



VMS Software

# **VSI OpenVMS x86-64 E9.2-4**

## **Release Notes**

**Publication Date:** September 2026

## VSI OpenVMS x86-64 E9.2-4 Release Notes



---

Copyright © 2026 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.

All other trademarks and registered trademarks mentioned in this document are the property of their respective holders.

# Table of Contents

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| 1. Introduction .....                                                                | 5  |
| 1.1. Field Test Build Considerations .....                                           | 5  |
| 1.2. Supported Platforms .....                                                       | 5  |
| 1.3. MD5 Checksum for the X86E924OE.ZIP File .....                                   | 6  |
| 1.4. x86-64 Licensing .....                                                          | 6  |
| 2. New in This Release .....                                                         | 6  |
| 2.1. New Features .....                                                              | 6  |
| 2.1.1. C Run-Time Library .....                                                      | 6  |
| 2.1.2. Guest Console Enhancements .....                                              | 7  |
| 2.1.3. HYPERSORT Utility Enabled by Default .....                                    | 7  |
| 2.1.4. Improved MSCP Throughput and Scalability .....                                | 7  |
| 2.1.5. iSCSI Data Disk Support .....                                                 | 7  |
| 2.1.6. OpenLDAP as the Supported LDAP Solution .....                                 | 8  |
| 2.1.7. OpenVMS Boot Manager Improvements .....                                       | 8  |
| 2.1.8. OpenVMS Debugger .....                                                        | 9  |
| 2.1.9. PEDRIVER Enhancements .....                                                   | 9  |
| 2.1.10. Signed Images .....                                                          | 10 |
| 2.1.11. Support for Virtio-SCSI Virtual CD/DVD-ROM Devices .....                     | 11 |
| 2.1.12. Support for VMware Paravirtual SCSI (PVSCSI) .....                           | 11 |
| 2.1.13. SYSBOOT Output Pagination .....                                              | 11 |
| 2.1.14. System Dump Configuration Enhancements .....                                 | 12 |
| 2.1.15. VirtIO Random Number Generator Support .....                                 | 12 |
| 2.1.16. VM Tools .....                                                               | 12 |
| 2.2. Fixed Issues .....                                                              | 13 |
| 2.2.1. C Run-Time Library .....                                                      | 13 |
| 2.2.2. OpenVMS Boot Manager .....                                                    | 14 |
| 2.2.3. OpenVMS Debugger .....                                                        | 14 |
| 2.2.4. PIPE Command Procedures in Batch Jobs .....                                   | 17 |
| 2.2.5. SET DUMP_OPTIONS Qualifier Negation .....                                     | 18 |
| 2.3. Issues Discovered in This Release .....                                         | 18 |
| 2.3.1. Cluster Common System Disks Not Recommended .....                             | 18 |
| 2.3.2. CLUSTER_CONFIG.COM Does Not Correctly Create Roots Above SYS99 .....          | 18 |
| 2.3.3. CLUSTER_CONFIG_LAN Node Deletion Leaves Files on Memory Disk .....            | 18 |
| 2.3.4. Cluster Quorum Disks Not Supported .....                                      | 19 |
| 2.3.5. C RTL Register Preservation and Invocation Context Issues .....               | 19 |
| 2.3.6. FAST_PATH_PORTS Setting Ignored for the iSCSI Port Driver .....               | 19 |
| 2.3.7. LANCP Commands May Exceed BYTLM Quota .....                                   | 19 |
| 2.3.8. Microsoft Windows iSCSI Target Server Compatibility .....                     | 20 |
| 2.3.9. Network Adapter Limits on Virtual Machines .....                              | 20 |
| 2.3.10. OpenLDAP Template Files May Generate PCSI-I-RETAIN Messages .....            | 20 |
| 2.3.11. Secure Boot Overrides IMAGE_SECURITY Settings .....                          | 20 |
| 2.3.12. SET PROCESS/DUMP=NOW Fails to Generate Process Dumps .....                   | 21 |
| 2.3.13. Virtio-NET Transitional Device Limit on KVM .....                            | 21 |
| 2.3.14. Virtio-SCSI Boot Manager Device Limit on KVM .....                           | 21 |
| 2.3.15. VSI Performance Management Fails if IMAGE_SECURITY is Higher Than<br>1 ..... | 21 |
| 3. Known Issues and Limitations .....                                                | 22 |
| 3.1. Features Not Available in VSI OpenVMS x86-64 E9.2-4 .....                       | 22 |
| 3.2. BACKUP/INITIALIZE to a Disk Mounted /FOREIGN Does Not Work .....                | 22 |

|                                                                                                 |    |
|-------------------------------------------------------------------------------------------------|----|
| 3.3. Cluster Nodes Running VSI OpenVMS V9.2 May Cause All x86-64 Cluster Members to Crash ..... | 23 |
| 3.4. Connecting to a Shared LD Container in a Mixed-Architecture Cluster .....                  | 23 |
| 3.5. DECnet MAC Address Change May Block Network Traffic .....                                  | 23 |
| 3.6. DECnet Over Wi-Fi .....                                                                    | 24 |
| 3.7. ENCRYPT Utility Does Not Work as Expected .....                                            | 24 |
| 3.8. LAN Failover MAC Address Change May Block Network Traffic .....                            | 24 |
| 3.9. MACRO-32 Unaligned Stack Issue .....                                                       | 24 |
| 3.10. Possible Issues with BOOT_OPTIONS.COM .....                                               | 24 |
| 3.11. Privileged Images Linked Using /SYSEXEC Should Be Relinked .....                          | 25 |
| 3.12. Symbolic Debugger .....                                                                   | 25 |
| 3.12.1. Supported Registers .....                                                               | 25 |
| 3.12.2. Language Support Limitations .....                                                      | 25 |
| 3.12.3. Source Line Correlation .....                                                           | 25 |
| 3.13. Viewing Call Stack in Pthread Debugger .....                                              | 26 |
| 3.14. VSI FTP Service Might Not Connect Correctly In Virtual Environments .....                 | 26 |
| 4. Platform-Specific Notes .....                                                                | 27 |
| 4.1. Memory Disks .....                                                                         | 27 |
| 4.2. Process Dumps .....                                                                        | 28 |
| 4.3. System Crash Dumps .....                                                                   | 28 |
| 4.4. User-Written x86-Assembler Modules .....                                                   | 31 |
| 5. Products Included With This Release .....                                                    | 32 |
| 5.1. ZIP/UNZIP Tools .....                                                                      | 32 |

# 1. Introduction

VMS Software, Inc. (VSI) is pleased to introduce VSI OpenVMS x86-64 E9.2-4.

VSI OpenVMS x86-64 E9.2-4 is the first OpenVMS x86-64 release built natively on OpenVMS x86-64 systems. Previous OpenVMS x86-64 releases were built on OpenVMS IA-64 systems using cross-compilation tools. This transition allows the operating system to be built using the x86-64 compiler environment and may benefit from improved optimization and code generation, potentially resulting in performance improvements in some areas.

This document is intended for all users of VSI OpenVMS x86-64 E9.2-4. Read this document before you install or use VSI OpenVMS x86-64 E9.2-4.

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

- *Installing and Upgrading OpenVMS x86-64* [<https://howto.vmssoftware.com/Installing/index.html>]
- *VSI OpenVMS x86-64 Boot Manager User Guide* [<https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide-922/>]
- *VSI OpenVMS on VMware ESXi Configuration Guide* [<https://howto.vmssoftware.com/ESXi/index.html>]
- *VSI OpenVMS on KVM Configuration Guide* [<https://howto.vmssoftware.com/KVM/index.html>]
- *VSI OpenVMS Calling Standard Manual* [<https://docs.vmssoftware.com/vsi-openvms-calling-standard/>]
- *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/>]

## 1.1. Field Test Build Considerations

The VSI OpenVMS x86-64 E9.2-4 field test build enables the `SYSTEM_CHECK`, `POOLCHECK`, and `BUGCHECKFATAL` system parameters by default.

These settings assist in identifying software defects during field testing. As a result, field test systems may bugcheck in situations that would not normally result in a system crash in a production release. In addition, the `SYSTEM_CHECK` and `POOLCHECK` settings may reduce system performance.

Users who find this behavior disruptive in their environment may choose to set these parameters to 0. However, this may reduce the ability of the field test build to detect and report software defects.

## 1.2. Supported Platforms

For information about supported platforms and configurations, refer to the [Supported Platforms](https://vmssoftware.com/products/supported-platforms/) [<https://vmssoftware.com/products/supported-platforms/>] page on the VMS Software website.

The following platforms have been tested by VSI with VSI OpenVMS x86-64 E9.2-4.

### 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 E9.2-4 with KVM on several Linux distributions. The following table lists the Linux distributions, Linux kernel versions, and QEMU versions used during testing:

| Linux Distribution                            | Linux Kernel Versions            | QEMU Versions<br>(package information) |
|-----------------------------------------------|----------------------------------|----------------------------------------|
| Oracle Linux (OL) Server 9.7                  | 5.15.0-320.202.8.5.el9uek.x86_64 | 9.1.0<br>(qemu-kvm-9.1.0-29.el9_7.6)   |
| Red Hat Enterprise Linux<br>(RHEL) 9.8 (Plow) | 5.14.0-687.12.1.el9_8.x86_64     | 9.1.0<br>(qemu-kvm-9.1.0-29.el9_7.6)   |
|                                               | 5.14.0-687.30.1.el9_8.x86_64     | 10.1.0<br>(qemu-kvm-10.1.0-17.el9_8)   |

## VMware ESXi

VSI has tested VSI OpenVMS x86-64 E9.2-4 with VMware ESXi V8.0.x.

## 1.3. MD5 Checksum for the X86E924OE.ZIP File

VSI recommends that you verify the MD5 checksum of the **X86E924OE.ZIP** file after it has been downloaded from the [VMS Software Service Platform \[https://sp.vmssoftware.com/packages/1819\]](https://sp.vmssoftware.com/packages/1819) to your target system. The MD5 checksum of **X86E924OE.ZIP** must correspond to the following value:

**A42888ACA06FE2046BD12B1B1F49B856**

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

## 1.4. 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 [Licensing on OpenVMS x86-64 \[https://howto.vmssoftware.com/Licensing/index.html\]](https://howto.vmssoftware.com/Licensing/index.html).

# 2. 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.

## 2.1. New Features

This section lists features introduced in the current release.

### 2.1.1. C Run-Time Library

The following new features and improvements relate to the C Run-Time Library (RTL).

#### 2.1.1.1. Enhanced `pipe()` Mailbox Access Control

Two OpenVMS-specific optional arguments, **promsk** and **acmode**, have been added to the `pipe()` function. These arguments allow applications to specify the protection mask and access mode, respectively, of mailboxes created by the function.

This enhancement improves interoperability between processes running with different privilege levels and provides greater control over pipe access permissions.

#### 2.1.1.2. Thread Scheduling Performance Improvements

Performance improvements have been made to the thread scheduling infrastructure, reducing thread context-switching overhead and improving the performance of scheduling operations such as `sched_yield()`.

### 2.1.2. Guest Console Enhancements

VSI OpenVMS x86-64 E9.2-4 introduces a number of enhancements to the OpenVMS Guest Console subsystem that improve stability, usability, and terminal compatibility.

Improvements include updated terminal emulation functionality, improved keyboard and keypad handling, corrections to screen redraw and scrolling behavior in full-screen applications such as TPU/EVE, and increased console stability during boot, shutdown, and console IPC operations.

#### 2.1.3. HYPERSORT Utility Enabled by Default

Starting with VSI OpenVMS x86-64 E9.2-4, the high-performance sort/merge utility (HYPERSORT) is enabled by default.

To switch back to the SORT utility, issue the following command:

```
$ DEFINE/NOLOG SORTSHR SYS$SHARE:SORTSHR
```

#### 2.1.4. Improved MSCP Throughput and Scalability

Starting with VSI OpenVMS x86-64 E9.2-4, the default value of the `MSCP_BUFFER` system parameter has been increased from 1024 to 4096, and the default value of the `MSCP_CREDITS` system parameter has been increased from 32 to 128. These changes increase the number of buffers available to MSCP servers and allow a greater number of concurrent I/O requests from client systems.

In addition, the DUDRIVER disk class driver has been rewritten in C. Together, these enhancements improve throughput between MSCP server and client nodes and reduce I/O stalls under heavy workloads.

#### 2.1.5. iSCSI Data Disk Support

VSI OpenVMS x86-64 E9.2-4 introduces iSCSI initiator support, allowing OpenVMS systems to connect to supported iSCSI storage targets and access remote block-storage devices as local disk volumes. Once configured, iSCSI storage devices are presented to the operating system as DZA devices and can be managed using standard OpenVMS disk-management commands.

The iSCSI initiator provides the following capabilities:

- Static and SendTargets discovery of iSCSI targets

- CHAP authentication, including one-way and mutual CHAP
- Multipath I/O support
- Logging and monitoring through iSCSI SDA commands
- FastPath support for optimized I/O performance

iSCSI support enables OpenVMS systems to use IP-based storage networks as an alternative to Fibre Channel infrastructure, allowing remote storage resources to be accessed through standard Ethernet networks.

For information about configuring and managing iSCSI storage, refer to the relevant chapter in the *Guidelines for OpenVMS Cluster Configurations* [[https://docs.vmssoftware.com/guidelines-for-openvms-cluster-configurations/#ISCSI\\_DATA\\_DISK\\_SUPPORT\\_CH](https://docs.vmssoftware.com/guidelines-for-openvms-cluster-configurations/#ISCSI_DATA_DISK_SUPPORT_CH)].

### 2.1.6. OpenLDAP as the Supported LDAP Solution

Starting with VSI OpenVMS x86-64 E9.2-4, VSI OpenLDAP is the supported LDAP solution for OpenVMS. VSI OpenLDAP provides a unified LDAP offering for OpenVMS systems.

The LDAP implementation integrated with OpenVMS (integrated LDAP) remains available in this release but is now considered deprecated and may be removed in a future release.

For additional information about OpenLDAP, refer to the *VSI OpenLDAP ACME Agent for OpenVMS Configuration and User Guide* [<https://docs.vmssoftware.com/vsi-openldap-acme-agent-config-and-user-guide/>] and the latest version of the *VSI OpenLDAP for OpenVMS Release Notes and Installation Guide* available at <https://docs.vmssoftware.com/>.

### 2.1.7. OpenVMS Boot Manager Improvements

VSI OpenVMS x86-64 E9.2-4 introduces a number of enhancements to the OpenVMS Boot Manager, including the following:

- The Boot Manager now supports Secure Boot environments. The Boot Manager executable is digitally signed, and SYSBOOT signature validation is supported during the boot process.
- New **SIGNED** and **NOSIGNED** Boot Manager commands have been added to control SYSBOOT signature validation.
- The Boot Manager supports disabling automatic booting by specifying **AUTOACTION 0**. This disables automatic booting and causes the Boot Manager to remain at the `BOOTMGR>` prompt when launched.
- Command history support has been enhanced. The last 15 commands entered are stored and can be recalled using the **Up** and **Down** arrow keys.
- Support has been added for booting from VMware Paravirtual SCSI (PVSCSI) devices and for Fibre Channel booting in KVM environments. PVSCSI devices are presented as standard SCSI devices and require no special Boot Manager configuration.

For more information, refer to the *VSI OpenVMS x86-64 Boot Manager User Guide* [<https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide/>].

## 2.1.8. OpenVMS Debugger

The following new features and improvements relate to the OpenVMS Debugger.

### 2.1.8.1. Build Support for PCA-Aware Modules

The DEBUG build procedure has been enhanced to compile PCA-aware modules in addition to the standard debugger modules. This change provides earlier validation of PCA-specific code paths and helps identify integration issues during the DEBUG build process rather than during subsequent PCA builds.

### 2.1.8.2. Default XMM Register Format

The debugger now supports a default display format for `%XMMn` registers on x86-64 systems. This capability is provided through the new **SET MODE XMM** command, which allows users to specify the default format used by the **EXAMINE** and **DEPOSIT** commands when working with XMM registers.

This enhancement simplifies the display and modification of XMM register contents during debugging sessions.

### 2.1.8.3. Improved Support for Optimized C++ Variables

The debugger has been enhanced to improve support for optimized C++ applications.

This enhancement improves the debugger's ability to access, evaluate, and display optimized variable values using compiler-generated debugging information.

### 2.1.8.4. Source Line Trimming

Source files created or edited on non-OpenVMS platforms can sometimes contain trailing control characters, such as carriage returns (`<CR>`), introduced during file transfer or translation to OpenVMS. Although these characters are ignored by the compiler, they may be displayed by the debugger when showing source lines.

The debugger now provides the **SET MODE TRIM** command. In this mode, the debugger trims trailing control characters from displayed source lines.

### 2.1.8.5. Support for C++ using Declarations

The debugger has been enhanced to support C++ `using` declarations and `using namespace` directives represented in compiler-generated debugging information.

This enhancement improves symbol resolution and breakpoint handling when debugging C++ applications that make use of imported declarations and namespaces.

## 2.1.9. PEDRIVER Enhancements

VSI OpenVMS x86-64 E9.2-4 introduces a number of enhancements to PEDRIVER, the OpenVMS Cluster LAN port driver. These enhancements are detailed in the following sections.

### 2.1.9.1. Expanded PEDRIVER Counters and Timestamps

PEDRIVER counters have been expanded from 16-bit and 32-bit values to 32-bit and 64-bit values, reducing the likelihood of counter overflow and allowing SCACP to display large counter values without truncation.

In addition, PEDRIVER now records timestamps for the four most recent errors and the 32 most recent ECS membership changes. Timestamp information can be displayed using the **/TIMESTAMP** qualifier on applicable SCACP commands.

### 2.1.9.2. PEDRIVER Interval Counter Enhancements

PEDRIVER interval counter processing has been enhanced. The new **/SNAPSHOT** qualifier allows a consistent snapshot of counters to be recorded across all buses, virtual circuits, and channels.

Subsequent SCACP **SHOW** commands can use the **/INTERVAL** qualifier to display activity that occurred since the most recent snapshot was taken.

### 2.1.9.3. PEDRIVER Compression, Checksum, and Transmit-Chaining Statistics

PEDRIVER now maintains global compression, checksum, and transmit-chaining statistics. These statistics can be displayed using the SDA **PE STATISTICS** command or by examining the associated PEDRIVER symbols.

The **PE STATISTICS** command also reports compression effectiveness metrics.

## 2.1.10. Signed Images

Starting with E9.2-4, VSI OpenVMS x86-64 cryptographically verifies the digital signatures of selected system images whenever they are loaded. When an image is loaded, OpenVMS computes a cryptographic digest of the image contents and uses the corresponding public key embedded in the operating system to verify the image signature.

This capability helps detect unauthorized or accidental modification of system images before they are loaded into the executive or into a running process, ensuring that signed system components originate from a trusted source and have not been altered since they were signed.

The following categories of images are subject to signature verification according to the **IMAGE\_SECURITY** policy:

- Executive loadable images
- Device drivers
- SDA extensions
- Images loaded using the **SYSMAN SYS\_LOADABLE** interface

For more information, refer to the relevant chapters in the *VSI OpenVMS System Manager's Manual, Volume 1: Essentials* [[https://docs.vmssoftware.com/vsi-openvms-system-manager-s-manual-volume-1-essentials/#IMAGE\\_SIGNING](https://docs.vmssoftware.com/vsi-openvms-system-manager-s-manual-volume-1-essentials/#IMAGE_SIGNING)] and *VSI OpenVMS Guide to System Security* [[https://docs.vmssoftware.com/vsi-openvms-guide-to-system-security/#SIGNING\\_CHAPTER](https://docs.vmssoftware.com/vsi-openvms-guide-to-system-security/#SIGNING_CHAPTER)].

### 2.1.10.1. IMAGE\_SECURITY System Parameter

The **IMAGE\_SECURITY** dynamic system parameter controls the behavior of system image signature verification.

The default value of **IMAGE\_SECURITY** is 1, which prevents loading system images with invalid signatures while permitting unsigned images to be loaded.

For additional information, refer to the relevant chapter in the *VSI OpenVMS Guide to System Security* [[https://docs.vmssoftware.com/vsi-openvms-guide-to-system-security/#SIGNING\\_CHAPTER](https://docs.vmssoftware.com/vsi-openvms-guide-to-system-security/#SIGNING_CHAPTER)].

### 2.1.10.2. IMGSEC Audit Event Support

The new IMGSEC audit event type can be used to review image-signing security events. These events can be examined using the following command:

```
$ ANALYZE/AUDIT/EVENT_TYPE=IMGSEC
```

This event type provides visibility into image-signing and signature-verification activity performed by the operating system.

### 2.1.11. Support for Virtio-SCSI Virtual CD/DVD-ROM Devices

Starting with E9.2-4, VSI OpenVMS x86-64 supports virtual CD/DVD-ROM devices attached through Virtio-SCSI controllers.

This enhancement provides an alternative to SATA-attached virtual CD/DVD-ROM devices and allows virtual media to be accessed through the Virtio-SCSI storage infrastructure.

### 2.1.12. Support for VMware Paravirtual SCSI (PVSCSI)

Starting with E9.2-4, VSI OpenVMS x86-64 supports VMware Paravirtual SCSI (PVSCSI) controllers on VMware ESXi systems. Support is provided by the PKPDRIVER port driver.

PVSCSI is VMware's high-performance para-virtualized storage adapter and is designed to provide improved I/O throughput, lower CPU overhead, and greater I/O efficiency than traditional emulated storage controllers such as LSI Logic. Prior releases of OpenVMS on VMware were limited to emulated SCSI storage controllers and could not take advantage of VMware's native PVSCSI functionality.

PVSCSI disks are presented as standard OpenVMS storage devices.

The PVSCSI implementation provides the following capabilities:

- Support for multiple PVSCSI controllers and logical units (LUNs).
- Installation of OpenVMS on PVSCSI-attached disks.
- Booting OpenVMS from PVSCSI-attached system disks.
- FastPath support for high-throughput I/O workloads.
- Support for OpenVMS Cluster environments.

PKPDRIVER provides functionality on VMware similar to that provided by VSPDRIVER for Virtio-SCSI storage devices on KVM systems.

OpenVMS includes the PKP SDA extension for diagnostics and troubleshooting of PVSCSI controllers. For more information, refer to the relevant sections in the *VSI OpenVMS System Analysis Tools Manual* [[https://docs.vmssoftware.com/vsi-openvms-system-analysis-tools-manual/#PKP\\_EXTENSION](https://docs.vmssoftware.com/vsi-openvms-system-analysis-tools-manual/#PKP_EXTENSION)] and the *VSI OpenVMS on VMware ESXi Configuration Guide* [[https://howto.vmssoftware.com/ESXi/common\\_tasks/adding\\_vmware\\_paravirtual\\_controller.html](https://howto.vmssoftware.com/ESXi/common_tasks/adding_vmware_paravirtual_controller.html)].

### 2.1.13. SYSBOOT Output Pagination

A new **/PAGE** qualifier has been added to the SYSBOOT **SHOW** command.

The **/PAGE** qualifier allows large SYSBOOT parameter listings to be displayed one screen at a time. This prevents output from scrolling off the screen during conversational boot and makes pre-boot configuration information easier to review.

## 2.1.14. System Dump Configuration Enhancements

Starting with VSI OpenVMS x86-64 E9.2-4, the OpenVMS dump subsystem has been enhanced to provide greater flexibility in configuring system memory dumps and error log dumps. These enhancements include support for directing error log dumps to Dump Off System Disk (DOSD) devices, preventing dump files from being written to the system disk, and improved reporting of dump configuration settings.

### 2.1.14.1. Enhanced Dump Configuration Reporting

The **SHOW DUMP\_OPTIONS** command has been enhanced to display the configuration status of the new **/ERROR\_LOGS** and **/SKIP\_SYSDISK** qualifiers, providing improved visibility into the active dump configuration.

### 2.1.14.2. Error Log Dumps to Dump Off System Disk Devices

A new **/ERROR\_LOGS** qualifier has been added to the **SET DUMP\_OPTIONS** command. When enabled, error log dump files (SYS\$ERRLOG.DMP) can be written to devices configured in the dump device list using the **/DEVICE** qualifier.

### 2.1.14.3. Preventing Dumps from Being Written to the System Disk

A new **/SKIP\_SYSDISK** qualifier has been added to the **SET DUMP\_OPTIONS** command. When enabled, the system disk is excluded from dump operations, preventing both system memory dumps and error log dumps from being written to the system disk.

## 2.1.15. VirtIO Random Number Generator Support

Starting with E9.2-4, VSI OpenVMS x86-64 provides VirtIO Random Number Generator (RNG) support on KVM systems.

VirtIO RNG provides a high-quality source of entropy for OpenVMS guest virtual machines by obtaining entropy from the host system. The **SHOW ENTROPY** command has also been enhanced to display the status of configured VirtIO RNG devices.

## 2.1.16. VM Tools

VSI OpenVMS x86-64 E9.2-4 introduces VM Tools, a collection of services designed to improve interaction between an OpenVMS x86-64 guest virtual machine and the host system. The initial implementation of VM Tools provides host time synchronization and graceful power operations. These features are described in the following sections.

For additional information about VM Tools, refer to the relevant chapter in the [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/#vm_tools) [https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-ii-m-z/#vm\_tools].

### 2.1.16.1. Host Time Synchronization

Starting with E9.2-4, VSI OpenVMS x86-64 supports host time synchronization on KVM and VMware ESXi virtual machines through VM Tools.

OpenVMS can automatically synchronize the guest clock with the host, correcting clock drift and maintaining accurate time after virtual machine pause and resume operations. Time synchronization supports both immediate clock corrections and gradual clock adjustments.

The new **SET TIME/HOST** and **SHOW TIME/HOST** commands can be used to synchronize the guest clock with the host, display the host time, and display the difference between the guest and host clocks.

When VM Tools time synchronization is enabled, DTSS and the TCP/IP NTP service must be disabled. For information about disabling these services, refer to [Installing and Upgrading OpenVMS x86-64](https://howto.vmssoftware.com/Installing/index.html) [https://howto.vmssoftware.com/Installing/index.html] and the relevant chapter in the [VSI TCP/IP Services for OpenVMS V6.0 Management Guide](https://docs.vmssoftware.com/vsi-tcpip-services-for-openvms-6-management/#NTP_CHAP) [https://docs.vmssoftware.com/vsi-tcpip-services-for-openvms-6-management/#NTP\_CHAP].

### 2.1.16.2. VMware Graceful Power Operations

Starting with E9.2-4, VSI OpenVMS x86-64 supports graceful handling of VMware shutdown, reboot, suspend, and resume requests on VMware ESXi systems through VM Tools.

When enabled, VM Tools allows OpenVMS to receive and process power operation requests initiated by the VMware host. This capability helps prevent data loss and file system corruption that can occur when a virtual machine is powered off or rebooted without an orderly guest shutdown.

Graceful power operations can be enabled using the VMTOOLS\$POWEROPS logical name.

## 2.2. Fixed Issues

This section lists issues that have been fixed with the current release.

### 2.2.1. C Run-Time Library

The following fixed issues relate to the C Run-Time Library (RTL).

#### 2.2.1.1. Deadlock in `flockfile()`

Previously, a deadlock could occur when a file stream locked with the `flockfile()` function was subsequently closed using `fclose()`. Under certain circumstances, the file lock was not released when the stream was closed. If the file was later reopened and accessed from another thread, the application could hang.

This issue has now been fixed.

#### 2.2.1.2. `mktime()` Does Not Update Time Zone Information

Previously, the `mktime()` function did not update the `tm_zone` and `tm_gmtoff` members of the `tm` structure. As a result, time zone and UTC offset information could remain unset after a call to `mktime()`.

This issue has now been fixed.

#### 2.2.1.3. Stack Overflow in `localtime_r()` When Called from a Child Thread

Previously, a stack overflow could occur if `localtime_r()` was first called from a child thread before being called from the main thread. This could result in an access violation (ACCVIO) and process termination.

This issue has now been fixed.

## 2.2.2. OpenVMS Boot Manager

The following issues in the OpenVMS Boot Manager have been corrected:

- In graphical mode, check-button states now correctly reflect the current state of the associated boot flags.
- Issues affecting automatic-action countdown processing have been corrected.
- Disk labels assigned using the **LABEL** command are now retained correctly across reboots.
- Improvements have been made to boot-device discovery reliability during system startup.

## 2.2.3. OpenVMS Debugger

The following fixed issues relate to the OpenVMS Debugger.

### 2.2.3.1. ACCVIO After STEP/INTO or GO

Previously, the debugger could encounter an access violation (ACCVIO) after issuing the **STEP/INTO** or **GO** command when execution entered code in a shared library.

This issue occurred because the debugger created a writable copy of a shared-library page without preserving its execute permissions. When program execution reached the copied page, the system generated an access violation.

This issue has now been fixed.

### 2.2.3.2. Anonymous Class Symbol Path

Previously, the debugger could display erroneous information for symbols associated with anonymous classes or anonymous routines. This issue was caused by incorrect handling of unnamed records when building symbol paths.

This issue has now been fixed.

### 2.2.3.3. Breakpoints in Unallocated Routines

Previously, the debugger would display the following error when setting a breakpoint in an unallocated routine:

```
%DEBUG-E-ERROR, debugger error or session corruption
```

This issue was caused by incorrect processing of trampoline information for unallocated routines.

This issue has now been fixed.

### 2.2.3.4. C++ Pointer-to-Member Types Not Supported

Previously, the debugger did not support C++ pointer-to-member types. As a result, expressions involving `.*` and `->*` operators could not be evaluated correctly.

This issue has now been fixed.

### 2.2.3.5. COBOL Programs Do Not Stop at the Main Entry Point

Previously, when debugging COBOL applications on x86-64 systems, execution would not stop at the program's main entry point. Instead, the debugger could stop at the initial transfer address

(ELF\$TFRADR), which is not associated with a source line. Users would then need to step through instructions manually until the debugger reached a location that could be mapped to the source code.

This issue has now been fixed.

#### **2.2.3.6. Explicit Inheritance in C++ Symbol Paths**

Previously, the debugger did not support explicit inheritance in C++ symbol paths (for example, *object.E::B::A::member*). This could prevent users from accessing members through a fully qualified inheritance hierarchy.

This issue has now been fixed.

#### **2.2.3.7. Floating-Point Fields in Packed Variant Records**

Previously, the debugger displayed floating-point fields in packed variant records as integer values. This occurred because floating-point type information was not preserved when the debugger created temporary types for bit-string fields.

This issue has now been fixed.

#### **2.2.3.8. Incorrect Alpha Pseudo-Register Handling**

Previously, the debugger did not correctly handle Alpha pseudo-registers. When executing a program under debugger control, some register values could become corrupted, and updates made to register values by the user were not always reflected in the running program.

This issue has now been fixed.

#### **2.2.3.9. Incorrect Array Element Type Display**

Previously, the debugger displayed array elements as integers regardless of their actual data type. This occurred because array type information was not preserved when building array element descriptors.

This issue has now been fixed.

#### **2.2.3.10. Incorrect Display of COBOL PIC 9(8) Values**

Previously, the debugger could display incorrect values when evaluating COBOL PIC 9(8) fields. This issue was caused by incorrect handling of VAX H-floating conversion during value display on x86-64 systems.

This issue has now been fixed.

#### **2.2.3.11. Incorrect Static Class Variable Offset Display**

Previously, the debugger could display an incorrect offset for static class variables due to the use of uninitialized data during debug information processing.

This issue has now been fixed.

#### **2.2.3.12. Incorrect Structure and Class Type Display**

Previously, the debugger could incorrectly identify structures as classes when displaying type information. This occurred because record-type information was not preserved correctly while internal debugger data structures were being built.

This issue has now been fixed.

### 2.2.3.13. Internal Memory Error on Debugger Exit

Previously, the debugger would sometimes return the following error when exiting certain programs:

```
%DEBUG-F-INTMEMERR, internal memory-pool error
```

This issue could occur when the program contained labels whose names consisted entirely of digits. The debugger did not process such labels correctly, resulting in an internal memory error.

This issue has now been fixed.

### 2.2.3.14. Local C++ Symbols Not Resolved Correctly

Previously, the debugger could fail to resolve some local C++ symbols from their addresses. As a result, commands such as **EXAMINE** could display memory addresses instead of symbolic variable names for certain local variables.

This issue was caused by incorrect processing of DWARF frame-base information when calculating local variable addresses.

This issue has now been fixed.

### 2.2.3.15. Member Lookup Through Derived Classes

Previously, the debugger could fail to locate C++ members when the specified member path included a derived class or structure. This occurred because derived types were not properly considered during symbol-path resolution.

This issue has now been fixed.

### 2.2.3.16. Routine Signatures in Scope Strings

Previously, the debugger did not allow users to specify a routine signature in a scope string and would display an error message. This occurred because support for parentheses in scope-string parsing had not been implemented.

This issue has now been fixed.

### 2.2.3.17. Screen Mode Display on Wide Terminals

Previously, the debugger's screen mode could display incorrectly when used in very wide terminal windows, causing screen layout inconsistencies.

This issue has now been fixed.

### 2.2.3.18. SHOW SYMBOL/FULL Error with Anonymous Classes

Previously, the debugger could display an error when examining a pointer to an anonymous class using the **SHOW SYMBOL/FULL** command. This occurred because the debugger did not correctly handle unnamed records.

This issue has now been fixed.

### 2.2.3.19. Static Const Member Display and Evaluation

Previously, the debugger did not correctly display the values of static const data members and would report only "address: 0 [static]". Evaluation of such members in debugger expressions could also fail.

This issue has now been fixed.

#### **2.2.3.20. STEP Command May Not Stop Execution**

Previously, under certain conditions, the debugger could continue program execution to completion after a **STEP** command rather than stopping at the next instruction or source line.

This issue was caused by a trace condition that was not processed correctly by the debugger.

This issue has now been fixed.

#### **2.2.3.21. STEP/INTO May Continue Execution to Program Completion**

Previously, when debugging C++ applications, the **STEP/INTO** command could fail to stop after returning from a routine and instead continue execution until program termination. This occurred because the debugger would make an incorrect decision when processing a return (RET) instruction.

This issue has now been fixed.

#### **2.2.3.22. STEP/RETURN Fails to Stop at Multi-Entry Routine Epilog**

Previously, the debugger could fail to stop at the routine epilog when the **STEP/RETURN** command was used within a multi-entry routine.

This issue has now been fixed.

#### **2.2.3.23. STEP/RETURN May Not Locate Routine Epilogues Correctly**

Previously, the debugger could intermittently fail to process the **STEP/RETURN** command correctly. This occurred because the debugger did not process the DWARF line-processing opcode DW\_LNS\_set\_epilogue\_begin and therefore could not always identify the beginning of a routine epilogue accurately.

This issue has now been fixed.

#### **2.2.3.24. STEP/RETURN Stops at Incorrect Routine**

Previously, if a breakpoint was set within a called routine, the **STEP/RETURN** command could stop at the end of that routine rather than at the end of the routine from which the command was issued.

This issue was caused by the debugger failing to evaluate all continuation conditions correctly.

This issue has now been fixed.

#### **2.2.3.25. Template Debug Information**

Previously, the debugger did not display template-related debug information, such as template type parameters and template value parameters, because template-specific debug tags were not processed.

This issue has now been fixed.

### **2.2.4. PIPE Command Procedures in Batch Jobs**

Previously, when the **PIPE** command executed one or more DCL command procedures in a batch job, subsequent commands in the calling command procedure could be skipped unexpectedly.

This issue could affect both simple **PIPE** commands that execute DCL command procedures and **PIPE** command sequences containing multiple indirect command invocations.

This issue has now been fixed.

### 2.2.5. SET DUMP\_OPTIONS Qualifier Negation

Previously, negating certain **SET DUMP\_OPTIONS** qualifiers, such as **/NODEVICE**, could result in dump processing continuing to use configuration values established during system startup until the next system reboot.

This issue has now been fixed. Changes made using qualifier negation are now reflected immediately in the active dump configuration and no longer require a reboot to take effect.

## 2.3. Issues Discovered in This Release

### 2.3.1. Cluster Common System Disks Not Recommended

Although Cluster Common System Disk (CCSD) capability is available in VSI OpenVMS x86-64 E9.2-4, the use of CCSDs is **not recommended** with this field test release.

A system disk integrity issue affecting CCSD configurations was identified shortly before the release of the E9.2-4 field test kit. We plan to release a field test update kit to address this issue. Until then, VSI strongly recommends that customers avoid deploying CCSDs.

### 2.3.2. CLUSTER\_CONFIG.COM Does Not Correctly Create Roots Above SYS99

When the SY\$MANAGER:CLUSTER\_CONFIG.COM command procedure is used to create a new system root with a name above SYS99, the operation does not complete correctly.

The root directory is created on the system disk, but the corresponding Memory Disk update fails. As a result, the new root is not available on the Memory Disk and cannot be used as expected.

The operation reports an error similar to the following:

```
%DCL-W-BUFOVF, command buffer overflow - shorten expression or command line
```

As a workaround, use root names in the range SYS0 through SYS99.

This limitation will be addressed in a future release.

### 2.3.3. CLUSTER\_CONFIG\_LAN Node Deletion Leaves Files on Memory Disk

When a node is added to a cluster using the SY\$MANAGER:CLUSTER\_CONFIG\_LAN.COM command procedure, a root-specific directory (SYS $n$ ) is created on both the system disk and the Memory Disk. Product kits may install root-specific files in these directories.

If the node is later deleted, the corresponding root-specific directory and files on the Memory Disk are not automatically removed.

To remove these files manually, execute the CLUSTER\_CONFIG.COM **DELETE** command again, specifying the Memory Disk (SY\$SYSDEVICE\_MD) and the same root name as the deleted node. This method will leave the deleted space available on the Memory Disk and does not specifically require a reboot.

Alternatively, if you wish to recover space in the Memory Disk, reducing its size, you can execute the `SYSS$UPDATE:SYSS$MD.COM` command procedure to rebuild the Memory Disk, which will then reflect only the system roots that exist on the system disk. A reboot is necessary to use the new version of the Memory Disk. You may also wish to execute the **PURGE** command against the Memory Disk container file `SYSS$LOADABLE_IMAGES:SYSS$MD.DSK` after rebooting. VSI recommends using this method only when the system can be taken offline and rebooted.

This issue will be resolved in a future release.

### 2.3.4. Cluster Quorum Disks Not Supported

Cluster quorum disks are not supported in VSI OpenVMS x86-64 E9.2-4.

This limitation will be addressed in a future release.

### 2.3.5. C RTL Register Preservation and Invocation Context Issues

The C RTL `set jmp()` and `long jmp()` routines may not correctly save and restore register values, which can lead to incorrect program behavior.

In addition, the `LIB$X86_GET_CURR_INVO_CONTEXT` and `LIB$X86_GET_PREV_INVO_CONTEXT` routines may save incorrect register information in the Invocation Context Block (ICB).

This issue will be addressed in a future release.

### 2.3.6. FAST\_PATH\_PORTS Setting Ignored for the iSCSI Port Driver

When the system parameter `FAST_PATH` is set to 1, the `FAST_PATH_PORTS` system parameter cannot be used to disable FastPath specifically for the iSCSI (PZA) port driver. The setting is accepted without error or warning, but FastPath remains enabled for the iSCSI port driver.

As a result, configurations that rely on disabling FastPath for the iSCSI port driver may not behave as expected.

To disable FastPath for the iSCSI port driver, FastPath must currently be disabled globally by setting `FAST_PATH` to 0.

This limitation may be addressed in a future release.

### 2.3.7. LANCP Commands May Exceed BYTLM Quota

Some LANCP commands may require more BYTLM than is available to the account used to execute them.

If the BYTLM quota is exceeded, the commands may produce incomplete output and return an error similar to the following:

```
%LANCP-E-EXBYTLM, Exceeded byte count quota (BYTLM)
```

To provide LANCP with sufficient BYTLM resources, increase the BYTLM quota of the affected account. For example:

```
UAF> SHOW account
```

```
UAF> MODIFY account/BYTLM=value
```

where *account* is the affected account and *value* is sufficient for the LANCP commands being executed.

### 2.3.8. Microsoft Windows iSCSI Target Server Compatibility

Microsoft Windows iSCSI Target Server implementations are not currently supported for use with the VSI OpenVMS x86-64 iSCSI initiator. Device autoconfiguration may fail when connecting to a Windows iSCSI Target Server, preventing the associated storage devices from being configured for use by OpenVMS.

### 2.3.9. Network Adapter Limits on Virtual Machines

VSI OpenVMS x86-64 currently supports a maximum of 26 EI network devices (EIA through EIZ). This limit is imposed by the OpenVMS device-naming scheme.

In addition to this OpenVMS limit, hypervisors may impose their own limits, which can vary depending on the network adapter type selected. If a virtual machine is configured with more adapters than are supported by the hypervisor, some adapters may be configured as offline or may not operate correctly.

To verify the network adapters configured by OpenVMS, issue the following command:

```
$ MCR LANCP SHOW CONFIGURATION
```

To verify the adapters detected by the system bus scan, issue the following commands:

```
$ ANALYZE/SYSTEM  
SDA> CLUE CONFIG/ADAPTER
```

If the required network configuration cannot be achieved, consider using a different network adapter type or contact VSI Support for assistance.

### 2.3.10. OpenLDAP Template Files May Generate PCSI-I-RETAIN Messages

In rare circumstances, PCSI may report PCSI-I-RETAIN messages for certain OpenLDAP template files during installation or upgrade operations.

These messages can occur because OpenLDAP currently ships certain template files using the same names as files previously provided by legacy LDAP components. This naming convention was retained to simplify migration between the legacy ACMELDAP agent and the OpenLDAP ACME agent.

The messages are informational and do not indicate a problem with the installation.

### 2.3.11. Secure Boot Overrides IMAGE\_SECURITY Settings

When Secure Boot is enabled, OpenVMS currently enforces image-signing verification behavior equivalent to IMAGE\_SECURITY=2, regardless of the configured value of the IMAGE\_SECURITY parameter.

As a result, systems configured with IMAGE\_SECURITY=3 do not receive the additional enforcement associated with that setting. In addition, administrators cannot temporarily reduce image-signing enforcement below level 2 while Secure Boot remains enabled.

This limitation will be addressed in a future release.

### 2.3.12. SET PROCESS/DUMP=NOW Fails to Generate Process Dumps

The **SET PROCESS/ID=*process-id*/DUMP=NOW** command will fail when used to generate a process dump for another process.

The target process reports an error similar to the following:

```
%SYSTEM-F-ACCVIO, access violation
```

This issue applies to the E9.2-4 field test release and will be fixed in VSI OpenVMS x86-64 V9.2-4.

### 2.3.13. Virtio-NET Transitional Device Limit on KVM

When using Virtio-NET devices on KVM, network adapters may be configured as either "virtio-net-pci-transitional" or "virtio-net-pci-non-transitional". The OpenVMS LAN driver treats both device types identically.

However, KVM limits the number of usable "virtio-net-pci-transitional" devices to nine. If more than nine transitional Virtio-NET devices are configured, the additional devices may fail initialization and remain offline. This limitation does not apply to "virtio-net-pci-non-transitional" devices.

Note that OpenVMS currently supports a maximum of 26 EI devices (EIA through EIZ). These devices may be any combination of e1000, e1000e, virtio-net-pci-transitional, and virtio-net-pci-non-transitional.

When configuring network adapters with tools such as **virt-manager** or **Cockpit**, the **virtio** option may map to either the transitional or non-transitional device type, depending on the Linux distribution and KVM version. If necessary, use:

```
virsh edit vm-name
```

to verify or modify the NIC model type.

For additional information, refer to the [VSI OpenVMS on KVM Configuration Guide \[https://howto.vmssoftware.com/KVM/index.html\]](https://howto.vmssoftware.com/KVM/index.html).

As a workaround, VSI recommends using virtio-net-pci-non-transitional devices when configuring large numbers of Virtio-NET interfaces.

### 2.3.14. Virtio-SCSI Boot Manager Device Limit on KVM

When using Virtio-SCSI storage devices on KVM systems that boot through OVMF firmware, the underlying UEFI firmware supports a maximum of eight logical units (LUNs) per Virtio-SCSI controller.

As a result, devices assigned to LUN numbers greater than 7 may not be visible to the OpenVMS Boot Manager and therefore cannot be selected as boot devices. This limitation originates in the OVMF firmware implementation and is not specific to OpenVMS or the OpenVMS Boot Manager.

As a workaround, VSI recommends configuring additional Virtio-SCSI controllers and distributing disks across multiple controllers.

### 2.3.15. VSI Performance Management Fails if IMAGE\_SECURITY is Higher Than 1

VSI Performance Management V3.1-1 will not function correctly on VSI OpenVMS x86-64 E9.2-4 when Secure Boot is enabled or the IMAGE\_SECURITY parameter is set to a value higher than 1.

To resolve this issue, install the X86VMS-PERFMGMT-V0301-1-ECO1 ECO kit before enabling Secure Boot or increasing the IMAGE\_SECURITY setting. The ECO kit is available [here \[https://sp.vmssoftware.com/packages/1812\]](https://sp.vmssoftware.com/packages/1812).

## 3. 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/].

### 3.1. Features Not Available in VSI OpenVMS x86-64 E9.2-4

The following functionalities, products, and commands are *not* available in VSI OpenVMS x86-64 E9.2-4:

- Process swapping
- 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

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

A **BACKUP** command of the form:

```
$ BACKUP/INITIALIZE input-disk:[directory-path...]*.*; output-disk:save-set.BCK/SAVE_SET
```

where the output volume is a disk that has been mounted **/FOREIGN**, does *not* work in VSI OpenVMS x86-64 E9.2-4 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:[directory-path...]*.*; output-disk:save-set.BCK/SAVE_SET
```

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.

### 3.3. 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.

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

In VSI OpenVMS x86-64, 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 IA-64 systems using a command with the following syntax:

```
$ LD CONNECT/ALLOCLASS=1/SHARE device-name:[directory-path]file-name.DSK LDA5:
```

2. Once all required connections on OpenVMS Alpha and/or OpenVMS IA-64 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 Alpha and/or OpenVMS IA-64 system will fail with an error message similar to the following:

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

This limitation will be addressed in a future update for OpenVMS Alpha and OpenVMS IA-64 systems.

### 3.5. DECnet MAC Address Change May Block Network Traffic

If DECnet is installed, network traffic may be blocked after DECnet changes the MAC address of a network interface. This can affect all network traffic, including TCP/IP.

Many hypervisors block traffic when a virtual machine begins transmitting with a MAC address that differs from the address configured for the virtual network interface.

If this issue occurs, consider one of the following actions:

- Configure the hypervisor to allow MAC address changes.
- Configure the virtual machine to use the DECnet-derived MAC address.
- Disable Phase IV-compatible addressing in DECnet-Plus.
- Disable DECnet if DECnet services are not required.

For information about calculating DECnet MAC addresses and configuring DECnet-Plus, refer to the relevant section in the *VSI DECnet-Plus for OpenVMS Installation and Configuration Manual* [[https://docs.vmssoftware.com/vsi-openvms-decnet-plus-installation-and-configuration/#BASIC\\_CONFIG\\_215](https://docs.vmssoftware.com/vsi-openvms-decnet-plus-installation-and-configuration/#BASIC_CONFIG_215)].

## 3.6. 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. For information on DECnet over TCP/IP, refer to the relevant section in the *VSI DECnet-Plus for OpenVMS Network Management Guide* [[https://docs.vmssoftware.com/vsi-decnet-plus-network-management-guide/#section\\_chb\\_vjr\\_gpb](https://docs.vmssoftware.com/vsi-decnet-plus-network-management-guide/#section_chb_vjr_gpb)].

## 3.7. 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.

## 3.8. LAN Failover MAC Address Change May Block Network Traffic

When a LAN Failover device is created and enabled, the LAN Failover driver changes the MAC address by replacing the first four bytes with "AA-00-04-00" while preserving the final two bytes of the original MAC address.

Some hypervisors block network traffic when a virtual machine transmits using a MAC address different from the one configured for the virtual network interface. As a result, network traffic to and from a LAN Failover device may be blocked.

If this issue occurs, configure the hypervisor to allow MAC address changes. For instructions on how to do this, refer to the hypervisor documentation. Typically, this involves enabling settings such as **Promiscuous Mode**, **MAC Address Spoofing**, **MAC Address Changes**, or **Forged Transmits** on the virtual network interface, a virtual switch, or both. The required settings vary between hypervisors and may also differ between versions of the same hypervisor.

## 3.9. MACRO-32 Unaligned Stack Issue

Some MACRO-32 calls leave the stack unaligned. Usually, this does not cause any issues. However, if the called routine (usually from C) attempts to save floating registers on the stack, the instruction requires an aligned stack. This results in a hardware fault.

The suggested workaround for this issue is to save the argument pointer or stack pointer to a temporary register and then use that during the argument push sequence.

## 3.10. Possible Issues with BOOT\_OPTIONS.COM

To ensure that boot options are applied and recorded consistently, VSI strongly recommends setting up and managing firmware boot options using the built-in platform firmware utilities, as opposed to the SY\$MANAGER:BOOT\_OPTIONS.COM procedure.

## 3.11. 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.

## 3.12. 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.

### 3.12.1. Supported Registers

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

**Table 1. Supported registers**

|                         |                                                                                                                      |
|-------------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>64-bit registers</b> | %rax, %rbx, %rcx, %rdx, %rsi, %rdi, %rsp, %rbp, %r8, %r9, %r10, %r11, %r12, %r13, %r14, %r15                         |
| <b>32-bit registers</b> | %eax, %ebx, %ecx, %edx, %esi, %edi, %esp, %ebp, %r8d, %r9d, %r10d, %r11d, %r12d, %r13d, %r14d, %r15d                 |
| <b>16-bit registers</b> | %ax, %bx, %cx, %dx, %si, %di, %sp, %bp, %r8w, %r9w, %r10w, %r11w, %r12w, %r13w, %r14w, %r15w                         |
| <b>8-bit registers</b>  | %al, %ah, %bl, %bh, %cl, %ch, %dl, %dh, %sil, %dil, %spl, %bpl, %r8b, %r9b, %r10b, %r11b, %r12b, %r13b, %r14b, %r15b |

### 3.12.2. 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.

### 3.12.3. 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 a notable 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.

## 3.13. Viewing Call Stack in Pthread Debugger

The Pthread debugger on VSI OpenVMS x86-64 supports viewing thread call stacks. 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
```

## 3.14. 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) [[https://docs.vmssoftware.com/vsi-tcpip-services-for-openvms-user-s-guide-60/#FTP\\_COMMANDS\\_SEC](https://docs.vmssoftware.com/vsi-tcpip-services-for-openvms-user-s-guide-60/#FTP_COMMANDS_SEC)].

## 4. Platform-Specific Notes

This section contains implementation details, architectural information, and operational considerations specific to VSI OpenVMS x86-64.

### 4.1. Memory Disks

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 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, SYSS\$UPDATE:SYSS\$MD.COM, keeps the Memory Disk up-to-date.

If any changes are made that affect the boot path or dump processing, the SYSS\$MD.COM command procedure must be run before rebooting. This includes modifications to files referenced or loaded during the boot process (up to and including activation of STARTUP) and files used by the dump kernel.

However, there are two exceptions to the above requirement. If you take either of the following actions that affect the boot path or dump processing, you do not need to run SYSS\$MD.COM:

- Use either of the following commands:

```
$ MCR SYSGEN WRITE CURRENT
$ MCR SYSMAN PARAMETERS WRITE CURRENT
```

These commands access the parameter file on the Memory Disk directly.

- Copy a file directly to the Memory Disk when specifically instructed to do so by VSI Support.

Use the following command exactly as specified:

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

After SYSS\$MD.COM completes, you must reboot the system.

Running SYSS\$MD.COM produces output similar to the following:

```
$ @SYSS$UPDATE:SYSS$MD.COM
Updating the system memory disk. Please do not interrupt this procedure...
```

```
Created memory disk $60$DKD1:[VMS$COMMON.SYSS$LDR]SYSS$MD.DSK;8
- using 171216 blocks in 6 extents with 18965 spare blocks
- mounted on $60$LDM9823: with volume label MD26228A50FH
```

- contains OpenVMS E9.2-4

After the next system reboot, purge obsolete versions of the Memory Disk using the following command:

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

---

## Notes

- Do not invoke SYS\$MD.COM unless you are following documented procedures or have been instructed to do so by VSI Support. For example, if you load a user-written execut by running SYS\$UPDATE:VMS\$SYSTEM\_IMAGES.COM, you must then invoke SYS\$UPDATE:SYS\$MD.COM.
- VSI does not recommend changing, moving, or otherwise manipulating SYS\$MD.DSK or SYS\$EFL.SYS (or the underlying EFI partition) in any way. These components are required to boot the system and are maintained by OpenVMS.

---

## 4.2. 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**, and **MAP**. Support for the Symbolic Debugger interface will be added in a future release of VSI OpenVMS x86-64.

## 4.3. 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 4.1, "Memory Disks"*.

### Interleaved Dumps

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 backed by Solid State Disk (SSD) storage, the dump can be written much faster, allowing the system to be rebooted sooner.

The DUMPSTYLE parameter specifies which dump type is used. Unless overridden in SYS\$SYSTEM:MODPARAMS.DAT, AUTOGEN selects a default value based on the number of CPUs in the system: 9 (Compressed Selective Dump) for single-CPU systems and 128 (Interleaved Dump) for systems with more than one CPU.

In OpenVMS x86-64, the only additional DUMPSTYLE bits that affect dump content are bit 1 (value 2), which enables full console output (registers, stack, and system image layout), and bit 5 (value 32), which limits the dump to key processes and global pages. Either or both of these values may be added to the default DUMPSTYLE values (9 for Compressed Selective Dump and 128 for Interleaved Dump). Bit 2 (value 4), which historically controlled Dump Off System Disk (DOSD) operation, is ignored on x86-64. Use the **SET DUMP/DEVICE** command to specify an alternate dump device.

## 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:[SYS*n*.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.EXE
SYSGEN> CREATE DKA100:[SYS0.SYSEXE]SYSDUMP.DMP /SIZE=16777216
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.

For additional information, refer to the [VSI OpenVMS x86-64 Boot Manager User Guide \[https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide/\]](https://docs.vmssoftware.com/vsi-openvms-x86-64-boot-manager-user-guide/).

## 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 SYS\$SYSTEM:SDA.EXE and SYS\$LIBRARY: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, **SYS\$LOADABLE\_IMAGES: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.

## 4.4. User-Written x86-Assembler Modules

User-written x86-Assembler code must follow the VSI OpenVMS calling standard (see the [VSI OpenVMS Calling Standard Manual \[https://docs.vmssoftware.com/vsi-openvms-calling-standard/#X86\\_64\\_CONVENTIONS\\_CH1\]](https://docs.vmssoftware.com/vsi-openvms-calling-standard/#X86_64_CONVENTIONS_CH1)) 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. Products Included With This Release

The following products are shipped by default with VSI OpenVMS x86-64 E9.2-4:

- VSI Availability Manager
- VSI DECnet Phase IV E9.2-4
- VSI DECnet-Plus V9.2-I
- VSI DECwindows Motif V1.8-2
- VSI Kerberos V3.3-5A
- VSI OpenLDAP V2.6-6B
- VSI OpenSSH V9.9-2C
- VSI OSAK V3.1-2
- VMS PERL V5.42-0
- VSI SSL3 V3.0-21
- VSI TCP/IP Services V6.0-31

Layered and 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/).

For information about the latest versions of open-source products for VSI OpenVMS x86-64, refer to the [VMS Software website \[https://products.vmssoftware.com/?platform=x86&open\\_source=Yes\]](https://products.vmssoftware.com/?platform=x86&open_source=Yes).

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

### 5.1. ZIP/UNZIP Tools

VSI provides freeware utilities for managing ZIP archives on OpenVMS x86-64 systems. In VSI OpenVMS x86-64 E9.2-4, the installation procedure automatically puts these files in the following directories on the system disk:

| Directory                              | Files                                                                           |
|----------------------------------------|---------------------------------------------------------------------------------|
| SYS\$COMMON:[SYSHLP.UNSUPPORTED.UNZIP] | UNZIP.EXE<br>UNZIPSFY.EXE<br>UNZIPSFY_CLI.EXE<br>UNZIP_CLI.EXE<br>UNZIP_MSG.EXE |
| SYS\$COMMON:[SYSHLP.UNSUPPORTED.ZIP]   | ZIP.EXE<br>ZIPCLOAK.EXE<br>ZIPNOTE.EXE<br>ZIPSPLIT.EXE                          |

| Directory | Files                      |
|-----------|----------------------------|
|           | ZIP_CLI.EXE<br>ZIP_MSG.EXE |