VMS Software

Paho-C Version 1.3.15 for VSI OpenVMS

Release Notes

Publication Date: December 2025

Operating Systems: VSI OpenVMS Alpha Version 8.4-2L1 or higher
VSI OpenVMS IA-64 Version 8.4-1H1 or higher
VSI OpenVMS x86-64 Version 9.2-3 or higher

Kit Names: VSI-AXPVMS-PAHO_C-V0103-15-1.PCSI
VSI-I64VMS-PAHO_C-V0103-15-1.PCSI
VSI-X86VMS-PAHO_C-V0103-15-1.PCSI

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

Table of Contents

1. Introduction

Thank you for your interest in this port of the Paho-C MQTT client API to OpenVMS. The current release of Paho-C for OpenVMS is based on the Paho-C 1.3.15 distribution.

The Paho MQTT C API is a fully featured MQTT client written in ANSI standard C. MQTT is a lightweight publish-subscribe protocol for use on top of the TCP/IP protocol. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers.

The MQTT publish-subscribe messaging pattern requires a message broker, which is responsible for distributing messages to interested clients based on the topic of a message. A comprehensive list of currently available MQTT brokers and the features they support can be found at https://github.com/mqtt/mqtt.github.io/wiki/Server%20support.

There are synchronous and asynchronous variants of the API. The synchronous API is generally simpler to use from a programming perspective, with I/O-related calls blocking until the operation in question has completed. In contrast, no calls ever block in the asynchronous API, with notifications of call results or completion being provided asynchronously via callbacks. The choice of API (synchronous or asynchronous) depends on the requirements of the application in question. From an OpenVMS perspective, it should be noted that asynchronous behaviour is achieved using POSIX threads as opposed to ASTs.

This OpenVMS port of the Paho-C MQTT API includes all functionality provided by the Open-Source release, including SSL/TLS support. Additional information about the Paho-C MQTT API and about MQTT in general can be found at https://www.eclipse.org/paho and http://mqtt.org respectively.

2. Acknowledgements

VMS Software Inc. would like to acknowledge the work of the Paho API development team and the Eclipse Foundation for their ongoing efforts in developing and supporting this Open-Source software.

3. What's New in This Release

For details of new features and bug fixes included in this release, please read the project home page (https://www.eclipse.org/paho/) and links contained therein, and review the project repository change log (https://github.com/eclipse-paho/paho.mqtt.c/releases).

This release of Paho-C for VSI OpenVMS includes a language-agnostic API that can be more readily used with common OpenVMS programming languages other than C/C++, such as COBOL, Pascal, FORTRAN, and BASIC. It should be noted that this API currently provides only a limited set of functionality; however, the facilities that are provided will generally be sufficient for most purposes, and it is anticipated that additional functionality will be included in future releases. This language-agnostic API is provided via the shareable image MQTT$CLIENT_SHR.EXE, which can be linked with application programs.

This release of Paho-C for OpenVMS also provides a new logical name PAHO_YIELD_TIMEOUT that can be used to define the timeout used by the function `MQTTClient_yield()`, which is most commonly used (either directly or indirectly) in conjunction with the synchronous API and QoS 1

or 2. By default the timeout used by this function is 100ms, which can effectively limit synchronous throughput to 10 messages per second. The logical name PAHO_YIELD_TIMEOUT can be defined to a smaller value (for example 10 milliseconds) to achieve higher throughput; however, care must be taken to ensure that the defined value is not so low as to cause erroneous timeout errors. The logical name may be defined at any level and the specified value must be a positive integer representing the number of milliseconds to be used for the timeout value.

SSL/TLS support is statically linked into the shareable images LIBMQTTV3CS$SHR.EXE and LIBMQTTV3AS$SHR.EXE and uses OpenSSL 3.0.

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 guarantee functionality if your system is running older versions.

- OpenVMS Alpha V8.4-2L1 or higher, OpenVMS IA-64 V8.4-1H1 or higher, OpenVMS x86-64 V9.2-3 or higher
- VSI TCP/IP
- C compiler – VSI C V7.4-001 or higher
- VSI SSL3 (statically linked into the supplied Paho-C shareable images)

Note

If you wish to statically link application code with the supplied object libraries and require SSL/TLS support, it will be necessary to link with a comparable OpenSSL distribution.

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 documentation on the Eclipse Foundation website (http://wiki.eclipse.org/Paho) and carefully examine the provided sample programs before using the software.

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 PAHO_C
```

The installation will then proceed as follows (output may differ slightly from that shown below depending on platform and other factors):

```
Performing product kit validation of signed kits ...

The following product has been selected:
    VSI I64VMS PAHO_C V1.3.15              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 I64VMS PAHO_C V1.3.15

    VMS Software Inc. & the Eclipse Foundation

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS PAHO_C V1.3.15             DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...10%...40%...50%...90%...100%

The following product has been installed:
    VSI I64VMS PAHO_C V1.3.15             Layered Product

VSI I64VMS PAHO_C V1.3.15

    Post-installation tasks are required.

    To enable the Paho-C runtime at system boot, add the following
    lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

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

    To disable the Paho-C runtime at system shutdown, add the
    following lines to SYS$MANAGER:SYSHUTDWN.COM:

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

6.1. 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 developers to use the software are defined system-wide at start-up.

Note that in addition to the logical name PAHO$ROOT (which points to the root of the Paho-C software installation), the following logical names are defined:

Logical Name	**Description**
LIBMQTTV3A$SHR	Paho-C synchronous API; no TLS/SSL support
LIBMQTTV3AS$SHR	Paho-C asynchronous API; includes TLS/SSL support
LIBMQTTV3C$SHR	Paho-C synchronous API; no TLS/SSL support
LIBMQTTV3CS$SHR	Paho-C synchronous API; includes TLS/SSL support
MQTT$CLIENT_SHR	Language-agnostic API

These logical names can be used by developers to link application code with the appropriate library. Alternatively, developers can statically link their code with the corresponding object libraries found in PAHO$ROOT:[LIB].

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 compiler option **`/NAMES=(AS_IS,SHORTENED)`** or include in their code appropriate `#pragma` directives to ensure that symbols are correctly resolved when linking. Developers will also need to include in their code the header files found in PAHO$ROOT:[INCLUDE].

6.2. Privileges and Quotas

While there are no special quota or privilege requirements for applications developed using the Paho-C library, a high BYTLM is recommended, and SYSPRV, BYPASS, or OPER privilege will be required if applications developed using the library 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
```

7. Sample Applications

The directory PAHO$ROOT:[EXAMPLES] contains several simple example programs that can be used to learn about the API or as a source of inspiration for the development of new applications. These examples can be compiled and linked using the provided build procedure (SAMPLES.COM). Once built, these programs are simple to run. In most cases it will be necessary to run the programs via an appropriately-defined foreign command, and running the program without any additional command line arguments will display usage information for some of the programs. There are also some useful comments included in the code. In order to run the example programs, it is necessary to have access to an MQTT broker.

It should be noted that some of the examples as currently built by SAMPLES.COM are statically linked to the non-TLS/SSL object libraries. To use TLS/SSL it is recommended that applications are linked with one of the shareable images LIBMQTTV3AS$SHR.EXE (asynchronous API) or LIBMQTTV3CS$SHR.EXE (synchronous API), which are statically linked with OpenSSL. These images reside in PAHO$ROOT:[LIB] and can be linked with application code using the logical names LIBMQTTV3AS$SHR and LIBMQTTV3CS$SHR respectively.

The examples directory includes several sample programs written in COBOL and FORTRAN that illustrate use of the language-agnostic API provided by the shareable image MQTT$CLIENT_SHR.EXE. The build script SAMPLES.COM illustrates how to compile and link these examples, and the function of the examples is summarised in the table below.

Program	**Notes**
cob_client_pub.cob, for_client_pub.for	A simple publisher example. A single message is published using QoS (Quality of Service) level 1 (at least once delivery).

Program	Notes
cob_client_pub_simple.cob, for_client_pub_simple.for	Another simple publisher example, similar to the above example, but using a somewhat simplified call sequence to publish the message.
cob_client_sub.cob, for_client_sub.for	A basic subscriber that can be used to consume messages published with either of the above publisher examples.
cob_client_sub_async.cob, for_client_sub_async.for	A simple asynchronous consumer that consumes messages on a separate thread from the main program. This example can be used to consume messages published by either of the publisher examples.