VMS Software

SQL Relay Version V2.0-0J for VSI OpenVMS IA-64 and x86-64

Release Notes

Publication Date: September 2026

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

Kit Names: VSI-X86VMS-SQLRELAY-V0200-0J-1.PCSI
VSI-I64VMS-SQLRELAY-V0200-0J-1.PCSI

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

Table of Contents

1. Introduction 3
2. Acknowledgements 3
3. What's New in This Release 3
4. Requirements 4
5. Recommended Reading 4
6. Installing the Kit 4
7. Post-Installation Steps 5
8. Privileges and Quotas 6
9. Sample Applications 7
10. Known Issues and Limitations 7

1. Introduction

Thank you for your interest in this port of the SQL Relay client API to VSI OpenVMS IA-64 and x86-64. The current release of the SQL Relay client API for OpenVMS is based on the SQL Relay 2.1.1 open-source distribution.

SQL Relay is an open-source database connection management solution that resides between your application and the database, providing functionality not typically provided by the database directly, including persistent database connection pooling, proxying, throttling, high availability, query routing, query filtering, query translation, and connection scheduling. Of particular interest from an OpenVMS perspective are the proxying capabilities of SQL Relay, which can be used to facilitate access to databases from unsupported platforms. Databases that can be accessed via SQL Relay using the client API include Oracle, Sybase, Microsoft SQL Server, IBM DB2, MySQL, MariaDB, PostgreSQL, Firebird, and SQLite, as well as ODBC data sources. For more information see http://sqlrelay.sourceforge.net/index.html.

This OpenVMS port of the SQL Relay client API includes all functionality provided by the open-source release, including SSL/TLS support (based on OpenSSL 3.0.5). Also included is a wrapper API that makes it easier to use the SQL Relay client API with OpenVMS programming languages other than C/C++, such as COBOL, FORTRAN, Pascal, and BASIC. Additionally, the kit provides several SQL Relay command line tools, including sqlrsh (an interactive tool similar to Oracle SQL*Plus), and sqlr-export and sqlr-import, which can be used to export and import data from XML files on a per-table basis. Additionally, this release includes a beta version of an embedded SQL pre-processor (PRESQLR.EXE) that can be used to port existing Oracle C and COBOL applications that use embedded SQL code to SQL Relay, in place of using the Oracle pre-processor tools and client API[1].

2. Acknowledgements

VMS Software Inc. would like to acknowledge the work of David Muse and the Firstworks SQL Relay development team (http://www.firstworks.com/index.html) for their ongoing efforts in developing and supporting this open source software.

3. What's New in This Release

SQL Relay for OpenVMS V2.0-0J includes the following improvements and bug fixes to the ESQL2 component:

- Username, password, host, and port can now be combined into a single connection string in ESQL2 for Cobol.
- Implemented C typedef statements support in ESQL2.
- Improved handling of the parameters in the SELECT queries.
- Improved handling of EXEC SQL INCLUDE commands in ProC.
- Previously, SQL Relay did not display any errors upon receiving an incorrect username or password. This issue has been fixed.

[1]It should be noted that the pre-processor in its current form is intended for use with embedded SQL code that will interact with an Oracle database. While it is possible to use the tool in conjunction with another database, correct operation of the resultant generated code is not guaranteed.

The previous release, SQL Relay for OpenVMS V2.0-0A, also included improvements and bug fixes to the ESQL2 component. For details, refer to *SQL Relay V2.0-0E for OpenVMS Release Notes* [https://docs.vmssoftware.com/sql-relay-v2-0-0e-for-vsi-openvms-ia-64-and-x86-64-release-notes/] .

For a detailed description of the features and bug fixes included in the open source SQL Relay 2.1.1, refer to the ChangeLog file in the source repository (git://git.code.sf.net/p/sqlrelay/sqlrelay) and the version 2.1.1 release announcement (http://software.firstworks.com/2025/05/sql-relay-211-release-announcement.html).

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 operating system or products listed, we cannot guarantee functionality if your system is running older versions.

- VSI OpenVMS x86-64 V9.2-3 or higher or VSI OpenVMS IA-64 Version 8.4-2L1 or higher
- Rudiments Library Version 2.1.0.
- VSI TCP/IP
- VSI SSL Version 3.0-18 or higher
- C, CXX, COBOL, or other language compiler (depending upon which language or languages you intend to use to develop applications using the SQL Relay client API)

Note that if you wish to statically link application code with the supplied SQL Relay 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 and administrators read the extensive documentation provided on the SQL Relay web site (http://sqlrelay.sourceforge.net/documentation.html) and that developers carefully examine the provided sample programs before using the software.

6. Installing the Kit

This field test release includes the following architecture-specific kits:

- VSI-I64VMS-SQLRELAY-V0200-0J-1.PCSI for OpenVMS IA-64
- VSI-X86VMS-SQLRELAY-V0200-0J-1.PCSI for OpenVMS x86-64

The kit can be installed by a suitably privileged user via the following command:

```
$ PRODUCT INSTALL SQLRELAY
```

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 ...
%PCSI-I-VSIVALPASSED, validation of $1$DGA80:[DEV.KITS]VSI-I64VMS-SQLRELAY-
V0200-0J-1.PCSI$COMPRESSED;1 succeeded
```

```
The following product has been selected:
    VSI I64VMS SQLRELAY V2.0-0J            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 SQLRELAY V2.0-0J

    VMS Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS SQLRELAY V2.0-0J            DISK$I64V842L1SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...30%...40%...50%...60%...70%...90%...100%

The following product has been installed:
    VSI I64VMS SQLRELAY V2.0-0J            Layered Product

VSI I64VMS SQLRELAY V2.0-0J

    Post-installation tasks are required.


    To enable SQLRelay at system boot time, add the following lines to
    SYS$MANAGER:SYSTARTUP_VMS.COM:

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

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

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

Note

In addition to installing the SQL Relay client API, on OpenVMS it is also necessary to install and configure the SQL Relay server on the Linux or Windows server hosting the target database, or alternatively to build and use the Docker container as described above.

A detailed description of installing and configuring the SQL Relay server software is beyond the scope of this document, and the reader should refer to the documentation available on the SQL Relay web site (http://sqlrelay.sourceforge.net/documentation.html).

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

In addition to the logical name SQLR$ROOT (which points to the root directory of the SQL Relay client software installation), the logical names SQLR$SHR and SQLR$LIBESQL_SHR are also defined. The logical name SQLR$SHR points to the shareable image SQLR$SHR.EXE, located in SQLR$ROOT:[LIB], which can be linked with application code that uses the SQL Relay client API. Alternatively, it is possible to statically link application code with the object libraries found in the SQLR$ROOT:[LIB] directory. The logical name SQLR$LIBESQL_SHR points to the shareable image SQLR$ROOT:[LIB]SQLR$LIBESQL_SHR.EXE, which must be linked with application code containing embedded SQL statements that is built using the PRESQLR.EXE embedded SQL pre-processor.

Other logical names defined by the SQL Relay start-up script are as follows:

Logical Name	**Purpose**
SQLR$BASIC	Location of the include file sqlrdef.bas for use with BASIC applications.
SQLR$BIN	Location of SQL Relay command line utilities (sqlr-export.exe, sqlr-import.exe, sqlrsh.exe).
SQLR$COBOL	Location of the include file sqlrdef.cob for use with COBOL applications.
SQLR$EXAMPLES	Location of example programs in BASIC, FORTRAN, Pascal, COBOL, and C.
SQLR$FORTRAN	Location of the include file sqlrdef.for for use with FORTRAN applications.
SQLR$LIB	Location of SQL Relay client API object libraries and shareable image.
SQLR$PASCAL	Location of the include file sqlrdef.pas for use with Pascal applications.

From a development perspective, for C/C++ programs it should be noted that symbols in the shareable image and object libraries are mixed-case, and developers should therefore use the C and 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 the appropriate language header file, from SQLR$ROOT:[INCLUDE]. Symbols for the wrapper API that can be used with other OpenVMS programming languages are all upper-case.

Note that the C header file SQLR$ROOT:[INCLUDE.SQLRELAY]SQLR_WRAP_LIB.H must be included by C source code modules containing embedded SQL code that is pre-processed by the PRESQLR.EXE embedded SQL pre-processor.

8. Privileges and Quotas

Generally speaking there are no special quota or privilege requirements for applications developed using the SQL Relay client API, although 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
```

9. Sample Applications

The directory SQLR$ROOT:[EXAMPLES] contains several simple example programs written in C, CXX, COBOL, FORTRAN, BASIC, and Pascal that serve to illustrate the usage of the API. The command procedure SQLR$ROOT:[EXAMPLES]BUILD_EXAMPLES.COM can be used to build all of the example programs, and assumes that you have all of the relevant language compilers installed on the system in question. If this is not the case, it will be necessary to modify the command procedure to remove or comment out any language examples that you do not wish to build.

These simple example programs are intended to provide an introduction to the SQL Relay API and to hopefully serve as a basis for the development of more sophisticated applications. Note that it will be necessary to have available an appropriately configured SQL Relay server environment in order to run the example programs, and the example programs will need to be modified to specify the correct username, password, network address, and database connection details for your environment.

The directory SQLR$ROOT:[EXAMPLES.ESQL] contains a simple example build procedure that illustrates how to build applications using the PRESQLR.EXE embedded SQL pre-processor. The example build procedure illustrates some of the command line options that may be used with PRESQLR.EXE; in order to see the full list of parameters and options, run the pre-processor without specifying any parameters.

Note that it is mandatory to specify an input file using the **/INAME** qualifier. All other qualifiers are optional, and will default to the values shown, where appropriate.

10. Known Issues and Limitations

The supplied kit for VSI OpenVMS includes all functionality supported by the open-source SQL Relay client API. In addition, the port includes a language-agnostic API that makes it straightforward to write applications using 3GL languages such as COBOL, FORTRAN, Pascal, and BASIC. This release of SQL Relay for VSI OpenVMS also includes a beta version of the PRESQLR.EXE embedded SQL pre-processor tool for COBOL and C.

The following list identifies any known issues and limitations associated with this release of SQL Relay for VSI OpenVMS:

- When a bind variable that is part of a COBOL record is used in SQL statements, the generated COBOL code will not be correct in all cases.

  This bug will be fixed in an upcoming update.

- Support for the use of FOR within the INSERT and UPDATE SQL statements has been implemented in a simplistic manner and may not be optimal for some situations.

- The PRESQLR.EXE embedded SQL pre-processor for SQL Relay on VSI OpenVMS currently supports only the COBOL and C programming languages. Support for additional languages (and in particular FORTRAN) is expected to be included in future releases.

- The PRESQLR.EXE embedded SQL pre-processor currently provides support for a subset of Oracle RDBMS SQL statements.[2] Currently supported statements include those for connection and transaction handling (CONNECT, COMMIT, ROLLBACK), cursors (DECLARE,OPEN, FETCH, CLOSE), basic DML, DQL, and DDL operations (INSERT, UPDATE, DELETE, SELECT, CREATE, and DROP), PL/SQL (with some limitations), partial support for BLOBs, and standard embedded SQL constructs (INCLUDE SQLCA, BEGIN DECLARE SECTION, END DECLARE SECTION). Support for additional Oracle RDBMS SQL statements will be provided in future releases of the software.

- The embedded SQL CONNECT statement for SQL Relay is somewhat different than the CONNECT statement for Oracle. Specifically, it is assumed that all databases are remote, and the connect string must contain connection details for the remote SQL Relay server in the form *host*:*port*, where *host* may be the name or IP address of the server running the SQL Relay server, and *port* is the port number being used by the SQL Relay server process. The connect string can be specified via a host variable or a string constant, or alternatively it can be specified via a PRESQLR.EXE command line parameter. It should be noted that SQL statements support named connections for parallel activity across the same or different databases using the AT clause, where named connections can be defined and created with the CONNECT statement.

Subsequent releases of the SQL Relay client for OpenVMS will include additional development tools and utilities, including embedded SQL processors for other programming languages and an enhanced interactive query tool. It is anticipated that the inclusion of these facilities and the scope of the functionality they provide will evolve over the course of several product releases.

[2]For example, declaring and using cursors for UPDATE and DELETE is not currently supported.