vms Software

SQL Relay Client for OpenVMS I64 and Alpha (Beta)

December 2021

VSI-AXPVMS-SQLRELAY-V0109-0H-1.PCSI
VSI-I64VMS-SQLRELAY-V0109-0H-1.PCSI

1. Introduction

Thank you for your interest in this port of the SQL Relay client API to VSI OpenVMS I64 and Alpha. The current release of the SQL Relay client API or OpenVMS is based on the SQL Relay 1.9.0 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 1.1.1l). 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

For a detailed description of the features and bug fixes included in this release, please refer to the `ChangeLog` file in the source repository (git://git.code.sf.net/p/sqlrelay/sqlrelay).

[1] It should be noted that the pre-processor in its current form is intended for use with 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.

As noted previously, this release of the SQL Relay client for VSI OpenVMS includes a beta version of an embedded SQL pre-processor that can be used to port existing COBOL and C applications that use embedded SQL code to SQL Relay, in place of using the Oracle pre-processor tools and client API. This release includes various bug fixes and enhancements, including improved logging of errors and warnings.

Due to changes to the underlying SQL Relay OpenVMS API made as a consequence of these and other enhancements, it is recommended that existing application programs be rebuilt using the updated libraries.

Additional new features included in this release are as follows (see release notes for previous version for details of new features added for those versions):

- Added support for the Oracle Communications Area (ORACA) with COBOL and C. Note that this support does not currently include cursor cache statistics.
- Added the `/VERBOSE` qualifier for the `PRESQLR` embedded SQL pre-processor utility. Permitted values for this qualifier are 0, 1, and 2, with the verbosity of output increasing with the value specified. A value of 0 causes errors only to be displayed; a value of 1 causes both errors and warnings to be reported; and a value of 2 displays error and warnings plus any parser errors. If this qualifier is not specified, then the default behavior is to display errors only.
- A listing file is now generated by `PRESQLR` that contains details of any errors encountered during processing. Errors are highlighted in the listing file for ease of identification.
- The `PRESQLR` utility was treating cursor names in a case sensitive manner, causing previously declared cursors not to be correctly identified. This issue has been resolved.
- `PRESQLR` now permits embedded SQL variable declarations to statements be specified in the COBOL working storage section.
- `PRESQLR` now correctly reports an error when code attempts to use a cursor that has not been previously declared.
- `PRESQLR` now continues processing to the end of the file after encountering any errors, making it easier to identify and correct multiple (and possibly related) problems.
- Updated examples.
- Miscellaneous bug fixes and enhancements.

Also provided with this release is a Dockerfile that can be used to build a Docker image containing the SQL Relay server and Oracle Instant Client. Deploying these components via a Docker container provides an easy way to set up and use the SQL Relay server, with configuration and use equating to little more than modifying the SQL Relay server configuration file to specify the correct connection details, and running a small handful of Docker commands to build and start the container image[2]. The Dockerfile and instructions for building the image can be found at https://github.com/vmssoftware/sqlr-ora.

[2] Docker must be installed on the system where you intend to run the container image. Either a Microsoft Windows, macOS, or Linux system may be used for this purpose. See https://www.docker.com/ for details regarding the installation and use of Docker.

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 say for sure that you will be so lucky if your system is running older versions.

- VSI OpenVMS Version 8.4-2L1 or higher
- VSI TCP/IP Services for OpenVMS, HPE TCP/IP Services for OpenVMS, or MultiNet TCP/IP
- C, COBOL, or other language compiler (depending upon which language or languages you intend to use to develop applications using the SQL Relay client API)
- OpenSSL 1.1.1l or higher (statically linked into the supplied SQL Relay client API shareable images)

  Note that if you wish to statically link application code requiring 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 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

The kit is provided as an OpenVMS PCSI kit (`VSI-I64VMS-SQLRELAY-V0109-0H-1.PCSI` or `VSI-AXPVMS-SQLRELAY-V0109-0H-1.PCSI`, depending on platform) that can be installed by a suitably privileged user using 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 ...

The following product has been selected:
    VSI I64VMS SQLRELAY V1.9-0H            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 V1.9-0H

    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 V1.9-0H            DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...20%...50%...70%...80%...90%...100%

The following product has been installed:
    VSI I64VMS SQLRELAY V1.9-0H            Layered Product

VSI I64VMS SQLRELAY V1.9-0H

    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).

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.

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.

6.2. 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
```

7. Sample applications

The directory SQLR$ROOT:[EXAMPLES] contains several simple example programs written in C, COBOL, FORTRAN, BASIC, and Pascal that serve to illustrate the usage of the API. The command procedure SQLR$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.

8. 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:

- 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) will be included in future releases.
- The `PRESQLR.EXE` embedded SQL pre-processor currently provides support only for a subset of Oracle RDBMS SQL statements[3]. 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) 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.

  Specific enhancements planned for subsequent releases include:

  - Additional statements:
    - `LOB WRITE`
    - `LOB READ`
    - `VAR`
    - `DECLARE` (partially implemented)
  - Types and pseudo-types:

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

    - VARYING (COBOL) / VARCHAR (C)
    - SQL-ROWID
    - SQL-CLOB
    - SQL-NCLOB
- 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 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 addition 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.