vms Software

gSOAP for OpenVMS I64, Alpha, and x86-64

April 2024

```
VSI-AXPVMS-GSOAP-V0208-129B-1.PCSI
VSI-I64VMS-GSOAP-V0208-129B-1.PCSI
VSI-X86VMS-GSOAP-V0208-129B-1.PCSI
```

The gSOAP Toolkit for SOAP Web Services and XML-Based Applications is a cross-platform open source C and C++ software development toolkit. It generates C/C++ RPC code, XML data bindings, and efficient schema-specific parsers for SOAP Web services and other applications that benefit from an XML interface.

1. Introduction

Thank you for your interest in this port of gSOAP to OpenVMS. The current release of the OpenVMS gSOAP port is based on the 2.8.129 gSOAP distribution.

The initial motivation for porting gSOAP to OpenVMS was a question from a customer asking how they could call a remote Web service from their OpenVMS COBOL/ACMS application. A number of OpenVMS users have crafted novel solutions to address this type of problem; however, gSOAP potentially provides a more complete, proven, and flexible solution. Since legacy applications can easily call C libraries, gSOAP is well positioned to act as a Web service client for legacy applications written in any OpenVMS 3GL.

In addition, gSOAP may be used to implement Web services on the OpenVMS platform. The other popular technology for implementing Web services on OpenVMS is the Java implementation of Apache AXIS2 used in conjunction with Apache Tomcat. These offerings provide OpenVMS users with web service technology based on their choice of programming language and environment.

The following notes briefly describe how to install the gSOAP kit and how to get started with the simple example applications. If you have any trouble with the kit, have suggestions for how it might be improved, or would like a distribution for another version of the operating system, please let us know, and we will do our best to oblige.

Finally, be sure to read the gSOAP documentation (see https://www.genivia.com/docs.html for details), specifically the gSOAP User Guide. Aside from the handful of OpenVMS-specific functions that have been added as part of the porting exercise, there should be little or no variance in terms of usage from what is described in the User Guide (and if there is, we would like to hear about it).

1.1. What's new in this release?

This release is based on gSOAP 2.8.129, which is a significant jump from the previous release. Please refer to the documentation at https://www.genivia.com/changelog.html for a description of these enhancements.

Specific changes in this VSI OpenVMS V2.8-129B kit include a fix to prevent undefined symbols when compiling and linking C++ examples, and calls to `setsockopt()` to set send and receive buffer sizes have been disabled, making it no longer required to have elevated privileges in order to run gSOAP applications.

1.2. Outstanding issues

- When using gSOAP on systems that employ On Disk Structure Level 2 (ODS-2), the `soapcpp2` utility will fail if generating sample XML message files and there are Web Service methods with names longer than 35 characters. The workaround to this problem is to specify the `-x` command line option, which will suppress the generation of sample XML message files. This issue may be resolved in future releases of gSOAP for OpenVMS; however, installation on an ODS-5 file system is recommended.
- When using gSOAP servers in CGI mode with either Apache HTTPD or WASD, problems can be encountered with the output of the CGI program being truncated. These problems are caused by inappropriate interpretation of carriage-control characters. To avoid these problems, include the following code before calling `soap_serve`():

```
stdout = freopen("SYS$OUTPUT", "w", stdout, "ctx=bin", "ctx=xplct");
```

2. Requirements

The kit you are receiving requires the following products and product versions to be installed for the software to operate correctly. In general, it will also be possible to install and use the software on higher versions of the operating system and other required products.

- OpenVMS 8.4-2L1 or higher (I64), OpenVMS V8.4-2L1 or higher (Alpha), OpenVMS 9.2-1 or higher (x86-64)
- VSI TCP/IP
- C compiler
- Optional components:
  - C++ compiler (required if you wish to do development in C++ as opposed to C)
  - VSI SSL3 V3.0-5 or higher (required if you wish to use SSL)
  - VSI CSWS V2.4-48 or higher (required if you wish to use gSOAP in conjunction with `MOD_GSOAP` and the Apache HTTPD web server)

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.

2.1. Optional products

While gSOAP was implemented to facilitate the development of Web services applications using C and C++, language integration issues aside, under OpenVMS in particular, there is no reason why it cannot be used to develop Web services applications using other 3GL languages such as COBOL, BASIC, Pascal, or FORTRAN. In addition, a simple OpenVMS-specific module is provided that can be used to implement a gSOAP-based Web services layer on top of an existing ACMS application. Optional products might therefore include your language compiler(s) of choice and ACMS.

Regardless of your programming language preference, it will invariably be necessary to write some C code; however (when working with languages other than C/C++), this will typically need to be little more than a thin veneer that takes care of language incompatibilities between C/C++ and your primary programming language.

This release of gSOAP for OpenVMS also includes support for FastCGI. See Section 4.5 below for details regarding FastCGI for OpenVMS and how to obtain a kit.

3. Recommended reading

Before getting too carried away, do be sure to read the very comprehensive gSOAP User Guide (https://www.genivia.com/doc/guide/html/index.html).

4. Installing the kit

The kit is provided as an OpenVMS PCSI kit (`VSI-AXPVMS-GSOAP-V0208-129B-1.PCSI`, `VSI-I64VMS-GSOAP-V0208-129B-1.PCSI`, or `VSI-X86VMS-GSOAP-V0208-129B-1.PCSI`, depending on platform) that can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL LIBMARIADB
```

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

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

The following product has been selected:
    VSI I64VMS GSOAP V2.8-129B              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 GSOAP V2.8-129B

    VMS Software Inc. & Genivia Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS GSOAP V2.8-129B              DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...20%...40%...50%...60%...70%...80%...90%...100%

The following product has been installed:
    VSI I64VMS GSOAP V2.8-129B              Layered Product

VSI I64VMS GSOAP V2.8-129B

    Post-installation tasks are required.

    To define logical names used by gSOAP at system boot time, add
    the following lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

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

    To deassign logical names used by gSOAP at system shutdown, add
    the following lines to SYS$MANAGER:SYSHUTDWN.COM:

        $ file := SYS$STARTUP:GSOAP$SHUTDOWN.COM
```

```
$ if f$search("''file'") .nes. "" then @'file'

Please be sure to review the gSOAP licensing information
described in the files GPLv2_license.txt, license.pdf, and
LICENSE.txt included with this kit in GSOAP$ROOT:[000000].
```

4.1. Post-installation steps

After the installation has successfully completed, include the commands displayed at the end of the installation procedure into `SYSTARTUP_VMS.COM` and `SYSHUTDWN.COM` to ensure that logical names required to use the software are defined system-wide at start-up and de-assigned on system shutdown.

Note: If you are upgrading from a previous release of gSOAP for OpenVMS and use `MOD_GSOAP`, be sure to replace the existing version of `MOD_GSOAP.EXE` in `APACHE$COMMON:[MODULES]` with the new version supplied with this release (`GSOAP$ROOT:[LIB]MOD_GSOAP.EXE`).

4.2. Privileges and quotas

A moderate level of privilege is required to run applications developed using the current release of gSOAP for OpenVMS.

Generally speaking, there are no special gSOAP-imposed quota requirements for applications developed using gSOAP, although a reasonably high `BYTLM` is recommended. The following quotas should be more than adequate for most purposes:

```
Maxjobs:          0  Fillm:      4096  Bytlm:      2000000
Maxacctjobs:      0  Shrfillm:      0  Pbytlm:           0
Maxdetach:        0  BIOlm:       900  JTquota:       8192
Prclm:            0  DIOlm:       900  WSdef:         4096
Prio:             4  ASTlm:       900  WSquo:        16384
Queprio:          0  TQElm:       900  WSextent:     32767
CPU:         (none)  Enqlm:      8192  Pgflquo:    2000000
```

4.3. Debug library

To assist with debugging of applications, this OpenVMS gSOAP release includes the object library `GSOAPDBG.OLB`. This library is essentially the same as `GSOAP.OLB`; however, the contents of the library have been compiled with the `SOAP_DEBUG` macro defined, which causes a considerable body of debug code to be included. Applications linked with the debug version of the object library will create three log files (`recv.log`, `sent.log`, and `test.log`) that will contain a considerable volume of potentially useful information about how the application is operating. Note that any of your application files that include `stdsoap2.h` should also be compiled with the macro `SOAP_DEBUG` defined. Please refer to the gSOAP documentation for more information regarding debugging.

4.4. SSL support

The supplied kit includes the object library `GSOAPSSL.OLB`, which may be used to build gSOAP applications that use OpenSSL to provide secure communication. In order to make use of this capability, the following points should be observed:

- VSI SSL3 V3.0-5 or higher must be installed and correctly configured on all relevant machines.
- All gSOAP-generated code or code that includes gSOAP header files must be compiled with the `WITH_OPENSSL` macro defined (`/define=WITH_OPENSSL`).
- Programs must be linked with the `GSOAPSSL.OLB` object library and with the OpenSSL shareable images `SSL3$LIBSSL_SHR32.EXE` and `SSL3$LIBCRYPTO_SHR32.EXE`, which reside in `SYS$LIBRARY`.

- Refer to the relevant sections of the gSOAP User Guide for details of how to modify your code to deal with SSL and certificates.

The sample build procedure `BUILD-CALC-SSL.COM` in `GSOAP$ROOT:[SAMPLES.CALC]` illustrates how to compile and link gSOAP applications to use OpenSSL in accordance with the notes presented above.

4.5. FastCGI support

In addition to the libraries described above, this release of gSOAP for OpenVMS includes the object libraries `GSOAPFCGI.OLB` (C) and `GSOAPXXFCGI.OLB` (C++), which may be used to build gSOAP applications that use FastCGI (see http://www.fastcgi.org for additional information). In order to make use of this capability, the following points should be noted:

- FastCGI for OpenVMS must be installed and configured
- All gSOAP C/C++ code or code that includes gSOAP header files must be compiled with the `WITH_FASTCGI` macro defined (`/define=WITH_FASTCGI`).

The sample code in `GSOAP$ROOT:[SAMPLES.FCGI]` illustrates how to compile and link gSOAP applications to use FastCGI. The reader should refer to the release notes provided with the FastCGI for OpenVMS for additional information regarding use of the software and the configuration of WASD or Apache HTTPD to act as a FastCGI server.

4.6. Possible issues for those not running ACMS

- As noted previously, one of the OpenVMS-specific extensions included with this kit is a module that can be used to implement a gSOAP-based Web services layer on top of an ACMS application. If you are not running ACMS on the OpenVMS system where you intend to use the gSOAP development kit, you may wish to remove the ACMS agent module from the pertinent object library to avoid linker warnings and other potential problems in the future:

  ```
  $ lib/delete="agent"/log gsoap$root:[lib]gsoap.olb
  ```

  Note the use of the double quotes around the module name in the above command.

- The gSOAP library code contains a number of function names that exceed 31 characters in length and function names that are in mixed case. As a consequence of this, developers need to take care with regard to the case of function names in any developed code, and it may be necessary to use the `CASE_SENSITIVE=YES|NO` directive in linker options files when linking applications. It may also be necessary to bracket some C header files with `#pragma` directives to control how the names of functions and variables specified in those header files are treated by the C compiler. For example, the `#pragma` directives below will ensure that the compiler does not upper-case function and variable names specified in the header files `soapH.h` and `add.nsmap`, but will instead preserve case of any function and variable names used therein (refer to the C compiler documentation for more information):

  ```
  #pragma names save
  #pragma names as_is
  #include "soapH.h"
  #include "add.nsmap"
  #pragma names restore
  ```

5. Sample applications

The `GSOAP$ROOT:[SAMPLES]` directory contains several very simple examples that illustrate various aspects of the gSOAP toolkit and several specific features of the OpenVMS port. The following text briefly discusses each of these examples. Each of the example directories contains simple

command procedures that can be run to build the example in question. These build procedures contain the minimum code to compile and link the samples.

Note that the examples below are built using samples directories installed with the kit. It is advisable to make copies of the files into a different directory to preserve the original kit contents.

1. `gsoap$root:[samples.calc]`

   This sample application is adapted from one of the examples provided with the open source gSOAP distribution. It has been extended to include a very simple COBOL client that calls one of the Web service methods.

   Before running the build procedure, have a look through the code and refer to the gSOAP User Guide to help you understand what is going on.

   Before running the build procedure, edit `calcclient.c` and `cobclient.cob`, and change the value of the URL variable. For example, in the COBOL client, change the URL in the following statement to something appropriate for your environment:

   ```
   move "http://10.10.224.180:8181/calcserver.exe" to url.
   ```

   Be sure to remember the port number, 8181 above, you have specified. If you now run the build procedure (`@build-calc`), you should have three executable programs: `calcclient.exe`, `calcserver.exe`, and `cobclient.exe` (assuming you have a COBOL compiler).

   Next, define a foreign command for `calcserver`:

   ```
   $ calcserver :== $gsoap$root:[samples.calc]calcserver.exe
   ```

   You should now be able to run `calcserver`, specifying the port number you used in your URL (see above) as the single command line parameter. For example, if decided to use port 8181 from the original sample code, you would type:

   ```
   $ calcserver 8181
   ```

   Assuming that all is well, `calcserver` will now be listening on port 8181 for incoming Web service requests. Try running `calcclient.exe` and `cobclient.exe` to verify that the service is working correctly.

   When looking through the code, you will have noticed that the COBOL client (`cobclient.cob`) calls several functions whose names begin with `GSOAP$`. You might also have noticed that the COBOL client does not call the generated gSOAP client stub directly, but instead calls a simple wrapper function written in C. The `GSOAP$` functions are OpenVMS-specific extensions to gSOAP.

   Finally, in the `[.calc]` directory you will also see a DCL command procedure called `build-calc-ssl.com`. As noted previously, this command procedure illustrates how to compile and link gSOAP applications to use OpenSSL. The [.calc] directory also includes a C++ implementation of the calculator example, which can be built using the command procedure `build-calc-cpp.com`.

2. `[.math]`

   This example is designed to implement the same functionality as the math example provided with the HP Web Services Integration Toolkit (WSIT). There is no client application provided with this example; tools such as soapUI can be used as a generic SOAP client to test the service. If you wish to try using soapUI with this sample application, you will need to copy the WSDL file (`demo.wsdl`) generated by gSOAP for the application to your PC.

3. `[.agent]`

   This example is only of relevance to users using ACMS. As noted previously, one of the OpenVMS extensions to the gSOAP toolkit provided in the OpenVMS gSOAP kit is a module that can be used to implement a gSOAP-based Web services layer on top of an ACMS application.

The contents of the [.agent] directory illustrate the usage of this module to call the "add" task implemented by the ACMS ADD example provided in the ACMSDI$EXAMPLES. Note that no client application is provided with this example; tools such as soapUI may be used to test and exercise the application.

Note that you will need to change the ACMS username used by the example as appropriate for your environment. Remember also that the username has to be defined in the ACMS User Definition File with the "/agent" privilege.

4. [.COBOL]

   This is a very simple example that illustrates how to create a Web Service that uses an existing COBOL application. The example does very little, but hopefully serves to illustrate some of the factors that need to be considered when developing such services.

5. [.FORTRAN]

   This example provides a simple illustration of how to use gSOAP to create a Web Service that uses existing FORTRAN code, and how to create a simple client program to call this service from FORTRAN code.

6. [.fcgi]

   This example illustrates how to use gSOAP with FastCGI. After building the application using the supplied command procedure, the reader should refer to the FastCGI for OpenVMS release notes for details of how to configure and run the application. See Section 4 for details for details on FastCGI for OpenVMS and how to obtain a kit.

7. [.wsse]

   This directory contains an example program that illustrates the use of the WSSE plug-in that is included with the kit. This example also uses the gSOAP DOM implementation and SSL. It should be noted that it will be necessary to obtain appropriate certificates in order to use the SSL functions.

6. Configuring and using Apache mod_gsoap

The COBOL example shipped with the kit can be built both for use with the gSOAP Standalone Server as well as with the Apache HTTPD mod_gsoap module. The following text describes how to install and configure the service built in this way to work with the Apache web server. Any service developed using gSOAP can be similarly built and deployed in this same manner,

The gSOAP Standalone Server is very useful for development and testing. However, the standalone server code provided with gSOAP is generally not as scalable or as fault tolerant as the Apache HTTPD, and the server functionality provided by the standalone server is also somewhat limited. Therefore, for production environments it is generally better to run services under Apache HTTPD using the mod_gsoap module.

Deploying services under Apache HTTPD and mod_gsoap is straightforward, although a reasonably high level of privilege is required in order to perform the installation. Assuming you have built the Web service to work with Apache HTTPD and mod_gsoap, the following steps need to be performed to install and configure the service:

- Copy gsoap$root:[lib]mod_gsoap.exe to apache$root:[modules] and ensure that the ownership and permissions on the copied file are the same as the other modules in the directory.
- Modify the Apache configuration file (apache$root:[conf]httpd.conf) to include the mod_gsoap module and a definition for your new service. For example, you might add the following statements at the bottom of the file:

```
LoadModule gsoap_module modules/mod_gsoap.exe
<Location /add>
```

```
    SetHandler gsoap-handler
    SOAPLibrary add_server
</Location>
```

The `LoadModule` statement instructs Apache to load the `mod_gsoap` module. The `<Location>` tag specifies an alias for and the location of our Web service. Assuming that Apache is running on node bugs.bunny.com on port 81, the `<Location /add>` tag means that we will access our service using the URL http://bugs.bunny.com:81/add. The statement "`SOAPLibrary add_server`" instructs `mod_gsoap` that the shareable image implementing our service is `add_server`. In this case, `add_server` is a logical name (see next step). You could alternatively specify the full path to the service (`add.exe`).

- Define a system-wide logical name for `add_server` that points to the service (the sharable image called `add.exe)`. There is no need to install the shareable image.
- Restart Apache to pick up the new service.

You should now be able to access the "add" service running under Apache using a URL similar to that specified above.

7. What's missing?

As noted previously, all standard gSOAP functionality is provided by this OpenVMS release. If there are additional components that you would like to see added to the distribution, please let us know, and we will endeavour to factor your requests into a subsequent release.