versions/updates. This additional debug information is not read by the image activator, so it will not slow down image activation. However, to make image files smaller, the linker was changed to include reduced traceback information. This affects the traceback output, as it no longer prints the routine name. Any other traceback output is unaffected. This feature can be enabled with the LINE_NUMBER keyword for the /TRACE qualifier. For details, see the *VSI OpenVMS Linker Manual* [https://docs.vmssoftware.com/vsi-openvms-linker-utility-manual/#LINK_COMM_REF_PART].

Traceback now prints the image name, routine name, and line numbers much like traceback on OpenVMS Alpha and OpenVMS Integrity server systems, with the following differences:

1. Traceback is unable to determine the module name, so instead it prints the "basename" of the source file used to create the module.
2. The position of the values in their respective columns may not line up with the header line.

These differences will be addressed in a future release of VSI OpenVMS x86-64.

5.14. Viewing Call Stack in Pthread Debugger

Starting with VSI OpenVMS x86-64, the call stack can be viewed in the Pthread debugger. To show the call stack, use the `-o C` option in the `threads` command. For example, if the debugger runs in the PTHREAD SDA extension, the command to show the call stack for thread id 3 is the following:

```
SDA> pthread threads -o "C" 3
Process name: SECURITY_SERVER   Extended PID: 0000008F   Thread data:
"threads -o "C" 3"
```

```
-----
thread 3 (blocked, timed-cond) "Process_Proxy_Task", created by pthread
Stack trace:
  0xfffff83000c83b1a5 (pc 0xfffff83000c83b1a5, sp 0x00000000024e3228)
  0xfffff83000c87b53a (pc 0xfffff83000c87b53a, sp 0x00000000024e3230)
  0xfffff83000c858493 (pc 0xfffff83000c858493, sp 0x00000000024e32e0)
  0xfffff83000c852b04 (pc 0xfffff83000c852b04, sp 0x00000000024e33f0)
  0xfffff83000c8499a3 (pc 0xfffff83000c8499a3, sp 0x00000000024e34d0)
  0xfffff83000c844e9a (pc 0xfffff83000c844e9a, sp 0x00000000024e3800)
  0x0000000080004ad8 (pc 0x0000000080004ad8, sp 0x00000000024e3900)
  0x0000000080007fe6 (pc 0x0000000080007fe6, sp 0x00000000024e3950)
  0xfffff83000c8887df (pc 0xfffff83000c8887df, sp 0x00000000024e3bd0)
  0xfffff83000c83b0ea (pc 0xfffff83000c83b0ea, sp 0x00000000024e3f00)
```

However, the output of the `threads -o u` Pthread debugger command that shows an **SDA SHOW CALL** command cannot yet be used in SDA.

```
SDA> pthread threads -o u 3
Process name: SECURITY_SERVER   Extended PID: 0000008F   Thread data:
"threads -o u 3"
```

```
-----
thread 3 (blocked, timed-cond) "Process_Proxy_Task", created by pthread
Unwind seed for SDA SHOW CALL 00000000024e2e40
SDA> SHOW CALL 00000000024e2e40
00000000.024E2E40 is no valid handle for a call stack start of
00000000.00000000
```

5.15. VSI DECram for OpenVMS

VSI DECram for OpenVMS, also referred to as a RAMdisk, is fully operational in VSI OpenVMS x86-64.

For details of the DECram disk characteristics and configuration, refer to the *DECram for OpenVMS User's Manual* [<https://docs.vmssoftware.com/vsi-decram-for-openvms-users-manual/>].

5.16. Symbolic Links and POSIX Pathname Support

Symbolic links (symlinks) and POSIX pathnames are documented in the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://docs.vmssoftware.com/vsi-c-run-time-library-reference-manual-for-openvms-systems/#SYMLINK_CHAP]. The release notes in this section augment that documentation.

5.16.1. Device Names in the POSIX Root

The POSIX file namespace begins at a single root directory. The location of the POSIX root in OpenVMS is defined by the system manager using the `SET ROOT` command. Files may be located via a path starting in the root using an absolute pathname that starts with the `/` character. For example, `/bin` identifies the **bin** directory located in the POSIX root directory. Additionally, all disk devices on an OpenVMS system may be located by using their device name as a name in the POSIX root. For example, the path `/DKA0/USER` identifies the directory `DKA0:[USER]`. The name after the `/` character may be an actual device name or a logical name that resolves to a device name (and possibly one or more directory names). Device names are not actually present in the POSIX root directory. In

resolving an absolute pathname, OpenVMS first searches for the name in the POSIX root. If it is not found, it tries to locate the name as a device or logical name.

5.16.2. /SYMLINK Qualifier in DCL Commands

A number of DCL commands that operate on files accept the /SYMLINK qualifier to control whether the command operates on a file that a symlink points to or on the symlink itself, and whether symlinks are followed in wildcard searches. For more information, refer to *VSI DCL Dictionary: A–M* [<https://docs.vmssoftware.com/vsi-openvms-dcl-dictionary-a-m/>] and *VSI DCL Dictionary: N–Z* [<https://docs.vmssoftware.com/vsi-openvms-dcl-dictionary-n-z/>].

Most commands require /SYMLINK to be used with a keyword. The valid keywords are as follows:

Keyword	Explanation
NOWILDCARD	Indicates that symlinks are disabled during directory wildcard searches.
WILDCARD	Indicates that symlinks are enabled during directory wildcard searches.
NOELLIPSIS	Indicates that symlinks are matched for all wildcard fields except for ellipsis.
ELLIPSIS	Equivalent to WILDCARD (included for command symmetry).
NOTARGET	Indicates that the command operates on the named file whether it is an ordinary file or a symlink.
TARGET	Indicates that if the named file is a symlink, the symlink is followed to operate on the symlink target.

Some commands, such as COPY, DIRECTORY, DUMP, or SET FILE, accept /SYMLINK with no keyword value. In such cases, the qualifier causes the command to operate on the symlink; by default, the command operates on the file pointed to by the symlink.

5.16.3. Symlink Support in COPY and CREATE

If the input file of a COPY command is a symlink and the /SYMLINK qualifier is specified, the command copies the symlink; otherwise, it copies the target of the symlink. If the output named in a COPY or CREATE command is an existing symlink, COPY creates a file as identified by the target name of the symlink. Thus, it is possible to create a symlink that points to a non-existent file and then create the file by specifying the symlink as the output of the command.

5.16.4. Symlink Support in RENAME

The RENAME command always operates on the file specified in the command. That is, if the input file is a symlink, the symlink is renamed. If a symlink corresponding to the output name exists, it will not be followed.

5.16.5. Symlink Support in BACKUP

The BACKUP command never follows symlinks and has no /SYMLINK qualifier. If the input file is a symlink, the symlink is saved or copied. When a save set is restored, BACKUP does not follow symlinks named as output files – instead, the specified name is created. Also, BACKUP does not follow symlinks

in its directory wildcarding operation. Any symlinks encountered during directory searches are saved or copied as symlinks.

5.16.6. Symlinks and File Versions

A symlink, while implemented as a type of file, may not have multiple versions. Depending on the usage, commands that create files (such as COPY, BACKUP, and RENAME) may attempt to create a new version of a symlink or create a symlink where one or more versions of a file exist. Operations of this kind fail with the following error:

```
NOSYMLINKVERS, cannot create multiple versions of a symlink
```

5.16.7. Symlinks Pointing to Multiple File Versions

Even though a symlink is limited to a single file version, it may point to a file that has multiple versions. When a DCL command that searches for multiple files (such as **DIRECTORY** or **SET FILE**) locates a file via a symlink, it returns the name of the symlink. As a result, even though multiple versions of the symlink target may exist, the DCL command operates only on the latest version of the target file.

For example, in the following sequence of commands:

```
$ CREATE/SYMLINK=FILE.TXT LINK.TXT
$ COPY FILE2.TXT LINK.TXT
$ COPY FILE2.TXT LINK.TXT
$ COPY FILE2.TXT LINK.TXT
```

three versions of the file FILE.TXT will exist, but a **DIRECTORY** command will show only the latest version in response to the name LINK.TXT. Likewise, the command `$ TYPE LINK.TXT ; *` will display only the latest version of FILE.TXT, not all three versions.

5.17. Symbolic Debugger

VSI OpenVMS x86-64 includes an initial implementation of the Symbolic Debugger. While many features work, there are some that do not work. These are listed in the following sections. VSI will address these issues in a future release. In addition, some of the missing features will require future native compilers, as the cross compilers are unable to provide sufficient debug information to the debugger.

5.17.1. Supported Registers

All integer registers (see the table below) are currently supported, including the 32, 16, and 8-bit variants.

Table 3. Supported registers

64-bit registers	%rax, %rbx, %rcx, %rdx, %rsi, %rdi, %rsp, %rbp, %r8, %r9, %r10, %r11, %r12, %r13, %r14, %r15
32-bit registers	%eax, %ebx, %ecx, %edx, %esi, %edi, %esp, %ebp, %r8d, %r9d, %r10d, %r11d, %r12d, %r13d, %r14d, %r15d
16-bit registers	%ax, %bx, %cx, %dx, %si, %di, %sp, %bp, %r8w, %r9w, %r10w, %r11w, %r12w, %r13w, %r14w, %r15w

8-bit registers

```
%al, %ah, %bl, %bh, %cl, %ch, %dl, %dh,
%sil, %dil, %spl, %bpl, %r8b, %r9b,
%r10b, %r11b, %r12b, %r13b, %r14b,
%r15b
```

5.17.2. Older Versions of Compilers Always Set Language to C

The issue with the x86-64 compilers setting the debug language to C by default has been resolved in the current release of VSI OpenVMS x86-64.

However, older versions of cross-compilers still set the debug language to C by default. This means that when the debugger regains control, the language is set to C, even if the module being debugged was written in another language.

The way to work around this problem is simply to use the `SET LANGUAGE` command to set the default language to that which is being debugged.

5.17.3. Language Support Limitations

There is some support for language-specific features when using the `EXAMINE` and `DEPOSIT` commands. However, some of the more complex data structures may not work correctly and can have unexpected and undefined behaviour.

As mentioned previously, the cross-compilers all set the language type to C in the debug output. This may appear to prevent language-specific features from working. Using the `SET LANGUAGE` command will resolve this.

5.17.4. Source Line Correlation

In this release, the debugger fully supports source line correlation. Note, however, that the current version of the debugger requires you to use the latest versions of cross-compilers and/or native compilers.

Previous versions of the debugger *do not* support source line correlation because the previous versions of compilers generate listing line numbers in their debug data, not source lines. In most instances, this will lead to quite a large disparity between the line numbers available to the debugger and the actual line numbers of the source file. This can manifest itself in messages similar to the following:

```
break at DANCE\pause_actions\%LINE 138517
%DEBUG-I-NOTORIGSRC, original version of source file not found
      file used is SYS$SYSDEVICE:[DEBUG.BLUEY]DANCE.C;1
%DEBUG-W-UNAREASRC, unable to read source file SYS$SYSDEVICE:
[DEBUG.BLUEY]DANCE.C;1
-RMS-E-EOF, end of file detected
```

To work around this issue, VSI recommends you use the debugger with accompanying source listings to locate relevant lines reported by break points and the `EXAMINE` command.

The lack of source line correlation support also means that the `STEP/LINE` command does not always work correctly and can behave like the `STEP/INSTRUCTION` command.

5.17.5. Floating-Point Support

There is currently no support for floating-point registers. Although it is possible to examine and deposit them, the contents are inaccurate and will not be updated.

5.18. User-Written x86-Assembler Modules

User-written x86-assembler code must follow the VSI OpenVMS calling standard (see the *VSI OpenVMS Calling Standard manual* [https://docs.vmssoftware.com/vsi-openvms-calling-standard/#X86_64_CONVENTIONS_CHI]) to provide exception handling information that is used by the exception handling mechanism to find a handler in one of the callers of the assembler module. Without that information, exception handling can only call the last chance handler, which means the application will likely not be able to handle the exception.

See the following example of a user-written x86-assembler module:

```
.vms_module "MYTEST", "X-01"
.text
.globl MYTESTRTN
.align 16
.type MYTESTRTN,@function

MYTESTRTN:
.cfi_startproc
pushq %rbp // Establish a frame pointer
.cfi_def_cfa_offset 16 // Record distance to saved PC
.cfi_offset %rbp, -16
movq %rsp,%rbp
.cfi_def_cfa_register %rbp

// function body

movq %rbp,%rsp // Restore stack and return
popq %rbp
.cfi_def_cfa %rsp, 8 // Adjust distance to saved PC
retq
.size MYTESTRTN, .-MYTESTRTN
.cfi_endproc
```

5.19. CD Audio Functionality Not Supported on x86-64

CD audio functionality is not supported on VSI OpenVMS running on x86-64. CD audio functionality has been deprecated for all x86 platforms and beyond.

This does *not* apply to VSI OpenVMS running on Alpha or Integrity platforms.

5.20. STABACKIT.COM Deprecated

SY\$UPDATE:STABACKIT.COM has been deprecated in VSI OpenVMS x86-64. The STABACKIT functionality originally supported Standalone Backup for VAX systems and has long been replaced by SY\$SYSTEM:AXPVMS\$PCSI_INSTALL_MIN.COM for OpenVMS Alpha and SY\$SYSTEM:I64VMS\$PCSI_INSTALL_MIN.COM for OpenVMS Integrity systems. The referenced command procedures produce a minimal installation of OpenVMS on a target device which may be booted to allow a full backup of the system disk. STABACKIT would offer to run the appropriate procedure for you.

VSI OpenVMS x86-64 E9.2-3 currently does not support minimal installation of the operating system on a non-system disk. Backup options for virtual machines include the ability to perform backup from the host operating system and the ability to checkpoint or clone within the hypervisor. One may also

boot from the installation media and choose the DCL subprocess menu entry to perform a backup of the system device.

5.21. SMP Timeout Parameters Increased

Starting with VSI OpenVMS V9.2-1, the default values for the system parameters `SMP_SPINWAIT` and `SMP_LNGSPINWAIT` have been increased. This change was made to avoid potential `CPUSPINWAIT` and `CLUEXIT` crashes occurring on virtual machines.

5.22. Improvements to System Memory Allocation

When allocating large amounts of system memory such as when allocating a large DECram disk, the operation could take a long time. These operations could result in a `CLUEXIT` or `CPUSPINWAIT` crash. Starting with VSI OpenVMS V9.2-1, improvements have been made to the system to improve the efficiency of the system memory allocation.

5.23. Contiguous Best Try Qualifier for SET FILE/ATTRIBUTES

The `SET FILE/ATTRIBUTES` command supports the Contiguous Best Try (CBT) keyword.

With CBT enabled, the file system will allocate additional blocks to a file contiguously on a best effort basis. The file system will disable the attribute if the best effort algorithm cannot complete the file extension with at most three additional extents.

To enable CBT, use the following command:

```
$ SET FILE/ATTRIBUTES=CBT
```

To disable CBT, use the following command:

```
$ SET FILE/ATTRIBUTES=NOCBT
```

5.24. /EXTENTS Qualifier for ANALYZE/DISK_STRUCTURE

The `ANALYZE/DISK_STRUCTURE` command now supports the `/EXTENTS` qualifier.

`ANALYZE/DISK_STRUCTURE/EXTENTS` will produce a report on the fragmentation of free space on a volume. By default, the only output is the number of extents of free space and the total number of free blocks. For additional details, specify one of the following additional qualifiers with the `ANALYZE/DISK_STRUCTURE/EXTENTS` command:

Qualifier	Syntax	Description
<code>/LARGEST</code>	<code>/LARGEST[=<i>n</i>]</code>	<p>Displays a list of block counts for the <i>n</i> largest extents of free space on the volume in descending size order. The default for <i>n</i> is 10. The qualifier is ignored if you specify zero or a negative number.</p> <p>The list is also saved in the DCL symbol <code>ANALYZE\$LARGEST_EXTENTS</code> (as a comma-separated list of decimal values). The symbol is set</p>

Qualifier	Syntax	Description
		to an empty string if there is no free space on the disk. There is no upper limit on <i>n</i> , but if the DCL symbol exceeds 1024 characters, the number of extents in the symbol will be reduced to ensure the symbol is no more than 1024 characters.
/LOCK_VOLUME	/LOCK_VOLUME /NOLOCK_VOLUME	Locks the volume against allocation while the data is being collected. By default, the volume is locked.
/OUTPUT	/OUTPUT[= <i>filespec</i>] /NOOUTPUT	Specifies the output file for the fragmentation report produced by the ANALYZE/DISK_STRUCTURE utility. If you omit the qualifier or the entire filename, SYSS\$OUTPUT will be used. The default filename is ANALYZE\$EXTENTS.LIS.
/REQUIRED	/REQUIRED= <i>n</i>	Displays the number of extents required to satisfy an allocation request of <i>n</i> blocks (starting with the largest extent). There is no default for <i>n</i> . If you specify zero or a negative number, the qualifier is ignored. The result is also saved in the DCL symbol ANALYZE\$REQUIRED_EXTENTS. The symbol is set to an empty string if there is insufficient space on the disk to satisfy the allocation request.

Consider the following example:

```
$ ANALYZE/DISK_STRUCTURE/EXTENTS/LARGEST/REQUIRED=20000 LDM9063:
```

```
Extent report for _X86VMS$LDM9063:
=====
```

The disk has 6 extents of free space for a total of 25262 free blocks.

The extent sizes are:

```
17176
5591
2469
15
9
2
```

2 extents are required for an allocation of 20000 blocks.

5.25. /OPTIONS qualifier for PRODUCT SHOW PRODUCT

The **PRODUCT SHOW PRODUCT** command supports the **/OPTIONS=keyword** qualifier. A keyword is required. Currently, the only defined value is **EXTENDED_DISPLAY**. If you specify

/OPTIONS=EXTENDED_DISPLAY, the system will output an additional line of information for each of the listed products, giving you the **DESTINATION** that was specified during the product installation. If no alternative destination was used during the product installation, the system disk will be shown as the destination. See the following example:

```
$ PRODUCT SHOW PRODUCT/OPTIONS=EXTENDED_DISPLAY *VMS
-----
PRODUCT                                KIT TYPE      STATE
-----
VSI X86VMS OPENVMS V9.2-2              Platform     Installed
    Destination: DISK$SYSTEM_DISK:[VMS$COMMON.]
VSI X86VMS VMS V9.2-2                  Oper System  Installed
    Destination: DISK$SYSTEM_DISK:[VMS$COMMON.]
-----
2 items found
```

The **/OPTIONS** qualifier has been available since V8.4-1H1. It can be combined with other qualifiers, for example, **/FULL**.

5.26. CHECKSUM Utility Supports SHA1 and SHA256 Algorithms

In VSI OpenVMS x86-64, the **CHECKSUM** utility supports the SHA1 and SHA256 secure hash algorithms to calculate file checksums. These algorithms calculate a checksum for all bytes within a file and ignore possible record structures.

Use the **CHECKSUM** command qualifier **/ALGORITHM=option** to specify the algorithm for the file checksum calculation.

For information about all supported checksum algorithms, refer to the *VSI OpenVMS DCL Dictionary: A-M* [<https://docs.vmssoftware.com/vsi-openvms-dcl-dictionary-a-m/#CHECKSUM>].

5.27. VSI C Run-Time Library (C RTL) Update

VSI OpenVMS x86-64 V9.2-x includes an updated VSI C Run-Time Library (C RTL). The update provides bug fixes, as well as new functions, including the additional C99 Standard functions, new constants, new and updated header files. For a complete list of changes and additions, refer to the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [<https://docs.vmssoftware.com/vsi-c-run-time-library-reference-manual-for-openvms-systems/>] [<https://docs.vmssoftware.com/vsi-c-run-time-library-reference-manual-for-openvms-systems/#C99ECO3>] and the release notes provided with your VSI C RTL ECO kit.

5.28. SCS Jumbo Packets Not Enabled on Boot

VSI OpenVMS x86-64 V9.2-x does not enable Jumbo Packets automatically during boot. You can enable Jumbo Packets manually. Note that after you do so, you must restart the LAN adapter communications.

5.29. Large Hardware Page Usage

VSI OpenVMS x86-64 V9.2-x takes advantage of 2 MB hardware pages in limited areas of the operating system. Usage of the larger page size allows for faster memory access. Larger pages are used by parts of the executive code and data along with some internal memory management data.

5.30. Entropy

VSI OpenVMS x86-64 V9.2-x collects information from stochastic system events and hardware-provided entropy sources, when available, to create a pool of random data which may be used as seeds for random number and cryptographic algorithms. The features associated with entropy collection include the `RANDOM_SOURCES` `SYSGEN` parameter and the `SY$GET_ENTROPY` system service.

Note

On hardware that supports the Intel RDRAND instruction, entropy collection will include random data from the RDRAND instruction as part of the entropy pool mix. However, in spite of the host hardware supporting RDRAND, not all hypervisors support this instruction, and hypervisor support may be configurable on a per-VM basis. To display the information about the status of the RDRAND instruction, use the new DCL command `SHOW ENTROPY` (see [SHOW ENTROPY](#)).

5.30.1. SYSGEN Parameter RANDOM_SOURCES

The `RANDOM_SOURCES` `SYSGEN` parameter allows you to select the software and hardware sources of data that will contribute to the entropy pool. The default value is `-1`, meaning all possible sources are enabled.

Note

Not all categories of sources are implemented at this time. It is expected that additional sources will be added in subsequent updates or operating system releases.

For a detailed description of `RANDOM_SOURCES`, execute the command `HELP SYS_PARAMETERS RANDOM_SOURCES`.

VSI recommends leaving `RANDOM_SOURCES` set to `-1` unless you are advised differently by VSI support or engineering. `RANDOM_SOURCES` is a dynamic parameter and *does not* require a reboot after a value change.

5.30.2. SY\$GET_ENTROPY System Service

SY\$GET_ENTROPY

`SY$GET_ENTROPY` — Returns up to 256 bytes of entropy from the system-wide entropy pool. This service accepts 64-bit addresses.

Format

`SY$GET_ENTROPY buffer, buflen`

C Prototype:

```
int sys$get_entropy(void * buffer, unsigned __int64 buflen);
```

Parameters

BUFFER

A 32- or 64-bit address of a buffer to receive the random data. The service will fill the buffer with random data up to buffer length number of bytes. The maximum amount of data returned in a single call is 256 bytes.

OpenVMS Usage:	address
Type:	address
Access:	write only
Mechanism:	by value

BUFLEN

Size of the buffer in bytes. The maximum value is 256.

OpenVMS Usage:	byte count
Type:	integer
Access:	read only
Mechanism:	by value

Description

The SYSS\$GET_ENTROPY service retrieves bytes of data from the system entropy pool and writes them to the address specified in the buffer parameter.

Required Privileges

None.

Required Quota

None.

Condition Values Returned

SS\$_NORMAL	Successful completion.
SS\$_ACCVIO	The supplied buffer is not writeable in the callers mode.
SS\$_BADBUFLEN	Buffer length is zero or larger than 256.

5.30.3. SHOW ENTROPY

The SHOW ENTROPY command provides information about the state of the entropy engine. For more information, see *VSI OpenVMS DCL Dictionary: N-Z* [https://docs.vmssoftware.com/vsi-openvms-dcl-dictionary-n-z/#BRASS_ENTROPY] and the VSI OpenVMS HELP entry for this command.

5.30.4. Symmetric Multiprocessing (SMP)

VSI OpenVMS x86-64 supports a maximum of 32 CPUs.

If you increase the number of CPUs in your virtual machine configuration, you will see messages like the following during system startup:


```
%SMP-I-CPUTRN, CPU #2 has joined the active set.  
%SMP-I-CPUTRN, CPU #1 has joined the active set.  
%SMP-I-CPUTRN, CPU #3 has joined the active set.
```

Once VSI OpenVMS x86-64 is up and running, the DCL command `SHOW CPU` will reflect your CPU count. For example:

```
$ show cpu  
System: X86VMS, VBOX    VBOXFACP  
CPU ownership sets:  
    Active           0-3  
    Configure        0-3  
CPU state sets:  
    Potential        0-3  
    Autostart        0-3  
    Powered Down     None  
    Not Present      None  
    Hard Excluded    None  
    Failover         None  
$
```

The DCL command `STOP/CPU n` will remove a CPU from the set of CPUs being used. For example:

```
$ stop/cpu 3  
%SMP-I-CPUTRN, CPU #3 was removed from the active set.
```

The DCL command `START/CPU n` will add a CPU to the set of CPUs being used. For example:

```
$ start/cpu 3  
%SMP-I-CPUTRN, CPU #3 has joined the active set.
```

6. Virtualization Notes

The notes in this section describe known issues and limitations when running VSI OpenVMS x86-64 E9.2-3 as a **guest operating system** in Oracle VM VirtualBox, KVM, and VMware virtual machines.

6.1. Changing Settings of a Running Virtual Machine May Cause a Hang

Even if the hypervisor appears to allow it, *do not* change the configuration of a VSI OpenVMS virtual machine while it is running, as this may cause it to hang. Before editing the settings, make sure to power off the virtual machine.

6.2. Time of Day May Not Be Maintained Correctly in Virtual Machine Environments

VSI OpenVMS x86-64 may not maintain the time of day correctly in virtual machine environments after certain events, such as booting, suspending, taking a snapshot of a virtual machine, or other similar events (depending on the virtual machine host). To keep the time of day accurate in such cases, use the `SET TIME` command. If this does not resolve the problem, enter the following command via the `SYSTEM` account:

```
$ @SYS$MANAGER:UTC$SET_TIME.COM
```

This issue will be addressed in a future release of VSI OpenVMS x86-64.

6.3. System Time on KVM Virtual Machines

On KVM/QEMU systems, when a virtual machine is powered on, the system time is set based on the value of the `CLOCK OFFSET` parameter in the configuration file of that virtual machine. The default value is `'utc'`. Depending on the physical location of the host system, this might lead to differences between system times of the VM and the host.

To resolve this problem, use the `virsh edit` command to edit the XML configuration file of your virtual machine and change the value of the `CLOCK OFFSET` parameter to `'localtime'`, like so:

```
<clock offset='localtime'>
```

For more information, see the official documentation for your Linux distribution.

6.4. VirtualBox and Hyper-V Compatibility on Windows 10 and 11 Hosts

Host systems running Windows 10 and 11 that have previously run Microsoft Hyper-V hypervisor may fail the CPU feature checks. The issue is that certain CPU features are supported on the host system (the `vmcheck.py` script passes), but not on the guest system (the OpenVMS Boot Manager check fails). Primarily, the XSAVE instruction may not be present on the guest system.

This issue persists even if the Hyper-V feature has been removed. This happens because certain Hyper-V services interfere with VirtualBox.

The VirtualBox developers are aware of this issue and are working to improve the interoperability with Hyper-V.

To explicitly disable execution of the Hyper-V services that interfere with VirtualBox, perform the following steps on your Windows host system:

1. Run Command Prompt as administrator.
2. In Command Prompt, execute the following command to disable Hyper-V:

```
bcdedit /set hypervisorlaunchtype off
```
3. Shut down your Windows host system by executing the following command:



```
shutdown -s -t 2
```
4. Power on and boot your Windows host system again.

The XSAVE instruction should now be available to your VirtualBox guest.

For more information about the CPU feature checks, see Section 2.4, “CPU Compatibility Checks for Virtual Machines” in the Section 2, “Read These Before You Start” section.

Tips on How To Determine If Hyper-V Services Impact Your VirtualBox VM

When you launch a VirtualBox guest, look for the icon in the guest window status bar.

- A green turtle icon () indicates that the VirtualBox host is running as a Hyper-V guest with diminished performance.
- An icon with a V symbol () indicates that you are running directly on a VirtualBox host.

View the log file VBOX.LOG.

1. To open the log file, in the VirtualBox VM Manager window, right-click on the virtual machine entry and select **Show Log** from the menu.
2. In the log file, search for “XSAVE”.
 - If it shows "1 (1)", your VM guest has XSAVE.
 - If it shows "0 (1)", your VM guest has Hyper-V services impacting it.
3. In the log file, search for “HM”. The following message also indicates that Hyper-V is active:

```
{timestamp} HM: HMR3Init: Attempting fall back to NEM: VT-x is not available  
{timestamp} NEM: WHvCapabilityCodeHypervisorPresent is TRUE, so this might work.
```

6.5. VirtualBox: TCP Ports May Become Unusable After Guest Is Terminated

When running VSI OpenVMS x86-64 as a guest in a VirtualBox VM, TCP console ports may become unusable after a guest session has been terminated. After that, you cannot connect to your VirtualBox VM again. These ports remain in the LISTEN state even after you have disconnected the remote terminal session.

As a workaround, use the following commands to free your TCP ports and connect to your VirtualBox VM:

```
vboxmanage controlvm <vmname> changeuartmode1 disconnected  
vboxmanage controlvm <vmname> changeuartmode1 tcpserver <port>
```

The VirtualBox developers have indicated that the fix will be provided in an upcoming VirtualBox maintenance release.

6.6. VMware Guest May Fail to Boot After Adding Second SATA Controller

It has been reported that a VMware guest does not boot when a second SATA controller is added to the configuration. In their case, removing the second SATA controller eliminates the issue.

VSI has not observed boot issues when adding a second SATA controller during testing. If you encounter this situation, please report your issue via the VMS Software Service Platform.

6.7. Boot Manager Displays Incomplete List of Bootable Devices

If you are running a VMware virtual machine with multiple buses, you may run into a problem with the list of bootable devices displayed by the Boot Manager.

If, after entering the `DEVICE` command, your Boot Manager display normally looks similar to the following:

```

BOOT MANAGER DEVICE:  DKB0
DEFAULT BOOT COMMAND:  BOOT DKB0 0 80000100

VIRTUAL MACHINE GUEST:  VMware (TM) No Mouse support; Use Commands or Arrow Keys

CONNECT A REMOTE TERMINAL SESSION NOW.
BOOTMGR> DEVICE
BOOTABLE DEVICES (System Disks, Installation Kits, other):

DKA0      (HD) = FS0   UEFI: E9_2      OpenVMS: CCS0_92_XG6F 20480 MB SCSI Disk
DKB0      (HD) = FS1   UEFI: E9_2      OpenVMS: 9266F_CARL02 16384 MB SATA Disk
DKB200    (HD) = FS2   UEFI: X9_2_XG69  OpenVMS: X86XG69      8192 MB SATA Disk
DKB100    (DVD) = FS3   UEFI: E9_2      OpenVMS: None         1263 MB SATA DVD

BOOTMGR>
    
```

but occasionally, upon shutting down your VMware virtual machine, it appears as shown below:

```

BOOT MANAGER DEVICE:  DKB0
DEFAULT BOOT COMMAND:  BOOT DKB0 0 80000100

VIRTUAL MACHINE GUEST:  VMware (TM) No Mouse support; Use Commands or Arrow Keys

CONNECT A REMOTE TERMINAL SESSION NOW.
BOOTMGR> DEVICE
BOOTABLE DEVICES (System Disks, Installation Kits, other):

DKB0      (HD) = FS0   UEFI: E9_2      OpenVMS: 92G6F_CARL02 16384 MB SATA Disk

BOOTMGR>
    
```

This means, not all of your devices and/or buses have been configured properly. Proceeding to boot your VMware virtual machine from this truncated configuration display may result in an incomplete configuration of your Virtual Machine’s buses and the disk devices on those buses.

This happens because, by default, VMware products do not allow the UEFI Shell to be launched by the platform firmware Boot Option and have the Quick Boot option enabled.

These problems can be resolved by setting the correct values for the `efi.shell.activeByDefault` and `efi.quickBoot.enabled` parameters. To do so, follow the procedure described in Section 6.8, “Possible Issues with VMware Virtual Machines”.

6.8. Possible Issues with VMware Virtual Machines

Virtual machines created in VMware hypervisors (ESXi, Workstation Pro, Player, Fusion) may not operate as intended until you manually set the parameters listed in the table below.

Depending on your specific configuration, there may be cases where you may not need to set one or more of these parameters. VSI provides this information in case you experience the issues these parameters address.

Key/Parameter Name	Value	Description
<code>efi.serialconsole.enabled</code>	TRUE	<p>This parameter enables serial console access to the UEFI Shell and VSI OpenVMS Boot Manager.</p> <p>By default, VMware disables serial port console access. Because of this, a remote serial connection only becomes active once OpenVMS is booted (SYSBOOT or beyond).</p>

Key/Parameter Name	Value	Description
efi.shell.activeByDefault	TRUE	By default, VMware products do not allow the UEFI Shell to be launched by the platform firmware Boot Option. Setting this parameter to TRUE will allow for automatic launching of the Shell.
efi.quickBoot.enabled	FALSE	By default, the Quick Boot option is enabled. With Quick Boot enabled, the VM attempts to map <i>only the essential</i> devices to speed up the boot process. However, to make sure that <i>all</i> devices are visible to the VSI OpenVMS Boot Manager, Quick Boot must be disabled. Be aware that on large configurations with hundreds of disks, the boot process can take several minutes.

Warning

Note that key names are *case-sensitive*.

To set these parameters for a VMware ESXi virtual machine, follow these steps:

1. Select your VM and click **Edit**.
2. Switch to the VM Options tab.
3. Expand the **Advanced** menu.
4. Under Configuration Parameters, click **Edit Configuration**.
5. Click **Add Parameter**.
6. Enter the new key (parameter name) and set the value according to the table above.

Note

Quotes around the values are *not* required on ESXi.

7. Click **OK**.
8. Repeat steps 5 through 7 to add the other two parameters.
9. Click **Save**.

To set these parameters for a virtual machine running under any other VMware product, follow these steps:

1. Determine the location of your VM configuration file. To do so, perform the following steps:
 - a. Select your VM and bring up its Settings window.
 - b. Switch to the **Options** tab.
 - c. The directory specified in the **Working Directory** field is where you will find your VM configuration file (it will be named *vm_name.vmx*).
2. Make sure your VM is powered off.

3. Open the folder that contains your VM configuration file.
4. Open the file in an editor.
5. Enter the new keys (parameter names) and set their values according to the table above.

Note

Quotes around the values *are required* when manually editing VMX files.

6. Save and close the file.

6.9. One VirtIO-SCSI Adapter Supported on KVM

On KVM/QEMU, only *one* VirtIO-SCSI adapter can be configured and used on a VSI OpenVMS x86-64 V9.2-x system. Note that this functionality requires QEMU version 6.2 or later.

6.10. OpenVMS Clusters on Virtual Machines

VSI OpenVMS x86-64 supports VirtualBox, KVM, and VMware virtual machines in OpenVMS clusters. Virtual disks can be shared in a cluster via MSCP. Starting with the E9.2-3 release, VSI OpenVMS x86-64 supports direct (non-MSCP) shared disk access via the VMware VMDirectPath I/O feature (see Section 3.1.3, “VMware VMDirectPath I/O Support for Data Disks”). Cluster common system disks are not supported.

VSI OpenVMS x86-64 was tested in a clustered configuration using V8.4-1H1, V8.4-2, V8.4-2L1, V8.4-2L2, and V8.4-2L3. VSI has tested 2-node and 3-node clusters with MSCP-served disks where appropriate, CLUSTER_CONFIG_LAN.COM, and many relevant SET, SHOW, and SYSMAN commands. Other configurations will be tested at a later date.

Adding a Node Using a Copy of an Existing System Disk

On VSI OpenVMS x86-64 systems, you must perform an additional step if you use a copy of an existing system disk as the initial system disk of a new node that is being added to a cluster.

In addition to tasks such as modifying the SCSNODE and SCSSYSTEMID parameters and changing the label on the new system disk, you must also change the label for the memory disk. Follow these steps:

Note

The following steps assume that the new system disk is DKA300:, and it is already mounted.

1. Invoke LD\$STARTUP.COM by using the following command:

```
@SYS$STARTUP:LD$STARTUP.COM
```

2. Connect and mount the memory disk container file using the following commands:

```
$ LD CONNECT DKA300:[VMS$COMMON.SYS$LDR]SYS$MD.DSK LDM LDDEV  
$ MOUNT/OVER=ID LDDEV
```

3. Note the label of the memory disk. It will be of the form “MD20345927FD”. Change the last letter to create a unique name. For example:

```
$ SET VOLUME LDDEV /LABEL=MD20345927FE
```

4. Dismount the memory disk before completing the other setup tasks for the new system disk.

```
$ DISMOUNT LDDEV  
$ LD DISCONNECT LDDEV
```

7. Layered and Open-Source Products Notes

Layered products and field-test versions of open-source products for VSI OpenVMS x86-64 can be downloaded individually from the [VMS Software Service Platform \[https://sp.vmssoftware.com/\]](https://sp.vmssoftware.com/).

Released versions of open-source products for VSI OpenVMS x86-64 can be downloaded from the [official VMS Software website \[https://vmssoftware.com/products/list/?license=Open%20Source\]](https://vmssoftware.com/products/list/?license=Open%20Source).

VSI recommends that you regularly check both resources for up-to-date versions of layered and open-source products for VSI OpenVMS x86-64.

For detailed information about the products, please refer to the associated product release notes bundled with the kits.