MatrixSSL Getting Started Guide

MatrixSSL 3.3

Software Manual

3129 Rev 1
23 01 12

AuthenTec, Inc.
100 Rialto Place, Suite 100
Melbourne, Florida 32901
321.308.1300
authentec.com

AuthenTec welcomes your input. We try to make our publications useful, interesting, and informative, and we hope you will take the time to help us improve them. Please send any comments or suggestions by mail or e-mail.

AuthenTec, Inc.
100 Rialto Place, Suite 100
Melbourne, Florida 32901
321-308-1300
www.authentec.com
apps@authentec.com

MatrixSSL Getting Started Guide
3129 Rev 1 (23 01 12)

Table of Contents

Overview

This Getting Started Guide explains how to quickly compile and test the MatrixSSL package on supported reference platforms. This guide also contains instructions on building and running the client and server applications provided in the package.

Who Is This Document For?

- Software developers working on a supported platform that want to create a development environment for integrating MatrixSSL security into a custom application
- Software developers who want to port MatrixSSL to a new platform
- Anyone wanting to learn more about MatrixSSL

Documentation Style Conventions

- File names and directory paths are *italicized*.
- C code literals are distinguished with the `Monaco font`.

Compiling and Testing MatrixSSL

POSIX Platforms with Makefiles

The POSIX classification in MatrixSSL encompasses support for several operating system platforms including Mac OSX 10.5 and most UNIX/LINUX varieties. This is the default platform for the *Makefile* system that is provided in the package and should be the first build option if you are unsure of your platform configuration.

Preparation

The development platform must have the following tools installed:

- The *tar* archiver for unzipping the package (or other decompression utility supporting *.TGZ* files)
- A C source code compiler and linker (*GCC* is the default in the provided Makefile system)
- The *make* tool

Building the source

1. From the command prompt, unpack the zipped tar image.

   ```
   $ tar -xzvf matrixssl-3-2-0-open.tgz
   ```

2. Change directory to the root of the package and build the MatrixSSL library .

   ```
   $ cd matrixssl-3-2-0-open
   $ make
   ```

3. Confirm there were no compile errors and that the MatrixSSL libraries have been built. A successful build will result in a libmatrixssl shared and static library. Applications interface with this library through the MatrixSSL public API set, which is documented in the MatrixSSL_API PDF file, included in the distribution.

Self-Test Application

Source code for a self-test application to exercise the SSL handshake and data exchange functionality of the MatrixSSL library is provided with the package. The following optional steps will enable the developer to build and run the test application to confirm the SSL protocol is fully functional.

1. Having successfully built the static library from the **Building the source** steps above, change directories to the *test* folder where the *sslTest.c* source is located and compile the application.

```
$ cd matrixssl/test
$ make
```

2. Run the *sslTest* application from the command line. This sample output shows a successful run of the test using the default configuration of the open source package.

```
$ ./sslTest
Testing TLS_RSA_WITH_AES_128_CBC_SHA suite
        Standard handshake test
                PASSED: Standard handshake
        Re-handshake test (client-initiated)
                PASSED: Re-handshake
        Resumed handshake test (new connection)
                PASSED: Resumed handshake
        Re-handshake test (server initiated)
                PASSED: Re-handshake
        Resumed Re-handshake test (client initiated)
                PASSED: Resumed Re-handshake
        Resumed Re-handshake test (server initiated)
                PASSED: Resumed Re-handshake
        Change cert callback Re-handshake test
                PASSED: Upgrade cert callback Re-handshake
        Change keys Re-handshake test
                PASSED: Upgrade keys Re-handshake
        Change cipher suite Re-handshake test
                PASSED: Change cipher suite Re-handshake

Testing SSL_RSA_WITH_3DES_EDE_CBC_SHA suite
        Standard handshake test
                PASSED: Standard handshake
        Re-handshake test (client-initiated)
                PASSED: Re-handshake
        Resumed handshake test (new connection)
                PASSED: Resumed handshake
        Re-handshake test (server initiated)
```

```
        PASSED: Re-handshake
Resumed Re-handshake test (client initiated)
        PASSED: Resumed Re-handshake
Resumed Re-handshake test (server initiated)
        PASSED: Resumed Re-handshake
Change cert callback Re-handshake test
        PASSED: Upgrade cert callback Re-handshake
Change keys Re-handshake test
        PASSED: Upgrade keys Re-handshake
Change cipher suite Re-handshake test
        PASSED: Change cipher suite Re-handshake
```

Sockets-Based Client and Server Applications

Source code for TCP/IP sockets-based client and server applications are provided with the MatrixSSL package. The following optional steps will enable the developer to build and run the applications to confirm the development platform is configured for MatrixSSL integration.

1. Having successfully built the static library from the **Building the source** steps above, change directories to the *apps* folder where the *client.c* and *server.c* source are located and compile the applications.

   ```
   $ cd apps
   $ make
   ```

2. Run the *server* application from the command line.

   ```
   $ ./server
   Listening on port 4433
   ```

3. 3. In a second shell environment, run the client application and verify two connections were made to the running server. Client trace in the successful case:

   ```
   $ ./client
           === INITIAL CLIENT SESSION ===
           Validated cert for: Sample Server Cert.
           SEND: [GET / HTTP/1.0
           User-Agent: MatrixSSL/3.x
           Accept: */*
           Content-Length: 0

           ]
           RECV: [HTTP/1.0 200 OK
           Server: AuthenTec Networks MatrixSSL/3.x
           Pragma: no-cache
           Cache-Control: no-cache
           Content-type: text/plain
           Content-length: 9

           MatrixSSL]
           SUCCESS: Received HTTP Response

           === CLIENT SESSION WITH CACHED SESSION ID ===
   ```

```
SEND: [GET / HTTP/1.0
User-Agent: MatrixSSL/3.x
Accept: */*
Content-Length: 0

]
RECV: [HTTP/1.0 200 OK
Server: AuthenTec Networks MatrixSSL/3.x
Pragma: no-cache
Cache-Control: no-cache
Content-type: text/plain
Content-length: 9

MatrixSSL]
SUCCESS: Received HTTP Response
$
```

Debug Builds vs. Release Builds

The default compiler options in the Makefile build system use the `-Os` optimization flag to create a size-optimized release quality MatrixSSL library. If you wish to create a debug version of the library (or applications) simply type `make debug` to activate the `-g` compiler flag.

WIN32 Platforms using Visual Studio Projects

Preparation

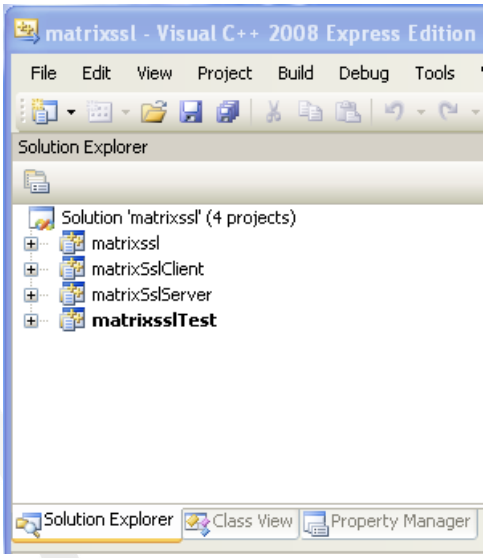
The Windows development platform must have the following installed:

- Microsoft Visual C++ 2010 Express Edition

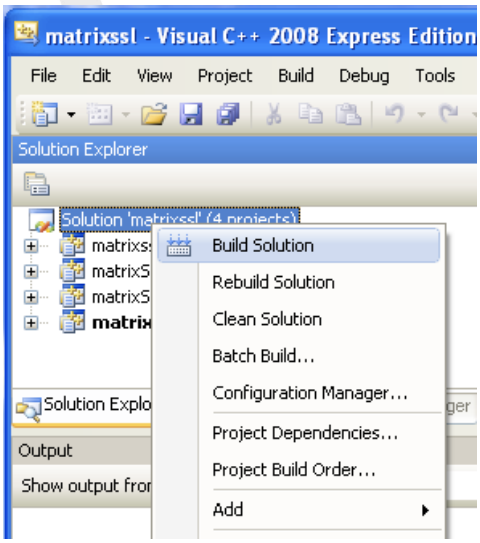
Building the source

1. Unpack the product image to the directory of your choosing.

2. Open the *matrixssl.sln* file at the top level of the directory structure by either double clicking the file or choosing it through the *File-->Open-->Project/Solution* menu option of Visual C++. The result should be a solution with four projects:



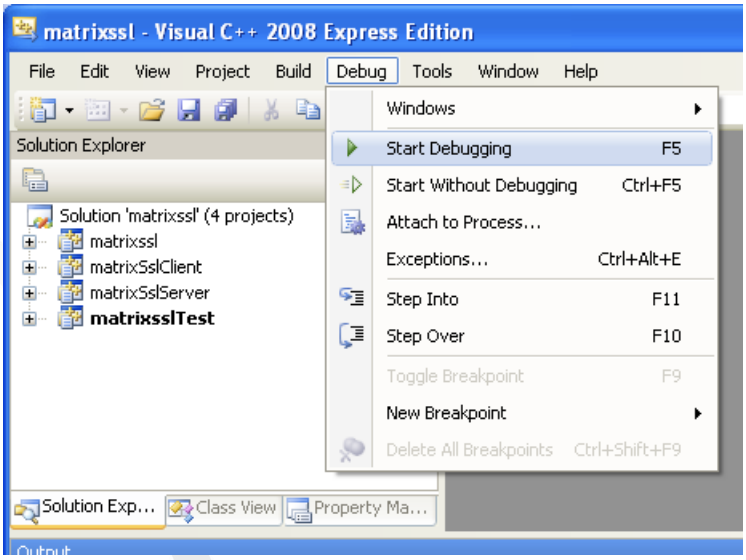
3. Build the solution by right clicking the solution and selecting Build Solution.



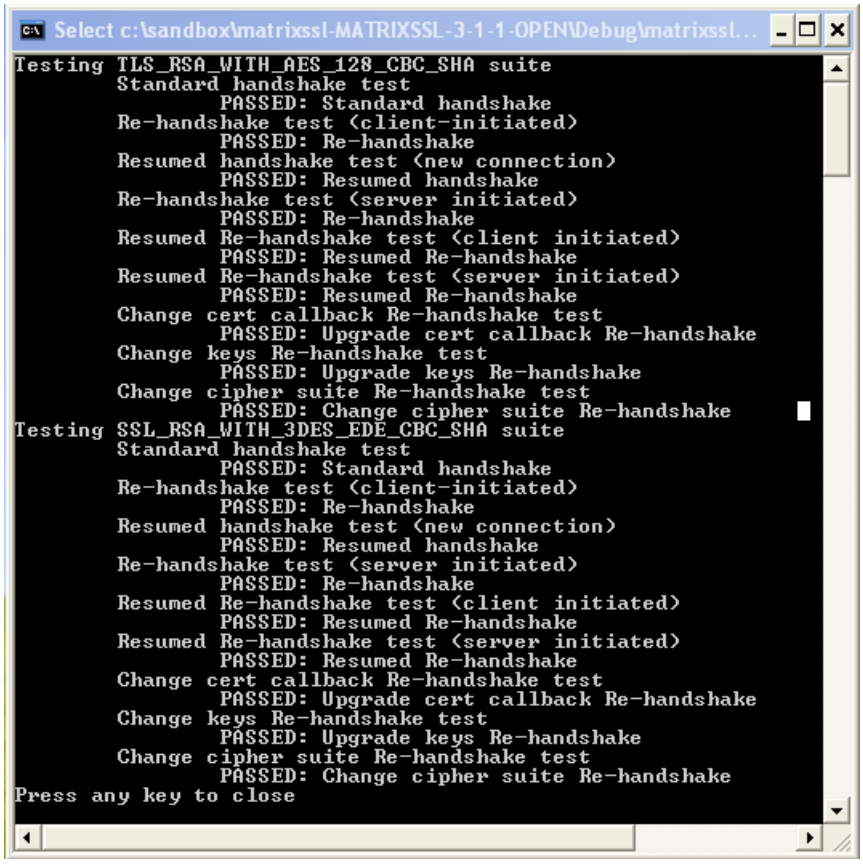
4. Confirm there were no compile errors and that the MatrixSSL library and applications have been built. A successful build will result in a *Debug* (or *Release*) directory being created at the top level directory with the object files, libraries, and executables. The *matrixssl.dll, matrixSslClient.exe, matrixSslServer.exe,* and *matrixsslTest.exe* files are the final outputs to look for.

Self-Test Application

The test application will have been built using the above steps. To run the test application, ensure the **matrixsslTest** project appears in bold in the *Solution Explorer* to indicate it is the *StartUp* project. Select *Debug->Start Debugging* from the menu to start the application.



The console output should look like this:

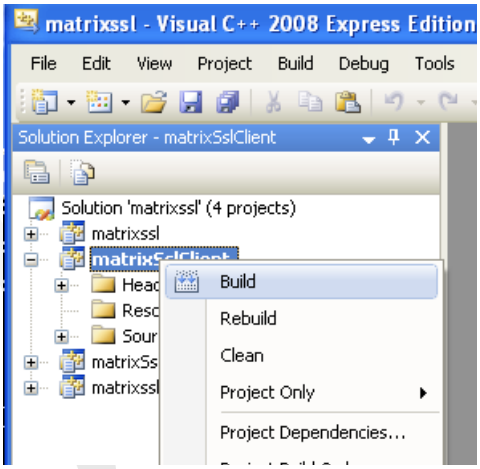


Sockets-Based Client and Server Applications

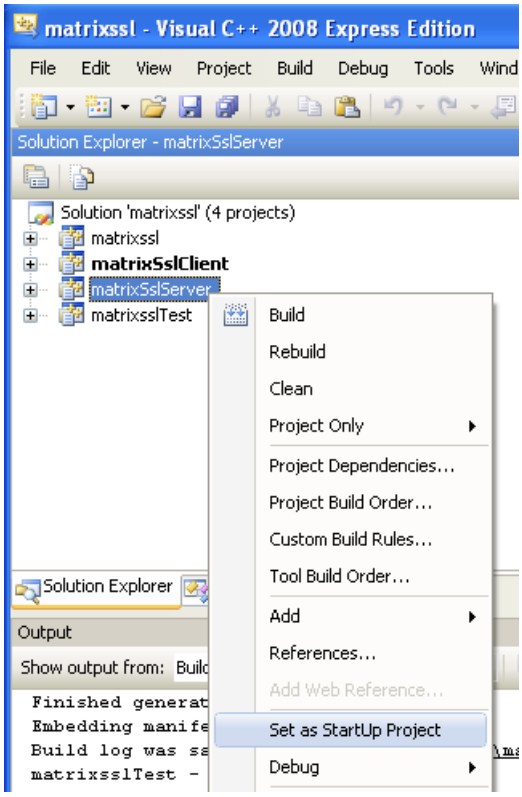
Source code for TCP/IP sockets-based client and server applications are provided with the

MatrixSSL package. The client and server applications will have been built when the Solution was built in the above steps.

To individually recompile a project, simply right click the project name and choose Build from the drop down list:

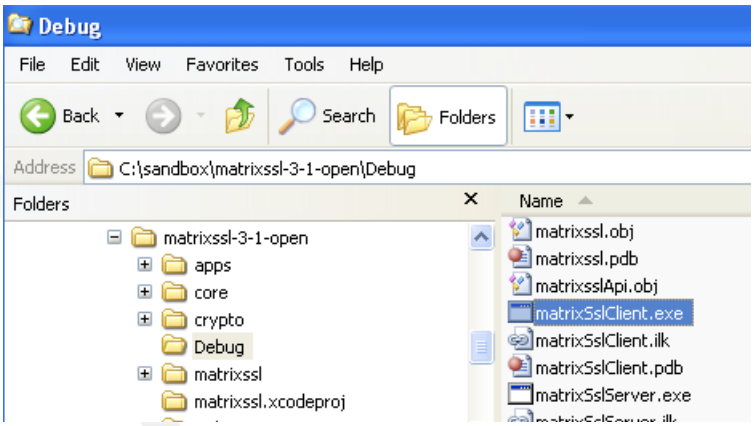


To run the client or server application in the Visual C++ environment, ensure the target project appears in bold in the Solution Explorer to indicate it is the StartUp project. Select Debug->Start Debugging from the menu to start the application.

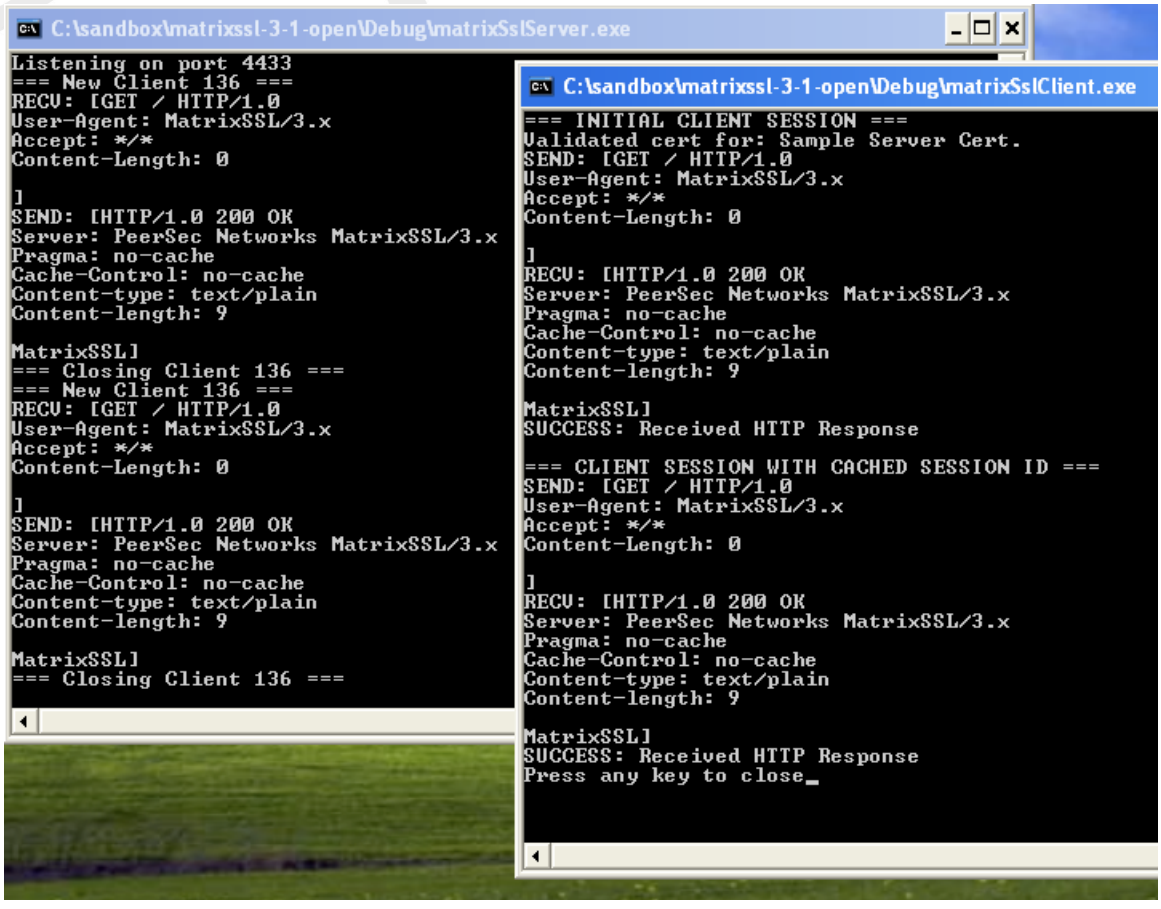


Only a single project can be executed from within the Visual C++ environment at a time so when testing the server and client applications against each other, at least one of them must be manually run by double clicking the executable file. The server application must

be started before the client.

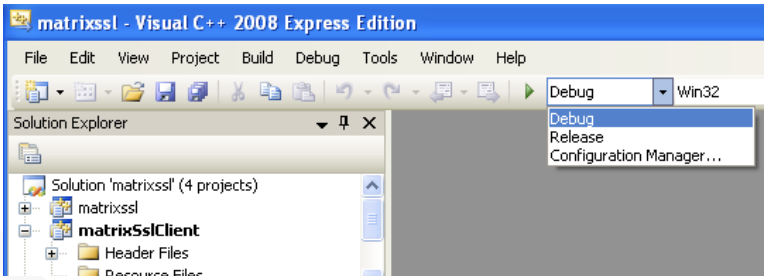


A successful test will result in console output like this:



Debug Builds vs. Release Builds

The default build configuration for the provided Visual Studio projects is set to Debug. If you wish to create release versions of the library or applications, simply select the Configuration pull-down list and select Release as shown below.



By default the output files will be created in a directory at the top level matching the name of the build (Debug or Release).

Mac OS X Platforms using Xcode Projects

Xcode projects have been included for building the MatrixSSL library, the test application, and the client/server applications. The Xcode version used to create the projects was 3.1.4. The version of Mac OS X used to test the projects was 10.5.8.

AuthenTec, Inc.
100 Rialto Place, Suite 100
Melbourne, Florida 32901
321-308-1300 (voice)
321-308-1430 (fax)
www.authentec.com
apps@authentec.com