VMS Software

ZeroMQ V4.1-8 for VSI OpenVMS

Release Notes

Publication Date: April 2025

Operating Systems: VSI OpenVMS x86-64 Version 9.2-2 or higher
VSI OpenVMS IA-64 Version 8.4-2L1 or higher
VSI OpenVMS Alpha Version 8.4-2L2

Kit Names: VSI-I64VMS-ZEROMQ-V0401-8A-1.PCSI
VSI-X86VMS-ZEROMQ-V0401-8A-1.PCSI
VSI-AXPVMS-ZEROMQ-V0401-8A-1.PCSI

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

Table of Contents

1. Introduction

Thank your for your interest in this port of ØMQ (ZeroMQ) to OpenVMS. The current release of ZeroMQ (ØMQ) for OpenVMS is based on the ØMQ 4.1.2 distribution.

ØMQ (http://www.zeromq.org) is a messaging system that aims to address many of the problems of more traditional enterprise messaging solutions such as complexity and bloat. ØMQ tackles these issues by taking a different approach. Instead of inventing new APIs and complex wire protocols, ØMQ extends the socket API, eliminating the learning curve and allowing a network programmer to master it in just a few of hours.

The wire protocols employed by ØMQ are deliberately simplistic, even trivial, and performance of ØMQ matches and often exceeds that of raw sockets. Speeds of over 8 million messages per second with a latency of some 12µs have been measured using standard Intel hardware and Linux together with Infiniband. Less spectacular results will be obtained with standard OpenVMS configurations; however good performance combined with the simplicity of the ØMQ programming model make the software an excellent option for the development of any TCP/IP sockets-based application.

This OpenVMS port of ØMQ includes almost all ØMQ. The port presently does not provide support for reliable multicast (via OpenPGM). It is anticipated this and other deficiencies will be addressed in future releases.

2. Acknowledgements

VMS Software Inc. would like to acknowledge the support and assistance of Pieter Hintjens and members of the ØMQ team in the creation of this release.

3. What's New in This Release

For a detailed description of the new features and bug fixes included in this release, please read https://raw.githubusercontent.com/zeromq/zeromq4-1/master/NEWS.

4. Requirements

The kit you are receiving has been compiled and built using the operating system and compiler versions listed below. While it is highly likely that you will have no problems installing and using the kit on systems running higher versions of the products listed, we cannot say for sure that you will be so lucky if your system is running older versions.

- VSI OpenVMS x86-64 Version 9.2-2 or higher
- VSI OpenVMS IA-64 Version 8.4-2L1 or higher
- VSI OpenVMS Alpha Version 8.4-2L2
- VSI TCP/IP Services
- C or C++ compiler for development

In addition to the above requirements, it is assumed that the reader has a good knowledge of OpenVMS and of software development in the OpenVMS environment.

5. Recommended Reading

It is recommended that developers read the very comprehensive documentation on the ØMQ web site (http://www.zeromq.org) before using the software. In addition to programming guides, there are whitepapers and assorted other documents that provide plenty of useful information on how ØMQ can be used.

6. Installing the Kit

The kit is provided as an OpenVMS PCSI kit that can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL ZEROMQ
```

The installation will then proceed as follows (output may differ slightly from that shown):

```
Performing product kit validation of signed kits ...
%PCSI-I-VSIVALPASSED, validation of X86$DKB200:[USER.zeromq]VSI-X86VMS-
ZEROMQ-V0401-8A-1.PCSI$COMPRESSED;1 succeeded

The following product has been selected:
    VSI X86VMS ZEROMQ V4.1-8A              Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and
 for
any products that may be installed to satisfy software dependency
 requirements.

Configuring VSI X86VMS ZEROMQ V4.1-8A

    VMS Software Inc. & iMatix Corporation

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI X86VMS ZEROMQ V4.1-8A              DISK$X86SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...50%...80%...90%...100%

The following product has been installed:
    VSI X86VMS ZEROMQ V4.1-8A              Layered Product

VSI X86VMS ZEROMQ V4.1-8A

    Post-installation tasks are required.


    To start ZeroMQ at system boot time, add the following lines to
    SYS$MANAGER:SYSTARTUP_VMS.COM:
```

```
        $ file := SYS$STARTUP:ZMQ$STARTUP.COM
        $ if f$search("'file'") .nes. "" then @'file'

    To stop ZeroMQ at system shutdown, add the following lines to
    SYS$MANAGER:SYSHUTDWN.COM:

        $ file := SYS$STARTUP:ZMQ$SHUTDOWN.COM
        $ if f$search("'file'") .nes. "" then @'file'
```

7. Post-Installation Steps

After the installation has successfully completed, include the commands displayed at the end of the installation procedure into SYSTARTUP_VMS.COM to ensure that the logical names required in order for users to use the software are defined system-wide at start-up.

Note that in addition to the logical name ZMQ$ROOT (which points to the root of the ØMQ installation tree), the logical name ZMQ$SHR32 and ZMQ$SHR64 is also defined. These logical names point to the shareable images ZMQ$ROOT:[LIB]ZMQ$SHR32.EXE, and ZMQ$ROOT:[LIB]ZMQ$SHR64.EXE which can be linked with application code compiled with 32-bit and 64-bit pointers respectively. Alternatively developers can statically link their code with the corresponding object libraries found in ZMQ$ROOT:[LIB]. Note that only the shareable image and object library are built with 32-bit pointers; use of 64-bit pointers is not supported in the release of ZeroMQ for OpenVMS Alpha.

From a development perspective, it should be noted that symbols in the shareable images and object libraries are mixed-case, and developers should use the C/C++ compiler option **`/NAMES=(AS_IS,SHORTENED)`** or include in their code appropriate `#pragma` directives (C only) to ensure that symbols are correctly resolved when linking. Developers will also need to include in their code header files found in ZMQ$ROOT:[INCLUDE].

8. Privileges and Quotas

Generally speaking there are no special quota or privilege requirements for applications developed using ØMQ, although a high BYTLM is recommended, and SYSPRV, BYPASS, or OPER privilege will be required if ØMQ processes need to utilise privileged ports (ports below 1024).

The following quotas should be more than adequate for most purposes:

```
Maxjobs:         0  Fillm:         256  Bytlm:     128000
Maxacctjobs:     0  Shrfillm:        0  Pbytlm:         0
Maxdetach:       0  BIOlm:         150  JTquota:     4096
Prclm:          50  DIOlm:         150  WSdef:       4096
Prio:            4  ASTlm:         300  WSquo:       8192
Queprio:         4  TQElm:         100  WSextent:   16384
CPU:        (none)  Enqlm:        4000  Pgflquo:   256000
```

9. Sample Applications

The directory ZMQ$ROOT:[PERF] contains several simple example programs that can be used to measure latency and throughput. These examples can be compiled and linked using the provided build procedure (BUILD.COM). Once built, these programs are simple to run. For example, for the latency example, to measure the latency for a 16-bytes message using a sample size of 10000 messages, on one machine we could run the following command:

```
$ MCR []LOCAL_LAT.EXE "TCP://10.1.1.250:5555" 16 10000
```

And on another machine we would enter the following command:

```
$ MCR []REMOTE_LAT.EXE "TCP://10.1.1.250:5555" 16 10000
```

When the run completes, REMOTE_LAT.EXE will display the results as follows (the latency will vary, depending on your specific hardware, operating system, and network configuration):

```
message size: 16 [B]
roundtrip count: 10000
average latency: 263.000 [us]
```

In addition to these examples, additional example code may be found on the ØMQ web site. For example, see https://zeromq.org/get-started/.

10. What's Missing?

As noted previously, the bulk of the ØMQ functionality is present, and it should be possible to do much of what is described on the ØMQ web site. Support for reliable multicast (via OpenPGM) is not currently supported.

ØMQ also supports a range of language bindings, including scripting languages such as Ruby, PHP, and Lua, and 3GL languages such as Ada and FORTRAN. VMS Software Inc. are working to provide similar options on OpenVMS.