vms Software

LIBPQ for OpenVMS I64

April 2023

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 11.0 open source distribution and includes the client API along with several utility programs such as the ECPG embedded SQL pre-processor utility 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 that negate issues associated with the maximum number of arguments that can be supplied to a function call on OpenVMS, making it possible perform 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.

Additional information about the LIBPQ C/C++ client API and the ECPG pre-processor utility can be found at https://www.postgresql.org/docs/11/static/.

2. Acknowledgements

VMS Software Inc. would like to acknowledge and thank Sector7 for contributing changes to the ECPG embedded SQL processor and associated API code that negate issues associated with the maximum number of arguments that can be specified in a function call on OpenVMS.

3. 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.

- OpenVMS 8.4-1H1 I64
- HP TCP/IP Services V5.7 ECO 2

  It has not been verified whether the kit works with the MultiNet TCP/IP stack, but there is a good chance that it will.
- C compiler - HP C V7.3-018

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. 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 the documentation for LIBPQ and the ECPG utility available at https://www.postgresql.org/docs/11/static/ 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 (`VSI-I64VMS-LIBPQ-V1100D-0-1.PCSI`) that can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL LIBPQ
```

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

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

The following product has been selected:
    VSI I64VMS LIBPQ V11.0-0D              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 V11.0-0D: Libpq for OpenVMS is based on
PostgreSQL 11

    © Copyright 2023 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 V11.0-0D              DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...20%...50%...60%...70%...90%...100%

The following product has been installed:
    VSI I64VMS LIBPQ V11.0-0D              Layered Product

VSI I64VMS LIBPQ V11.0-0D: Libpq for OpenVMS is based on PostgreSQL
11

    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` to ensure that the logical names required in order for users to use the software are defined system-wide at start-up.

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, it should be noted that symbols in the shareable image and object libraries are mixed-case, and application developers must therefore use the compiler option `/NAMES=(AS_IS)` or include in their code appropriate directives to ensure that symbols are correctly resolved when 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

Generally speaking there are no special quota or privilege requirements for applications developed using LIBPQ, although a reasonably high `BYTLM` is recommended, particularly if database operations will transfer large amounts of data. 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
```

6. Sample applications

The directory `LIBPQ$ROOT:[EXAMPLES]` contains several simple example programs that can be used to learn about the API or as a source of inspiration for the development of new applications. These examples can be compiled and linked using the provided build procedures (`EXAMPLES.COM`), which also serves to illustrate how applications using LIBPQ must be compiled and linked. All examples are linked with the `LIBPQ$SHR` shareable image; however they could instead be statically linked with the various object libraries found in `LIBPQ$ROOT:[LIB]`.

Note that from a production deployment perspective it may in fact be preferable to statically link applications, as this will avoid the need to install LIBPQ on production systems. An example is provided (see comments in `EXAMPLES.COM`) that illustrates how to statically link your application code with the relevant object libraries. Note that it is necessary to link in

SSL. This is not required when linking with `LIBPQ$SHR.EXE` as the shareable image statically links in these libraries.

7. What's 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

- It should be mentioned that LIBPQ has been built using IEEE floating point and linking the libraries with code compiled to use another floating point formats may produce unexpected results. It is recommended that all code use 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.