TLS Test Tool User Manual V1.15.99

achelos

achelos GmbH
Vattmannstraße 1
33100 Paderborn
Germany

Phone +49 5251 14212-0
info@achelos.de
www.achelos.de

Contents

1 Introduction

The TLS Test Tool is able to test a huge variety of TLS clients and servers.

This user manual gives an overview over the structure of the TLS Test Tool and its usage. Furthermore, interfaces on different levels as well as the input and output formats for their correct usage are explained.

The system overview (Chapter 2) describes the interaction between the TLS Test Tool and its user. Then, the development environment setup and the steps required to build the TLS Test Tool from source (Chapter 3) are given. Afterwards, the format of the command line arguments and configuration files (Chapter 4) is defined. The changes in the last version (Chapter **??**) contain relevant information for users of the TLS Test Tool.

1.1 Operating Principle

When the configuration contains no manipulation, the TLS Test Tool behaves like a normal TLS peer. As client, it establishes a TCP/IP connection and starts a TLS handshake by sending a ClientHello message. As server, it binds to a TCP port, waits for an incoming connection and an incoming ClientHello message that is responded with a ServerHello message. Then, the TLS handshake takes place. After the handshake, the TLS Test Tool sends a Closure Alert, closes the TCP/IP connection, and exits.

The user can influence this default behavior by using one or more of the provided manipulations.

2 System overview

The TLS Test Tool tests a TLS implementation over a TCP/IP connection. Internally, a TLS library is used. Currently, this TLS library is mbed TLS. The usage of this library is encapsulated by an abstraction layer that allows changing the library in the future with little effort. The TLS Test Tool can be used by a human user as well as by an automatic test suite. Figure 2.1 gives an overview of the TLS Test Tool and its environment.

The input for the TLS Test Tool, its command line arguments and configuration, is described in Chapter 4. The output, the TLS Test Tool's return value and log output, is described in Chapter 5.

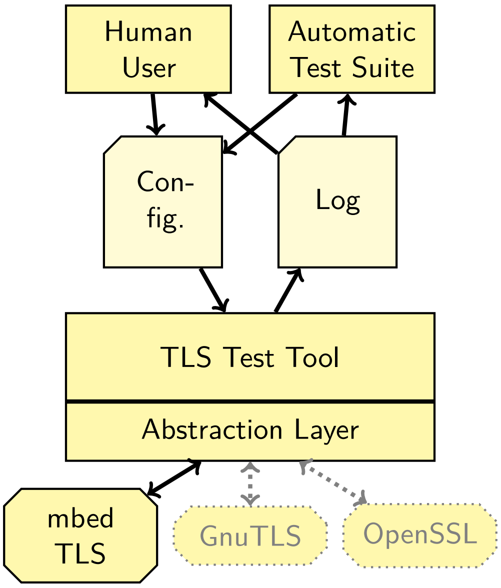

Figure 2.1: Structure of the TLS Test Tool.

3 Building from source

This chapter describes the steps necessary for setting up a development environment to build the TLS Test Tool. If you received the TLS Test Tool in binary form (e.g., an executable called `TlsTestTool`), you can skip this chapter.

3.1 Building on Linux

This section describes the build on a GNU/Linux system.

3.1.1 Required software

The TLS Test Tool uses CMake as its build system and requires a C++14-compatible compiler. For building external libraries, Perl and patch are required. For example, on a Debian GNU/Linux system, the required software can be acquired by installing the following packages:

- build-essential
- cmake

3.1.2 Building third-party libraries

There is a special part of the build that downloads and builds the required third-party libraries. It is located inside the `third_party` subdirectory. To build it, perform the following steps.

1. Open a shell
2. `cd [...]/TLS_Test_Tool/third_party`
3. `mkdir build`
4. `cd build`
5. `cmake -DBUILD_ASIO=ON -DBUILD_CPPUTEST=ON -DBUILD_MBEDTLS=ON -DBUILD_ZLIB=ON ..`
6. `cmake --build .`

3.1.3 Building the tool

To build the tool from the command line, perform the following steps.

1. Open a shell.
2. `cd [...]/TLS_Test_Tool`
3. `mkdir build`
4. `cd build`
5. `cmake -DCMAKE_BUILD_TYPE=Release ..`
6. `cmake --build .`
7. You can run the tests with: `cmake --build . --target test`

3.2 Building on Windows

In the following, the build procedure is described specifically for a Windows host machine.

3.2.1 Required software

The software required for building the TLS Test Tool is MSYS2/MinGW64 (e.g., msys2-x86_64-20160921.exe) and can be found on the Internet.

Install and start Start MSYS MinGW 64-bit and make sure `MINGW64` is displayed in the shell:

- Make sure to install the required packages with the following command:
  `pacman --force -S make mingw-w64-x86_64-binutils mingw-w64-x86_64-cmake mingw-w64-x86_64-gcc mingw-w64-x86_64-make mingw-w64-x86_64-perl patch`
- Please install the required package for clang-format with the following command:
  `pacman -S mingw-w64-x86_64-clang`
- (Optional) Please install the required package for debugging with the following command:
  `pacman -S mingw-w64-x86_64-gdb`

If you do not put the tools into your `PATH`, make sure to use full paths in the steps below.
But it is recommended to add `[...]/msys64/mingw64/bin` to the PATH.

3.2.2 Building third-party libraries

There is a special part of the build that downloads and builds the required third-party libraries. It is located inside the `third_party` subdirectory. To build it, perform the following steps.

1. Open MSYS2/MinGW64.
2. `cd [...]/TLS_Test_Tool/third_party`
3. `mkdir build`
4. `cd build`
5. `cmake -G "MSYS Makefiles" -DBUILD_ASIO=ON -DBUILD_CPPUTEST=ON -DBUILD_MBEDTLS=ON -DBUILD_ZLIB=ON ..`
6. `make`

3.2.3 Building the tool

To build the tool from the command line, perform the following steps.

1. Open MSYS2/MinGW64.
2. `cd [...]/TLS_Test_Tool`
3. `mkdir build`
4. `cd build`
5. `cmake -G "MSYS Makefiles" -DCMAKE_BUILD_TYPE=Release ..`
6. `make`
7. You can run the tests with: `make test`

3.3 Using Eclipse

For example, you can use the Eclipse IDE for C/C++ Developers (e.g., eclipse-cpp-2018-09-win32-x86_64.zip). For using Eclipse, CMake has to generate an Eclipse project. The build directory should **not be** located inside the source directory for usage with Eclipse.
If it hasn't been done yet, put `[...]/msys64/mingw64/bin` to the PATH. It is also important that the `sh.exe` **is not** in your PATH. Otherwise the following steps will not work.

1. Open a command window.
2. `cd [...]/TLS_Test_Tool/..`
3. `mkdir build-eclipse`
4. `cd build-eclipse`
5.
```
cmake -G "Eclipse CDT4 - MinGW Makefiles" -DCMAKE_BUILD_TYPE=RelWithDebInfo \
-DCMAKE_ECLIPSE_EXECUTABLE=[...]/eclipse-cpp-win32-x86_64/eclipse.exe \
../TLS_Test_Tool
```

6. You can check, if the build works outside Eclipse, with: `cmake --build .`

You can now use the newly created project.

1. Open Eclipse CDT.
2. Use “File”, “Import...”, “General”/“Existing Projects into Workspace” to locate your build directory.
3. Import the Eclipse project from the build directory.
4. In the Project Explorer, you now see the imported project with “[Source directory]” containing the sources and “Binaries” showing the binaries after the build.

3.4 Using Qt Creator

As another example, you can use Qt Creator (e.g., contained in qt-opensource-windows-x86-msvc2015_64-5.6.1.exe).

1. Open the file `[...]/TLS_Test_Tool/CMakeLists.txt` in Qt Creator.
2. In the project configuration dialog that is shown, click the button to manage the available kits.
3. You can skip this step, if you have already done it before. Create a kit that does not use Qt, uses your MinGW64 compiler, and your CMake installation.

   - Configuration of the compiler if it has not yet been auto detected:
     Options->Build & Run->Compilers->Add->MinGW
     Compiler path: `[...]/msys64/mingw64/bin/g++.exe`
     ABI: x86-windows-msys-pe-64bit
   - Configuration of CMake if it has not yet been auto detected:
     Options->Build & Run->CMake->Add
     Name: CMake
     Path: `[...]/msys64/mingw64/bin/cmake.exe`
   - (Optional) Configuration of the Debugger if it has not yet been auto detected:
     Options->Build & Run->Debuggers->Add
     Name: GDB
     Path: `[...]/msys64/mingw64/bin/gdb.exe`
   - Configuration of the Kit if it has not yet been auto detected:
     Options->Build & Run->Kits->Add
     Name: MinGW
     Compiler: MinGW
     Debugger: GDB (optional)
     CMake Tool: CMake
     CMake Generator: CodeBlocks - MinGW Makefiles

   Mark the kit as default by pressing the button “Make Default” and close the settings dialog.
4. Select the newly created kit for the TLS Test Tool project.

5. In “Details”, select only the release build with debug information and specify a build directory of your choice (e.g., `[...]/TLS_Test_Tool/build-qtcreator`).
6. Click “Configure project”.
7. In the following dialog, leave the “Arguments” as is and choose the “MinGW Generator”.
8. Build & Run the project.
9. If you want to test the TLS Test Tool: Build & Run “TlsTestToolTest” instead of “TlsTestTool”.

Use clang-format with QT Creator (a detailed tutorial can be found here clang-format).

1. Select Help->Plugins->C++->Beautifier to enable the plugin.
2. Restart Qt Creator to be able to use the plugin.
3. Select Tools->Options->Beautifier->Clang Format
   Clang Format command: [...]/msys64/mingw64/bin/clang-format.exe
   Options - Use predefined style: File

There is an external tutorial on how to integrate Perforce in QT Creator.

4 Configuration

This chapter describes the available arguments for configuring the TLS Test Tool. Values that a user has to provide are denoted in square brackets (e.g., `[length]` for a value named `length`).

4.1 Command line arguments

The TLS Test Tool expects at least one argument on the command line.

- `--configFile=[configuration file path]`
  Specify the path to a configuration file. When this argument is given multiple times, the given configuration files are read. Options from configuration files that are given later on the command line will overwrite options from those given earlier.

Examples:

- `TlsTestTool --configFile=config/TestCase27.conf`
- `TlsTestTool --configFile=device.conf --configFile=tlsOptions.conf`

4.2 Configuration file

The configuration for the TLS Test Tool is given in a configuration file. The configuration file is a plain text file. Lines that start with the hash sign (#) are treated as comments and ignored. Arguments are given as name-value pairs separated with the equals sign (=). The following arguments are known.

4.2.1 Input of binary data

Binary data is given in hexadecimal form. The bytes of a byte array have to be encoded separately and printed separated by a space character. Each byte is represented by two digits from `0-9a-f`. For example, the array of the two bytes `0xc0 0x30` has to be given as `c0 30`. In the following, the word `HEXSTRING` is used as placeholder for an arbitrary byte array. Please note that an empty byte array is possible and has to be represented by an empty string.

4.2.2 Network options

- `mode=[mode]`
  (required, with `mode` either `client` or `server`)

  Specify the mode for the TLS Test Tool. If `mode=client`, the TLS Test Tool will run as a TLS client and connect to a server using TCP/IP. If `mode=server`, the TLS Test Tool will run as a TLS server and listen for incoming TCP/IP connections.

- `host=[host]`
  (required, if `mode=client`, with string `host`)

  If `mode=client`, specify a host name or IP address that the TLS Test Tool should connect to. Ignored, if `mode=server`.

- `port=[port]`
  (required, with decimal integer `port`)

  If `mode=client`, the TCP port of a service to connect to. If `mode=server`, the TCP port to bind and listen to on the local host.

- `listenTimeout=[timeout]`
  (with decimal integer `timeout`)

  If `mode=server`, the TLS Test Tool will exit if no incoming TCP/IP connection is received within `timeout` seconds. If not specified or `timeout` equals zero, the TLS Test Tool will listen forever. Ignored, if `mode=client`.

- `waitBeforeClose=[timeout]`
  (with decimal integer `timeout`)

  Specify the `timeout` in seconds that the tool waits for incoming data after a run before closing the TCP/IP connection.

- `receiveTimeout=[timeout]`
  (with decimal integer `timeout`)

  Specify the `timeout` in seconds that the tool waits for incoming TCP/IP packets during a receive operation.

4.2.3 Logging options

- `logLevel=[level]`
  (with `level` from {`off`, `low`, `medium`, `high`}, default `off`)

  Amount of log output on the command line (see Chapter 5).

  - `high`
    Much debug output (e.g., additional hex dumps).
  - `medium`
    Medium amount of debug output (e.g., additional output of sizes of received packages).
  - `low`
    Little debug output (e.g., print actions that are performed).

  - `off`
    No output.

- `logFilterRegEx=[regEx]`
  (with a regular expression `regEx`)

  Match log messages against the regular expression `regEx`. If a message `MESSAGE` matches, a new log entry with the message `Matched message: MESSAGE` is written to the log.

4.2.4 TLS options

- `caCertificateFile=[path]`
  (with `path` pointing to a PEM- or DER-encoded file)

  File containing a X.509 CA certificate that will be used to verify peer certificates.

- `certificateFile=[path]`
  (with `path` pointing to a PEM- or DER-encoded file)

  File containing a X.509 certificate that will be used as server or client certificate, respectively, depending on the mode.

- `privateKeyFile=[path]`
  (with `path` pointing to a PEM- or DER-encoded file)

  File containing a private key that matches the certificate's public key.

- `tlsVersion=([major],[minor])`
  (with decimal integers `major` equal to 3 and `minor` from [1, 3])

  If `mode=client`, send the specified version in ClientHello.client_version. If `mode=server`, accept only the specified version and send it in ServerHello.server_version. Use `(3,1)` for TLS v1.0, `(3,2)` for TLS v1.1, `(3,3)` for TLS v1.2. If not specified, all three TLS versions are accepted by a server and the highest version is sent by a client.

- `tlsVerifyPeer=[verify]`
  (with Boolean value `verify` either `true` or `false`, default `false`)

  If `false`, a peer certificate is not verified. If `true`, a valid peer certificate is required. If no valid peer certificate is presented, the TLS handshake is aborted.

- `tlsCipherSuites=([valueUpper],[valueLower])[,([valueUpper],[valueLower])...]`
  (with hexadecimal integers `valueUpper` and `valueLower` preceded with `0x`)

  Specify a list of supported TLS cipher suites in decreasing order of preference. If this option is set, at least one TLS cipher suite has to be given. If `mode=client`, send the list in ClientHello.cipher_suites. If `mode=server`, use this list to find a matching TLS cipher suite to send in ServerHello.cipher_suite. The values correspond to the values from the TLS Cipher Suite Registry. For example, the value `(0xC0,0x2C)` corresponds to TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. If not specified, a default list of TLS cipher suites is used.

- `tlsServerDHParams=[predefined]`
  (with `predefined` equal to a key defined below)

  If `mode=server`, configure the Diffie-Hellman group that will be used. The value of `predefined` can be one of the keys given in the following table.

Key	Definition
`rfc3526_1536`	RFC 3526 – 1536-bit MODP Group
`rfc3526_2048`	RFC 3526 – 2048-bit MODP Group
`rfc3526_3072`	RFC 3526 – 3072-bit MODP Group
`rfc3526_4096`	RFC 3526 – 4096-bit MODP Group
`rfc3526_6144`	RFC 3526 – 6144-bit MODP Group
`rfc3526_8192`	RFC 3526 – 8192-bit MODP Group
`rfc5114_1024_160`	RFC 5114 – 1024-bit MODP Group with 160-bit Prime Order Subgroup
`rfc5114_2048_224`	RFC 5114 – 2048-bit MODP Group with 224-bit Prime Order Subgroup
`rfc5114_2048_256`	RFC 5114 – 2048-bit MODP Group with 256-bit Prime Order Subgroup

Ignored, if `mode=client`. If not specified, a default Diffie-Hellman group is used.

- `tlsSecretFile=[path]`
  (with `path` pointing to an output file)

  Append the master_secret in the NSS Key Log Format to a plain text file. This file can be used by Wireshark to decrypt TLS packets.

- `tlsEncryptThenMac=[enabled]`
  (with Boolean value `enabled` either `true` or `false`, default `true`)

  Enable (`enabled=true`) or disable (`enabled=false`) the usage of Encrypt-then-MAC.

4.2.5 Procedure manipulations

- `manipulateSkipChangeCipherSpec=[ignored]`
  (with an arbitrary, possibly empty value `ignored`)

  Skip sending a ChangeCipherSpec message and directly send a Finished message.

- `manipulateSkipFinished=[ignored]`
  (with an arbitrary, possibly empty value `ignored`)

  If `mode=client`, skip sending a Finished message. Ignored, if `mode=server`.

4.2.6 Miscellaneous manipulations

- `manipulatePreMasterSecretRandom=[ignored]`
  (with an arbitrary, possibly empty value `ignored`)

  If `mode=client`, replace the field PreMasterSecret.random in a ClientKeyExchange message with non-zero random bytes. The manipulation is done before performing the encrypting that results in EncryptedPreMasterSecret. Please note that the structure EncryptedPreMasterSecret is created only if the key exchange method is RSA. Ignored, if `mode=server`.

- `manipulatePreMasterSecretRandomByte=[index]`
  (with positive, decimal integer `index`)

  If `mode=client`, replace the byte with the given index in the field PreMasterSecret.random with a zero byte. Since the length of the field is 46 bytes, the maximum allowed index is 45. The manipulation is done before performing the encrypting that results in EncryptedPreMasterSecret.

Please note that the structure EncryptedPreMasterSecret is created only if the key exchange method is RSA. Ignored, if `mode=server`.

- `manipulatePreMasterSecretVersion=([major],[minor])`
  (with hexadecimal integers `major` and `minor` preceded with `0x`)

  If `mode=client`, replace the field PreMasterSecret.client_version in a ClientKeyExchange message with the two bytes given in `major` and `minor`. The manipulation is done before performing the encrypting that results in EncryptedPreMasterSecret. Please note that the structure EncryptedPreMasterSecret is created only if the key exchange method is RSA. Ignored, if `mode=server`.

- `manipulateRsaesPkcs1V15EncryptPadding=[firstByte],[blockType],[padding]`
  (with 3 hexadecimal integer `firstByte`, `blockType` and `padding` preceded with `0x`)

  If `mode=client`, overwrite the first byte with the value of `firstByte`, the block type byte with the value of `blockType` and the value of the byte between PS and M with the value of `padding` in RSAES-PKCS1-V1_5-ENCRYPT when using RSA to create the EncryptedPreMasterSecret in a ClientKeyExchange message. Ignored, if `mode=server`.

4.2.7 Examples

Example for running as a TLS client:

```
# Connect to a local web server
mode=client
host=192.168.0.1
port=443
logLevel=high
caCertificateFile=path/to/ca_certificate.pem
```

Example for running as a TLS server:

```
# Run locally on the LDAPS port
mode=server
port=636
logLevel=low
certificateFile=server/certificate.pem
privateKeyFile=server/private/key.pem
```

5 Reporting

The different kinds of output of the TLS Test Tool and their formats are defined in the following sections.

5.1 Return value

On success, the TLS Test Tool returns zero. On fatal error, e.g., a configuration file cannot be parsed, the TLS Test Tool returns one.

5.2 Output

The TLS Test Tool writes its output to standard output.

5.3 Log format

The most important output of the TLS Test Tool is its log. The log entries are written line by line. The lines are terminated by a newline character (`'\n'`, ASCII code 10). The format of each log entry is described in Section 5.3.1.

5.3.1 Log entry format

A log entry is represented by one line in the log. Each log line consists of four columns. The columns are separated by a horizontal tab character (`'\t'`, ASCII code 9). The contents of the columns are as follows:

1. The log entry's timestamp (see Section 5.3.2).
2. The log entry's severity (see Section 5.3.3).
3. The log entry's origin (see Section 5.3.4).
4. The log entry's message (see Section 5.3.5).

The structural format of a log line can be seen below.

```
Timestamp       Severity        Origin  Message
```

Below, an example of an actual log line is given.

```
2016-05-04T08:43:17.356Z        HIGH    TLS(TlsLogFilter.cpp:119)       ServerHello.server_version=03 03
```

5.3.2 A log entry's timestamp

A log entry's timestamp is given in the format `YYYY-MM-DDThh:mm:ss.fffZ`. This format follows the definitions of ISO 8601. `YYYY` is the year as a 4 digit decimal number. `MM` is the month as a 2 digit decimal number from $[1, 12]$. `DD` is the day of the month as a 2 digit decimal number from $[1, 31]$. Year, month, and day are separated by a hyphen character (`'-'`, ASCII code 45). `hh` is the hour as a 2 digit decimal number in 24 hour clock format from $[0, 23]$. `mm` is the minute as a 2 digit decimal number from $[0, 59]$. `ss` is the second as a 2 digit decimal number from $[0, 59]$. `fff` is the decimal fraction of a second as a 3 digit decimal number from $[0, 999]$. Hour, minute, and second are separated by a colon character (`':'`, ASCII code 58). Second and the decimal fraction of a second are separated by a dot character (`'.'`, ASCII code 46). This means that the resolution of the timestamps is 1 millisecond. The date and the time are separated by a `'T'` character (ASCII code 84). The time is given in UTC. Therefore, the last character is `'Z'` (ASCII code 90) denoting the time zone with zero UTC offset. An actual timestamp looks like this: `2016-05-04T08:43:17.356Z`.

5.3.3 A log entry's severity

A log entry's severity is either `HIGH`, `MEDIUM`, or `LOW`.

5.3.4 A log entry's origin

A log entry's origin identifies the author of a log entry. It may contain an arbitrary string. In case of the TLS Test Tool, it is given as the combination of a category, a file name, and a line number. The category identifies the module that generated the log entry (e.g., `TLS`). It is followed by a left parenthesis (`'('`, ASCII code 40). The file name identifies the source code file that generated the log entry. It is separated from the line number inside this file by a colon character (`':'`, ASCII code 58). The origin is finished with a right parenthesis (`')'`, ASCII code 41). An actual origin looks like this: `TLS(TlsLogFilter.cpp:119)`.

5.3.5 A log entry's message

A log entry's message contains free text. Many messages are not defined to have a fixed format (e.g., packet dumps, informational output by the TLS library). Some messages generated by the TLS Test Tool have a fixed format. See Section 5.4 for a definition of these messages.

5.3.6 Output of binary data

In the log output, binary data is printed in hexadecimal form. The format is defined in Section 4.2.1. As defined there, the word `HEXSTRING` is used in the following.

5.4 Definition of messages

A set of fixed messages (see Section 5.3.5) is defined for the output of the TLS Test Tool. These messages can be seen as a public interface of the TLS Test Tool. At first, general messages are defined. Following, message definitions for specific messages of a TLS handshake are given.

5.4.1 Networking

When running the TLS Test Tool as server, it will log a log message of the format "Waiting for TCP/IP connection on port PORT." when it is listening on port PORT. When a client with IP address CLIENT_IP_ADDRESS and port CLIENT_PORT has established a connection, a log message of the format "TCP/IP connection from CLIENT_IP_ADDRESS:CLIENT_PORT received." is written. An actual example of a log output when running the TLS Test Tool as server is given below.

```
2016-05-12T14:38:34.220Z        HIGH    Network(TlsTestTool.cpp:135)    Waiting for TCP/IP connection on
     port 443.
2016-05-12T14:38:40.881Z        HIGH    Network(TlsTestTool.cpp:140)    TCP/IP connection from
     127.0.0.1:53266 received.
```

When running the TLS Test Tool as client, it will log a message of the format "TCP/IP connection to SERVER_IP_ADDRESS:SERVER_PORT established." when it has established a connection to the server at the IP address SERVER_IP_ADDRESS and port SERVER_PORT. As an actual example, below a log output when running the TLS Test Tool as client is given.

```
2016-05-13T06:49:32.405Z        HIGH    Network(TlsTestTool.cpp:95)     TCP/IP connection to
     62.214.75.58:443 established.
```

If the TCP/IP connection is closed, the message "TCP/IP connection is closed." is logged. For example, a log output for the closing of a TCP/IP connection looks like this:

```
2016-05-13T06:49:32.487Z        HIGH    Network(TlsTestTool.cpp:65)     TCP/IP connection is closed.
```

Message size and time stamp

A size and a time stamp will be reported for messages that are sent or received by the TLS Test Tool. The size is given as the number of bytes that are written to or read from the TCP/IP connection, respectively. The time stamp is given as the number of nanoseconds (10^{-9} s) elapsed since an arbitrary point in time. It does not necessarily represent wall clock time. Therefore, a time stamp should be used only in comparison to another time stamp.

When a message with a size of SIZE bytes is sent, a log message of the format "Write.size=SIZE" is written. When at a time TIME a message is sent, a log message of the format "Write.timestamp=TIME" is written. An actual example of a log output when sending a message looks like this:

```
2016-12-01T07:51:35.032Z        HIGH    Network(TimestampObserver.cpp:101)      Write.size=385
2016-12-01T07:51:35.032Z        HIGH    Network(TimestampObserver.cpp:132)
     Write.timestamp=1480578695032601755
```

When a message with a size of SIZE bytes is received, a log message of the format "Read.size =SIZE" is written. When at a time TIME a message is received, a log message of the format "Read.timestamp=TIME" is written. An actual example of a log output when receiving a message is given below:

```
2016-12-01T07:51:35.067Z        HIGH    Network(TimestampObserver.cpp:140)      Read.size=5
2016-12-01T07:51:35.067Z        HIGH    Network(TimestampObserver.cpp:142)
     Read.timestamp=1480578695067526150
```

Both SIZE and TIME are given as non-negative decimal numbers.

5.4.2 Receiving of TLS messages

The successful or unsuccessful receiving and parsing of a TLS message is documented in the log. After successfully parsing an incoming TLS message of type `TYPE`, the log message "`Valid TYPE message received.`" is written. When a failure while parsing an incoming TLS message of type `TYPE` occurs, the log message "`Bad TYPE message received.`" is written.

The following table gives an overview of all log messages created for the receiving of TLS messages.

TLS message type	Successful parsing	Log message format
ClientHello	Yes	`Valid ClientHello message received.`
ClientHello	No	`Bad ClientHello message received.`
ServerHello	Yes	`Valid ServerHello message received.`
ServerHello	No	`Bad ServerHello message received.`
Certificate	Yes	`Valid Certificate message received.`
Certificate	No	`Bad Certificate message received.`
ServerKeyExchange	Yes	`Valid ServerKeyExchange message received.`
ServerKeyExchange	No	`Bad ServerKeyExchange message received.`
CertificateRequest	Yes	`Valid CertificateRequest message received.`
CertificateRequest	No	`Bad CertificateRequest message received.`
ServerHelloDone	Yes	`Valid ServerHelloDone message received.`
ServerHelloDone	No	`Bad ServerHelloDone message received.`
ClientKeyExchange	Yes	`Valid ClientKeyExchange message received.`
ClientKeyExchange	No	`Bad ClientKeyExchange message received.`
CertificateVerify	Yes	`Valid CertificateVerify message received.`
CertificateVerify	No	`Bad CertificateVerify message received.`
ChangeCipherSpec	Yes	`Valid ChangeCipherSpec message received.`
ChangeCipherSpec	No	`Bad ChangeCipherSpec message received.`
Finished	Yes	`Valid Finished message received.`
Finished	No	`Bad Finished message received.`

5.4.3 Transmitting of TLS messages

The transmitting of a TLS message is documented in the log. After an outgoing TLS message of type `TYPE` has been sent, the log message "`TYPE message transmitted.`" is written.

The following table gives an overview of all log messages created for the transmitting of TLS messages.

TLS message type	Log message format
ClientHello	`ClientHello message transmitted.`
ServerHello	`ServerHello message transmitted.`
Certificate	`Certificate message transmitted.`
ServerKeyExchange	`ServerKeyExchange message transmitted.`
CertificateRequest	`CertificateRequest message transmitted.`
ServerHelloDone	`ServerHelloDone message transmitted.`
ClientKeyExchange	`ClientKeyExchange message transmitted.`
CertificateVerify	`CertificateVerify message transmitted.`
ChangeCipherSpec	`ChangeCipherSpec message transmitted.`
Finished	`Finished message transmitted.`

5.4.4 ClientHello

The following fields of a ClientHello message are printed in the given format to the log.

TLS message field	Log message format
client_version	`ClientHello.client_version=HEXSTRING`
random	`ClientHello.random=HEXSTRING`
session_id	`ClientHello.session_id=HEXSTRING`
cipher_suites	`ClientHello.cipher_suites=HEXSTRING`
compression_methods	`ClientHello.compression_methods=HEXSTRING`
extensions	`ClientHello.extensions=HEXSTRING`

Below, an output example of an actual ClientHello message is given.

```
2016-05-09T08:58:31.854Z        HIGH    TLS(TlsLogFilter.cpp:143)       ClientHello.client_version=03 03
2016-05-09T08:58:31.855Z        HIGH    TLS(TlsLogFilter.cpp:143)       ClientHello.random=2e 89 d9 24 33
    f2 70 b0 2f 22 a5 e8 bb ed 39 ef 08 3a a0 28 45 68 ce a5 82 8a 52 4e 29 1a 08 ab
2016-05-09T08:58:31.856Z        HIGH    TLS(TlsLogFilter.cpp:143)       ClientHello.session_id=ec 49 ea
    4d 5b 5f af 12 43 8b 0b 0d 86 c4 c4 2b 76 b1 b3 7f f4 8e e5 13 b4 de ac f5 fa 22 dc 86
2016-05-09T08:58:31.857Z        HIGH    TLS(TlsLogFilter.cpp:143)       ClientHello.cipher_suites=c0 2b
    c0 2f c0 0a c0 09 c0 13 c0 14 00 33 00 39 00 2f 00 35 00 0a
2016-05-09T08:58:31.858Z        HIGH    TLS(TlsLogFilter.cpp:143)       ClientHello.compression_methods=00
2016-05-09T08:58:31.862Z        HIGH    TLS(TlsLogFilter.cpp:143)       ClientHello.extensions=00 00 00
    0e 00 0c 00 00 09 6c 6f 63 61 6c 68 6f 73 74 00 17 00 00 ff 01 00 01 00 00 0a 00 08 00 06 00 17 00
    18 00 19 00 0b 00 02 01 00 00 23 00 00 33 74 00 00 00 10 00 17 00 15 02 68 32 08 73 70 64 79 2f 33
    2e 31 08 68 74 74 70 2f 31 2e 31 00 05 00 05 01 00 00 00 00 00 0d 00 16 00 14 04 01 05 01 06 01 02
    01 04 03 05 03 06 03 02 03 04 02 02 02
```

5.4.5 ServerHello

The following fields of a ServerHello message are printed in the given format to the log.

TLS message field	Log message format
server_version	`ServerHello.server_version=HEXSTRING`
random	`ServerHello.random=HEXSTRING`
session_id	`ServerHello.session_id=HEXSTRING`
cipher_suite	`ServerHello.cipher_suite=HEXSTRING`
compression_method	`ServerHello.compression_method=HEXSTRING`
extensions	`ServerHello.extensions=HEXSTRING`

Below, an output example of an actual ServerHello message is given.

```
2016-06-06T06:51:24.994Z        HIGH    TLS(TlsLogFilter.cpp:226)       ServerHello.server_version=03 03
2016-06-06T06:51:25.009Z        HIGH    TLS(TlsLogFilter.cpp:226)       ServerHello.random=d9 06 62 c7 c9
    0f 31 10 ed fb c5 82 de e5 ba 11 b9 2b 5d a7 76 0a 78 01 7b eb 4f 17 b8 fb 3a f9
2016-06-06T06:51:25.009Z        HIGH    TLS(TlsLogFilter.cpp:226)       ServerHello.session_id=
2016-06-06T06:51:25.025Z        HIGH    TLS(TlsLogFilter.cpp:126)       ServerHello.cipher_suite=00 33
2016-06-06T06:51:25.025Z        HIGH    TLS(TlsLogFilter.cpp:145)       ServerHello.compression_method=00
2016-06-06T06:51:25.041Z        HIGH    TLS(TlsLogFilter.cpp:226)       ServerHello.extensions=ff 01 00
    01 00 00 23 00 00
```

5.4.6 ServerKeyExchange

The following fields of a ServerKeyExchange message are printed in the given format to the log.

TLS message field	Log message format
params.dh_p	`ServerKeyExchange.params.dh_p=HEXSTRING`
params.dh_g	`ServerKeyExchange.params.dh_g=HEXSTRING`
params.dh_Ys	`ServerKeyExchange.params.dh_Ys=HEXSTRING`
signed_params.algorithm.hash	`ServerKeyExchange.signed_params.algorithm.hash=HEXSTRING`
signed_params.algorithm.signature	`ServerKeyExchange.signed_params.algorithm.signature=HEXSTRING`
signed_params.signature	`ServerKeyExchange.signed_params.signature=HEXSTRING`
signed_params.md5_hash	`ServerKeyExchange.signed_params.md5_hash=HEXSTRING`
signed_params.sha_hash	`ServerKeyExchange.signed_params.sha_hash=HEXSTRING`
params.curve_params.namedcurve	`ServerKeyExchange.params.curve_params.namedcurve=HEXSTRING`
params.public	`ServerKeyExchange.params.public=HEXSTRING`

Please note that the values of the fields `md5_hash` and `sha_hash` are computed by the TLS Test Tool and not read from an incoming ServerKeyExchange message. Therefore, the reported values only equal those from an incoming ServerKeyExchange message, if the signature is validated successfully.

Below, an output example of an actual ServerKeyExchange message for DHE and TLS 1.1 is given.

```
2016-06-08T14:02:01.562Z        HIGH    TLS(TlsLogFilter.cpp:271)       ServerKeyExchange.params.dh_p=ff
    ff ff ff ff ff ff ff c9 0f da a2 21 68 c2 34 c4 c6 62 8b 80 dc 1c d1 29 02 4e 08 8a 67 cc 74 ...
2016-06-08T14:02:01.570Z        HIGH    TLS(TlsLogFilter.cpp:271)       ServerKeyExchange.params.dh_g=02
2016-06-08T14:02:01.691Z        HIGH    TLS(TlsLogFilter.cpp:271)       ServerKeyExchange.params.dh_Ys=4e
    ac 93 08 47 0b 05 5b 0c 7b dd 2c 53 c1 ed 11 53 50 0a 94 fe 1e d7 9e 82 90 0e c4 97 77 26 cb ...
2016-06-08T14:02:01.803Z        HIGH    TLS(TlsLogFilter.cpp:232)
    ServerKeyExchange.signed_params.signature=98 c3 36 7f 82 dd 2e c2 f2 32 30 14 45 68 1e e1 0a a6 b0
    1a 20 da a2 87 1a 7e 3a 63 8e 22 26 30 ...
2016-06-08T14:02:01.811Z        HIGH    TLS(TlsLogFilter.cpp:232)
    ServerKeyExchange.signed_params.md5_hash=23 8d dc 3a 0b 0d 0d 2c 73 b4 20 44 b0 00 4c 6b
2016-06-08T14:02:01.822Z        HIGH    TLS(TlsLogFilter.cpp:232)
    ServerKeyExchange.signed_params.sha_hash=f4 49 36 c0 0f 61 c3 cb b3 01 7e 69 d3 69 68 f3 8c 8d 6e 2a
```

Below, an output example of an actual ServerKeyExchange message for DHE and TLS 1.2 is given.

```
2016-06-03T13:17:19.879Z        HIGH    TLS(TlsLogFilter.cpp:263)       ServerKeyExchange.params.dh_p=ff
     ff ff ff ff ff ff ff c9 0f da a2 21 68 c2 34 c4 c6 62 8b 80 dc 1c d1 29 02 4e 08 8a 67 cc 74 ...
2016-06-03T13:17:19.879Z        HIGH    TLS(TlsLogFilter.cpp:263)       ServerKeyExchange.params.dh_g=02
2016-06-03T13:17:20.010Z        HIGH    TLS(TlsLogFilter.cpp:263)       ServerKeyExchange.params.dh_Ys=1f
     88 7d a3 2b 4c 8c 80 77 47 f8 f5 41 ea bc 6d fe cb 8a 0e bb 2a cf 6e c4 7b 61 fb 0b 1a 10 6b ...
2016-06-03T13:17:20.010Z        HIGH    TLS(TlsLogFilter.cpp:145)
     ServerKeyExchange.signed_params.algorithm.signature=01
2016-06-03T13:17:20.012Z        HIGH    TLS(TlsLogFilter.cpp:145)
     ServerKeyExchange.signed_params.algorithm.hash=06
2016-06-03T13:17:20.163Z        HIGH    TLS(TlsLogFilter.cpp:224)
     ServerKeyExchange.signed_params.signature=94 79 b9 45 e0 3c 96 62 b0 7b fd c5 6c 1d 0b 0d d0 9b db
     3d 86 82 99 e0 51 1d 03 95 81 c9 45 c8 ...
```

Below, an output example of an actual ServerKeyExchange message for ECDHE is given.

```
2016-05-10T11:30:36.420Z        HIGH    TLS(TlsLogFilter.cpp:208)
     ServerKeyExchange.params.curve_params.namedcurve=17
2016-05-10T11:30:36.420Z        HIGH    TLS(TlsLogFilter.cpp:210)
     ServerKeyExchange.params.public=04 7c 85 8f b8 0c 28 58 d5 67 f0 fe e9 96 d2 03 90 aa 0c e2 65 9c ff
     a4 52 ae 07 67 31 a2 b0 fe 48 ec 37 e0 56 c9 c5 ce 07 da 68 c8 44 ac 83 16 ff 2d a8 b4 4e fc 7a 1f
     6b d7 04 17 fa e0 4f 76 61
```

5.4.7 CertificateRequest

The following fields of a CertificateRequest message are printed in the given format to the log.

TLS message field	Log message format
certificate_types	`CertificateRequest.certificate_types=HEXSTRING`
supported_signature_algorithms	`CertificateRequest.supported_signature_algorithms=HEXSTRING`
certificate_authorities	`CertificateRequest.certificate_authorities=HEXSTRING`

Please note that the field `supported_signature_algorithms` is available only if TLS 1.2 is used.

Below, an output example of an actual CertificateRequest message for TLS 1.1 is given.

```
2016-07-22T07:35:24.821Z        HIGH    TLS(TlsLogFilter.cpp:80)
     CertificateRequest.certificate_types=01 40
2016-07-22T07:35:24.821Z        HIGH    TLS(TlsLogFilter.cpp:96)
     CertificateRequest.certificate_authorities=00 50 30 4e 31 10 30 0e 06 03 55 04 03 13 07 54 65 73 74
     20 43 41 31 16 30 14 06 03 55 04 0b 13 0d 54 4c 53 20 54 65 73 74 20 54 6f 6f 6c 31 15 30 13 06 03
     55 04 0a 13 0c 61 63 68 65 6c 6f 73 20 47 6d 62 48 31 0b 30 09 06 03 55 04 06 13 02 44 45
```

Below, an output example of an actual CertificateRequest message for TLS 1.2 is given.

```
2016-07-22T07:30:54.721Z        HIGH    TLS(TlsLogFilter.cpp:80)
     CertificateRequest.certificate_types=01 40
2016-07-22T07:30:54.721Z        HIGH    TLS(TlsLogFilter.cpp:87)
     CertificateRequest.supported_signature_algorithms=05 01 05 03
2016-07-22T07:30:54.721Z        HIGH    TLS(TlsLogFilter.cpp:96)
     CertificateRequest.certificate_authorities=00 50 30 4e 31 10 30 0e 06 03 55 04 03 13 07 54 65 73 74
     20 43 41 31 16 30 14 06 03 55 04 0b 13 0d 54 4c 53 20 54 65 73 74 20 54 6f 6f 6c 31 15 30 13 06 03
     55 04 0a 13 0c 61 63 68 65 6c 6f 73 20 47 6d 62 48 31 0b 30 09 06 03 55 04 06 13 02 44 45
```

5.4.8 ClientKeyExchange

The following fields of a ClientKeyExchange message are printed in the given format to the log.

TLS message field	Log message format
exchange_keys.pre_master_secret	`ClientKeyExchange.exchange_keys.pre_master_secret =HEXSTRING`
exchange_keys.master_secret	`ClientKeyExchange.exchange_keys.master_secret= HEXSTRING`

Below, an output example of an actual ClientKeyExchange message for ECDHE is given.

```
2016-06-06T06:35:24.194Z        HIGH    TLS(TlsLogFilter.cpp:225)
    ClientKeyExchange.exchange_keys.pre_master_secret=03 03 4e 3f d8 25 b2 e7 a0 db 40 d1 55 47 8e 09 88
    e5 b4 b9 73 0f ce fa 0d 59 26 a8 ed 2f c6 9b f9 f4 e8 c0 8c 8a f3 0f 87 a0 f5 8e 3c a2 4e ac
2016-06-06T06:35:24.194Z        HIGH    TLS(TlsLogFilter.cpp:225)
    ClientKeyExchange.exchange_keys.master_secret=c5 1f ab 0d 51 20 ff de b9 1b e8 1e 1c e9 42 a8 13 ef
    40 12 58 28 93 2c cb 2c 62 7a 97 40 c8 a0 ef c7 c7 52 f9 bf bd 8c ec 8c 86 3b e4 d7 06 ff
```

5.4.9 Certificate

The following fields of a Certificate message are printed in the given format to the log.

TLS message field	Log message format

certificate_list	`Certificate.certificate_list.size=NUMBER`
	`Certificate.certificate_list[NUMBER]=HEXSTRING`

Below, an output example of an actual Certificate message is given.

```
2016-05-19T14:53:10.002Z        HIGH    TLS(TlsLogFilter.cpp:174)
    Certificate.certificate_list[0]=30 82 05 e4 30 82 04 cc a0 03 02 01 02 02 10 3d f2 5b 3c ad 86 ...
    9d 09 f6 6f 52 2f da db ba 48 f7 6b 12 da f9 18 f4 3a 04 d9 30 a5 35 32 02 70 84 77 f6 ad
2016-05-19T14:53:10.002Z        HIGH    TLS(TlsLogFilter.cpp:174)
    Certificate.certificate_list[1]=30 82 05 e5 30 82 03 cd a0 03 02 01 02 02 10 6a 5d c3 e5 3b 4e ...
    26 e2 f5 09 a9 4b 37 36 92 e3 cd 6e b5 7c 3e f6 d3 2c 85 ee a5 f6 45 16 3d 1d f6 6a 5a 16
2016-05-19T14:53:10.003Z        HIGH    TLS(TlsLogFilter.cpp:181)
    Certificate.certificate_list.size=2
```

5.4.10 Finished

The following fields of a TLSCiphertext record containing a Finished message are printed in the given format to the log.

TLS message field	Log message format
GenericBlockCipher.IV	`Finished.GenericBlockCipher.IV=HEXSTRING`
GenericBlockCipher.padding_length	`Finished.GenericBlockCipher.padding_length= HEXSTRING`

Below, an output example of an actual Finished message is given.

```
2016-05-23T07:57:45.639Z        HIGH    TLS(TlsLogFilter.cpp:127)
     Finished.GenericBlockCipher.padding_length=0b
...
2016-05-23T07:57:45.672Z        HIGH    TLS(TlsSession.cpp:243) Finished.GenericBlockCipher.IV=4e 08 8a
     36 e2 26 b7 0d ba 6f 68 5f bc aa 05 2c
```

5.4.11 Alert

When receiving an Alert message, the log message "`Alert message received.`" is written. Additionally, the following fields of an Alert message are printed in the given format to the log.

TLS message field	Log message format
level	`Alert.level=HEXSTRING`
description	`Alert.description=HEXSTRING`

Below, an output example of an actual Alert message is given.

```
2016-05-12T11:01:13.535Z        HIGH    TLS(TlsLogFilter.cpp:62)        Alert message received.
2016-05-12T11:01:13.535Z        HIGH    TLS(TlsLogFilter.cpp:85)        Alert.level=02
2016-05-12T11:01:13.535Z        HIGH    TLS(TlsLogFilter.cpp:85)        Alert.description=28
```

5.4.12 NewSessionTicket

The following fields of a NewSessionTicket message are printed in the given format to the log.

TLS message field	Log message format
ticket	`NewSessionTicket.ticket=HEXSTRING`

Below, an output example of an actual NewSessionTicket message is given.

```
2018-05-14 15:28:09.645 TLS(TlsLogFilter.cpp:270) NewSessionTicket.ticket=c0 a2 c9 db 26 37 f9 60 d8 90
     a0 fa 9c 6e 86 1e 2b 53 9d 87 5f 8f d8 3c fb 11 f6 5d a8 0b db a2 6a 2d 32 81 2a 48 34 f7 4f 4c 45
     b4 d1 88 5b e3 2a cc 03 f6 f8 15 ee f5 04 46 b0 96 e5 9b d0 89 4f 07 9f 7e 4f 30 db a5 73 f7 5b 99
     e6 54 a3 ad 1b 02 9b 58 49 a3 a3 52 54 21 cf 08 13 30 5f 12 8d 90 cd e3 6a 62 9b 16 c6 19 ee 17 11
     22 91 48 ce 19 9f f3 a9 73 d7 fb 4e 7f 47 96 63 09 f2 5d b8 87 cb cd ad c3 4a 55 04 38 2b 61 d5 87
     ee 42 2e 26 6b 1b c0 6c 12 d7 77 72 11 5c 5b 8c 36 a8
```

6 Licenses

The licenses of 3rd party software used within TLS Test Tool are listed below.

6.1 Mbed TLS version 2.2.1

https://tls.mbed.org
Source: https://tls.mbed.org/download/mbedtls-2.2.1-apache.tgz

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.
```

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You

institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

```
   APPENDIX: How to apply the Apache License to your work.

      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "[]"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright [yyyy] [name of copyright owner]

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

6.2 Zlib version 1.2.11

https://www.zlib.net
Source: http://download.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz

```
  zlib.h -- interface of the 'zlib' general purpose compression library
  version 1.2.11, January 15th, 2017

  Copyright (C) 1995-2017 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
```

6.3 CppUTest version 3.8

CppUTest unit testing and mocking framework for C/C++. CppUTest is licensed under the BSD 3-Clause "New" or "Revised" License. A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the project or its contributors to promote derived products without written consent.

https://cpputest.github.io
Source: https://github.com/cpputest/cpputest/releases/download/v3.8/cpputest-3.8.tar.gz