VMS Software

VSI OpenVMS x86-64 V9.2 Release Notes

Document Number: DO-RLNV92-01A

Publication Date: July 2022

Operating System and Version: VSI OpenVMS x86-64 V9.2

VMS Software, Inc. (VSI)
Burlington, Massachusetts, USA

VSI OpenVMS x86-64 V9.2 Release Notes

VMS Software

Copyright © 2022 VMS Software, Inc. (VSI), Burlington, Massachusetts, USA

Legal Notice

Preface

1. Introduction

VMS Software, Inc. (VSI) is pleased to introduce VSI OpenVMS x86-64 V9.2.

2. Intended Audience

This document is intended for all users of VSI OpenVMS x86-64 V9.2. Read this document before you install or use VSI OpenVMS x86-64 V9.2.

3. Document Structure

This document contains the following sections:

- Before You Start... Read These First

- Release Notes

- VSI C Run-Time Library (C RTL) Notes

- Security Enhancements for VSI TCP/IP Services X6.0-16 FTPS

4. Related Documents

The following documents provide additional information in support of this release. They are included in the **V92_DOCS.zip** file that is available for download from the VSI Services Portal.

- *VSI OpenVMS x86-64 V9.2 Installation Guide*

- *VSI OpenVMS x86-64 Boot Manager User Guide*

- *VSI OpenVMS Calling Standard Manual*

- *VSI OpenVMS Linker Utility Manual*

- *VSI OpenVMS x86-64 Cross-Tools Kit Installation and Startup Guide*

Chapter 1. Before You Start... Read These First

Before you download the VSI OpenVMS x86-64 V9.2 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. The notes also describe the new boot method called MemoryDisk, and available networking options.

1.1. Supported Disk Types

VSI OpenVMS x86-64 V9.2 supports SATA disks.

An issue with OpenVMS x86-64 V9.2 prevents support for LSI Logic Parallel SCSI controllers on ESXi. A fix for this issue is expected to be available shortly after the release of VSI OpenVMS x86-64 V9.2.

Support for other disk types will be added in future releases of VSI OpenVMS x86-64.

1.2. Tested Platforms

VSI OpenVMS x86-64 V9.2 can be installed as a guest operating system on Oracle VM VirtualBox, KVM, and VMware virtual machines using the **X86092OE.ISO** file.

VirtualBox

VSI tests with VirtualBox V6.1 and regularly installs updates when they are available.

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 V9.2 with KVM on several Linux distributions. The following table includes the Linux distribution, version, and the QEMU version:

Linux Distribution	QEMU Version (package information)
CentOS Linux Release 7.9.2009 (Core)	2.12.0 (qemu-kvm-ev-2.12.0-44.1.el7_8.1)
Linux Mint 20.2	4.2.1 (Debian 1:4.2-3ubuntu6.21)
OpenSUSE Leap 15.3	5.2.0 (qemu-5.2.0-150300.112.4.src)
Rocky Linux 8.5	4.2.0 (qemu-kvm-4.2.0-59.module+el8.5.0+726+ce09ee88.1)
Ubuntu 18.04 LTS	2.11.1 (Debian 1:2.11+dfsg-1ubuntu7.39)
Ubuntu 20.04.4 LTS	4.2.1 (Debian 1:4.2-3ubuntu6.23)

VMware

VSI has tested VSI OpenVMS x86-64 V9.2 with the following VMware products:

VMware Product	Version Tested by VSI
Workstation Pro	V15.5.7
Workstation Player	V16.1.0
Fusion Pro	V11.5.7
Fusion Player	V12.2.3
ESXi	V6.7.0, V7.0.2

Important

Note that *not* all VMware license types are currently supported for running VSI OpenVMS x86-64 V9.2. The following table lists VMware license types that have been tested by VSI.

VMware License	VSI Tested
Enterprise Plus	ESXi V6.7.0 and V7.0.2 as part of vSphere Enterprise Plus
Enterprise	Not tested
Essentials Plus	Not currently supported
Essentials	Not currently supported
Standard	Not currently supported
Hypervisor	Not currently supported

The VMware licenses that are marked as "Not currently supported" do not support the use of virtual serial lines in a virtual machine. OpenVMS requires a serial port connection with a terminal emulator and therefore VMware systems with these licenses are not currently supported for running OpenVMS. This support will be added in a future release of VSI OpenVMS x86-64.

1.3. MD5 Checksum for the X86092OE.ZIP File

VSI recommends that you verify the MD5 checksum of the **X86092OE.ZIP** file after it has been downloaded from the VMS Software Service Platform to your target system. The MD5 checksum of **X86092OE.ZIP** must correspond to the following value:

27AA8FC61E82F16BEFF98253DCB9F310

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

1.4. Non-Intel Processors Are Not Currently Supported

VSI OpenVMS x86-64 V9.2 runs on Intel processors only.

Support for AMD processors is planned for a future version of VSI OpenVMS x86-64.

1.5. 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 V9.2 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 OpenVMS V9.2 web page [https://vmssoftware.com/about/openvmsx86/].

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 V9.2, 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.

1.6. Terminal Emulator Settings

When you are in the Boot Manager and before proceeding with the installation of OpenVMS x86-64, you are required to access the system through a serial port connection with a terminal emulator such as PuTTY. You may need to experiment in order to find the appropriate setting for your emulator.

Refer to the *OpenVMS x86-64 Boot Manager User Guide* for more details on 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.

1.7. MemoryDisk and the Command Procedure SYS$MD.COM

VSI OpenVMS x86-64 uses a new boot method called MemoryDisk that simplifies the boot process by eliminating boot complexity and decoupling the operating system Loader (the Boot Manager) from a specific device or version of VSI OpenVMS x86-64. VSI provides a pre-packaged MemoryDisk container file (SYS$MD.DSK) on the distribution kit and on every bootable OpenVMS system device.

The MemoryDisk 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 or patch installations require the operating system to execute a procedure to update the MemoryDisk container, thus assuring that the next boot will use the new images. A command procedure, SYS$MD.COM, keeps the MemoryDisk 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 2.1.14 of this document.

Note

Do not rename or move SYS$MD.DSK or SYS$EFI.SYS (the UEFI partition). Doing so will invalidate the boot blocks and render the system unbootable.

1.8. x86-64 Licensing

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

A PAK is represented as a text structure containing a series of named fields and unique values that were generated by VSI. You have the option of deferring PAK entry until after the installation and entering them using the OpenVMS LICENSE utility. If you choose to enter your PAKs during the installation, you can either type the values of each requested field, or copy-and-paste the values into the console (assuming your console connection supports this action, which terminal emulators do).

Below is an example of a typical PAK:

```
$ LICENSE REGISTER OPENVMS-X86-HAOE -
/ISSUER=VSI -
/AUTHORIZATION=1-VSI-20220608-00000 -
/PRODUCER=VSI -
/UNITS=1 -
/TERMINATION_DATE=31-OCT-2022 -
/OPTIONS=(PCL,X86_64) -
/CHECKSUM=2-XXXX-XXXX-XXXX-XXXX
```

1.8.1. Using a VMware vSphere Hypervisor Basic License

Use this procedure to run VMware vSphere Hypervisor on an ESXi server with a basic license (not Enterprise or Enterprise Plus). You must use serial ports within the same ESX server.
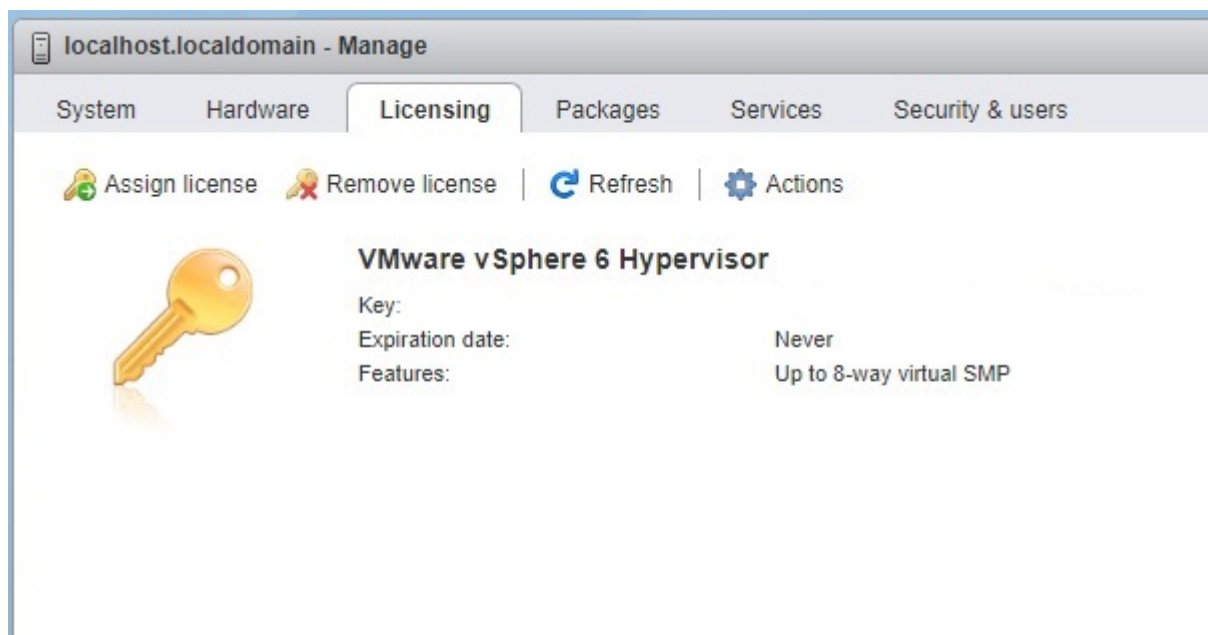
Using the named pipe functionality, map COM1/OPA0: on the VMS virtual machine to a pipe on a management server on which a terminal emulator is installed.

With the VM system in client mode, use the following syntax: `\\.\pipe_xxx`

With the management system in server mode, use the following syntax: `\\.\pipe_xxx`, where `xxx` is a unique string for each VM.

The terminal emulator should be set for serial connection at 115200 baud.

The two figures below show how to set up pipes for a local terminal-emulator-based console.
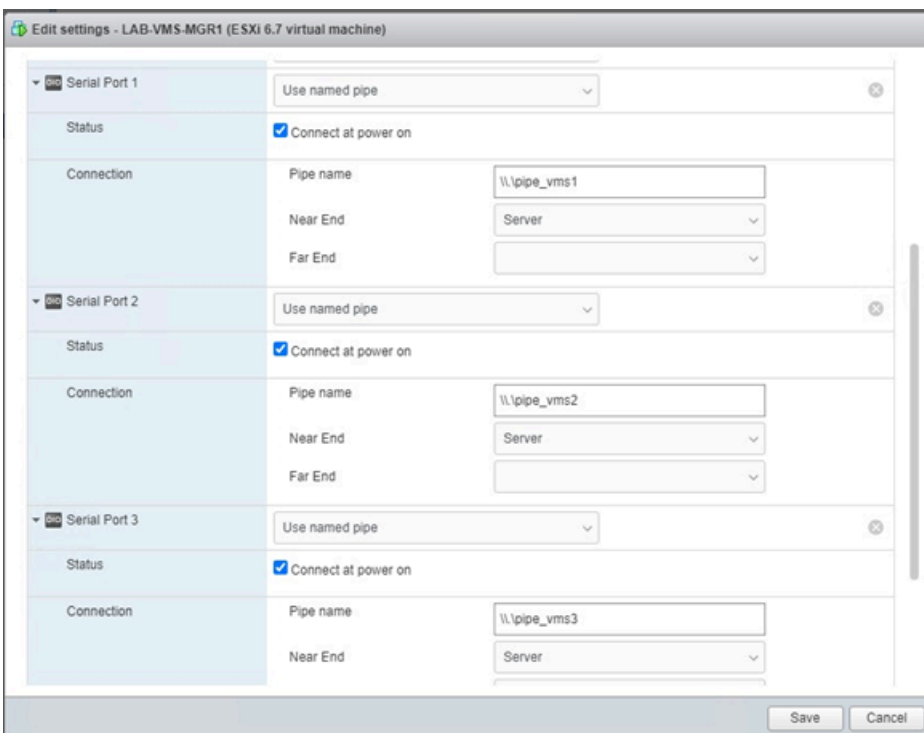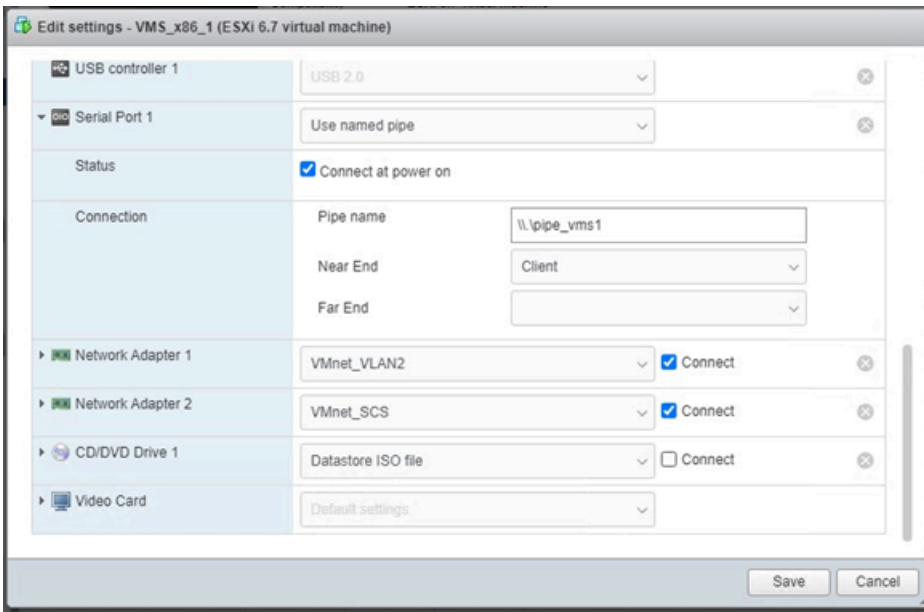
The two figures below show how to set up pipes between two local virtual machines where one plays the role of VMS console. This could be a virtual machine guest running any OS that supports a terminal emulator.
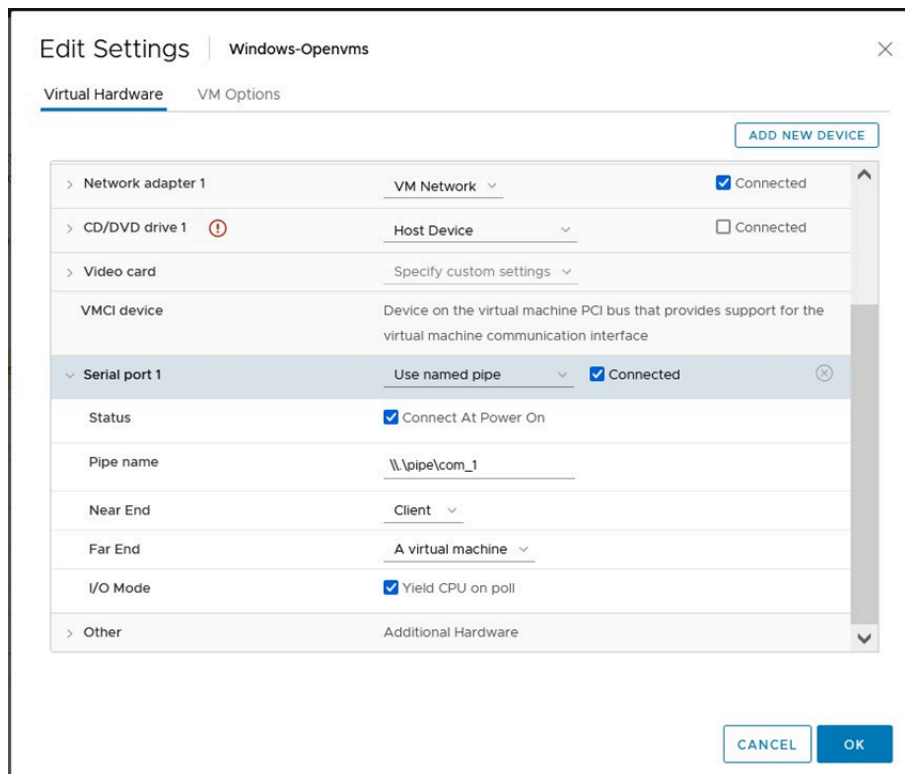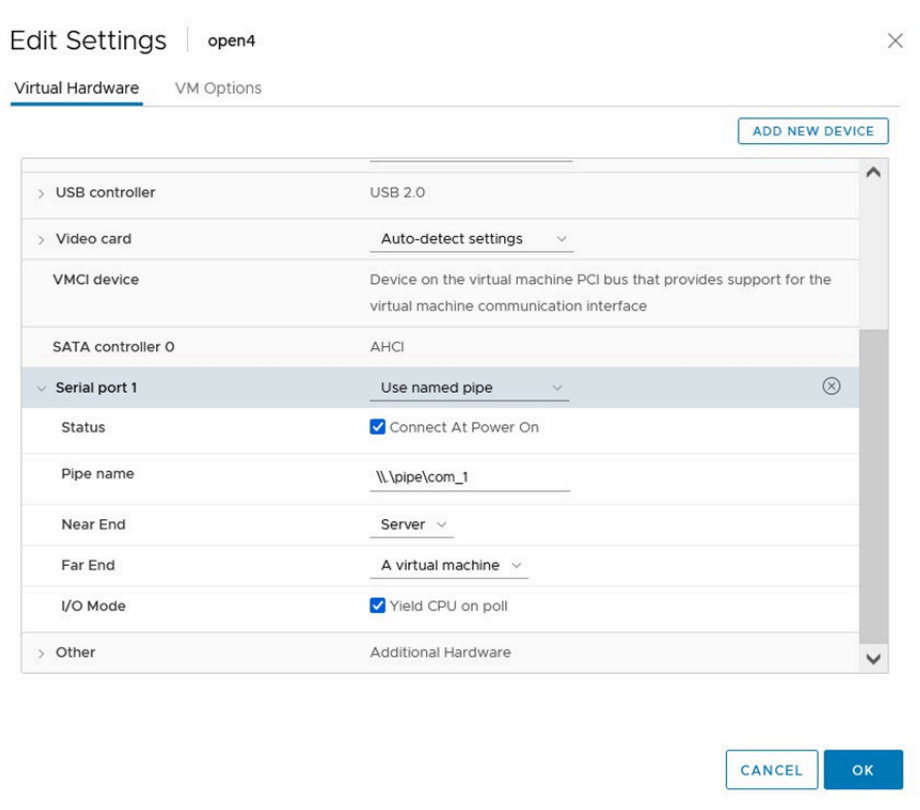
1.9. Networking Options

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

- VSI TCP/IP Services

- VSI DECnet Phase IV

- VSI DECnet-Plus

VSI TCP/IP Services X6.0-16 is part of the OpenVMS x86-64 V9.2 installation and will be installed along with the operating system.

In order to use DECnet, you have to choose to install either VSI DECnet Phase IV or VSI DECnet-Plus. Note that both of the DECnet products *cannot* be installed on your system.

1.9.1. TCP/IP Services Uses BIND 9.11.37 Server

The current version of VSI TCP/IP Services for OpenVMS uses the BIND 9.11.37 Server.

Using Bind 9.11.37 is documented in the *VSI TCP/IP Services for OpenVMS Management* [https:// docs.vmssoftware.com/vsi-tcpip-services-for-openvms-management/] manual. VSI also recommends that users refer to the *Internet Systems Consortium (ISC) website* [https://www.isc.org/bind/] for the latest updates to Bind 9 configurations and resources.

1.9.2. VSI DECnet

Install either VSI DECnet Phase IV or VSI DECnet-Plus on VSI OpenVMS x86-64 V9.2 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.

1.9.3. Empty File for DECnet-Plus

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

1.9.4. VSI TCP/IP Services X6.0-16

VSI OpenVMS x86-64 V9.2 includes VSI TCP/IP Services X6.0-16. The following services are available in X6.0-16. Note that many of these services are yet to be qualified by VSI, however based on the testing done so far, they are ready for initial external testing. If you encounter any issues using VSI TCP/IP Services X6.0-16, please report them to VSI via the VMS Software Service Platform.

- BIND

- FTP

- FTPS

- Finger

- FailSafe IP

- LBROKER

- LPR/LPD

- NFS

- NTP4

- POP

- Remote (R) Commands

- SMTP

- SNMP

- Socket API

- TELNET (except Kerberos authentication)

With VSI TCP/IP Services X6.0-16, VSI introduces security enhancements for FTPS (FTP over SSL). For details, refer to Appendix B of this document.

Before starting VSI TCP/IP Services, you must run the TCPIP$CONFIG configuration procedure. To start TCPIP$CONFIG, enter the following command:

```
$ @SYS$MANAGER:TCPIP$CONFIG.COM
```

To start the network stack after configuring it, enter the following command:

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

For detailed information on running the TCPIP$CONFIG configuration procedure, refer to the *VSI TCP/IP Services for OpenVMS Installation and Configuration* [https://vmssoftware.com/docs/VSI_TCPIP_SERVICES_INSTALL_CONFIG.pdf] manual.

For a configuration example for FTP and TELNET, refer to the *VSI OpenVMS x86-64 V9.2 Installation Guide*.

Note

If FTP 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 SYS$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://vmssoftware.com/docs/VSI_TCPIP_SERVICES_UG.pdf].

1.9.5. 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.

1.10. VSI SSL111 V1.1-1N for OpenVMS

VSI OpenVMS x86-64 V9.2 includes VSI SSL111 V1.1-1N for OpenVMS that is based on OpenSSL 1.1.1n.

OpenSSL is used by many operating system functions, networking products, OpenVMS layered products and open source applications.

1.11. VSI OpenSSH V8.9 for OpenVMS

VSI has ported OpenSSH V8.9-1B to OpenVMS. In this release, OpenSSH has been integrated into the VSI OpenVMS x86-64 V9.2 kit as a required layered product, that will be installed unconditionally with the OS. For post-installation and configuration instructions, refer to the *VSI OpenVMS x86-64 V9.2 Installation Guide*.

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

1.12. VSI Kerberos V3.3-2 for OpenVMS

VSI OpenVMS x86-64 V9.2 includes VSI Kerberos V3.3-2 for OpenVMS.

1.13. VSI DECwindows Motif V1.8 for OpenVMS

VSI OpenVMS x86-64 V9.2 includes VSI DECwindows Motif V1.8. VSI DECwindows Motif V1.8 is a fully working DECwindows Motif kit that replaces V1.7-X, which was previously used as a placeholder to allow building DECwindows objects against it.

1.14. Required Layered Products

The following layered products will be installed unconditionally with VSI OpenVMS x86-64 V9.2:

- VSI TCP/IP Services

- VSI Kerberos

- VSI SSL111

- VSI OpenSSH

- VMSPORTS x86VMS PERL534 T5.34-0

Chapter 2. Release Notes

2.1. Operating System Notes

The notes in this section announce support for new functionality and also describe known issues and limitations in VSI OpenVMS x86-64 V9.2.

2.1.1. New Features

The notes in this section describe features that are new in VSI OpenVMS x86-64 V9.2 compared to VSI OpenVMS x86-64 E9.2 (the field test version).

2.1.1.1. Contiguous Best Try Qualifier for SET FILE/ATTRIBUTES

The SET FILE/ATTRIBUTES command now 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
```

2.1.1.2. /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
/LARGEST	/LARGEST[=n]	Displays a list of block counts for the n largest extents of free space on the volume in descending size order. The default for n is 10. The qualifier is ignored if you specify zero or a negative number. The list is also saved in the DCL symbol ANALYZE$LARGEST_EXTENTS (as a comma-separated list of decimal values). The symbol is set to an empty string if there is no free space on the disk. There is no upper limit on n, but if the DCL symbol exceeds 1024 characters, the number of

Qualifier	Syntax	Description
		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[=*filespec*] /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, SYS$OUTPUT will be used. The default filename is ANALYZE$EXTENTS.LIS.
/REQUIRED	/REQUIRED=n	Displays the number of extents required to satisfy an allocation request of *n* blocks (starting with the largest extent). There is no default for *n*. 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.
```

2.1.1.3. /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              Platform    Installed
    Destination: DISK$SYSTEM_DISK:[VMS$COMMON.]
VSI X86VMS VMS V9.2                  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.

2.1.1.4. 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 CHECKSUM command help or the *VSI OpenVMS DCL Dictionary: A-M* [https://vmssoftware.com/docs/VSI_DCL_DICT_VOL_I.pdf].

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

VSI OpenVMS x86-64 V9.2 includes the 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.

See Appendix A of this document for more detailed information.

2.1.1.6. ZIP/UNZIP Tools

VSI provides the Freeware executables for managing ZIP archives on OpenVMS x86-64 systems. In VSI OpenVMS x86-64 V9.2, the installation procedure automatically puts these files in the following directories on the system disk:

Files	Folder
UNZIP.EXE	SYS$COMMON:[SYSHLP.UNSUPPORTED.UNZIP]
UNZIPSFX.EXE	
UNZIPSFX_CLI.EXE	
UNZIP_CLI.EXE	
UNZIP_MSG.EXE	
ZIP.EXE	SYS$COMMON:[SYSHLP.UNSUPPORTED.ZIP]

Files	Folder
ZIPCLOAK.EXE	
ZIPNOTE.EXE	
ZIPSPLIT.EXE	
ZIP_CLI.EXE	
ZIP_MSG.EXE	

2.1.1.7. Entropy

VSI OpenVMS x86-64 V9.2 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. New features associated with entropy collection include a new SYSGEN parameter named RANDOM_SOURCES and a new system service named SYS$GET_ENTROPY.

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. At boot time, the system tests for hardware support and hypervisor support of the RDRAND instruction, and will display a message giving the state of the RDRAND support for the boot.

2.1.1.7.1. New SYSGEN Parameter RANDOM_SOURCES

The new SYSGEN parameter RANDOM_SOURCES 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 desciption of RANDOM_SOURCES, execute the command `HELP SYS_PARAMETERS RANDOM_SOURCES` in the SYSGEN utility.

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 changes *do not* require a reboot.

2.1.1.7.2. New System Service SYS$GET_ENTROPY

SYS$GET_ENTROPY

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

Format

```
SYS$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 SYS$GET_ENTROPY service retrieves bytes of data from the system entropy pool, writing 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.

2.1.2. Features Not Available in VSI OpenVMS x86-64 V9.2

The following functionalities, products, and commands are *not* available in VSI OpenVMS x86-64 V9.2:

- ACME_SERVER (only UAF Login is available)

- Availability Manager

- DECdtm Services

- Process swapping (see Section 2.1.16 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

2.1.3. Additional Prompt During OpenVMS x86-64 Installation

During the installation of OpenVMS x86-64 V9.2, 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 X6.0-16 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 with the default response (YES), otherwise the installation will be terminated.

Later in the installation, you will see the following messages, which may be safely ignored:

```
%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 the system reboot message twice.

VSI will address this issue in a future release.

2.1.4. AUTOGEN Warning That Appears During AUTOGEN Boot May Be Safely Ignored

AUTOGEN issues a warning message during the AUTOGEN Boot portion of the OpenVMS system installation. This message may be safely ignored. The problem will be fixed in a future release of VSI OpenVMS x86-64.

```
******************
%AUTOGEN-W-REPORT, Warnings were detected by AUTOGEN. Please review the
         information given in the file SYS$SYSTEM:AGEN$PARAMS.REPORT
******************
```

2.1.5. 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 V9.2 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://vmssoftware.com/docs/VSI_SYS_MGMT_MANUAL_VOL_I.PDF].

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 INITIALIZE and MOUNT/BIND commands.

2.1.6. Cross-Tools Kit Update

With VSI OpenVMS x86-64 V9.2, use the new **V9.2-XG9N** cross-tools kit (VSI-I64VMS-X86_XTOOLS-V0902-XG9N-1.ZIP).

For details on the changes to the cross-tools in the **V9.2-XG9N** kit, refer to the release notes bundled with the kit. Before the cross-tools kit installation, the release notes file can be extracted from the PCSI kit with the following command:

```
$ product extract release_notes x86_xtools /log [/source=ddcu:[dir]]
```

Refer to the *VSI OpenVMS x86-64 Cross-Tools Kit Installation and Startup Guide* for complete information on installing the cross-tools kit.

2.1.7. Display of License Charge Information for x86-64 Nodes

In a cluster with x86-64 nodes running VSI OpenVMS x86-64 V9.2 and Alpha or Integrity nodes running previous versions of OpenVMS, the SHOW LICENSE/CLUSTER/CHARGE command, run from a non-x86-64 node, displays the existing x86-64 nodes but does *not* display the license charge information for x86-64 systems.

This issue will be fixed in a future update for previous VSI OpenVMS versions.

2.1.8. 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.

2.1.9. Extended File Cache (XFC)

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

2.1.10. 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
```

2.1.11. New 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, which is 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$INITIALIZE module were declared. This change also does not affect any existing link commands, which explicitly include the LIB$INITIALIZE module from STARLET.OLB.

2.1.12. Linker: New Informational Messages

When the linker encounters writable code sections, with PSECT attributes set to WRT and EXE, it now 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.

2.1.13. Different Image Layout on x86-64 and IA64

When porting an application from IA64 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 IA64, the compiler may generate **short data**, which is accessed in an efficient way. The IA64 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 can not write to it.

2.1.14. 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 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 Engineers to do so.

Use the following command exactly as specified here:

```
$ @sys$update:sys$md
```

No parameters are needed, since 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

X86VMS$DKA0:[VMS$COMMON.SYS$LDR]SYS$MD.DSK;3 created (158451 blocks in 1 LBN range),
        mounted on X86VMS$LDM2: (volume label SYS$MD20133C) with 25013 free blocks,
        containing OpenVMS V9.2.

$
```

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

```
$ purge sys$loadable_images:sys$md.dsk
```

2.1.15. OpenVMS Clusters on Virtual Machines

VSI OpenVMS x86-64 V9.2 supports VirtualBox, KVM, and VMware virtual machines in OpenVMS clusters. However, shared disk access on virtual machines is not supported. Note that this means no shared data disks or cluster common system disks.

VSI OpenVMS x86-64 V9.2 has been 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
```

2.1.16. OpenVMS x86-64 Will Not Support Swap Files

VSI OpenVMS x86-64 will 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 V9.2 release, the SYSGEN INSTALL command no longer supports 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.

2.1.17. Privileged Images Linked /SYSEXE Should Be Relinked

VSI recommends that any privileged image linked /SYSEXE should be relinked for the V9.2 release because data structures and interfaces are subject to change for each new version.

2.1.18. 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.

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

When you run a VSI OpenVMS x86-64 image on VSI OpenVMS Integrity, 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.

2.1.20. Symmetric Multiprocessing (SMP)

VSI OpenVMS x86-64 V9.2 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.
$
```

2.1.21. 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://vmssoftware.com/docs/
VSI_SYS_MAN_UTIL_REF_VOL_II.pdf].

Table 2.1. 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$SYSTEM:X86_64VMSSYS.PAR.
WRITE	CURRENT	Specifies that source information is to be written to the current system parameter file on disk. The new values will take effect the next time the system is booted. On OpenVMS x86-64 systems, command modifies the current system parameter on disk, SYS$SYSTEM:X86_64VMSSYS.PAR.

Table 2.2. 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. This parameter does not apply to OpenVMS Alpha or OpenVMS Integrity systems.
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. This parameter does not apply to OpenVMS Alpha or OpenVMS Integrity systems.
DISABLE_X86_FT	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. It is used to decide which variant of the SYSTEM_PRIMITIVES execlet gets loaded. Setting all bits (disabling the use of all optional features) results in SYSTEM_PRIMITIVES_0 being loaded. The following bits are defined: <table><tr><th>Bit</th><th>Definition</th></tr><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></table> DISABLE_x86_FT is a STATIC parameter. This parameter does not apply to Alpha or Integrity systems.
GH_EXEC_CODE_S2	On x86-64 systems, GH_EXEC_CODE_S2 specifies the size in pages of the execlet code granularity hint region in S2 space. GH_EXEC_CODE_S2 has the AUTOGEN and FEEDBACK attributes. This parameter does not apply to Alpha or Integrity systems.
GH_EXEC_DATA_S2	On x86-64 systems, GH_EXEC_DATA_S2 specifies the size in pages of the execlet data granularity hint region in S2 space. GH_EXEC_DATA_S2 has the AUTOGEN and FEEDBACK parameters.

Parameter	Description
	This parameter does not apply to Alpha or Integrity systems.
GH_RES_DATA_S2	On x86-64 systems, GH_RES_DATA_S2 specifies the size in pages of the resident image data granularity hint region in S2 space. GH_RES_DATA_S2 has the AUTOGEN and FEEDBACK attributes. This parameter does not apply to Alpha or Integrity systems.
GH_RES_CODE	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. GH_RES_CODE has the AUTOGEN and FEEDBACK attributes.
GH_RO_EXEC_S0	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. GH_RO_EXEC_S0 has the AUTOGEN and FEEDBACK attributes. This parameter does not apply to Alpha or Integrity systems.
GH_RO_RES_S0	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. GH_RO_EXEC_S0 has the AUTOGEN and FEEDBACK attributes. This parameter does not apply to Alpha or Integrity systems.
LOAD_SYS_IMAGES	This special parameter is used by VSI and is subject to change. Do not change this parameter unless VSI recommends that you do so. LOAD_SYS_IMAGES controls the loading of system images described in the system image data file, VMS $SYSTEM_IMAGES. This parameter is a bit mask. The following bits are defined: <table><tr><td>**Bit**</td><td>**Description**</td></tr><tr><td>0 (SGN$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></table>

Parameter	Description
	These bits are on by default. Using conversational bootstrap exec slicing can be disabled. LOAD_SYS_IMAGES is an AUTOGEN parameter.
RAD_SUPPORT	RAD_SUPPORT enables RAD-aware code to be executed on systems that support Resource Affinity Domains (RADs). 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.
SCSBUFFCNT	SCSBUFFCNT is reserved for VSI use only. On x86-64, Alpha, and Integrity servers, the system communication services (SCS) buffers are allocated as needed, and SCSBUFFCNT is not used.
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.

The static system parameter table for VCC_FLAGS:

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 nonzero 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.

Note

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.

Parameter	Description
	VCC_FLAGS is an AUTOGEN parameter.

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.

2.1.22. System Crash Dumps

VSI OpenVMS x86-64 supports a single system crash dump of the compressed selective format. Bits 0 and 3 in the system parameter DUMPSTYLE must both be set. The value 9 is the default setting.

VSI OpenVMS x86-64 system crash dumps are written using a minimal VMS environment called the dump kernel. All the files used by the dump kernel are included in the MemoryDisk, described in Section 2.1.14 of this document.

Dump Off System Disk

Crash dumps can be written to the system disk or to an alternate disk designated for the 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 following example that specifies DKA100: as the desired dump device.

Note

On VSI OpenVMS x86-64, the *Dump off System Disk* bit of the DUMPSTYLE system parameter is no longer required.

1. Create a dump file on DKA100: using the SYSGEN utility.

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

3. Enter the command:

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

You can confirm the setting using the SHOW DUMP_OPTIONS command.

The change is effective immediately, without requiring a reboot.

System Dump Analysis

VSI strongly recommends that the version of SDA.EXE and SDA$SHARE.EXE used to analyze a system dump should be exactly the same as the version of OpenVMS in use when the system crash occurred. However, it is often possible to use SDA images from a different version of OpenVMS if there are no major differences between the versions, and ignore the warnings output by SDA (either %SDA-W-LINKTIMEMISM, or %SDA-W-SDALINKMISM, or both).

2.1.23. System Service Intercept (SSI)

System Service Intercept (SSI) is available on VSI OpenVMS x86-64. SSI enables system services to be intercepted and user-specified code to run before, after, or instead of the intercepted system service.

2.1.24. 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 Linker manual.

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 reports that the line numbers displayed are source line numbers. That is incorrect. The line numbers are in fact listing line numbers just like on OpenVMS Alpha and OpenVMS Integrity server systems.

2. Traceback is unable to determine the module name so instead it prints the "basename" of the source file used to create the module.

3. 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.

2.1.25. VAX Floating-Point Format Support

VSI OpenVMS x86-64 V9.2 includes support for VAX floating point format in most of the system-supplied libraries. However, there are some known issues that will be resolved in a future update. These include the inability to print VAX floating point values from C applications and missing support for VAX floating point in the current Fortran cross-compiler.

2.1.26. Viewing Call Stack in Pthread Debugger

In VSI OpenVMS x86-64 V9.2, 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:
     0xffff83000c83b1a5 (pc 0xffff83000c83b1a5, sp 0x00000000024e3228)
     0xffff83000c87b53a (pc 0xffff83000c87b53a, sp 0x00000000024e3230)
     0xffff83000c858493 (pc 0xffff83000c858493, sp 0x00000000024e32e0)
     0xffff83000c852b04 (pc 0xffff83000c852b04, sp 0x00000000024e33f0)
     0xffff83000c8499a3 (pc 0xffff83000c8499a3, sp 0x00000000024e34d0)
     0xffff83000c844e9a (pc 0xffff83000c844e9a, sp 0x00000000024e3800)
```

```
        0x0000000080004ad8 (pc 0x0000000080004ad8, sp 0x00000000024e3900)
        0x0000000080007fe6 (pc 0x0000000080007fe6, sp 0x00000000024e3950)
        0xffff83000c8887df (pc 0xffff83000c8887df, sp 0x00000000024e3bd0)
        0xffff83000c83b0ea (pc 0xffff83000c83b0ea, sp 0x00000000024e3f00)
```

However, the output of the Pthread debugger command `threads -o u`, which 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
SDA>
```

2.1.27. 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/].

2.1.28. Symbolic Links and POSIX Pathname Support

Symbolic links (symlinks) and POSIX pathnames are documented in Chapter 13 of 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/]. The release notes in this section augment that
documentation.

2.1.28.1. 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.

2.1.28.2. /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.

2.1.28.3. 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 nonexistent file, and then create the file by specifying the symlink as the output of the command.

2.1.28.4. 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.

2.1.28.5. 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.

2.1.28.6. 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
```

2.1.28.7. 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 even though it is operating on the target file. 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.

2.1.29. Symbolic Debugger

VSI OpenVMS x86-64 V9.2 now 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.

2.1.29.1. Supported Registers

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

Table 2.3. 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

2.1.29.2. Supported Commands

The following list details the currently supported commands specific to the x86-64 architecture. Non-architecture-specific commands, such as setting defaults, REPEAT, and others, are expected to work correctly.

- STEP [/INSTRUCTION]

- (SET,ACTIVATE,DEACTIVATE,CANCEL,SHOW)

- BREAK

- EVALUATE

- EXAMINE

- DEPOSIT

- SHOW CALLS [/IMAGE]

- SHOW IMAGE

- SHOW MODULE

- SHOW STACK [/START_LEVEL=n] [n]

- SHOW SYMBOL

2.1.29.3. Language Always Set to C

Currently, the x86-64 cross compilers 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.

2.1.29.4. 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.

2.1.29.5. Source Line Correlation

In this release, the debugger does not support source line correlation. This is because the current 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.

2.1.29.6. 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.

2.1.29.7. Not Supported or Not Working Features

Below is the list of commands and functionalities that are currently not supported or do not function correctly:

* SET WATCH will wedge the process

  Currently, SET WATCH does not work and will result in the debugger becoming unstable. This issue will be resolved in a future release.

* SET TRACE is unavaliable

* SET MODE SCREEN is not available

  SET MODE SCREEN (and associated commands) are not currently available. Executing the SET MODE SCREEN command will result in undefined behaviour that may cause the debugger to become unstable.

* STEP/INTO can result in %DEBUG-W-BADSTACK

  Using the STEP/INTO command can result in the following warning message:

  ```
  %DEBUG-W-BADSTACK, WARNING: stack corrupted; session integrity
  not guaranteed primary_handler: running + OTHERWISE 000287EB
  ```

2.1.30. Recompile and Relink Images that Include VCBDEF, FCBDEF or WCBDEF Data Structures

Additional fields have been added to these structures. Fields added to the WCB structure, in particular, are not compatible with those in previous versions of OpenVMS due to the layout of the structure. You must recompile and relink any images that access these structures to include these changes.

The file and volume major version in SYS$BASE_IMAGE.EXE has been incremented to reflect these changes.

If you activate an image that has not been recompiled and relinked, the system displays the following error message:

```
%SYSTEM-W-SYSVERDIF, system version mismatch; please relink
```

2.1.31. New Restriction on NOLOCK File Access

VSI OpenVMS I/O User's Reference Manual documents the file access control flag FIB$V_NOLOCK. This flag allows a privileged user to open a file even if another user has locked that file against all

other access. However, VSI OpenVMS x86-64 V9.2 adds the restriction that FIB$V_NOLOCK is only allowed on a read-only access basis. This means that the FIB$V_WRITE flag must be clear (along with the FIB$V_NOREAD and FIB$V_NOWRITE flags). Attempting to access a file with *both* FIB$V_NOLOCK set and FIB$V_WRITE set will result in the following error:

```
SYSTEM-F-BADPARAM, bad parameter value
```

FIB$V_NOLOCK is not used by most applications, but this restriction may affect disk maintenance utilities such as defragmenters.

2.1.32. Running DHCP Client and failSAFE IP are not Compatible on the Same NIC

You cannot run the DHCP and the failSAFE IP client on the same NIC on VSI TCP/IP Services for OpenVMS X6.x. If a customer is running the DHCP client on a NIC, then failSAFE IP should not be configured on this NIC. Since the assigned address is actually controlled by DHCP, VSI TCP/IP Services for OpenVMS should not reassign this address. If a customer needs to run the DHCP client and provide a failover mechanism, they should configure the NIC in a lan failover set.

2.1.33. ASTFLT Error when Using SFTP or SCP

When using SFTP or SCP with the default options, you may occasionally see an ASTFLT message followed by a stack dump.

The ASTFLT only happens when the progress meter is shown, which is the default. To avoid this problem, we recommend using the quiet switch (-q), so that the progress meter is not shown.

This problem will be addressed in a future version of OpenVMS.

2.1.34. User-Written x86-Assembler Modules

User-written x86-assembler code must follow the VSI OpenVMS calling standard (see the VSI OpenVMS Calling Standard manual, chapter "OpenVMS x86-64 Conventions" [https://docs.vmssoftware.com/vsi-openvms-calling-standard/#X86_64_CONVENTIONS_CH]) 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
```

2.1.35. Connecting to a Shared LD Container in a Mixed Architecture Cluster

In OpenVMS x86-64 V9.2, 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 first, using a command such as:

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

2. Once all required connections on OpenVMS Alpha and/or OpenVMS 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 Alpha and/or OpenVMS 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 Alpha and OpenVMS Integrity systems that are running VSI OpenVMS V8.4-1H1 or later, that will allow the connections to the shared container file to be performed in any order.

2.1.36. LSI Logic Controllers Are Not Supported on ESXi for VSI OpenVMS x86-64 V9.2

LSI Logic SAS controllers are not supported on ESXi due to a shared interrupt issue between one emulated LSI Logic SAS controller and another non-LSI Logic SAS controller on the PCI bus. The LSI Logic SAS controller may be supported in a future release of VSI OpenVMS.

An issue with VSI OpenVMS x86-64 V9.2 also prevents support for LSI Logic Parallel SCSI controllers on ESXi. A fix for this issue is expected to be available shortly after the release of VSI OpenVMS x86-64 V9.2. The fix is expected to allow configuration of up to four (4) LSI Logic Parallel SCSI controllers per ESXi virtual machine.

2.1.37. 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.

2.1.38. STABACKIT.COM Deprecated

SYS$UPDATE:STABACKIT.COM has been been deprecated in VSI OpenVMS x86-64 V9.2. The STABACKIT functionality originally supported Standalone Backup for VAX systems and has long been replaced by SYS$SYSTEM:AXPVMS$PCSI_INSTALL_MIN.COM for OpenVMS Alpha and SYS$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 V9.2 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.

VSI may implement minimal installation for OpenVMS x86-64 in a future release.

2.1.39. Minicopy Using HBMM Convertible Bitmaps Will Result in a System Crash on VSI OpenVMS x86-64 V9.2

Running the minicopy procedure using HBMM convertible bitmaps will result in a system crash on VSI OpenVMS x86-64 V9.2. However, running this procedure using *traditional* minicopy bitmaps will *not* result in a crash. If you want to take advantage of quicker minicopy (as opposed to full copy), please ensure that you are using *traditional* minicopy bitmaps following the dismount of a shadow set member.

To verify that HBMM convertible bitmaps are *not* in use, follow these steps:

1. Enter the following command:

    ```
    $ show device dsa2
    ```

2. Ensure that your system output looks similar to the following:

    ```
    Device                Device         Error   Volume      Free  Trans Mnt
     Name                 Status         Count    Label     Blocks Count Cnt
    DSA2:                 Mounted alloc      0 SHAD1         4401840     1   1
    $1$DGA162:  (ROUTE2)  ShadowSetMember    0 (member of DSA2:)
    $1$DGA163:  (ROUTE2)  ShadowSetMember    0 (member of DSA2:)
    ```

3. Type the following commands to dismount 1DGA162:

    ```
    $ dismount $1$dga162/pol=minicopy
    $ show device/bitmap
    ```

4. Check the system output. If it looks similar to the one below, convertible bitmaps are *not* in use, and the minicopy will succeed.

    ```
    $ show device/bitmap
    Device        BitMap     Size     Percent    Type of     Master   Active
      Name          ID     (Bytes)  Populated    Bitmap       Node
    DSA2:        00030001     9612      0.01%   Mini Copy   ROUTE2     Yes
    ```

However, if the output shows that the type of bit map is `Multiuse`, as in the example below, then HBMM *must* be disabled before remounting the device into the shadow set. Failure to disable HBMM in this case will result in a crash on VSI OpenVMS x86-64 V9.2. To avoid the crash, proceed to step 5.

```
Device          BitMap      Size     Percent     Type of    Master    Active
  Name            ID       (Bytes)  Populated    Bitmap      Node
DSA2:          00010001     9612      0.01%     Multiuse    ROUTE2    Yes
```

5. In order to avoid the crash when 1DGA162 is remounted, use the following commands:

```
$ set shadow dsa/disable=hbmm
$ show device/bitmap
```

The following output after the dismount means that convertible bitmaps are not in use, and the minicopy will succeed:

```
Device       BitMap    Size      Percent      Type of    Master   Active
  Name         ID     (Bytes)   Populated     Bitmap      Node
DSA2:       No Bitmaps Active
```

This issue will be fixed in a future patch kit.

2.1.40. Potential Crash When Using File or Page File Backed Global Sections

Recently, an issue was discovered when using file and page file backed writable global sections. The issue can result in an INCONMMGT or PGFIPLHI system crash. A fix for this issue is expected to be available shortly after the release of VSI OpenVMS x86-64 V9.2. If you experience a system crash, you are encouraged to report the problem and provide the dump for analysis.

2.1.41. LM Containers Do Not Work on VSI OpenVMS x86-64 V9.2

When connecting to a logical tape, you will get the following error:

```
$ LM CONNECT tape.tap LMA1:
%LM-F-INVGEOMETRY, Invalid geometry specified
```

Despite what the error message says, this is not a geometry issue, but a lock-related error that appears when the container is opened.

Because of this, logical tapes are not supported on VSI OpenVMS x86-64 V9.2.

A fix for this issue is expected to be available shortly after the release of VSI OpenVMS x86-64 V9.2.

2.1.42. TRACEBACK Sometimes Prints Extra Message

The TRACEBACK facility sometimes prints an extra message such as the RMS-S-STALL message in the example below. The error is not real and can be safely ignored.

```
$ run cnf_check3 OK AB OK %RMS-S-STALL, synchronize with operation
 completion
```

```
%PAS-F-OPNDASSCOM, operands are not assignment compatible %TRACE-F-
TRACEBACK, symbolic stack
dump follows image module routine line rel PC abs PC
```

2.1.43. MSCP Client Node May Crash Upon Completion of Mount Verification of a Device on the Client Node

Under certain conditions, if an MSCP serving node goes away and a device on a client node comes out of Mount Verification, the client system may crash.

This issue will be addressed in a future release.

2.1.44. C Run-Time Library Issues

Below is the list of known C Run-Time Library issues in VSI OpenVMS x86-64 V9.2:

• Writing 0 bytes to a mailbox fails and returns an error.

   Writing 0 bytes to a mailbox created with SYS$CREMBX incorrectly returns an error instead of sending an EOF to the mailbox.

• Several `poll()` problems:

   • Using `poll()` as a high-precision `sleep()` doesn't work.

      Calling `poll(NULL, 0, timeout)` to sleep for timeout seconds results in the process consuming CPU time

   • Polling sockets can sometimes result in excessive buffered I/Os

   • Requesting non-blocking I/O when transferring large buffers on sockets results in an unexpected network error.

      The C RTL now supports non-blocking I/O (`fcntl()` with O_NONBLOCK flag) but a network error will be reported if requesting non-blocking I/O on sockets and transferring buffers larger than 62696 bytes.

These issues will be addressed in a future release.

2.1.45. Unexpected Access Violations in pthreads

Applications that are linked against the pthreads library may crash because of an access violation occurring in the pthreads code. This issue is under investigation and will be fixed in a future update.

2.1.46. TCPIP$BIND_CONF.TEMPLATE_FORWARD Requires Adjustment in Environments Not Supporting DNSsec

The following lines in the TCPIP$BIND_CONF.TEMPLATE_FORWARD template file set up the forwarders' addresses and the DNSSEC validation:

```
//Specifies the IP addresses to be used for forwarding.
//The default is the empty list (no forwarding).
forwarders {
    8.8.8.8;
    8.8.4.4;
};

dnssec-validation auto; //Enable DNSSEC validation.
                        //Note dnssec-enable also needs to be set to
                        //yes to be effective. The default is yes.
```

However, if forwarders are changed to DNS servers that do not support DNSSEC or have it disabled, DNS lookup replies will be discarded when the DNSSEC validation fails.

To avoid this, please comment out the line of `dnssec-validation auto`.

2.1.47. NTPDATE No Longer Supported

NTPDATE is no longer supported and will be removed from an upcoming release of VSI TCP/IP Services. To perform the equivalent of NTPDATE, run `NTPD` making use of the `-q` and `"-G"` options.

```
$ ntpd :== $tcpip$ntp
$ ntpd "-G" -q
ntp.exe[538969120]: ntpd 4.2.8p15@1.3728 Fri Sep 22 07:00:58 UTC 2020 (2):
 Starting
ntp.exe[538969120]: Command line: tbd$dka200:[sys0.syscommon.][sysexe]tcpip
$ntp.exe -G -q -4
ntp.exe[538969120]: --------------------------------------------------
ntp.exe[538969120]: ntp-4 is maintained by Network Time Foundation,
ntp.exe[538969120]: Inc. (NTF), a non-profit 501(c)(3) public-benefit
ntp.exe[538969120]: corporation.  Support and training for ntp-4 are
ntp.exe[538969120]: available at https://www.nwtime.org/support
ntp.exe[538969120]: --------------------------------------------------
ntp.exe[538969120]: proto: precision = 1000.000 usec (-10)
ntp.exe[538969120]: proto: fuzz beneath 0.710 usec
ntp.exe[538969120]: basedate set to 2022-05-21
ntp.exe[538969120]: gps base set to 2022-05-22 (week 2211)
ntp.exe[538969120]: Listen and drop on 0 v4wildcard 0.0.0.0:123
ntp.exe[538969120]: Listen normally on 1 LO0 127.0.0.1:123
ntp.exe[538969120]: Listen normally on 2 WE0 10.10.116.182:123
ntp.exe[538969120]: Listening on routing socket on fd #4 for interface
 updates
ntp.exe[538969120]: ntpd: time set -50.590756 s
ntpd: time set -50.590756s
$
```

2.1.48. TCPIP$XDM Fails To Start

The TCPIP service XDM fails to start due to a version mismatch of a library against which it is built. The following error message will appear in SYS$SPECIFIC:[TCPIP$XDM]TCPIP$XDM_RUN.LOG:

```
%DCL-W-ACTIMAGE, error activating image DECW$XLIBSHR
-CLI-E-IMGNAME, image file X86VMS$DKA0:[SYS0.SYSCOMMON.][SYSLIB]DECW$XLIBSHR.EXE
-SYSTEM-F-SHRIDMISMAT, ident mismatch with shareable image
```

This issue will be addressed in an upcoming release of VSI TCP/IP Services.

2.2. Virtualization Notes

The notes in this section describe known issues and limitations when running VSI OpenVMS x86-64 V9.2 as a **guest operating system** in Oracle VM VirtualBox, KVM, and VMware virtual machines.

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

VSI OpenVMS x86-64 may not correctly maintain the time of day in virtual machine environments after certain events. To keep the time of day accurate, the system manager may need to issue a SET TIME command after booting, suspending, taking a snapshot of a virtual machine, or any other similar events, depending on the virtual machine host. This will be addressed in a future release of VSI OpenVMS x86-64.

2.2.2. VirtualBox and Hyper-V Compatibility on Windows 10 Hosts

Host systems running Windows 10 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 **vmscheck.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 10 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 10 host system by executing the following command:

   ```
   shutdown -s -t 2
   ```

4. Power on and boot your Windows 10 host system again.

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

For more information about the CPU feature checks, see Section 1.5 in the *Before You Start... Read These First* 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.
   ```

2.2.3. 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.

2.2.4. 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.

2.2.5. 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:

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

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, in turn, can significantly impact the performance of your VMware virtual machine.

Furthermore, proceeding to boot from such display will often result in an incorrect output on the serial console connection, requiring a reset of your terminal emulator's terminal settings.

To avoid this, proceed as follows (instead of booting):

1. Exit to the EFI Boot Manager by typing `EXIT` or `SHELL` and hitting several ESCape characters to get to the following screen (note that the list of EFI items will be different on your system):
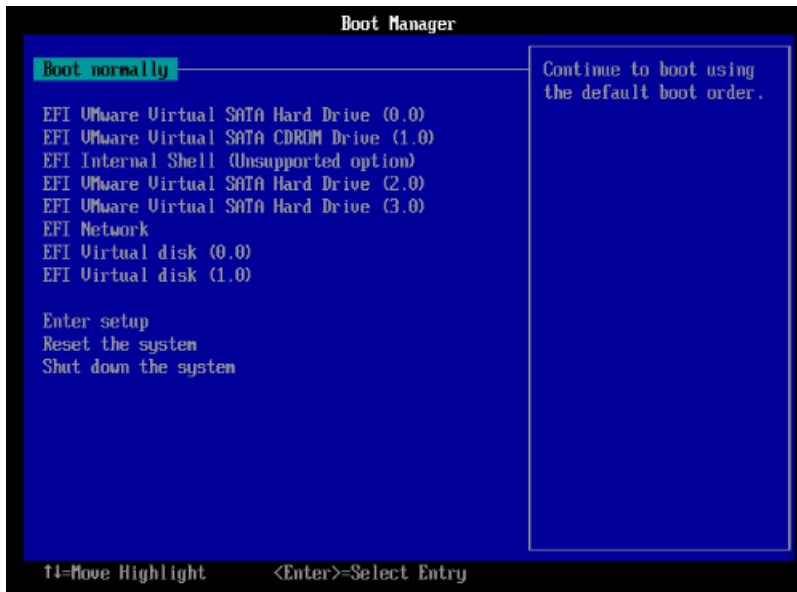
2. On this screen, press Enter to return to the main Boot Manager screen.

3. Enter the `DEVICE` command again, and you should be presented with a full configuration display, similar to the one in the first screen shot.

4. If you see a complete configuration screen at this point, proceed to boot your device of choice.

2.3. Layered and Open Source Products Notes

Layered and open source products for VSI OpenVMS x86-64 V9.2 can be downloaded individually from the VMS Software Service Platform. For detailed information about the products, please refer to the associated product release notes bundled with the kits.

Appendix A. VSI C Run-Time Library (C RTL) Notes

A.1. C99 Update

VSI OpenVMS x86-64 V9.2 includes the updated C RTL that provides additional C99 Standard functions and functionality that were not previously available.

These functions are also available on the following VSI OpenVMS versions:

- VSI OpenVMS Integrity V8.4-2L1 and V8.4-2L3

- VSI OpenVMS Alpha V8.4-2L1 and V8.4-2L2

To utilize C99 Standard functions, compile your applications with the /STANDARD=C99, /STANDARD=LATEST or /STANDARD=RELAXED (default) switches. See the section Section A.1.1 for a list of functions.

The value of the __CRTL_VER macro, predefined by the VSI C Compiler, has been changed from 80400000 to 80500000.

Note

If you develop an application on a system with the CRTL C99 or any later kit installed and intend it to be run on a system without those kits, you must compile your application with the switch /DEFINE=(__CRTL_VER_OVERRIDE=80400000).

This release also includes changes to some header files to make them more consistent with the standards.

MATH.H, FP.H and FLOAT.H	Definitions have been moved around/between these header files to match the C99 Standard requirements.
STDINT.H and INTTYPES.H	Definitions from INTTYPES.H have been moved into a new header file, STDINT.H, to match the standard's requirements. INTTYPES.H now contains '#include <STDINT.h>' so that existing applications will continue to compile without any changes. In addition, some new names have been defined for data types to match the C99 Standard. For example, int64_t.

Possible errors when compiling applications

With the addition of new data type and function definitions, it is possible that applications may incur compilation errors if the applications include definitions that conflict with the definitions now provided in the system header files. For example, if an application contains a definition of int64_t that differs from the definition included in STDINT.H, the compiler generates a %CC-E-NOLINKAGE error. Conflicting function definitions can result in various %CC errors or warnings. To diagnose such problems, compile the application using /LIST/SHOW=INCLUDE and then examine the listing file.

There are different ways to resolve such problems. Some examples are following:

- Remove the application-specific definition if the system-provided definition provides the proper functionality.

- Undefine the system-provided definition before making the application-specific definition. For example:

```
#ifdef alloca
#undefine alloca
#endif
<application-specific definition of alloca>
```

- Guard the application-specific definition. For example:

```
#ifndef alloca
<application-specific definition of alloca>
#endif
```

Possible informational and warning messages when linking applications

The implementations of isnan() and isnormal() have changed and now utilize functions in the Math Run-Time Library (DPML$SHR.EXE). If your application includes references to isnan() or isnormal() and you encounter the %ILINK-I-UDFSYM and %ILINK-W-USEUNDEF messages for MATH$ symbols when linking your application, you may add SYS$LIBRARY:DPML$SHR/SHAREABLE to your options file as one way of resolving undefined symbolic references.

UNSUPCONVSPEC warning

When using the new format specifiers with print and scan (see the section Section A.1.1.12) the system will generate a %CC-W-UNSUPCONVSPEC warning.

You can eliminate the warnings by adding #pragma message disable UNSUPCONVSPEC to your code or by compiling your code with the switch, /WARNING=DISABLE=UNSUPCONVSPEC. This warning will be removed in a future update to the C compiler.

va_copy()

va_copy() will be enabled with a future VSI C Compiler Version 7.5.

Online Help

A future version of VSI OpenVMS x86-64 will update the Online Help contents of the C RTL with the functions listed in this document.

A.1.1. C99 Functions

This section describes the C99 functions that have been added to the C RTL. For VSI OpenVMS x86-64, these functions are included in the C RTL.

For VSI OpenVMS Integrity and VSI OpenVMS Alpha systems, these functions are included in the following kits:

- C99 V1.0

- C99 V2.0

- RTL V2.0

A.1.1.1. fpclassify

Format

```
#include <math.h>
int fpclassify (real-floating x);
```

Description

The `fpclassify` macro classifies its argument value as NaN, infinite, normal, subnormal, zero, or into another implementation-defined category. First, an argument represented in a format wider than its semantic type is converted to its semantic type. Then classification is based on the type of the argument.

Returns

The `fpclassify` macro returns the value of the number classification macro appropriate to the value of its argument.

A.1.1.2. isblank, iswblank

Format

```
#include <ctype.h>
int isblank (int c);
#include <wctype.h>
int iswblank (wint_t wc);
```

Description

The `isblank` function tests for any character that is a standard blank character or is one of a locale-specific set of characters for which `isspace` is true and that is used to separate words within a line of text. The standard blank characters are the following: space (' '), and horizontal tab ('\t'). In the "C" locale, `isblank` returns true only for the standard blank characters.

The `iswblank` function tests for any wide character that is a standard blank wide character or is one of a locale-specific set of wide characters for which `iswspace` is true and that is used to separate words within a line of text. The standard blank wide characters are the following: space (L' '), and horizontal tab (L'\t'). In the "C" locale, `iswblank` returns true only for the standard blank characters.

Returns

These functions return true if and only if the value of the character or wide character has the property described in the description.

A.1.1.3. isgreater, isgreaterequal, isless, islessequal, islessgreater, isunordered

Format

```
#include <math.h>
int isgreater (x, y);
int isgreaterequal (x, y);
int isless (x, y);
int islessequal (x, y);
int islessgreater (x, y);
```

```
int isunordered (x, y);
```

Description

The normal relation operations (like <, "less than") will fail if one of the operands is NaN. This will cause an exception. To avoid this, C99 defines the macros listed below.

These macros are guaranteed to evaluate their arguments only once. The arguments must be of real floating-point type (note: do not pass integer values as arguments to these macros, since the arguments will not be promoted to real-floating types).

isgreater ()	determines (x) > (y) without an exception if x or y is NaN.
isgreaterequal ()	determines (x) >= (y) without an exception if x or y is NaN.
isless ()	determines (x) < (y) without an exception if x or y is NaN.
islessequal ()	determines (x) <= (y) without an exception if x or y is NaN.
islessgreater ()	determines (x) < (y) || (x) > (y) without an exception if x or y is NaN. This macro is not equivalent to x != y because that expression is true if x or y is NaN.
isunordered ()	determines whether its arguments are unordered, that is, whether at least one of the arguments is a NaN.

Returns

The macros other than `isunordered()` return the result of the relational comparison; these macros return 0 if either argument is a NaN.

`isunordered()` returns 1 if x or y is NaN and 0 otherwise.

A.1.1.4. llrint, llrintf, llrintl

Format

```
#include <math.h>
long long int llrint (double x);
long long int llrintf (float x);
long long int llrintl (long double x);
```

Description

The `llrint` functions round their argument to the nearest integer value, rounding according to the current rounding direction. If the rounded value is outside the range of the return type, the numeric result is unspecified and a domain error or range error may occur.

Returns

The `llrint` functions return the rounded integer value.

A.1.1.5. llround, llroundf, llroundl

Format

```
#include <math.h>
```

```
long long int llround (double x);
long long int llroundf (float x);
long long int llroundl (long double x);
```

Description

The `llround` functions round their argument to the nearest integer value, rounding halfway cases away from zero, regardless of the current rounding direction. If the rounded value is outside the range of the return type, the numeric result is unspecified and a domain error or range error may occur.

Returns

The `llround` functions return the rounded integer value.

A.1.1.6. nearbyint, nearbyintf, nearbyintl

Format

```
#include <math.h>
double nearbyint (double x);
float nearbyintf (float x);
long double nearbyintl (long double x);
```

Description

The `nearbyint` functions round their argument to an integer value in floating-point format, using the current rounding direction and without raising the "inexact" floating-point exception.

Returns

The `nearbyint` functions return the rounded integer value.

A.1.1.7. round, roundf, roundl

Format

```
#include <math.h>
double round (double x);
float roundf (float x);
long double roundl (long double x);
```

Description

The round functions round their argument to the nearest integer value in floating-point format, rounding halfway cases away from zero, regardless of the current rounding direction.

Returns

The round functions return the rounded integer value.

A.1.1.8. scalbln, scalblnf, scalblnl, scalbn, scalbnf, scalbnl

Format

```
#include <math.h>
double scalbln (double x, long int n);
float scalblnf (float x, long int n);
```

```
long double scalblnl (long double x, long int n);
double scalbn(double x, int n);
float scalbnf(float x, int n);
long double scalbnl(long double x, int n);
```

Description

These functions multiply their first argument *x* by FLT_RADIX (probably 2) to the power of *n*, which is:

```
x * FLT_RADIX ** n
```

The definition of FLT_RADIX can be obtained by including <float.h>.

Returns

On success, these functions return x × FLT_RADIX ** n.

If *x* is a NaN, a NaN is returned.

If *x* is positive or negative infinity, positive or negative infinity is returned.

If *x* is +/- 0, +/- 0 is returned.

If the result overflows, a range error occurs, and the functions return HUGE_VAL, HUGE_VALF, or HUGE_VALL, respectively, with a sign the same as *x*.

If the result underflows, a range error occurs, and the functions return zero, with a sign the same as *x*.

A.1.1.9. strtof, strtold, wcstof, wcstold

Format

```
#include <stdlib.h>
float strtof (const char * restrict nptr, char ** restrict endptr);
long double strtold (const char * restrict nptr, char ** restrict endptr);
#include <wchar.h>
float wcstof (const wchar_t * restrict nptr,
wchar_t ** restrict endptr);
long double wcstold (const wchar_t * restrict nptr,
wchar_t ** restrict endptr);
```

Function Variants

The `strtof` function has variants named _strtof32 and _strtof64 for use with 32-bit and 64-bit pointer sizes, respectively. The `strtold` function has variants named _strtold32 and _strtold64 for use with 32-bit and 64-bit pointer sizes, respectively. The `wcstof` function has variants named _wcstof32 and _wcstof64 for use with 32-bit and 64-bit pointer sizes, respectively. The `wcstold` function has variants named _wcstold32 and _wcstold64 for use with 32-bit and 64-bit pointer sizes, respectively. See *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

These functions convert the initial portion of the string or wide string pointed to by *nptr* to float, and long double representation, respectively. First, they decompose the input string into three parts: an

initial, possibly empty, sequence of white-space characters (as specified by the `isspace` function), a subject sequence resembling a floating-point constant or representing an infinity or NaN, and a final string of one or more unrecognized characters, including the terminating null character of the input string. Then, they attempt to convert the subject sequence to a floating-point number, and return the result.

The expected form of the (initial portion of the) string or wide string is optional leading white space, an optional plus ('+') or minus sign ('-') and then either (i) a decimal number, or (ii) a hexadecimal number, or (iii) an infinity, or (iv) a NAN (not-a-number).

Returns

The functions return the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, plus or minus HUGE_VAL, HUGE_VALF, or HUGE_VALL is returned (according to the return type and sign of the value), and the value of the macro ERANGE is stored in *errno*. If the result underflows, the functions return a value whose magnitude is no greater than the smallest normalized positive number in the return type; whether *errno* acquires the value ERANGE is implementation-defined.

A.1.1.10. va_copy

Format

```
#include <stdarg.h>
void va_copy (va_list dest, va_list src);
```

Description

The `va_copy` macro initializes *dest* as a copy of *src*, as if the va_start macro had been applied to *dest* followed by the same sequence of uses of the `va_arg` macro as had previously been used to reach the present state of *src*. Neither the `va_copy` nor `va_start` macro shall be invoked to reinitialize *dest* without an intervening invocation of the `va_end` macro for the same *dest*.

This macro will be enabled with a future VSI C Compiler Version 7.5.

Returns

The `va_copy` macro returns no value.

A.1.1.11. wcstoll, wcstoull

Format

```
#include <wchar.h>
long long int wcstoll (const wchar_t * restrict nptr,
wchar_t ** restrict endptr, int base);
unsigned long long int wcstoull (const wchar_t * restrict nptr,
wchar_t ** restrict endptr, int base);
```

Function Variants

The `wcstoll` function has a variant named _wcstoll64 for use with 64-bit pointer sizes. The wcstuoll function has a variant named _wcstoull64 for use with 64-bit pointer sizes. See the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

The `wcstoll` and `wcstoull` functions convert the initial portion of the wide string pointed to by *nptr* to long long int and unsigned long long int representation, respectively. First, they decompose the input string into three parts: an initial, possibly empty, sequence of white-space wide characters (as specified by the `iswspace` function), a subject sequence resembling an integer represented in some radix determined by the value of base and a final wide string of one or more unrecognized wide characters, including the terminating null wide character of the input wide string. Then, they attempt to convert the subject sequence to an integer, and return the result.

Returns

The functions return the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, LONG_MIN, LONG_MAX, LLONG_MIN, LLONG_MAX, ULONG_MAX, or ULLONG_MAX is returned (according to the return type sign of the value, if any), and the value of the macro ERANGE is stored in *errno*.

A.1.1.12. Print and scan conversion specifier and argument types

The C RTL now supports the F conversion specifier and the hh, t, j and z argument types in print and scan.

F	Similar to f.
hh	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a signed char or unsigned char argument (the argument will have been promoted according to the integer promotions, but its value shall be converted to signed char or unsigned char before printing); or that a following n conversion specifier applies to a pointer to a signed char argument.
t	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a *ptrdiff_t* or the corresponding unsigned integer type argument; or that a following n conversion specifier applies to a pointer to a *ptrdiff_t* argument.
J	Specifies that a following d, i, o, u, x, or X conversion specifier applies to an *intmax_t* or *uintmax_t* argument; or that a following n conversion specifier applies to a pointer to an *intmax_t* argument.
z	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a *size_t* or the corresponding signed integer type argument; or that a following n conversion specifier applies to a pointer to a signed integer type corresponding to *size_t* argument.

A.1.1.13. strftime, wcsftime, strptime – additional conversion specifiers

Description

The following conversion specifiers have been added to `strftime`, `wcsftime` and `strptime`:

%F	is equivalent to "%Y−%m−%d" (the ISO 8601 date format). [tm_year, tm_mon, tm_mday]
%g	is replaced by the last 2 digits of the week-based year as a decimal number (00−99). [tm_year, tm_wday, tm_yday]

%G	is replaced by the week-based year as a decimal number (e.g., 1997). [tm_year, tm_wday, tm_yday]
%k	The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by a blank.
%l	The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank.
%P	Like %p but in lowercase: "am" or "pm" or a corresponding string for the current locale.
%s	The number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC).
%u	The day of the week as a decimal, range 1 to 7, with Monday being 1.
%z	The +*hhmm* or -*hhmm* numeric timezone (that is, the hour and minute offset from UTC).
%Z	The timezone name.
%+	The date and time in date format.

A.2. CRTL ECO V3.0 Changes

For VSI OpenVMS Integrity and VSI OpenVMS Alpha systems, VSI provides the CRTL ECO V3.0 kit that includes bug fixes, new functions, new constants and a new header file.

For VSI OpenVMS x86-64, all these changes are included in the C RTL.

A.2.1. Bug Fixes

• Calling the function l64a with an invalid argument no longer causes a memory leak.

• Calling the function l64a_r with a null buffer pointer no longer causes an ACCVIO.

• Calling the functions `readv` or `writev` with an invalid file descriptor no longer causes a memory leak.

• Fixed a possible memory leak in `realpath`.

• Fixed possible undefined behavior in `make_cli_comm`.

• Fixed a memory leak in the return path of `newwin`.

• Fixed definitions of `isnan`, `isnanf`, and `isnanl`.

• Fixed `fstat` to return the proper value in the stat field, `st_ino`, when FID$W_SEQ field has the high bit set and _USE_STD_STAT has been defined.

• Fixed headers to define `isblank` and `iswblank` when compiling with /STANDARD=C99.

• Fixed the definition of C99 routines when compiling with /STANDARD=RELAXED.

• Fixed headers so that nan, nanf, and nanl are only defined when using IEEE floating point.

• Fixed headers so that `va_copy` is only defined when using the latest compiler.

• Fixed SEMAPHORE.H so that it no longer generates a compiler error when compiled with /STANDARD=ANSI89 or /STANDARD=VAXC.

A.2.2. New Constants

The following constants were added to LIMITS.H:

- LLONG_MAX – Maximum value for an object of type long long int.

- LLONG_MIN – Minimum value for an object of type long long int.

- ULLONG_MAX – Maximum value for an object of type unsigned long long int.

A.2.3. New Flags

The following flags were added to DLFCN.H:

- RTLD_GLOBAL

- RTLD_LOCAL

A.2.4. New Datatypes

The following type was added to SOCKET.H:

- socklen_t – Socket address length type.

The following types were added to DECC$TYPES.H:

- typedef const unsigned int * __const_u_int_ptr64;

- typedef int * __int_ptr64;

- typedef const int * __const_int_ptr64;

A.2.5. New Header

This ECO includes MALLOC.H.

A.2.6. Interface Change

The interface for the function `isatty` has been modified.

Previously, in case of an error, the function returned -1. This is not compatible with the POSIX 1003.1 standard. This leads to errors that are hard to find. With this release, in case of an error, the function returns 0 and stores the error in *errno*.

If your code assumes a return value of 0, this means that the fd is not a tty. If your code assumes a return value of -1, this means an error, you will need to change the code. See the following example:

Existing code:

```
int val = isatty(fd);
if (val == 1) {
 // fd is tty
}
else if (val == 0) {
 // fd is not tty
```

```
}
else if (val == -1) {
 // error
}
```

Changed code:

```
int val = isatty(fd);
if (val == 1) {
 // fd is tty
}
else if (val == 0) {
 if (errno) {
   // error
}
else {
   // fd is not tty
}
}
```

A.2.7. New Feature Logical: DECC$PRN_PRE_BYTE

A change introduced by Hewlett Packard Enterprise (HPE) during OpenVMS V8.4 maintenance allowed systems that used the CIFS product (SAMBA) to display files in the appropriate format. However, that change affected files with Print File Carriage Control (also known as Fortran Carriage Control). For some environments, the print codes that are removed when transferring files between systems cause incorrect printing behavior resulting in form feeds being lost.

A new C RTL feature logical name, DECC$PRN_PRE_BYTE, when enabled, converts the print codes in files with Print File Carriage Control to their ASCII control code equivalents. CIFS then sends them to the client.

Enabling this new logical, in addition to enabling the logical DECC$TERM_REC_CRLF, which is used by CIFS, correctly includes the print codes on transferred files.

To enable the DECC$PRN_PRE_BYTE feature, use:

```
$ DEFINE/SYSTEM DECC$PRN_PRE_BYTE ENABLE
```

A.2.8. New Functions

This section describes the functions that have been added to the C RTL. For VSI OpenVMS x86-64, they are included in the C RTL.

For VSI OpenVMS Integrity and VSI OpenVMS Alpha systems, these functions are included in the RTL V3.0 kit.

A.2.8.1. freeifaddrs

Format

```
#include <ifaddrs.h>
void freeifaddrs(struct ifaddrs *ifp);
```

Description

The `freeifaddrs` function frees the dynamically allocated data returned by the `getifaddrs` function. *ifp* is the address returned by a previous call to `getifaddrs`. If *ifp* is a NULL pointer no action occurs.

A.2.8.2. getgrent_r

Format

```
#include <grp.h>
int getgrent_r(struct group *grp, char *buffer, size_t bufsize, struct
 group **result);
```

Function Variant

The `getgrent_r` function has a variant named `__getgrent_r64` and for use with 64-bit pointers. See the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https:// vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

The `getgrent_r` function is the reentrant version of `getgrent`. The `getgrent_r` function returns a pointer to a structure containing the broken-out fields of a record in the group database. When first called, `getgrent_r` returns a pointer to a group structure containing the first entry in the group database. Thereafter, it returns a pointer to the next group structure in the group database, so successive calls can be used to search the entire database. It updates the group structure pointed to by *grp* and stores a pointer to that structure at the location pointed to by *result*. Storage referenced by the group structure is allocated from the memory provided with the *buffer* argument, which is *bufsize* characters in size. The maximum size needed for this buffer can be determined with the _SC_GETGR_R_SIZE_MAX parameter of the `sysconf` function.

If the requested entry is not found or an error is encountered, a NULL pointer is returned at the location pointed to by *result*.

Returns

On success, the function returns 0 and *result is a pointer to the struct group. On error, the function returns an error value and *result is NULL.

A.2.8.3. gethostbyname_r

Format

```
#include <netdb.h>
int gethostbyname_r(const char *name, struct hostent *ret, char *buffer,
size_t buflen, struct hostent **result, int *h_errnop);
```

Description

The `gethostbyname_r` function is the reentrant version of `gethostbyname`. The caller supplies a *hostent* structure *ret* which will be filled in on success, and a temporary work buffer *buffer* of size *buflen*. After the call, *result* will point to the result on success. In case of an error or if no entry is found *result* will be NULL. The functions return 0 on success and a nonzero error number on failure. In addition to the errors returned by the nonreentrant version, if *buffer* is too small, the functions will return ERANGE, and the call should be retried with a larger *buffer*. The global variable *h_errno* is not modified, but the address of a variable in which to store error numbers is passed in *h_errnop*.

Returns

The functions return 0 on success and a nonzero error number on failure. The global variable *h_errno* is not modified, but the address of a variable in which to store error numbers is passed in *h_errnop*.

Note

Modules which include calls to `gethostbyname` or `gethostbyname_r` must be compiled with the C switch **/PREFIX=ALL**.

A.2.8.4. getifaddrs

Format

```
#include <sys/socket.h>
#include <ifaddrs.h>
int getifaddrs(struct ifaddrs **ifap);
```

Function Variants

The `getifaddrs` function has variants named `_getifaddrs32` and `_getifaddrs64` for use with 32-bit and 64-bit pointer sizes, respectively. See the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

The `getifaddrs` function creates a linked list of structures describing the network interfaces, one for each network interface on the host machine. The `getifaddrs` function stores a reference to a linked list of the network interfaces on the local machine in the memory referenced by *ifap*. The list consists of `ifaddrs` structures, as defined in the include file <ifaddrs.h>. The `ifaddrs` structure contains the following entries:

```
struct    ifaddrs  *ifa_next;          /* Pointer to next struct */
char               *ifa_name;          /* Interface name */
u_int               ifa_flags;         /* Interface flags */
struct    sockaddr *ifa_addr;          /* Interface address */
struct    sockaddr *ifa_netmask;       /* Interface netmask */
struct    sockaddr *ifa_broadaddr;     /* Interface broadcast address */
struct    sockaddr *ifa_dstaddr;       /* P2P interface destination */
void               *ifa_data;          /* unused */
```

The data returned by `getifaddrs` is dynamically allocated and should be freed using `freeifaddrs` when no longer needed.

Returns

The `getifaddrs` function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

A.2.8.5. getrusage

Format

```
#include <sys/resource.h>
int getrusage(int who, struct rusage *r_usage);
```

Description

The `getrusage` function provides measures of the resources used by the current process or its terminated and waited-for child processes. If the value of the *who* argument is RUSAGE_SELF, information is returned about resources used by the current process. If the value of the *who* argument is RUSAGE_CHILDREN, information is returned about resources used by the terminated and waited-for children of the current process. If the child is never waited for, the resource information for the child process is discarded and not included in the resource information provided by `getrusage`.

Currently, only getting elapsed user time (`ru_utime`) and maximum resident memory (`ru_maxrss`) is supported.

Returns

Upon successful completion, `getrusage` returns 0; otherwise, -1 is returned and *errno* set to indicate the error.

A.2.8.6. stpcpy

Format

```
#include <string.h>
char *stpcpy(char *dest, const char *src);
```

Function Variants

The `stpcpy` function has variants named `_stpcpy32` and `_stpcpy64` for use with 32-bit and 64-bit pointer sizes, respectively. See the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

The function `stpcpy` uses `strlen` to determine the length of *src* then copies the *src* to *dest.* The difference from the strcpy function is that stpcpy returns a pointer to the final '\0', and not to the beginning of the line.

Returns

Pointer to the end of the string *dest.*

A.2.8.7. strerror_r

Format

```
#include <string.h>
int strerror_r(int error_code, char *buf, size_t buflen);
```

Description

The `strerror_r` function is the reentrant version of `strerror`. The `strerror_r` function uses the error number in *error_code* to retrieve the appropriate locale dependent error message. The contents of the error message strings are determined by the LC_MESSAGES category of the program's current locale.

If *error_code* is EVMSERR the function looks at vaxc$errno to get the OpenVMS error condition.

Returns

Upon successful completion, `strerror_r` returns 0 and puts the error message in the character array pointed to by *buf*. The array is *buflen* characters long and should have space for the error message and the terminating null character.

A.2.8.8. strtoimax, strtoumax

Format

```
#include <inttypes.h>
intmax_t strtoimax(const char *nptr, char **endptr, int base);
uintmax_t strtoumax(const char *nptr, char **endptr, int base);
```

Function Variants

The `strtoimax` function has variants named `_strtoimax32` and `_strtoimax64` for use with 32-bit and 64-bit pointer sizes, respectively. The `strtoumax` function has variants named `_strtoumax32` and `_strtoumax64` for use with 32-bit and 64-bit pointer sizes, respectively. See the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

The `strtoimax` and `strtoumax` functions converts strings of ASCII characters pointed to by *nptr* to the appropriate signed and unsigned numeric values. `strtoimax` is a synonym for `strtoll`, `strtoumax` is a synonym for `strtoull`. The functions recognizes strings in various formats, depending on the value of the *base*. Any leading white-space characters (as defined by `isspace` in <ctype.h>) in the given string are ignored. The function recognizes an optional plus or minus sign, then a sequence of digits or letters that may represent an integer constant according to the value of the base. The first unrecognized character ends the conversion and is pointed to by *endptr*.

Leading zeros after the optional sign are ignored, and 0x or 0X is ignored if the base is 16.

If base is 0, the sequence of characters is interpreted by the same rules used to interpret an integer constant: after the optional sign, a leading 0 indicates octal conversion, a leading 0x or 0X indicates hexadecimal conversion, and any other combination of leading characters indicates decimal conversion.

Returns

- If successful, an integer value corresponding to the contents of *nptr* is returned.

- If the converted value falls out of range of corresponding return type, a range error occurs (setting *errno* to ERANGE) and INTMAX_MAX, INTMAX_MIN, UINTMAX_MAX or 0 is returned, as appropriate.

- If no conversion can be performed, 0 is returned.

A.2.8.9. strndup

Format

```
#include <string.h>
char *strndup(const char *s, size_t size);
```

Function Variants

The `strndup` function has variants named `_strndup32` and `_strndup64` for use with 32-bit and 64-bit pointer sizes, respectively. See the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_CRTL_REF.pdf] for more information on using pointer-size-specific functions.

Description

The `strndup` function duplicates a specific number of bytes from a string. The `strndup` function is equivalent to the `strdup` function, duplicating the provided string in a new block of memory allocated as if by using `malloc`, with the exception that `strndup` copies at most *size* plus one bytes into the newly allocated memory, terminating the new string with a NUL character. If the length of *s* is larger than *size*, only *size* bytes will be duplicated. If *size* is larger than the length of *s*, all bytes in *s* will be copied into the new memory buffer, including the terminating NUL character. The newly created string will always be properly terminated.

Returns

A pointer to the resulting string or NULL if there is an error.

A.3. C RTL Changes

VSI OpenVMS x86-64 includes the updated C RTL that provides additional functions, updates to some functions, bug fixes, and a new header.

For VSI OpenVMS Integrity and VSI OpenVMS Alpha systems, the next CRTL ECO kit will be released to provide these changes.

Possible errors when compiling applications

With the addition of new data type and function definitions, it is possible that applications may incur compilation errors if the applications include definitions that conflict with the definitions now provided in the system header files. For example, if an application contains a definition of int64_t that differs from the definition included in STDINT.H, the compiler generates a %CC-E-NOLINKAGE error. Conflicting function definitions can result in various %CC errors or warnings. To diagnose such problems, compile the application using /LIST/SHOW=INCLUDE and then examine the listing file.

There are different ways to resolve such problems. Consider the following examples:

- Remove the application-specific definition if the system-provided definition provides the proper functionality.

- Undefine the system-provided definition before making the application-specific definition. For example:

```
#ifdef alloca
#undefine alloca
#endif
<application-specific definition of alloca>
```

- Guard the application-specific definition. For example:

```
#ifndef alloca
<application-specific definition of alloca>
#endif
```

Manipulating Variable Argument Lists on x86-64

The implementation of variable argument lists on x86-64 is different than on Integrity and Alpha and may require source code changes, depending on how the lists are used.

On Integrity and Alpha, it is possible to copy one variable argument list to another using an assignment operator. For example:

```
va2 = va1
```

On x86-64, this does not work. Use the va_copy function for this purpose. For example:

```
va_copy (va2, va1)
```

On Integrity and Alpha, it is also possible to reference specific entries in the variable argument list using the subscript notation. For example:

```
int arg2 = va[1]
```

On x86-64, this does not work. Use the `va_arg` function for this purpose. For example:

```
int arg2 = va_arg(va,int)
```

A.3.1. New Functions

This section describes the new functions that have been added to the C RTL.

A.3.1.1. alloca

Format

```
#include <alloca.h>
void *alloca (unsigned int size);
```

Description

The `alloca` function allocates *size* bytes from the stack frame of the caller. See the *VSI C User's Guide for OpenVMS Systems* [https://vmssoftware.com/docs/VSI_C_USER.pdf] for the __ALLOCA macro.

Returns

The `alloca` function returns a pointer to the allocated memory.

A.3.1.2. mempcpy

Format

```
#include <string.h>
void *mempcpy (void *dest, const void *source, size_t size);
```

Function Variants

The `mempcpy` function has variants named _mempcpy32 and _mempcpy64 for use with 32-bit and 64-bit pointer sizes, respectively.

Description

The `mempcpy` function, similar to the `memcpy` function, copies *size* bytes from the object pointed to by *source* to the object pointed to by *dest*; it does not check for the overflow of the receiving memory area (*dest*).

Returns

The function returns a pointer to the byte following the last written byte.

A.3.1.3. getline, getwline, getdelim, getwdelim

Format

```
#include <stdio.h>
ssize_t getline (char **lineptr, size_t *n, FILE *stream);
ssize_t getwline (wchar_t **lineptr, size_t *n, FILE *stream);
ssize_t getdelim (char **lineptr, size_t *n, int delimiter, FILE *stream);
ssize_t getwdelim (wchar_t **lineptr, size_t *n, wint_t delimiter, FILE *stream);
```

Function Variants

The `getline` function has variants named `_getline32` and `_getline64` for use with 32-bit and 64-bit pointer sizes, respectively.

The `getwline` function has variants named `_getwline32` and `_getwline64` for use with 32-bit and 64-bit pointer sizes, respectively.

The `getdelim` function has variants named `_getdelim32` and `_getdelim64` for use with 32-bit and 64-bit pointer sizes, respectively.

The `getwdelim` function has variants named `_getwdelim32` and `_getwdelim64` for use with 32-bit and 64-bit pointer sizes, respectively.

Description

`getline` and `getwline` read an entire line from *stream*, storing the address of the buffer containing the text into **lineptr*. The buffer is null-terminated and includes the newline character, if one was found.

If **lineptr* is NULL, then `getline` will allocate a buffer for storing the line, which should be freed by the user program. (In this case, the value in **n* is ignored.)

Alternatively, before calling `getline`, **lineptr* can contain a pointer to a `malloc` allocated buffer **n* bytes in size. If the buffer is not large enough to hold the line, `getline` resizes it with `realloc`, updating **lineptr* and **n* as necessary.

`getdelim` and `getwdelim` work like `getline` and `getwline`, except that a line delimiter other than newline can be specified as the delimiter argument. As with `getline` and `getwline` a delimiter character is not added if one was not present in the input before end of file was reached.

Returns

On success, all functions return the number of characters read, including the delimiter character, but not including the terminating null byte.

A.3.1.4. qsort_r

Format

```
#include <stdlib.h>
void qsort_r (void *base, size_t nmemb, size_t size, int (*compar)
(const void *, const void *, void *), void *arg)
```

Function Variants

The `qsort_r` function has variants named `_qsort_r32` and `_qsort_r64` for use with 32-bit and 64-bit pointer sizes, respectively.

Description

The `qsort_r` function is the reentrant version of `qsort`. See the `qsort` description in the *VSI C Run-Time Library Reference Manual for OpenVMS Systems* [https://vmssoftware.com/docs/ VSI_CRTL_REF.pdf]. The comparison function takes a third argument. A pointer is passed to the comparison function via *arg*.

Returns

`qsort_r` returns no value.

A.3.1.5. mkostemp

Format

```
#include <stdlib.h>
int mkostemp (char *template, int flags)
```

Description

The `mkostemp` function replaces the six trailing Xs of the string pointed to by *template* with a unique set of characters, and returns a file descriptor for the file opened using the flags specified in *flags*.

The string pointed to by *template* should look like a filename with six trailing X's. The `mkostemp` function replaces each X with a character from the portable file-name character set, making sure not to duplicate an existing filename.

If the string pointed to by *template* does not contain six trailing Xs, -1 is returned.

Returns

A file descriptor for the open file.

-1 indicates an error.

A.3.2. Updates to Functions

* Added support for close on exit to the `open`, `fopen`, and `popen` functions. The open function now supports the O_CLOEXEC flag. The `fopen` and `popen` now support "e" in the access mode.

* Added support for the O_NONBLOCK flag in `fcntl` in the F_SETFL and F_GETFL modes.

* The functions `setbuf` and `setvbuf` now take 64-bit arguments. However, the *buffer* parameter must contain a 32-bit memory buffer, so when compiling the application with /POINTER=64 or / POINTER=LONG, `_malloc32` must be used to allocate the buffer.

A.3.3. Bug Fixes

- The `open` function now works properly when opening `/dev/null` and `/dev/tty` when DECC$POSIX_COMPLIANT_PATHNAMES is defined as 1, 2, or 3.

- Multiple processes or multiple threads attempting to open a file for append at the same time now correctly open the same file.

- If the `fopen` function is called with the O_TRUNC flag and the file specification includes a file version number, the function truncates the file when open rather than returning an error.

- The `shmget` function can be called a second time with the same key value and a size of 0.

- The `stat` function now returns the correct value for `st_blocks` when the file allocation value is greater than 65536 blocks.

- MATH$FP_CLASS_<n>X functions, added as part of the C99 work, have been added to STARLET.OLB.

A.3.4. New Header

ALLOCA.H.

Appendix B. Security Enhancements for VSI TCP/IP Services X6.0-16 FTPS

FTPS (FTP over SSL) allows for an encrypted data connection when using FTP. FTPS is run by using either FTP /SSL or COPY /FTP /SSL commands.

B.1. Changes in Connection Behavior

With TCP/IP Services V5.7 and prior versions, if you use FTPS and the FTP server is not set up to run SSL by not having the proper certificate, the following messages will be displayed, and the connection will continue in plain text:

```
TCPIP$_FTP_SSLERR, SSL not enabled on server
TCPIP$_FTP_SSLERR, Session will continue in plain text
```

See the following example:

```
$ ftp /ssl node1
220 node1.domain.com FTP Server (Version 5.7) Ready.
Connected to node1.
500 AUTH command unsuccessful.
TCPIP$_FTP_SSLERR, SSL not enabled on server
TCPIP$_FTP_SSLERR, Session will continue in plain text
Name (node1:username):

$ copy /ftp /ssl /log node2"username password"::file.txt *.*
TCPIP$_FTP_SSLERR, SSL not enabled on server
TCPIP$_FTP_SSLERR, Session will continue in plain text

%TCPIP-S-FTP_COPIED, NODE2.DOMAIN.COM"username
password"::file.txt copied to DISK:[USERNAME]FILE.TXT;7
(968408 bytes)
```

With VSI TCP/IP Services X6.0-16, if you use FTPS and the FTP server is not set up to run SSL, the connection will be terminated. See the following examples:

```
$ ftp /ssl node1
220 node1.domain.com FTP Server (Version 5.7) Ready.
Connected to node1.
500 AUTH command unsuccessful.
%TCPIP-E-SSLERR, SSL not enabled on server

$ copy /ftp /ssl /log node2"username password"::file.txt *.*
%TCPIP-E-SSLERR, SSL not enabled on server
```

You must either connect to an SSL-enabled FTP server or reissue the command without the /SSL qualifier.

B.2. Changes in Certificate Verification

VSI TCP/IP Services V5.7 and prior versions only check for certificate integrity but do not perform the full server certificate verification. Blindly using a self-signed certificate is not a secure practice.

In the following example, VSI TCP/IP Services V5.7 allows the connection to the FTP server without notifying about the self-signed certificate used by the server.

```
$ ftp /ssl node3
220 node3.domain.com FTP Server (Version 5.7) Ready.
Connected to node3.
234 AUTH command successful.
200 PBSZ command successful.
200 PROT command successful.
Name (node3:username):

$ copy /ftp /ssl /log node3"username password"::file.txt *.*
%TCPIP-S-FTP_COPIED, node3"username password"::FILE.TXT;18 copied
to DISK$WORK:[USERNAME]FILE.TXT;19 (1476 bytes)
```

VSI TCP/IP Services X6.0-16 includes a check for a self-signed or expired server certificate and outputs the appropriate message if such certificates are encountered. You can use a self-signed certificate if you trust the certificate and accept to use it.

The following example shows the connection to the FTP server with a self-signed certificate using VSI TCP/IP Services X6.0-16:

```
$ ftp /ssl node4
220 node4.domain.com FTP Server (Version 6.0) Ready.
Connected to node4.
234 AUTH command successful.
200 PBSZ command successful.
200 PROT command successful.

%TCPIP-F-SSLERR, self signed certificate

        Country: US
          State: MA
       Locality: Boston
   Organization: Certificate Company
           Name: company.com
         E-Mail: first.last@company.com
     Valid from: 30-Apr-2021 22:57
        Expires: 30-Apr-2022 22:57

If you trust the certificate, re-issue the command with the /TRUST
 qualifier.

$ copy /ftp /ssl node3"username password"::file.txt *.*
%TCPIP-F-SSLERR, self signed certificate

        Country: US
          State: MA
       Locality: Boston
   Organization: Certificate Company
           Name: company.com
         E-Mail: first.last@company.com
     Valid from: 30-Apr-2021 22:57
        Expires: 30-Apr-2022 22:57

If you trust the certificate, re-issue the command with the /TRUST
 qualifier.
```

Add the /TRUST qualifier to the command to proceed with the FTPS connection as in the following example:

```
$ ftp /ssl /trust node4
220 node4.domain.com FTP Server (Version 6.0) Ready.
Connected to node4.
234 AUTH command successful.
200 PBSZ command successful.
200 PROT command successful.
%TCPIP-I-SSLERR, self signed certificate
%TCPIP-I-SSLERR, TRUST specified; FTP/SSL continuing...
Name (node4:username):

$ copy /ftp /ssl /log /trust node4"username password"::file.txt *.*
%TCPIP-I-SSLERR, self signed certificate
%TCPIP-I-SSLERR, TRUST specified; FTP/SSL continuing...

%TCPIP-S-FTP_COPIED, node4"username password"::FILE.TXT;18 copied to
 DISK:FILE.TXT;22 (1476 bytes)
```