

VSI OpenVMS x86-64 V9.1 Field Test Release Notes

June 2021

V9.1 Field Test

Contents

Preface	5
Introduction	5
Intended Audience	5
Document Structure.....	5
Related Documents.....	5
Before You Start... Read These First	6
1. Tested Virtual Environments	6
1.1 Tested VMware Products and License Types	6
2. Hardware Support	7
3. MD5 Checksum for the X86091OE.ISO File.....	7
4. Non-Intel Processors Are Not Currently Supported	7
5. CPU Compatibility Checks for Virtual Machines	7
6. Terminal Emulator Settings	8
7. x86-64 Licensing	9
8. License PAKs for VSI OpenVMS x86-64 V9.1.....	9
9. MemoryDisk and the Command Procedure SYS\$MD.COM	9
10. Networking Options.....	10
10.1 VSI DECnet Phase IV for OpenVMS.....	10
10.2 VSI TCP/IP Services X6.0-9 TELNET and FTP Available in VSI OpenVMS x86-64 V9.1	11
11. VSI SSL111 V1.1-1K for OpenVMS	12
12. VSI Kerberos V3.3-1 for OpenVMS.....	12
13. VSI DECwindows Motif V1.7-X for OpenVMS	12
14. Scripts for Configuring and Running VSI OpenVMS x86-64 in Virtual Machines ..	13
15. Documentation Notes.....	13
Release Notes	14
1. Operating System Notes	14
1.1 Features Not Available in VSI OpenVMS x86-64 V9.1	14
1.2 Access Violation.....	14
1.3 AUTHORIZE Utility: Exit Sometimes Results in System Crash.....	14
1.4 AUTOGEN Warning That Appears During AUTOGEN Boot May Be Safely Ignored.....	15
1.5 BACKUP Utility: Verification Errors for SYS\$EFI.SYS When Copying an x86-64 System Disk	15
1.6 CHECKSUM Utility Supports SHA1 and SHA256 Algorithms.....	16
1.7 Copying Files Between Integrity Systems and x86-64 Systems	16
1.8 Cross-Tools Kit Update	16
1.9 Display of License Charge Information for x86-64 Nodes.....	16
1.10 ENCRYPT Utility Does Not Work as Expected.....	17
1.11 Extended File Cache (XFC)	17
1.12 HYPERSORT Utility Available.....	17
1.13 Idle CPU Power Saving Mechanism.....	17
1.14 Images Linked /SYSEXE Require to Be Relinked	17
1.15 INSTALL Utility Supports INSTALL /RESIDENT and /SHARED=ADDRESS_DATA	18

1.16	ISO 9660 Formatted Volume Can Not Be Mounted on x86-64 Systems.....	18
1.17	LIB\$INITIALIZE Requires LINK Qualifier	19
1.18	Linker: New Informational Messages	19
1.19	Memory Disks	19
1.20	Mount Verification for Tape Devices Causes System Crash.....	20
1.21	MSCP Served Disks.....	20
1.22	OpenVMS Clusters	21
1.23	OpenVMS Cluster Usage of LAN Failover and VLAN Devices	21
1.24	OpenVMS x86-64 Will Not Support Swap Files.....	22
1.25	Parallel Processing Library (PPL\$).....	22
1.26	POSIX Threads Library	22
1.27	Process Dumps.....	23
1.28	Security Server	23
1.29	Reserved Memory.....	23
1.30	Spurious Error Message During Shutdown	24
1.31	Storage Controllers Without Attached Disks May Have Incorrect Device Names.....	24
1.32	Supported Disk Types.....	24
1.33	Symmetric Multiprocessing (SMP)	24
1.34	SYSGEN Parameter Changes	25
1.35	System Crash Dumps	28
1.36	System Service Intercept (SSI)	29
1.37	Text Editors.....	30
1.38	Traceback Support.....	30
1.39	Volume Shadowing	30
1.40	VSI C Run-Time Library (C RTL) Update	30
1.41	VSI DECram for OpenVMS	31
1.42	VSI TCP/IP Services: Use of SHOW INTERFACE and IFCONFIG -a on x86-64 Systems Under Heavy Traffic.....	31
1.43	ZIP/UNZIP Tools.....	31
2.	Virtualization Notes	32
2.1	Time of Day May Not Be Correctly Maintained in Virtual Machine Environments	32
2.2	VirtualBox and Hyper-V Compatibility on Windows 10 Hosts	32
2.3	VirtualBox: TCP Ports May Become Unusable After Guest Is Terminated ..	33
2.4	VMware Guest May Fail to Boot After Adding Second SATA Controller.....	34
2.5	Wall-Clock Time Sometimes Runs Slow on Virtual Machine Guests	34
3.	Layered and Open Source Products Notes	35
Appendix A: VSI C Run-Time Library (C RTL) Notes		36
	C99 Update	36
	C99 Functions	38
	CRTL ECO V3.0 Changes.....	46
	Bug Fixes	46
	New Constants	46
	New Flags	46
	New Datatypes.....	47
	New Header	47
	Interface Change.....	47
	New Feature Logical: DECC\$PRN_PRE_BYTE	48

- New Functions.....48
- C RTL Changes.....54
 - New Functions.....55
 - Updates to Functions.....57
 - Bug Fixes57
 - New Header58

V9.1 Field Test

Preface

Introduction

VMS Software, Inc. (VSI) is pleased to introduce VSI OpenVMS x86-64 V9.1 Field Test. This release of OpenVMS for x86-64 is intended for testing purposes only.

With this release, VSI introduces the installation procedure for VSI OpenVMS x86-64 V9.1 **as a guest operating system** on Oracle VM VirtualBox, Red Hat KVM, and VMware virtual machines.

Intended Audience

This document is intended for all users of VSI OpenVMS x86-64 V9.1. Read this document before you install or use VSI OpenVMS x86-64 V9.1.

Document Structure

This document contains the following sections:

- [Before You Start...Read These First](#)
- [Release Notes](#)

Related Documents

The following documents provide additional information in support of this release. They are included in the **V91_DOCS.zip** file that is available for download from the VSI Services Portal.

- *VSI OpenVMS x86-64 V9.1 Field Test Installation Guide*
- *OpenVMS V9.0 x86-64 Boot Manager User Guide with Virtual Machine Setup* (see the documentation notes in [Section 15](#) in the *Before You Start... Read These First* section)
- *VSI OpenVMS Calling Standard Manual*
- *VSI OpenVMS Linker Utility Manual*
- *VSI OpenVMS x86-64 Cross-Tools Kit Installation and Startup Guide*

Before You Start... Read These First

Before you begin to download the VSI OpenVMS x86-64 V9.1 installation kit, VSI strongly recommends that you read the notes in this section. The notes provide information about the virtual machine environments tested by VSI, CPU feature checks, terminal emulator settings, licensing on OpenVMS x86-64 systems, describe the new boot method called MemoryDisk, and available networking options.

1. Tested Virtual Environments

VSI OpenVMS x86-64 V9.1 can be installed as a guest operating system on Oracle VM VirtualBox, Red Hat KVM, and VMware virtual machines using the **X86091OE.ISO** file.

For the installation procedure, refer to the *VSI OpenVMS x86-64 V9.1 Field Test Installation Guide*.

VSI has been testing with VirtualBox V6.1.18 or later.

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.

For VMware products, see [Section 1.1](#) below.

1.1 Tested VMware Products and License Types

VSI has tested VSI OpenVMS x86-64 V9.1 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	V12.1.0
ESXi	V6.7.0

Important: Note that *not* all VMware license types are currently supported for running VSI OpenVMS x86-64 V9.1. The following table lists VMware license types that have been tested by VSI:

VMware License	VSI Tested
Enterprise Plus	ESXi V6.7.0 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.

2. Hardware Support

Direct support for x86-64 hardware systems (models to be specified) will be added in later releases. USB support will be addressed after we provide x86-64 hardware system support.

3. MD5 Checksum for the X86091OE.ISO File

VSI recommends that you verify the MD5 checksum of the **X86091OE.ISO** file after it has been downloaded to the target system, on which you will run your virtual machine. The MD5 checksum of **X86091OE.ISO** must correspond to the following value:

BB4163E2BC783410E05B1CC917515BE3

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

4. Non-Intel Processors Are Not Currently Supported

Currently, non-Intel processors are not supported for running VSI OpenVMS x86-64 V9.1. In virtual machine environments, this means that non-Intel processors are not supported on host systems or guest virtual machines for running VSI OpenVMS x86-64.

The support for non-Intel processors will be added in future versions of VSI OpenVMS x86-64.

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 V9.1 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*, is included in the **Helpful_Scripts.zip** file.

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 V9.1, you can issue the following Boot Manager command to list the compatibility details:

```
BOOTMGR> DEVICE CPU
```

VSI OpenVMS x86-64 V9.1 cannot be booted on the system that fails either of the CPU feature checks – the host system check (via the **vmscheck.py** script) or 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.

6. Terminal Emulator Settings

The *OpenVMS V9.0 x86-64 Boot Manager User Guide with Virtual Machine Setup* indicates that 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 Chapter 25 titled “Terminal Emulator Tips” in the *VSI OpenVMS x86-64 Boot Manager User Guide with Virtual Machine Setup* for more details about the settings for emulators.

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:

Session

Connection type: **Raw**

Terminal

Uncheck **Implicit CR in every LF**

Uncheck **Implicit LF in every CR**

Local echo: **Force off**

Local line editing: **Force off** (character mode)

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

Session

Connection type: **Telnet**

Connection → Telnet

Telnet negotiation mode: Switch from **Active** to **Passive** (This yields a connection to a PuTTY window.)

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.

7. x86-64 Licensing

VSI OpenVMS x86-64 V9.1 introduces support for licensing on x86-64 systems. Only Product Authorization Keys (PAKs) with the new X86_64 option keyword will load when running on x86-64 systems.

The LICENSE REGISTER command has been updated to add the X86_64 keyword for the /OPTIONS qualifier.

8. License PAKs for VSI OpenVMS x86-64 V9.1

VSI OpenVMS x86-64 V9.1 includes a pre-populated license database for Field Test. Please note that not all products included in the license database may be available during Field Test.

9. 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 execlt by running SYS\$UPDATE:VMS\$SYSTEM_IMAGES.COM, you must then invoke SYS\$UPDATE:SYS\$MD.COM. For more details, see [Section 1.19](#) in the *Release Notes* section.

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.

10. Networking Options

VSI OpenVMS x86-64 V9.1 provides support for VSI TCP/IP Services and VSI DECnet Phase IV.

VSI TCP/IP Services X6.0-9 is a part of the VSI OpenVMS x86-64 V9.1 installation and will be installed along with the operating system.

VSI DECnet Phase IV can be optionally selected to include when you install the VSI OpenVMS x86-64 V9.1 operating system. The following prompt will be displayed:

```
Do you want to install DECnet Phase IV for OpenVMS X86-64 V9.1?  
(Yes/No) [Yes]
```

VSI recommends that you answer the default `Yes` and install VSI DECnet Phase IV along with the operating system.

10.1 VSI DECnet Phase IV for OpenVMS

VSI OpenVMS x86-64 V9.1 includes support for VSI DECnet Phase IV. Select VSI DECnet Phase IV to include when you install the OpenVMS x86-64 operating system and then configure the product just as you would for an OpenVMS Alpha or Integrity release.

Note: Configuration on a virtual machine requires careful configuration of the NICs. For details, refer to the following documents that are included in the **Helpful_Scripts.zip** file:

- For VirtualBox and KVM, refer to the *How To Configure and Run VSI OpenVMS x86-64 V9.0* document for your VM platform. Each of these documents has a section entitled “Network Configuration – DECnet Phase IV”.
- For VMware, refer to the “Network Configuration – DECnet Phase IV” section in the *VMware, Instructions for Importing the Appliance and Network Setup* document.

After VSI DECnet Phase IV has been installed and configured, you can set host and copy files to/from other Integrity or x86-64 systems running DECnet.

Note: After you install VSI DECnet Phase IV, you *must* update the memory disk to ensure 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
```

Additional information about VSI DECnet Phase IV for OpenVMS can be found on the [VMS Software Documentation](#) webpage.

10.2 VSI TCP/IP Services X6.0-9 TELNET and FTP Available in VSI OpenVMS x86-64 V9.1

VSI OpenVMS x86-64 V9.1 includes VSI TCP/IP Services X6.0-9. The only services supported in X6.0-9 are TELNET and FTP. In order to use secure services such as SSH and SFTP it is necessary to install VSI OpenSSH after you have installed and configured VSI TCP/IP Services X6.0-9. The VSI OpenSSH kit is available for download from the VSI Services Portal.

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
```

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

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

In the [VSI TCP/IP Services for OpenVMS Installation and Configuration](#) manual, refer to Chapter 3 titled “Configuring TCP/IP Services” for detailed information on running the TCPIP\$CONFIG configuration procedure.

In the [VSI TCP/IP Services for OpenVMS Management](#) manual, refer to Chapter 15 titled “Configuring and Managing TELNET” and Chapter 16 titled “Configuring and Managing FTP” for detailed information on TELNET and FTP.

The information in these manuals is applicable to the x86-64 port of VSI TCP/IP Services.

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`. Refer to the [VSI TCP/IP Services for OpenVMS User's Guide](#) for the full description of the SET PASSIVE command.

11. VSI SSL111 V1.1-1K for OpenVMS

VSI OpenVMS x86-64 V9.1 includes VSI SSL111 V1.1-1K for OpenVMS that is based on OpenSSL 1.1.1k.

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

12. VSI Kerberos V3.3-1 for OpenVMS

VSI OpenVMS x86-64 V9.1 includes VSI Kerberos V3.3-1 for OpenVMS.

13. VSI DECwindows Motif V1.7-X for OpenVMS

VSI DECwindows Motif V1.7-X for OpenVMS x86-64 kit is not a standard DECwindows Motif kit. It is provided as a temporary measure until an actual VSI DECwindows Motif for x86-64 systems is available.

This kit contains shareable images and header files that have been designed to allow developers to compile and link their applications cleanly. The images contain the same global symbols as their Alpha and Integrity versions; if called, routines simply return the status `SS$_UNSUPPORTED`.

VSI DECwindows Motif V1.7-X can be optionally selected to include when you install the VSI OpenVMS x86-64 V9.1 operating system. The following prompt will be displayed:

```
Do you want to install DECwindows Motif for OpenVMS X86-64 V1.7-X?  
(Yes/No) [Yes]
```

VSI recommends that you answer the default `Yes` and install VSI DECwindows Motif for OpenVMS x86-64 along with the operating system.

VSI DECwindows Motif V1.7-X for OpenVMS x86-64 kit is needed if you want to run a main program that provides a DECwindows interface and a command line interface. Otherwise, the main program will fail to start due to the lack of the DECwindows shareable images, even though the command line interface is being used. The image activator recognizes that the main program was linked against the DECwindows shareable images and causes the main program to fail.

14. Scripts for Configuring and Running VSI OpenVMS x86-64 in Virtual Machines

The **Helpful_Scripts.zip** file contains the scripts for configuring and creating virtual machines to run VSI OpenVMS x86-64 using the pre-configured virtual appliances (the .OVA files) and scripts for removing the corresponding virtual machines. The zip file also includes the following documents that provide the details on using the scripts and importing the virtual appliances:

- *How to Configure and Run VSI OpenVMS x86-64 V9.0 in a KVM Virtual Machine on Linux*
- *How to Configure and Run VSI OpenVMS x86-64 V9.0 in an Oracle VM VirtualBox Virtual Machine on Linux*
- *How to Configure and Run VSI OpenVMS x86-64 V9.0 in an Oracle VM VirtualBox Virtual Machine on Windows 10*
- *VMware, Instructions for Importing the Appliance and Network Setup*

Note: VSI provided the pre-configured virtual appliances for the V9.0 EAK release series. For VSI OpenVMS x86-64 V9.1, the virtual appliances have been replaced with the ISO installation kit (**X86091OE.ISO**). Therefore, you may refer to these scripts and the related documents for details of the virtual machine configurations, such as the configuration of optional disks, network adapters, or serial devices, but the information about the VSI-supplied virtual appliances should be ignored.

15. Documentation Notes

The **V91_DOCS.zip** includes the *OpenVMS V9.0 x86-64 Boot Manager User Guide with Virtual Machine Setup*. Note that some of the information in this guide is specific to the V9.0 EAK release series. With those releases, VSI supplied the pre-configured virtual appliances (the .OVA files) to run OpenVMS x86-64 in virtual machines.

For VSI OpenVMS x86-64 V9.1, the virtual appliances have been replaced with the ISO installation kit (**X86091OE.ISO**). Therefore, please ignore the information specific to the VSI-supplied virtual appliances.

Please also disregard the commands for configuring and starting VSI TCP/IP that appear in this guide. VSI OpenVMS x86-64 V9.1 includes the new networking product VSI TCP/IP Services X6.0-9. For details, see [Section 10.2](#) of this document.

Refer to *OpenVMS V9.0 x86-64 Boot Manager User Guide with Virtual Machine Setup* for details of the boot process (including the operation of the OpenVMS Boot Manager), virtual machine configurations, and tips for terminal emulator settings. This information remains relevant for the V9.1 release.

Release Notes

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 V9.1.

1.1 Features Not Available in VSI OpenVMS x86-64 V9.1

The following functionality, products, and commands are *not* available in VSI OpenVMS x86-64 V9.1.

- ACME_SERVER (only UAF Login is available)
- Availability Manager
- DEBUG (user-mode symbolic debugger)
- DECdtm Services
- DECnet-Plus
- DECwindows server
- Process swapping (see [Section 1.24](#) 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
- The current cross-compilers do not support VAX floating-point. Do not specify VAX floating-point arguments on any compiler command. VAX floating-point support will be available in a future update for all compilers other than C++.
- Due to the lack of VAX floating-point support, the system routines such as LIB\$WAIT (even using the IEEE input option), CVT\$CONVERT_FLOAT, CVT\$FTOF, and many others will not work as intended since the underlying bit pattern of the floating-point arguments do not match the VAX layout.

1.2 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.

1.3 AUTHORIZE Utility: Exit Sometimes Results in System Crash

When you exit the AUTHORIZE utility after performing a conversational boot with SET/STARTUP OPA0: the system may crash. VSI has observed a few crashes after the following conditions have been met:

1. Perform a conversational boot via SET/STARTUP OPA0:
2. Invoke the AUTHORIZE utility
3. Exit the AUTHORIZE utility

Upon exiting, the system crashes. This problem has only been seen following a conversational boot, using SET/STARTUP OPA0:. It has not been observed when using a FULL or MIN startup.

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

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
*****
```

1.5 BACKUP Utility: Verification Errors for SYS\$EFI.SYS When Copying an x86-64 System Disk

The BACKUP utility will output a verification error message when copying an x86-64 system disk. For example:

```
$ backup/ignore=interlock/image/verify dka0: dka600:
.
.
.
%BACKUP-I-STARTVERIFY, starting verification pass at
1-JUN-2021 16:10:04.28
.
.
.
%BACKUP-E-VERIFYERR, verification error for block 1 of
DKA600: [VMS$COMMON.SYS$LDR]SYS$EFI.SYS;1
.
.
.
$
```

This error message will also be generated when a BACKUP/COMPARE operation is performed.

The message can be safely ignored as it reflects an expected change in the contents of the boot block in the EFI partition of the system disk. VSI will provide an update to BACKUP that recognizes this difference as acceptable in a future release of VSI OpenVMS x86-64.

1.6 CHECKSUM Utility Supports SHA1 and SHA256 Algorithms

In VSI OpenVMS x86-64 V9.1, 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.

Refer to the CHECKSUM command help or the [VSI OpenVMS DCL Dictionary: A-M](#) for information about all supported checksum algorithms.

1.7 Copying Files Between Integrity Systems and x86-64 Systems

Please refer to Chapter 21 in the *OpenVMS V9.0 x86-64 Boot Manager User Guide with Virtual Machine Setup* for information about how to copy files between Integrity systems and x86-64 systems.

1.8 Cross-Tools Kit Update

With VSI OpenVMS x86-64 V9.1, use the new VSI x86-64 cross-tools kit (VSI-I64VMS-X86_XTOOLS-V0901-XFZK-1.ZIP).

For detailed information on the cross-tools included in the V9.1-XFZK kit, refer to the release notes bundled with the kit.

The cross-tools kit also includes non-functional DECwindows Motif sharable images and header files. They have been designed to allow developers to compile and link applications, which call DECwindows Motif routines, without major modifications to the compilation and linking processes used on Itanium systems.

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

1.9 Display of License Charge Information for x86-64 Nodes

In a cluster with x86-64 nodes running VSI OpenVMS V9.1 and Alpha or I64 nodes running previous versions of OpenVMS, the `SHOW LICENCE/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.

1.10 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.

1.11 Extended File Cache (XFC)

VSI OpenVMS x86-64 V9.1 has extended file caching (XFC) enabled by default.

1.12 HYPERSORT Utility Available

The high-performance Sort/Merge utility (HYPERSORT) is available in VSI OpenVMS x86-64 V9.1. Enable the utility with the following command:

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

1.13 Idle CPU Power Saving Mechanism

VSI OpenVMS x86-64 V9.1 is capable of putting the CPU into a low-power (C1) state when it is idle. The power saving mechanism is controlled by the CPU_POWER_MGMT and CPU_POWER_THRSH system parameters as on VSI OpenVMS Integrity systems. This will help reduce power consumption as well as the host CPU utilization in a virtual machine environment.

1.14 Images Linked /SYSEXEC Require to Be Relinked

SY\$BASE_IMAGE.EXE contains a version array that defines compatibility for any images linked /SYSEXEC. In VSI OpenVMS x86-64 V9.1, the version number for all memory management cells has been incremented, requiring all images that link /SYSEXEC and touch memory management cells (for example, MMG\$GQ_PAGE_SIZE) be relinked.

All images that are included in VSI OpenVMS x86-64 V9.1 (including any layered products) have been relinked. If you create or use any additional images that linked /SYSEXEC and reference memory management cells in the base image, you will need to relink them. (If in doubt, relink any image linked /SYSEXEC).

If you do not relink and you try to activate such an image, you will see this error message:

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

Note that there is no functional change associated with the version number change. It is necessary because an internal data structure has been reorganized.

1.15 INSTALL Utility Supports INSTALL /RESIDENT and /SHARED=ADDRESS_DATA

Starting with V9.1, INSTALL /RESIDENT and /SHARED=ADDRESS_DATA are supported and functional on OpenVMS x86-64.

Note that on OpenVMS x86-64, installing images as resident images requires shared address data. This differs from other OpenVMS platforms, where shared address data is not a requirement for images being installed as resident images.

On all OpenVMS platforms, installing an image with shared address data requires that all images, which this image depends on, are installed with shared address data. On OpenVMS x86-64, this means that an image cannot be installed with /RESIDENT if this image depends on a shareable image that is not or cannot be installed with shared address data.

If INSTALL is run with only the /RESIDENT qualifier specified on the command line, /SHARED=ADDRESS_DATA is automatically added, and the following informational message is displayed:

```
%INSTALL-I-SHRADRADDED, '/RESIDENT requires /SHARED=ADDRESS_DATA,  
added for <image_name>'
```

This has been implemented to help to identify problems with installing resident images on x86-64. To avoid the message, either add /SHARED=ADDRESS_DATA, if the image can be installed with shared address data or remove /RESIDENT, if the image cannot be installed with shared address data.

1.16 ISO 9660 Formatted Volume Can Not Be Mounted on x86-64 Systems

The attempt to mount an ISO 9660 formatted volume (on a physical, logical, or virtual device) on VSI OpenVMS x86-64 V9.1 results in failure or system crash.

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

1.17 LIB\$INITIALIZE Requires LINK Qualifier

Programs that use the LIB\$INITIALIZE startup mechanism must explicitly include the LIB\$INITIALIZE module from STARLET.OLB when linking. Traditionally, source programs simply declared an external reference to that module, and the linker automatically included it. 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. This results in the linker not bringing in the LIB\$INITIALIZE module to process the startup routines.

Pascal programs that use the [INITIALIZE] attribute will experience the same behavior since the compiler uses LIB\$INITIALIZE as the underlying mechanism.

As a workaround, specify "SYS\$LIBRARY:STARLET.OLB/INCLUDE=LIB\$INITIALIZE" with your LINK command or options file.

1.18 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.

1.19 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 V9.1 there are three exceptions to the above statement. If you make any of the following changes that affect the boot path or dumping, you need not 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. Modify dump options using the SET DUMP_OPTIONS command. The copy of the data file SYS\$DUMP_CONFIG.DAT on the memory disk is updated directly.
3. Copy a file directly to the memory disk when specifically advised by VSI Support Engineers to do so.

For the V9.1 release, 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 XFKC-N4A.

$
```

1.20 Mount Verification for Tape Devices Causes System Crash

Any operation that invokes mount verification for tape devices including Logical Magtape (LM) causes a system crash with an INCONSTATE bugcheck.

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

1.21 MSCP Served Disks

MSCP served disks are supported on VSI OpenVMS x86-64 V9.1.

Refer to the [VSI OpenVMS Cluster Systems](#) manual for more information on using the MSCP server to make locally connected disks available to all cluster members.

1.22 OpenVMS Clusters

VSI OpenVMS x86-64 V9.1 can be clustered with any OpenVMS system running Version 7.3 or above. VSI has tested 2-node and 3-node clusters, booting from a common system disk, MSCP-served disks where appropriate, CLUSTER_CONFIG.COM, and many relevant SET, SHOW, and SYSMAN commands. Many configurations and options are yet to be tested but the basic capabilities are working and ready for external testing.

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 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, which assume that the new system disk is DKA300: and is already mounted.

1. 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
```

2. 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
```

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

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

1.23 OpenVMS Cluster Usage of LAN Failover and VLAN Devices

When an OpenVMS x86-64 system is in a cluster, LAN Failover and VLAN devices should be configured early in the boot process, but this does not happen. The result is that the cluster software configures PEDRIVER on the members of the LAN Failover set, preventing these devices from joining the LAN Failover set after boot. Also, PEDRIVER does not start on VLAN devices as expected.

The LAN Failover set issue can be worked around by doing an SCACP STOP LAN on the LAN Failover set members, then SCACP START LAN on the LAN Failover device. If any additional protocols have started on the LAN Failover set members, it is necessary to stop these protocols as well.

The LAN VLAN issue can be worked around by doing an SCACP START LAN on the VLAN devices after boot.

Both issues will be addressed in a future release of VSI OpenVMS.

1.24 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 will 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.

The SYSGEN INSTALL /SWAPFILE command is still present on the system and will install a swap file if used. However, OpenVMS x86-64 will never attempt to swap out a process to the swap file. The ability to install a swap file will be disabled in a future release of VSI OpenVMS x86-64.

Processes may be seen in the computable out swapped (COMO) state. This is a transient state for newly created processes. Processes will never appear in the local event flat wait out swapped (LEFO) or hibernate out swapped (HIBO) states. All performance counters associated with swapping are still present in the system. Various MONITOR displays will show swapping metrics. The swapping metrics may be removed from the displays in a future release of VSI OpenVMS x86-64.

1.25 Parallel Processing Library (PPL\$)

The Parallel Processing Library (PPL\$) is available in VSI OpenVMS x86-64 V9.1.

1.26 POSIX Threads Library

The POSIX Threads Library (formerly DECthreads) is available along with kernel threads and upcall support in VSI OpenVMS x86-64 V9.1. To optimally use POSIX threads in applications, compile them with /REENTRANCY=MULTITHREAD and link with /THREADS_ENABLE.

Refer to the [Guide to the POSIX Threads Library](#) (AA-QSBPD-TE) for details. The information in this guide is applicable to the x86-64 port.

1.27 Process Dumps

VSI OpenVMS x86-64 V9.1 provides support for Process Dumps, with the following limitations:

- 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.
- In a threaded process, only the state of the active thread is saved. All memory of the process is saved, but registers in use in other threads may not be available. This support will be added in a future release of VSI OpenVMS x86-64.

1.28 Security Server

The Security Server is enabled for VSI OpenVMS x86-64 V9.1.

Error Message When Displaying Intrusion Database

When issuing the SHOW INTRUSION command after a login failure followed by a successful login, the following message is displayed:

```
%SYSTEM-F-BADCONTEXT, invalid or corrupted context encountered
```

The message is caused by a problem in creating certain intrusion records. This problem will be fixed in a future update for VSI OpenVMS x86-64.

1.29 Reserved Memory

VSI OpenVMS x86-64 V9.1 provides the Reserved Memory support. Use the SYSMAN RESERVED_MEMORY commands to manage the Reserved Memory Registry.

For more information about the Reserved Memory Registry, refer to the [VSI OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems](#).

1.30 Spurious Error Message During Shutdown

When booted from the kit disk, the following message is displayed upon the system shutdown:

```
%DISM-F-SYSDEV, The system device cannot be dismounted  
  
    Shutting down the system  
        .  
        .  
        .  
SYSTEM SHUTDOWN COMPLETE
```

This message may be safely ignored. It will be removed in a future release of VSI OpenVMS x86-64.

1.31 Storage Controllers Without Attached Disks May Have Incorrect Device Names

The OpenVMS Boot Manager may assign incorrect controller letters when enumerating storage controllers that have no attached disks. For example, device names such as DKA100 may appear as DKB100 when OpenVMS x86-64 is booted.

When configuring your virtual machine guest, please avoid inclusion of storage controllers with no disks attached. If you have existing storage controllers with no disks attached, they should be removed from your virtual machine configuration, or have at least one disk attached so that they will be included during boot device enumeration. Additionally, you should avoid defining storage controllers for disk types which are not yet supported by OpenVMS x86-64 as these will also cause enumeration issues. See [Section 1.32](#) below for supported disk types.

If device naming appears incorrect, the Boot Manager allows you to specify the UEFI File System device name (i.e. FS4:) as the boot device in a boot command. For example, BOOTMGR> BOOT FS0:

1.32 Supported Disk Types

VSI OpenVMS x86-64 V9.1 only supports SATA disks. Support for other disk types will be added in future releases of VSI OpenVMS x86-64.

1.33 Symmetric Multiprocessing (SMP)

If you have more than one CPU in your virtual machine configuration, VSI recommends that you keep the CPU count on the virtual machine at least 1 or 2 smaller than the number of cores on your host system. Refer to the *OpenVMS x86-64 V9.0 Boot Manager User Guide with Virtual Machine Setup* for details on how to change the number of CPUs in your virtual machine configuration.

Note: Due to an unresolved issue, you should not configure your system with more than 24 CPUs. VSI is working on resolving this issue.

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 V9.1 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* is not currently supported.

1.34 SYSGEN Parameter Changes

The following changes and additions have been made to the SYSGEN Utility for VSI OpenVMS x86-64 V9.1. For more information about SYSGEN qualifiers and parameters, please see the [VSI OpenVMS System Management Utilities Reference Manual, Volume II: M-Z](#).

Table 1: SYSGEN Qualifiers Used for VSI OpenVMS x86-64

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

Table 2: System Parameters

Parameter	Description								
BOOT_BITMAP1	On x86 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. If this value is too small, the system may be unable to boot. This parameter does not apply to Alpha or Integrity systems.								
BOOT_BITMAP2	On x86 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. If this value is too small, the system may be unable to boot. This parameter does not apply to Alpha or Integrity systems.								
DISABLE_X86_FT	On x86 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 execlret 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 border="1" data-bbox="583 1327 1433 1516"> <thead> <tr> <th>Bit</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>If 1, do not use the XSAVEOPT instruction.</td> </tr> <tr> <td>1</td> <td>If 1, do not use the RDFSBASE, WRFSBASE, RDGSBASE or WRGSBASE instructions.</td> </tr> <tr> <td>2</td> <td>If 1, do not provide software mitigation against the Intel MDS vulnerabilities.</td> </tr> </tbody> </table> DISABLE_x86_FT is a STATIC parameter. This parameter does not apply to Alpha or Integrity systems.	Bit	Definition	0	If 1, do not use the XSAVEOPT instruction.	1	If 1, do not use the RDFSBASE, WRFSBASE, RDGSBASE or WRGSBASE instructions.	2	If 1, do not provide software mitigation against the Intel MDS vulnerabilities.
Bit	Definition								
0	If 1, do not use the XSAVEOPT instruction.								
1	If 1, do not use the RDFSBASE, WRFSBASE, RDGSBASE or WRGSBASE instructions.								
2	If 1, do not provide software mitigation against the Intel MDS vulnerabilities.								
GH_EXEC_CODE_S2	On x86 systems, GH_EXEC_CODE_S2 specifies the size in pages of the execlret 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	<p>On x86 systems, GH_EXEC_DATA_S2 specifies the size in pages of the execler data granularity hint region in S2 space. GH_EXEC_DATA_S2 has the AUTOGEN and FEEDBACK parameters.</p> <p>This parameter does not apply to Alpha or Integrity systems.</p>								
GH_RES_DATA_S2	<p>On x86 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.</p> <p>This parameter does not apply to Alpha or Integrity systems.</p>								
GH_RES_CODE	<p>This parameter now applies to x86 systems. On x86, Integrity, and Alpha systems, GH_RES_CODE specifies the size in pages of the resident image code granularity hint region in S0 space.</p> <p>GH_RES_CODE has the AUTOGEN and FEEDBACK attributes.</p>								
GH_RO_EXEC_S0	<p>On x86 systems, GH_RO_EXEC_S0 specifies the size in pages of the read-only execler data granularity hint region in S0 space. GH_RO_EXEC_S0 has the AUTOGEN and FEEDBACK attributes.</p> <p>This parameter does not apply to Alpha or Integrity systems.</p>								
GH_RO_RES_S0	<p>On x86 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.</p> <p>This parameter does not apply to Alpha or Integrity systems.</p>								
LOAD_SYS_IMAGES	<p>This special parameter is used by VSI and is subject to change. Do not change this parameter unless VSI recommends that you do so.</p> <p>LOAD_SYS_IMAGES controls the loading of system images described in the system image data file, VMS\$SYSTEM_IMAGES. This parameter is a bit mask.</p> <p>The following bits are defined:</p> <table border="1" data-bbox="586 1245 1544 1541"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0 (SGN\$V_LOAD_SYS_IMAGES)</td> <td>Enables loading alternate execlers 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 systems.</td> </tr> <tr> <td>2 (SGN\$V_RELEASE_PFNS)</td> <td>Enables releasing unused portions of granularity hint regions on Alpha servers.</td> </tr> </tbody> </table> <p>These bits are on by default. Using conversational bootstrap exec slicing can be disabled.</p> <p>LOAD_SYS_IMAGES is an AUTOGEN parameter.</p>	Bit	Description	0 (SGN\$V_LOAD_SYS_IMAGES)	Enables loading alternate execlers specified in VMS\$SYSTEM_IMAGES.DATA.	1 (SGN\$V_EXEC_SLICING)	Enables executive slicing. Note that executive slicing is always enabled on x86 systems.	2 (SGN\$V_RELEASE_PFNS)	Enables releasing unused portions of granularity hint regions on Alpha servers.
Bit	Description								
0 (SGN\$V_LOAD_SYS_IMAGES)	Enables loading alternate execlers specified in VMS\$SYSTEM_IMAGES.DATA.								
1 (SGN\$V_EXEC_SLICING)	Enables executive slicing. Note that executive slicing is always enabled on x86 systems.								
2 (SGN\$V_RELEASE_PFNS)	Enables releasing unused portions of granularity hint regions on Alpha servers.								
RAD_SUPPORT	<p>RAD_SUPPORT enables RAD-aware code to be executed on systems that support Resource Affinity Domains (RADs).</p> <p>On x86 systems, the default, minimum, and maximum values for RAD_SUPPORT are all zeros because RAD support is not currently available on that platform.</p>								

SCSBUFFCNT	<p>SCSBUFFCNT is reserved for VSI use only.</p> <p>On x86, Alpha, and Integrity servers, the system communication services (SCS) buffers are allocated as needed, and SCSBUFFCNT is not used.</p>								
VCC_FLAGS	<p>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.</p> <table border="1" data-bbox="586 474 1433 846"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>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.</td> </tr> <tr> <td>1</td> <td>Enables file system data caching and selects the Virtual I/O Cache. This value is relevant only for Alpha systems.</td> </tr> <tr> <td>2</td> <td>Enables file system data caching and selects the extended file cache.</td> </tr> </tbody> </table> <p>Note: On x86 and Integrity servers, the volume caching product [SYS\$LDR]SYSS\$VCC.EXE is not available. XFC caching is the default caching mechanism. Setting the VCC_FLAGS parameter to 1 is equivalent to not loading caching at all or to setting VCC_FLAGS to 0.</p> <p>VCC_FLAGS is an AUTOGEN parameter.</p>	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.
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.								

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.

1.35 System Crash Dumps

VSI OpenVMS x86-64 V9.1 supports a single system crash dump type, 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 V9.1 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 1.19](#) 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.SYSEX]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.

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

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

2. Modify the system parameter DUMPSTYLE to set bit 2. For this example, assume that DUMPSTYLE is at its default setting of 9:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SET DUMPSTYLE 13
SYSGEN> WRITE CURRENT
SYSGEN> EXIT
$
```

Update SYS\$SYSTEM:MODPARAMS.DAT to reflect this change.

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).

In VSI OpenVMS x86-64 V9.1, there is a layout change in the format of a system crash dump, such that older SDA images cannot be used to analyze V9.1 system dumps, and vice-versa. Please be sure to use the correct SDA images for system dump analysis.

1.36 System Service Intercept (SSI)

System Service Intercept (SSI) is available on VSI OpenVMS x86-64 V9.1. SSI enables system services to be intercepted and user-specified code to run before, after, or instead of the intercepted system service.

1.37 Text Editors

The EDT and TPU editors are available in VSI OpenVMS x86-64 V9.1.

1.38 Traceback Support

The linker has been updated to include sufficient traceback information in the image file for a functional symbolic traceback. As a result, the image file may be larger than in previous updates. This additional debug information is not read by the image activator, so it will not slow down normal image activation. This is the same behavior as on OpenVMS Alpha and OpenVMS Integrity server systems.

Traceback now prints the image name, routine name, and line numbers in 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.

1.39 Volume Shadowing

Volume Shadowing is supported on VSI OpenVMS x86-64 V9.1. VSI has tested many configurations including multi-volume shadow sets, booting with a shadowed system disk, dynamic volume expansion, many relevant SET and SHOW commands, and a 6-member shadow set mounted clusterwide using MSCP serving. Many configurations and options are yet to be tested but the basic capabilities are working and ready for external testing.

Refer to the [VSI OpenVMS Volume Shadowing Guide](#) for more information.

1.40 VSI C Run-Time Library (C RTL) Update

VSI OpenVMS x86-64 V9.1 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](#) in this document for more detailed information.

1.41 VSI DECram for OpenVMS

VSI DECram for OpenVMS, also referred to as a RAMdisk, is now fully operational in VSI OpenVMS x86-64 V9.1.

Refer to the [DECram for OpenVMS User's Manual](#) for details of the DECram disk characteristics and configuration.

1.42 VSI TCP/IP Services: Use of SHOW INTERFACE and IFCONFIG -a on x86-64 Systems Under Heavy Traffic

When you issue the TCP/IP commands SHOW INTERFACE or IFCONFIG -a on the OpenVMS x86-64 system that is heavily loaded with FTP traffic, the multiple error messages %LIB-E-KEYNOTFOU, key not found in tree are displayed and followed by the correct command output information. These multiple error messages can be safely ignored.

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

1.43 ZIP/UNZIP Tools

With VSI OpenVMS x86-64 V9.1, VSI provides the Freeware executables for managing ZIP archives on OpenVMS x86-64 systems. These files are located on the **distribution kit** in the following directories:

```
SYSSYSDEVICE:[VMS$COMMON.SYSHLP.UNSUPPORTED.ZIP]
```

```
zip.exe  
zipcloak  
zipnote.exe  
zipsplit.exe  
zip_cli.exe  
zip_msg.exe
```

```
SYSSYSDEVICE:[VMS$COMMON.SYSHLP.UNSUPPORTED.UNZIP]
```

```
unzip.exe  
unzipsfx.exe  
unzipsfx_cli.exe  
unzip_cli.exe  
unzip_msg.exe
```

In order to use these files, copy them from the distribution kit to an appropriate location on your system after you have installed VSI OpenVMS x86-64.

2. Virtualization Notes

The notes in this section describe known issues and limitations when running VSI OpenVMS x86-64 V9.1 as a **guest operating system** in Oracle VM VirtualBox, Red Hat KVM, and VMware virtual machines.

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

VSI OpenVMS x86-64 V9.1 may not correctly maintain the time of day in virtual machine environments. 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 operations with a virtual machine, depending on the virtual machine host. VSI is working on this issue and will deliver an update in a future release of VSI OpenVMS x86-64.

When running VSI OpenVMS x86-64 V9.1 in KVM virtual machines, the system manager should set the SYSGEN parameter PLATF_SPT_D3 to 30 to keep the OpenVMS system time accurate after it has been initially set. This will be addressed in a future release of VSI OpenVMS x86-64.

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 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 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.
```

2.3 VirtualBox: TCP Ports May Become Unusable After Guest Is Terminated

When running VSI OpenVMS x86-64 V9.1 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> changeuartmodel disconnected
vboxmanage controlvm <vmname> changeuartmodel tcpserver <port>
```

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

2.4 VMware Guest May Fail to Boot After Adding Second SATA Controller

It has been reported by an EAK user that their 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 VSI Services Portal.

2.5 Wall-Clock Time Sometimes Runs Slow on Virtual Machine Guests

Wall-clock time may advance slightly slowly on virtual machine guests. This is the time displayed by the DCL command SHOW TIME or obtained by reading EXE\$GQ_SYSTIME.

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

3. Layered and Open Source Products Notes

Layered and open source products for VSI OpenVMS x86-64 V9.1 can be downloaded individually from the VSI Services Portal. For detailed information about the products, please refer to the associated product release notes bundled with the kits.

V9.1 Field Test

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

C99 Update

VSI OpenVMS x86-64 V9.1 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 [C99 Functions](#) 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
#undef 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 %LINK-I-UDFSYM and %LINK-W-USEUNDEF messages for MATH\$ symbols when linking your application, you may add `SYSLIBRARY: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 [Print and scan conversion specifier and argument types](#)) 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 will update the Online Help contents of the C RTL with the functions listed in this document.

C99 Functions

This section describes the C99 functions that have been added to the C RTL. For VSI OpenVMS x86-64 V9.1, 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

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.

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.

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  $(x) \geq (y)$  without an exception if  $x$  or  $y$  is NaN.
isless ()
    determines  $(x) < (y)$  without an exception if  $x$  or  $y$  is NaN.
islessequal ()
    determines  $(x) \leq (y)$  without an exception if  $x$  or  $y$  is NaN.
islessgreater ()
    determines  $(x) < (y) \parallel (x) > (y)$  without an exception if  $x$  or  $y$  is NaN. This macro is not
    equivalent to  $x \neq y$  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.

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.

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.

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.

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.

scalbn, scalbnf, scalbnl, scalbn, scalbnf, scalbnl

Format

```
#include <math.h>
double scalbn (double x, long int n);
float scalbnf (float x, long int n);
long double scalbnl (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 .

strtouf, strtold, wcstof, wcstold

Format

```
#include <stdlib.h>
float strtouf (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 `strtouf` function has variants named `_strtouf32` and `_strtouf64` 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 Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) 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.

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.

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 `wcstoull` function has a variant named `_wcstoull64` for use with 64-bit pointer sizes. See Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) 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*.

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.

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.

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 V9.1, all these changes are included in the C RTL.

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`.

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.

New Flags

The following flags were added to `DLFCN.H`:

- `RTLD_GLOBAL`
- `RTLD_LOCAL`

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;`

New Header

This ECO includes MALLOC.H.

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 and a return value of -1 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
    }
}
```

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
```

New Functions

This section describes the functions that have been added to the C RTL. For VSI OpenVMS V9.0 EAK, 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.

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.

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 Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) 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.

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`.

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 Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) 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_broadcast; /* 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.

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.

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 Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) 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`.

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.

strtoimax, strtoumax

Format

```
#include <inttypes.h>
intmax_t strtoumax(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 Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) for more information on using pointer-size-specific functions.

Description

The `strtoimax` and `strtoumax` functions convert 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 recognize 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.

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 Section 1.9 in [VSI C Run-Time Library Reference Manual for OpenVMS Systems](#) 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.

C RTL Changes

VSI OpenVMS x86-64 V9.1 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. 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
#undef 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)
```

New Functions

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

`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](#) for the `__ALLOCA` macro.

Returns

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

`memcpy`

Format

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

Function Variants

The `memcpy` function has variants named `_memcpy32` and `_memcpy64` for use with 32-bit and 64-bit pointer sizes, respectively.

Description

The `memcpy` 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.

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.

qsort_r

Format

```
#include <stdlib.h>
void qsort_r (void *base, size_t nmem, 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](#). The comparison function takes a third argument. A pointer is passed to the comparison function via *arg*.

Returns

`qsort_r` returns no value.

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.

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.

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.

- 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`

New Header

`ALLOCA.H`.

V9.1 Field Test

Copyright © 2021 VMS Software, Inc., Burlington, Massachusetts, USA

Legal Notice

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

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

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

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

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Apple and macOS are registered trademarks of Apple Computer Inc.

VirtualBox is a registered trademark of Oracle Corporation.

KVM is a registered trademark of Red Hat Inc.

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

PuTTY is copyrighted by Simon Tatham.

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

Motif is a registered trademark of The Open Group.

POSIX is a trademark of The IEEE.

Kerberos is a trademark of the Massachusetts Institute of Technology.

OpenSSL is a registered trademark owned by OpenSSL Software Foundation.