LIBPQ Version 18.1 for OpenVMS

Release Notes

Publication Date: January 2026

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

Kit Names: VSI-AXPVMS-LIBPQ-V1801-0-1.PCSI
VSI-I64VMS-LIBPQ-V1801-0-1.PCSI
VSI-X86VMS-LIBPQ-V1801-0-1.PCSI

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

LIBPQ Version 18.1 for OpenVMS Release Notes

VMS Software

Table of Contents

1. Introduction

Thank you for your interest in the LIBPQ PostgreSQL client API and embedded SQL pre-processor utility for OpenVMS. LIBPQ is the C application programmer's interface to PostgreSQL and includes a set of library functions that allow client programs to pass queries to the PostgreSQL backend server and to receive the results of these queries.

This release of LIBPQ for OpenVMS is based on the PostgreSQL 18.1 open-source distribution. It includes the client API and several utility programs, such as the ECPG embedded SQL pre-processor for developing PostgreSQL client applications in C/C++ using embedded SQL statements and the PSQL interactive query tool.

This release also includes updates provided by Sector7. The updates resolve issues related to the maximum number of arguments allowed in OpenVMS function calls. This enables database queries that return large numbers of columns and null-indicator values. Additionally, this release includes bug fixes for problems observed with the PostgreSQL timestamp data type, and the shareable image LIBPQ$SHR provides additional functions required by certain client applications.

More information about the LIBPQ C/C++ client API and the ECPG pre-processor utility can be found at https://www.postgresql.org/docs/18/index.html.

2. Acknowledgements

VMS Software Inc. acknowledges the Sector7 development team for their contributions to the ECPG embedded SQL processor and associated API code. These changes negate issues related to the maximum number of arguments that can be specified in a function call on OpenVMS.

3. Requirements

This version of LIBPQ has been compiled and built using the operating system and compiler versions listed below. Although the kit is expected to perform correctly on systems with newer versions of the operating system or products listed, VSI cannot guarantee successful installation or operation on older versions.

- OpenVMS 8.4-2L1 or higher (IA-64), OpenVMS V8.4-2L1 or higher (Alpha), OpenVMS 9.2-3 or higher (x86-64)
- VSI TCP/IP
- C compiler for application development

In addition to the above requirements, it is assumed that the reader has a solid understanding of OpenVMS and its software development environment. It is also assumed that the reader is familiar with the use of the PostgreSQL client API and the embedded SQL pre-processor utility.

4. Recommended Reading

It is recommended that developers read online the documentation for LIBPQ and the ECPG utility available at https://www.postgresql.org/docs/18/index.html and examine the samples programs provided with the LIBPQ OpenVMS kit before using the software.

5. Installing the Kit

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

```
$ PRODUCT INSTALL LIBPQ
```

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

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

The following product has been selected:
    VSI I64VMS LIBPQ V18.1-0              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 LIBPQ V18.1-0: Libpq for OpenVMS is based on
 PostgreSQL 18.1

    © Copyright 2025 VMS Software Inc.

    VSI Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS LIBPQ V18.1-0              DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...20%...40%...50%...70%...90%...100%

The following product has been installed:
    VSI I64VMS LIBPQ V18.1-0              Layered Product

VSI I64VMS LIBPQ V18.1-0: Libpq for OpenVMS is based on PostgreSQL 18.1

    Post-installation tasks are required.
To start the Libpq runtime at system boot time, add the following
    lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

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

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

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

5.1. Post-installation Steps

After the installation has successfully completed, include the commands displayed at the end of the installation procedure into SYSTARTUP_VMS.COM. This ensures that the logical names required for users to run the software are defined system-wide at startup.

In addition to the system logical name LIBPQ$ROOT (which points to root directory of the LIBPQ installation tree), the logical name LIBPQ$SHR is also defined. This logical name points to the shareable image LIBPQ$ROOT:[LIB]LIBPQ$SHR.EXE, which can be linked with application code. Alternatively, it is possible to statically link application code with the object libraries found in the LIBPQ$ROOT:[LIB] directory.

From a development perspective, the shareable image and object libraries use mixed-case symbols. The application developers must therefore use the **`/NAMES=(AS_IS)`** compiler option or include appropriate directives in their code to ensure correct symbol resolution during linking. Developers will also need to include in their code one or more of the header files found in LIBPQ$ROOT:[INCLUDE]. The example build procedure (see LIBPQ$ROOT:[EXAMPLES]EXAMPLES.COM) illustrates how programs must be compiled and linked when using the library.

5.2. Privileges and Quotas

In general, applications developed using LIBPQ do not require special quotas or privileges. However, a reasonably high BYTLM quota is recommended, particularly for database operations that transfer large volumes of data. The following quotas should be sufficient for most use cases:

```
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
```

6. Sample Applications

The LIBPQ$ROOT:[EXAMPLES] directory provides simple example programs for learning the API or serving as a basis for new application development. These examples can be compiled and linked using the provided build procedures (EXAMPLES.COM), which also illustrates the required compile and link process for LIBPQ applications. All examples are linked with the LIBPQ$SHR shareable image; however they could instead be statically linked with the various object libraries found in LIBPQ$ROOT:[LIB].

For production deployments, it may be preferable to statically link applications. This approach avoids the need to install LIBPQ on production systems. An example demonstrating static linking with the required object libraries is provided in the comments of EXAMPLES.COM. When using this approach, SSL libraries must be explicitly linked. This is not required when linking with LIBPQ$SHR.EXE, as the shareable image statically links in these libraries.

7. What is Missing?

The supplied kit for OpenVMS includes all functionality supported by the PostgreSQL C/C++ client API and embedded SQL pre-processor, including Oracle Pro*C compatibility. Future releases of the software may include additional OpenVMS-specific functionality, such as a wrapper API to facilitate using the API from languages other than C/C++ such as COBOL, FORTRAN, Pascal, and BASIC.

8. Other Points to Note

- LIBPQ is built using IEEE floating-point format. Linking these libraries with code compiled using a different floating-point format may produce unexpected results. It is recommended that all code use the IEEE format to avoid such situations.
- To ensure correct parsing of command line arguments when using the ECPG and PSQL tools, it is recommended that users set process parse style to `extended` (`set process/parse_style=extended`) or enclose command line arguments and options in double quotes.