PeerSec Networks

MatrixSSL Integration for GoAhead WebServer 2.5

GoAhead WebServer 2.5 includes a fully supported layer to the PeerSec MatrixSSL 3 library. This document describes how to compile and configure an SSL-enabled Web server.

PeerSec Networks

Building an SSL-enabled Web server with MatrixSSL

MatrixSSL source is not included in the WebServer package. The MatrixSSL library must first be downloaded from http://www.matrixssl.org.

Makefile-based LINUX/MACOSX/POSIX builds

The WebServer package contains maintained Makefile build systems for the LINUX and MAC OS X platforms. These supported Makefiles include mechanisms for detecting a MatrixSSL source directory at the top level WebServer directory and automatically enabling the compile and link options to generate an SSL-enabled server.

1. Unzip/untar the WebServer 2.5 package to the directory of your choosing.
2. Unzip/untar the MatrixSSL 3 package to the top directory of that WebServer directory:

```
$> cd webs-2-5
$> tar -xzvf matrixssl-3-1-3-open.tgz
$> ls -d matrix*
matrixSSLSocket.c           matrixssl-3-1-3-open
matrixSSLSocket.h           matrixssl-3-1-3-open.tgz
```

3. Change directory to the matrixssl source, compile the package, and confirm the library was successfully created:

```
$> cd matrixssl-3-1-3-open
$> make
$> ls libmatrixssl.a
libmatrixssl.a
```

4. Return to the WebServer source directory in the proper OS_Type subdirectory and compile. This example shows the LINUX OS_Type:

```
$> cd ../LINUX
$> pwd
/Users/john/webs-2-5/LINUX
$> make
```

5. Confirm through the build output that there were no errors, that the webs objects were generated with the **WEBS_SSL_SUPPORT** preprocessor define enabled, and that the executable was linked with the libmatrixssl.a static object. For example:

```
cc -c -o ../webs.o -Os -DWEBS -DUEMF -DPOSIX -DOS="MACOSX" -DMACOSX -
DUSER_MANAGEMENT_SUPPORT -DDIGEST_ACCESS_SUPPORT -DWEBS_SSL_SUPPORT -I../
matrixssl-3-1-2-open/  -I..  ../webs.c

cc -o webs -Os -DWEBS -DUEMF -DPOSIX -DOS="MACOSX" -DMACOSX -DUSER_MANAGEMENT_SUPPORT -
DDIGEST_ACCESS_SUPPORT -DWEBS_SSL_SUPPORT -I../matrixssl-3-1-2-open/  -I..  main.o
libwebs.a ../matrixssl-3-1-2-open/libmatrixssl.a
```

6. Confirm SSL support by executing the WebServer server application and connecting a local Web browser using HTTPS on port 4433 (https://localhost:4433).

Troubleshooting

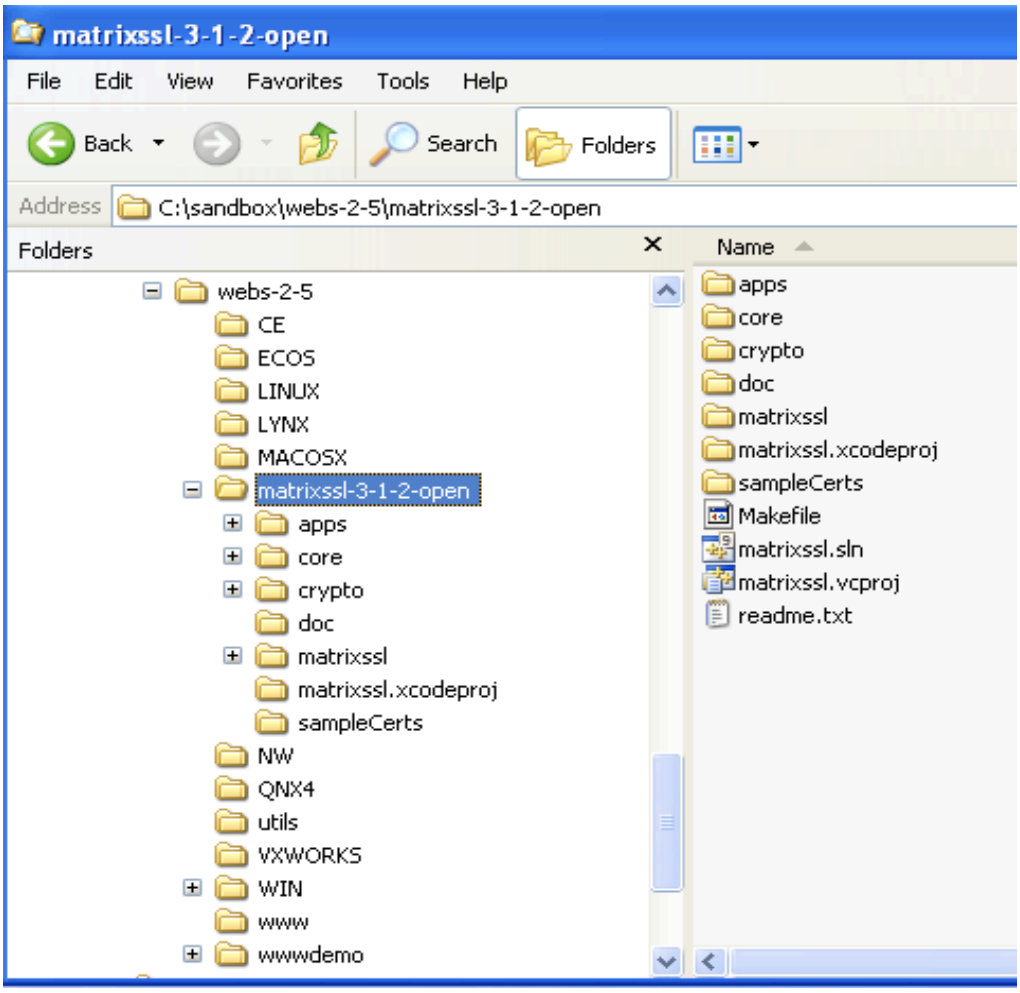
The mechanism for detecting the presence of a MatrixSSL source directory is based on a simple ‘ls’ shell command. If the MatrixSSL source directory is not being correctly detected, manually edit the Makefile so that the SSLINC, SSLLIB, SSLWS, and OPT_FILES macros are defined as shown in the file.

Visual Studio Express Windows Builds

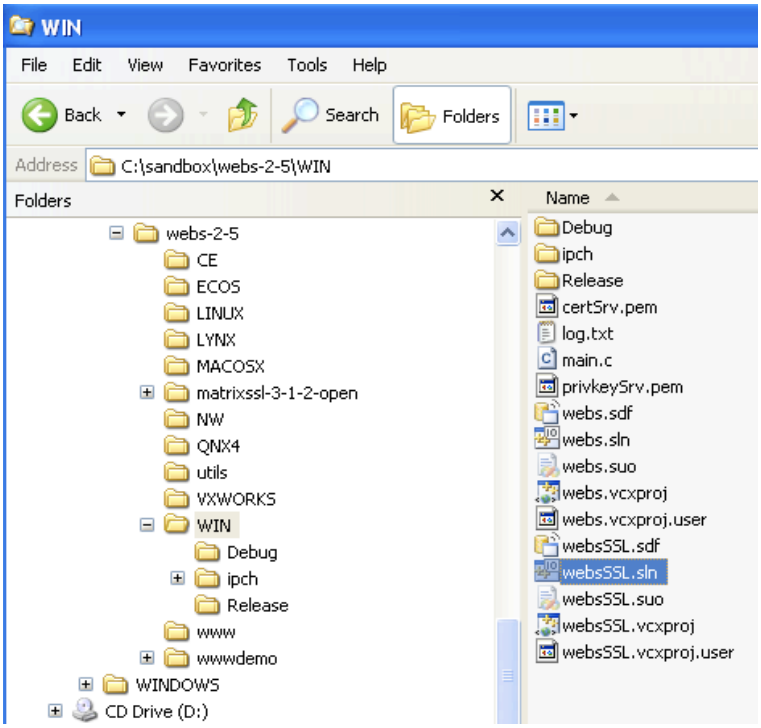
The WebServer and MatrixSSL packages both contain Visual Studio Express project files. The WebServer package contains a solution file that should be used to add both the project files to. Once the MatrixSSL project file is loaded, there are a few manual changes the user must make for the WebServer solution to succeed.

1. Extract the WebServer package to a directory of your choosing.

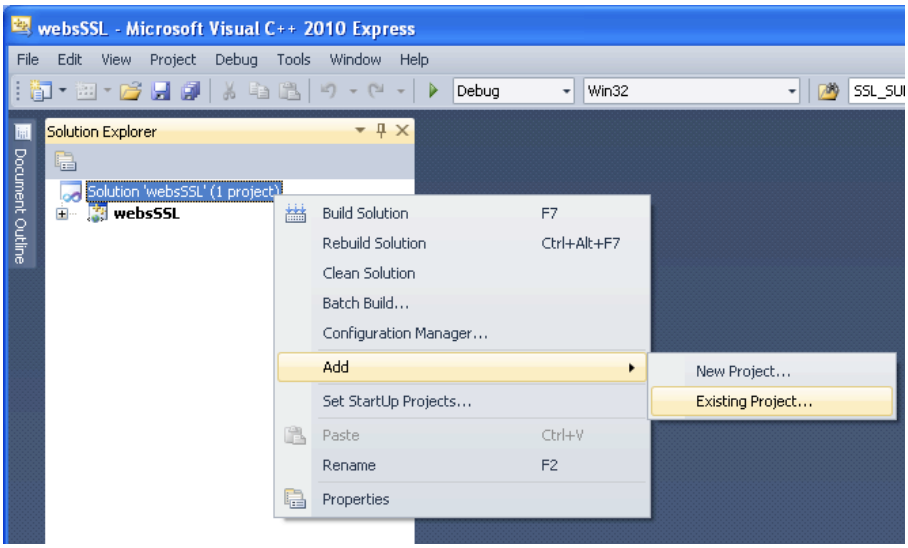
2. Extract the MatrixSSL 3 package to the top level directory of the WebServer source. The directory structure should now look like the following:



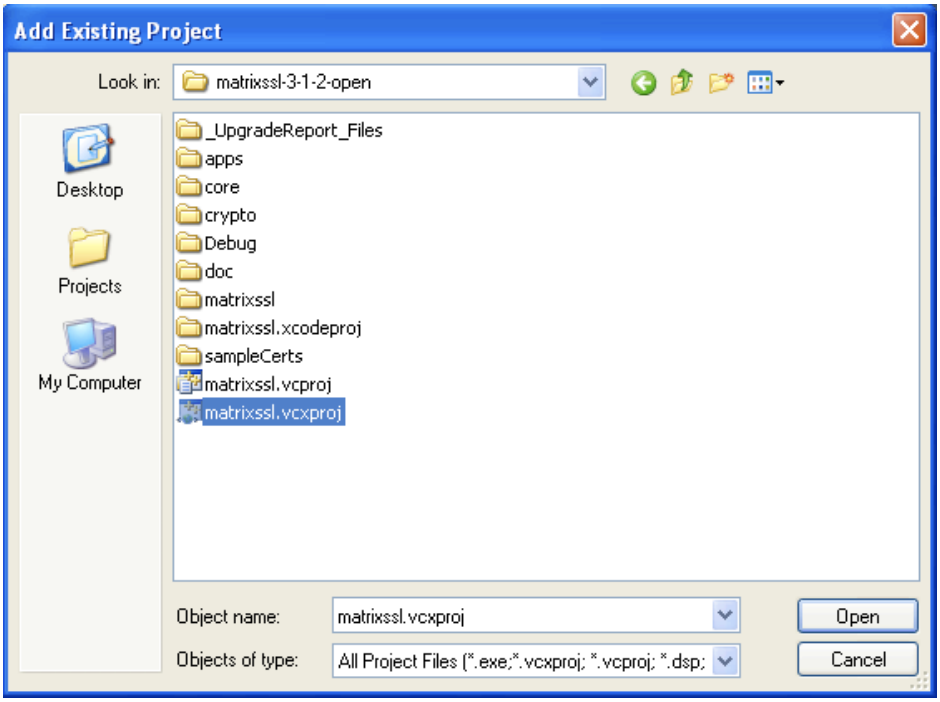
3. Navigate to the WIN directory of the WebServer package and open the SSL version of the Visual Studio solution, *websSSL.sln.*



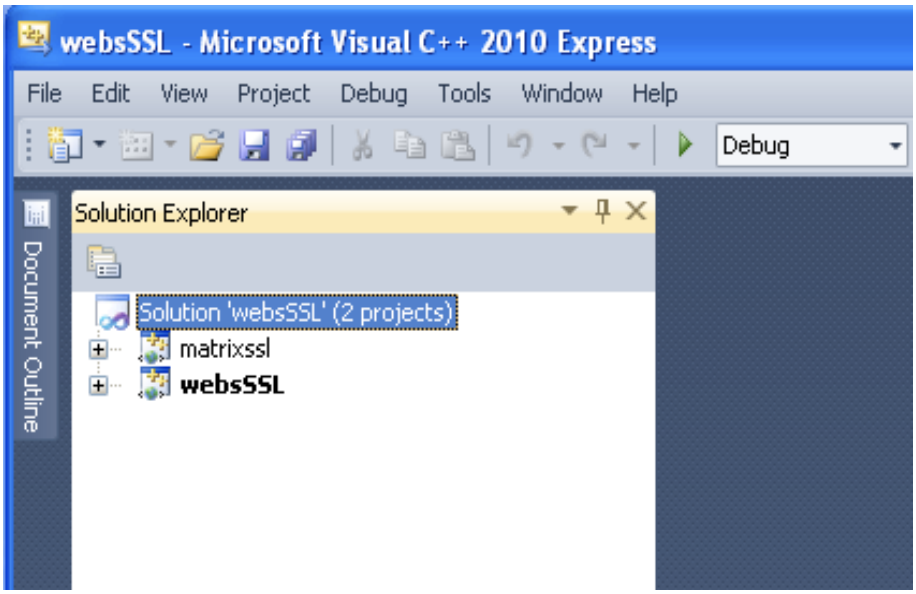
4. Add the MatrixSSL project to the solution by right-clicking the websSSL solution, clicking Add, and then clicking Existing Project.



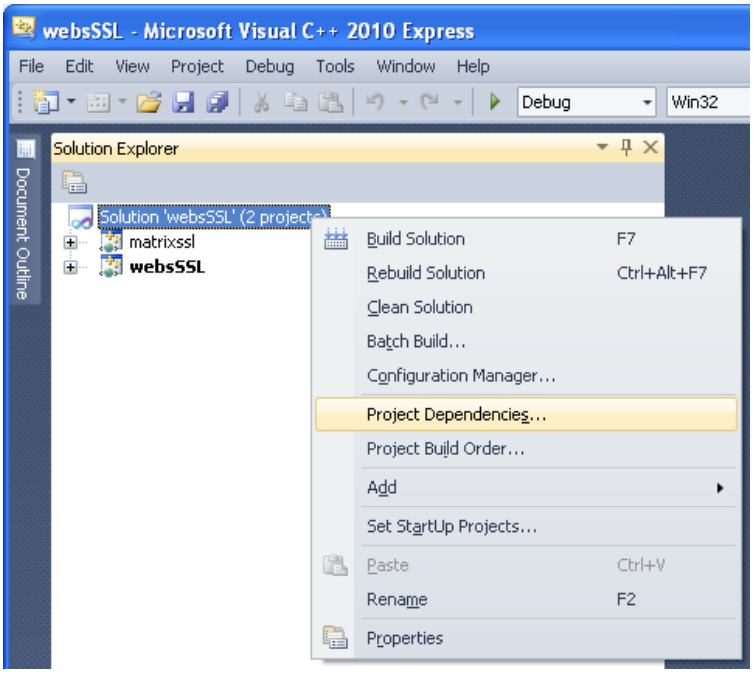
5. Navigate to the ./webs-2-5/matrixssl-3-* directory and choose matrixssl.vcxproj.



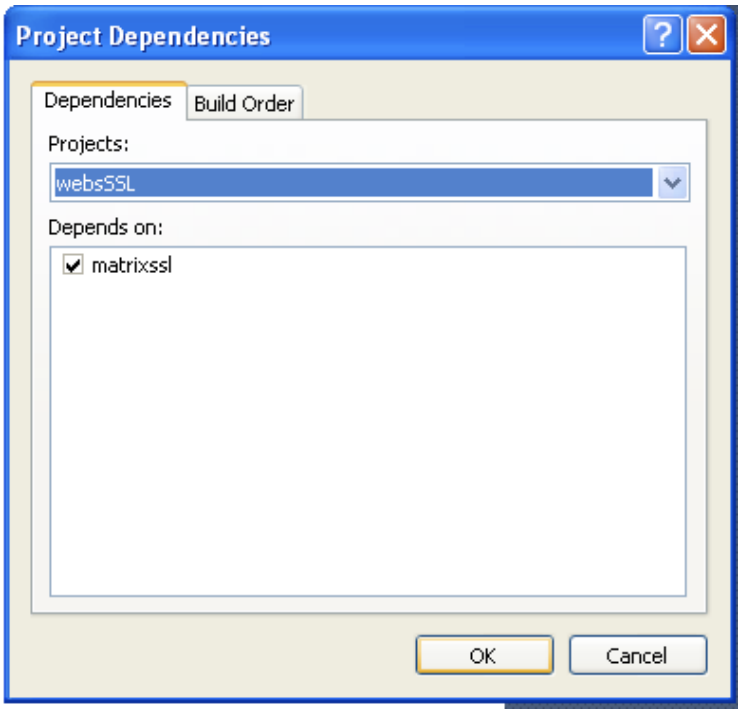
The webSSL and matrixssl projects should now both be available in the Solution Explorer window of Visual Studio:



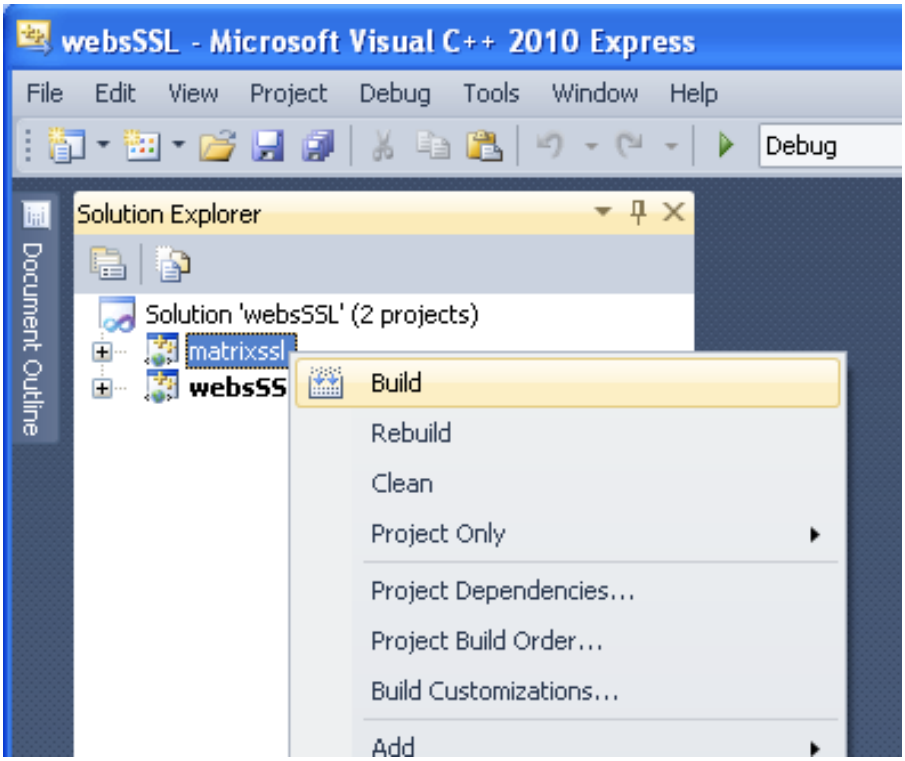
6. Set the Project Dependencies so that webSSL depends on matrixssl by right-clicking the webSSL solution and selecting Project Dependencies.



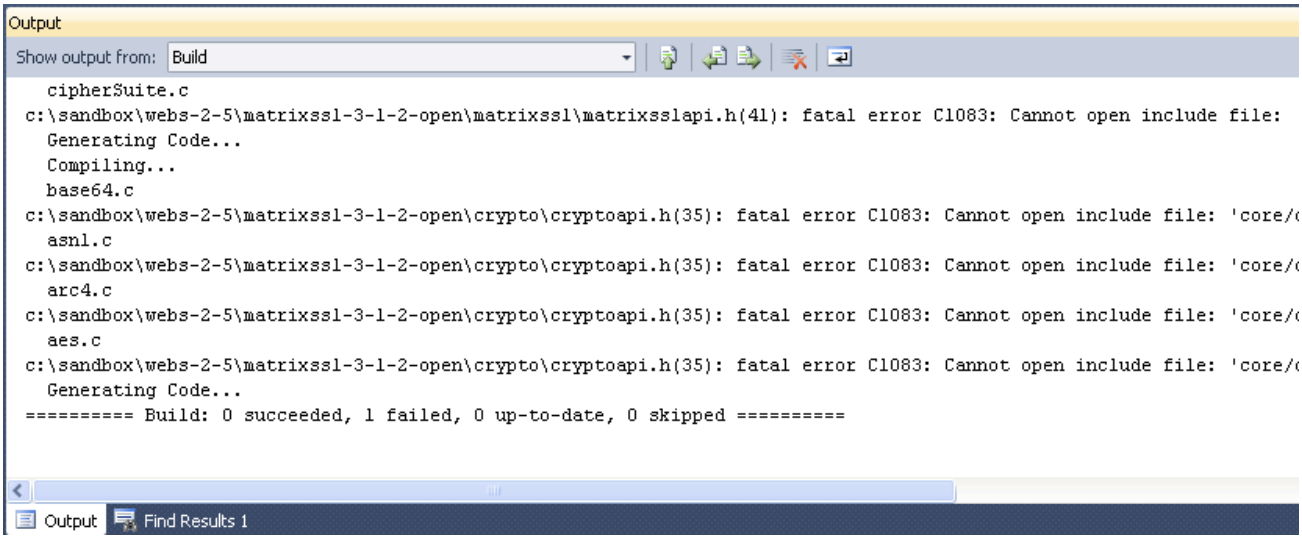
Choose webSSL from the drop-down as the Project and then select the checkbox next to matrixssl to set the Build Order. Click OK



7. Build the matrixssl library by right-clicking the matrixssl project and selecting Build

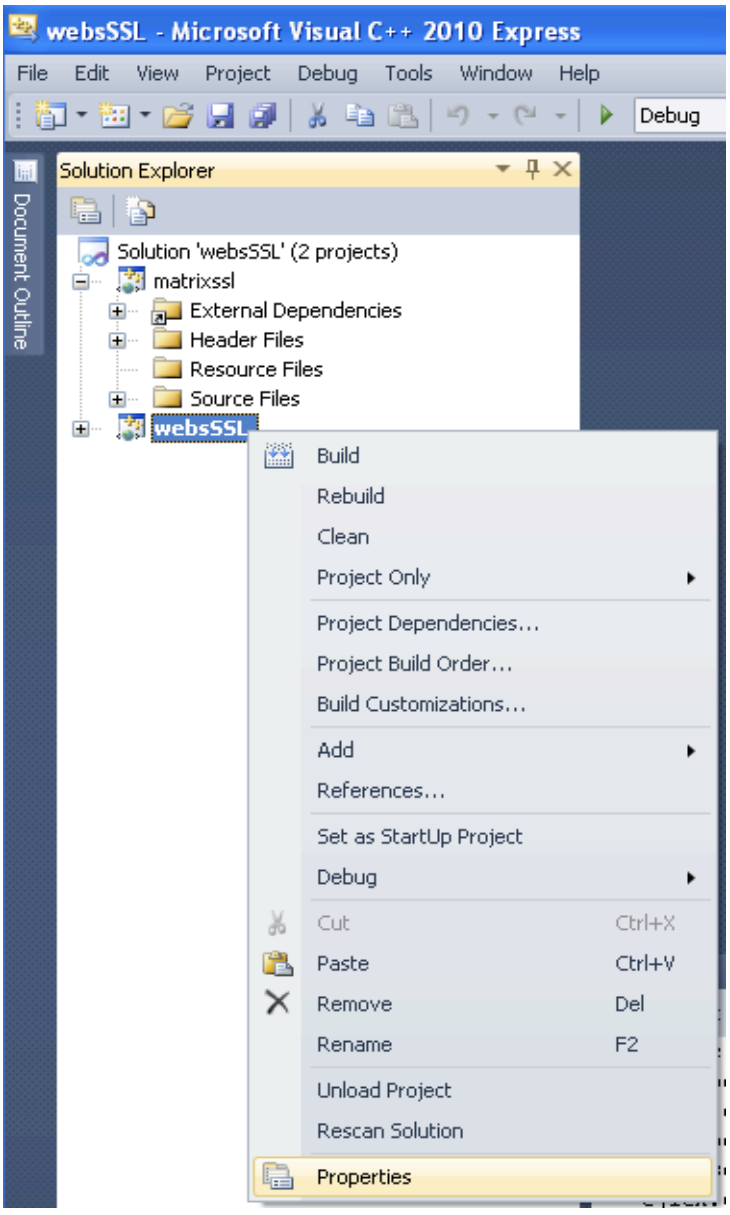


The build should succeed but if the Output window is reporting fatal errors about not being able to open header include files, you will need to edit the include path of the matrixssl project.

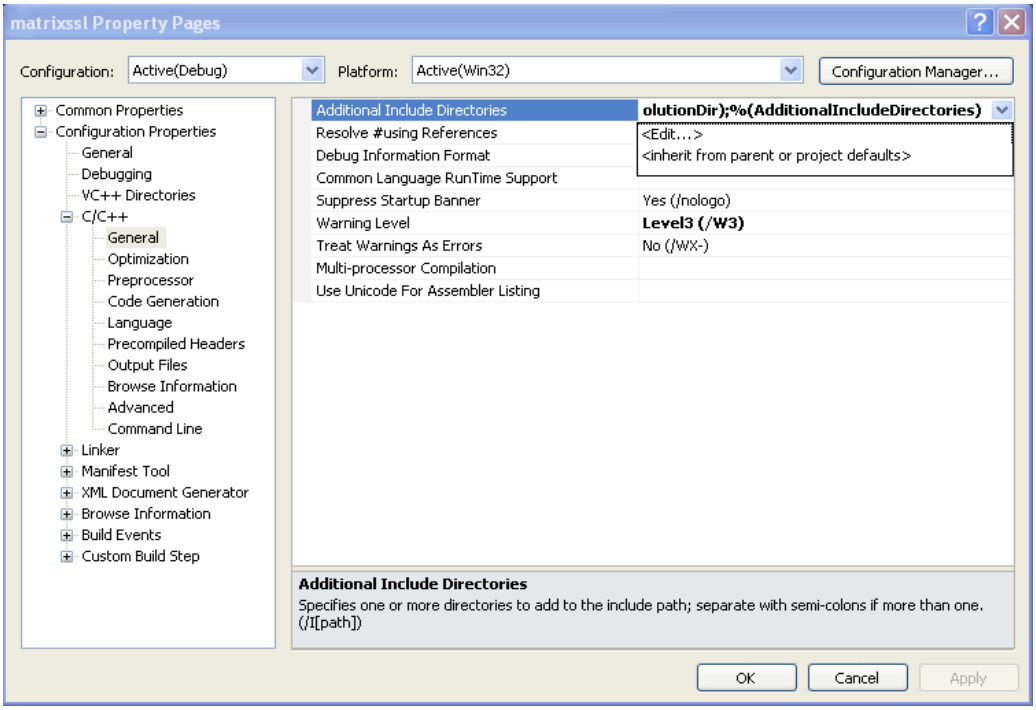


8. Modify the websSSL project to correctly include the matrixssl header files.

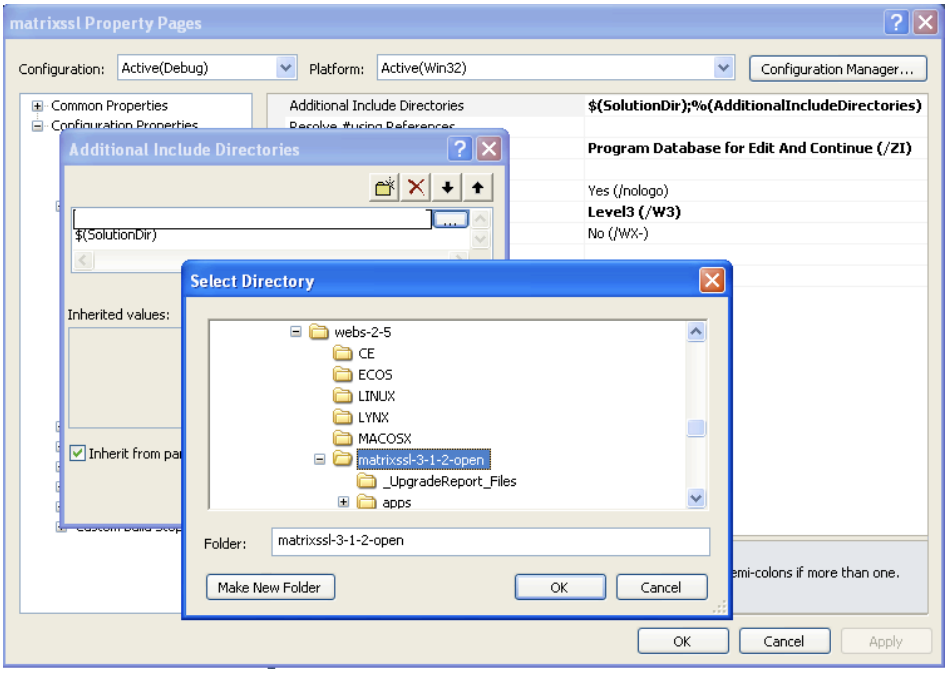
   a. Open the webSSL project Properties by right-clicking the websSSL project and selecting Properties:



b. In the websSSL Property Pages dialog, expand Configuration Properties, expand C/C++, select General, select Additional Include Directories, and then select Edit.

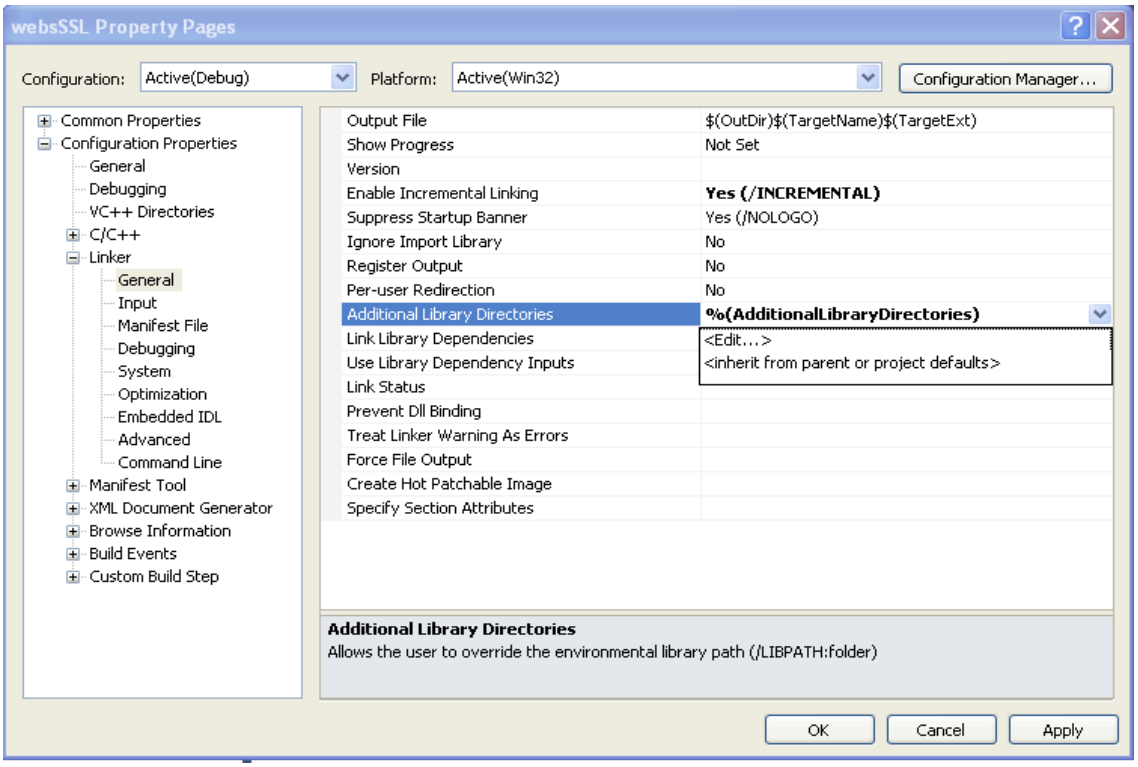


c. In the Additional Include Directories dialog, click the New Folder icon, and then in the Select Directory dialog, select the matrixssl source directory. Apply the changes.

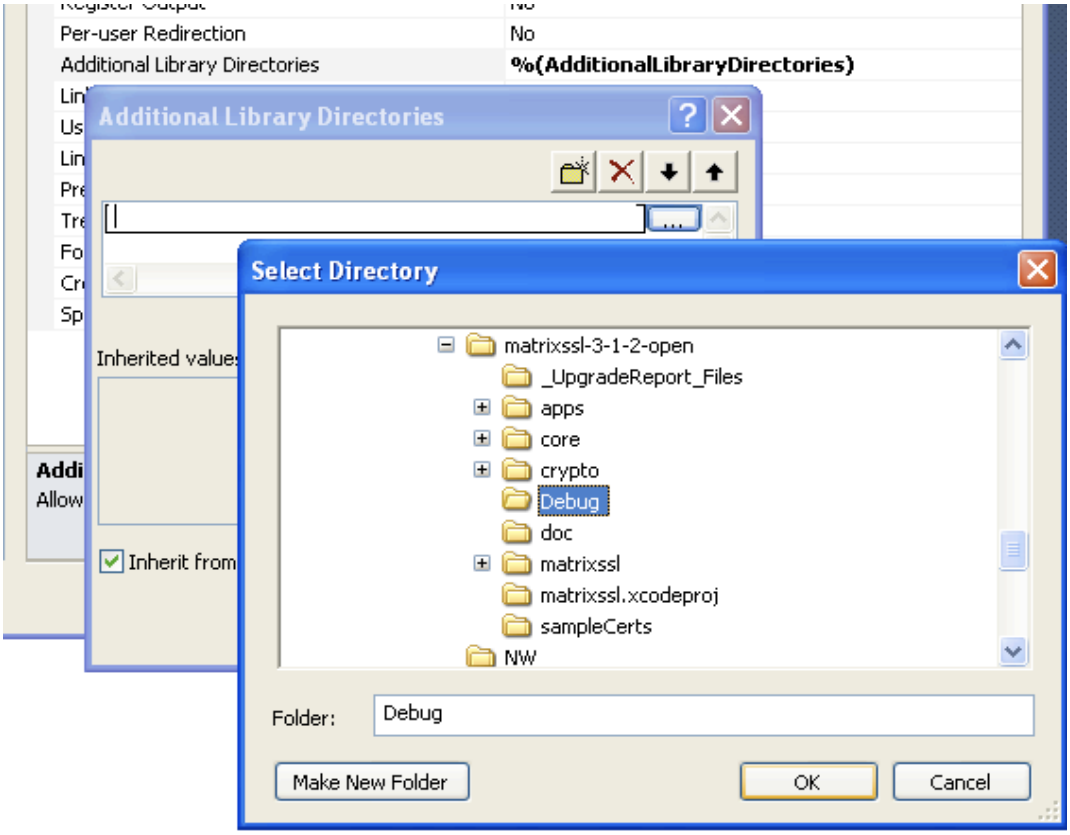


9. Modify the websSSL project to link with the matrixssl library by adding it to the linker input:

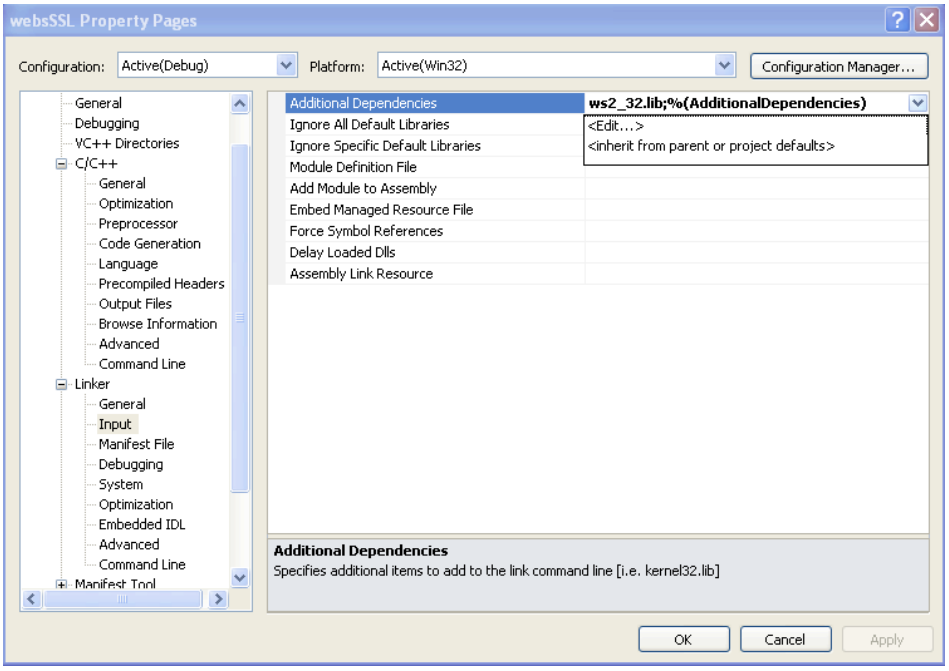
   a. First add the directory path. In the websSSL Property Pages dialog, expand Configuration Properties, expand Linker, select General, select Additional Library Directories, and then select Edit.



Click the New Folder icon to navigate to the matrixssl.lib directory. **Note**: The matrixssl.lib file should have been built in step 7 above and can be found in the Debug or Release subdirectory (depending on the desired Configuration) of the matrixssl source code directory.

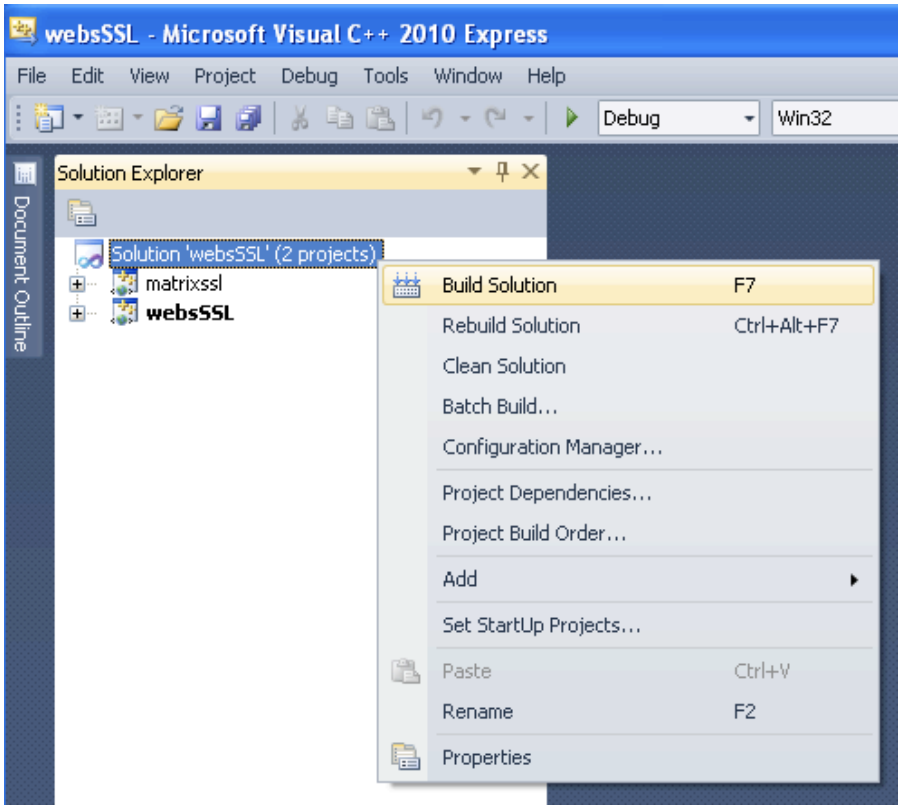


b. Next add the library file itself. In the webSSL Property Pages dialog under Linker, select Input, select Additional Dependencies, and then select Edit.



Manually add *matrixssl.lib* to the semi-colon delimited list. Apply all changes and exit the property settings for the project.

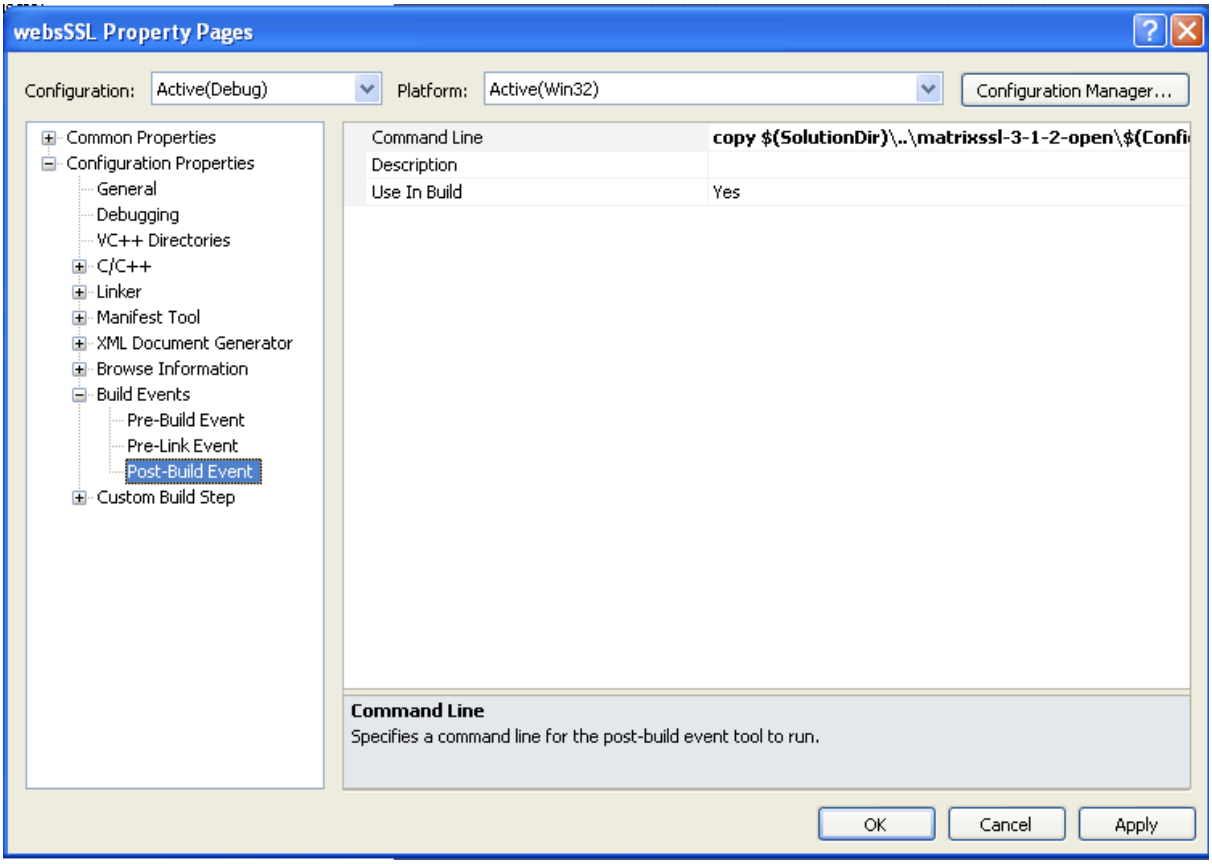
10. Build the Solution by right-clicking the websSSL Solution and selecting Build Solution from the drop-down list.



11. Confirm there were no build errors and that a *websSSL* executable has been generated.

12. In order to execute the websSSL server from inside the Visual Studio environment, the MatrixSSL DLL must be copied to the same directory as the executable. One automated way to do this is to create a Post-Build event into the websSSL project as follows.

    a. Open the Property Pages to websSSL and expand Configuration Properties, expand Build Events and select Post-Build Event



    b. Edit the Command Line to copy the *matrixssl.dll* file from the matrixssl build directory to the webSSL build directory. It should be:

    copy $(SolutionDir)\..\matrixssl-3-1-3-open\$(Configuration)\matrixssl.dll $(SolutionDir)\$(Configuration)

Testing the Secure Web server

To access the running Web server over an encrypted connection, open a browser to https://localhost:4433/.

Browsers such as Internet Explorer and Safari will present a warning about unsigned certificates and allow you to continue loading the page. Some browsers, such as Firefox 3.0 and above, will return an error message indicating that the certificate is not trusted:

```
Alert
localhost:4433 uses an invalid security certificate.
The certificate is not trusted because the issuer certificate is unknown.
The certificate is only valid for Sample Server Cert

(Error code: sec_error_unknown_issuer)
```

This error message is given because the sample certificate provided with the WebServer distribution has not been issued by a Certificate Authority that the browser recognizes (such as Verisign or GoDaddy). Of course, in this case you are running a server on your local network, so it is okay to add an exception for this certificate in your browser. The exception should be removed once you have generated or obtained your own certificate and private key for the server.

In Firefox, you can provide an exception for the site while testing under:

```
Preferences->Advanced->Encryption->View Certificates->Servers->Add Exception
```

The dialog will ask for the host URL, which in this example is "https://localhost:4433". Enter the name of your host and port, and click "Get Certificate". The certificate is loaded and you have the option to have an exception for this certificate on this host. Click Ok, and you will be able to access the Web server via https.

Increasingly, browsers are making it difficult to accept certificates that are self-signed, or ones where the host name in the certificate doesn't match the host name you are accessing. This adds a layer of security and requires the extra measures described above when testing with sample certificates.

Configuring the secure Web server

SSL Options

The following options are set in websSSL.h and matrixssl/src/matrixConfig.h

CFlag	Effect
Server Certificate File `DEFAULT_CERT_FILE`	Defines the .pem file containing the X.509 server certificate. The location can be absolute, or relative to the current working directory. This certificate contains the server's public key and may be self signed or signed by a root certificate chain. (websSSL.h)
Server Private Key File `DEFAULT_KEY_FILE`	Defines the .pem file containing the server's private key. The location can be absolute, or relative to the current working directory. This certificate contains the server's private key and should not be publicly accessible. (websSSL.h)
Directories Requiring SSL `websRequireSSL(url)`	This API designates a URL directory that can only be accessed through https. It may be called multiple times to secure multiple directories. (Defined in websSSL.h)

SSL Ciphers
`USE_SSL_RSA_WITH_RC4_128_MD5`
`USE_SSL_RSA_WITH_RC4_128_SHA`
`USE_SSL_RSA_WITH_3DES_EDE_CBC_SHA`
	These three ciphers are defined in matrixssl/src/matrixConfig.h. RC4_MD5 is the fastest and least secure. 3DES_SHA is the most secure. Comment out any ciphers you do not want, and recompile MatrixSSL.