General Information Guide

Cross-Domain Development Kit XDK110
Platform for Application Development

Bosch Connected Devices and Solutions

BOSCH
Invented for life

XDK110: Data Sheet

Document revision	1.0
Document release date	18.05.17
Document number	BCDS-XDK110-GUIDE HTTPS
Technical reference code(s)	
Notes	Data in this document is subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product's appearance. **Subject to change without notice**

XDK HTTPS Guide

Platform for Application Development

HTTPS is a common protocol to securely transfer data and files over a network. Since the XDK does not support HTTPS natively with an own high-level API a HTTPS connection needs to be configured manually. This guide will provide an introduction on how to establish a HTTPS socket connection.

Table of Contents

This guide postulates a basic understanding of the XDK and according Workspace. For new users we recommend going through the following guides at xdk.io/guides first:

- *Workbench Installation*
- *Workbench First Steps*
- *XDK Guide FreeRTOS*
- *XDK Guide Wi-Fi*

Furthermore this guide assumes a basic understanding of the HTTP protocol and its usage. For users that would prefer an introduction to HTTP first, we recommend the following guide:

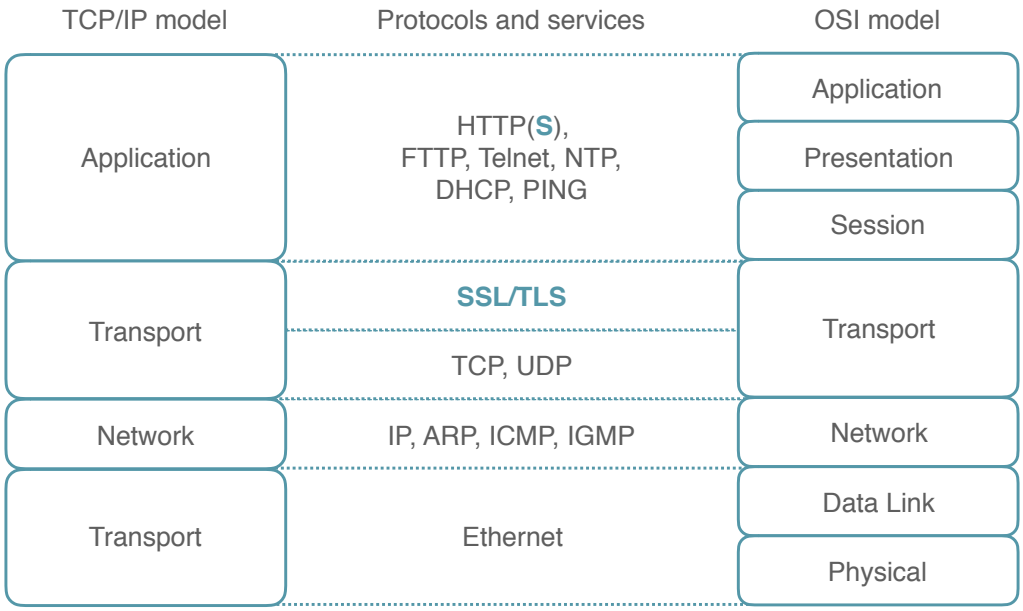
- *XDK Guide HTTP*

1 The HTTPS Protocol

HTTPS (HyperText Transfer Protocol Secure) is a data transfer protocol that runs on top of the TCP/IP protocol stack as shown in Picture 1. Regarding the transmission of data, HTTPS uses the same technologies and paradigms as HTTP, but unlike plain HTTP it has an extra layer for encryption.
This additional layer handles digital certificates that are used for authentication via a socket by using the protocols SSL (Secure Socket Layer) or TLS (Transport Layer Security).

The default port for HTTPS is the port 443.

Picture 1. HTTP in the OSI and TCP/IP stack



Note: The XDK uses version 1.1 of the HTTP protocol (“HTTP/1.1”). The complete specification of this version is available in the form of an RFC document: https://tools.ietf.org/html/rfc2616

2 API Overview

In general, it is recommended to develop an application based on the highest API level the XDK framework supports. The opposite will be necessary in this guide, since there is no high level HTTPS API, yet.

Since the XDK is equipped with a Texas Instruments CC3100 Wi-Fi chip, the XDK comes with the TI Simple Link API[1] that provides a low level access to the chip and also includes some other APIs, like the socket interface, which is necessary for implementing an HTTPS connection. Below, you can find a list of the main functions that will be used in this guide. For a full description please refer to the corresponding API[2].

Table 1. Socket interface functions (excerpt)

Function	Description
`sl_Socket`	Create a socket instance. Returns a handle for the created socket (in form of an integer identifier)
`sl_SetSockOpt`	Generic function for setting any options for the passed socket. This guide will show how to set the certificate and the secure method with this function. See the API documentation to learn which other settings can be made.
`sl_Connect`	Use this function to open the socket connection.
`sl_Close`	Use this function to close the socket connection.
`sl_Send`	Use this function to send a request via the given socket.
`sl_Recv`	Use this function to receive data from the given socket.

[1] http://xdk.bosch-connectivity.com/xdk_docs/html/group__ti__networking.html

[2] http://xdk.bosch-connectivity.com/xdk_docs/html/socket_8h.html

3 Preparation

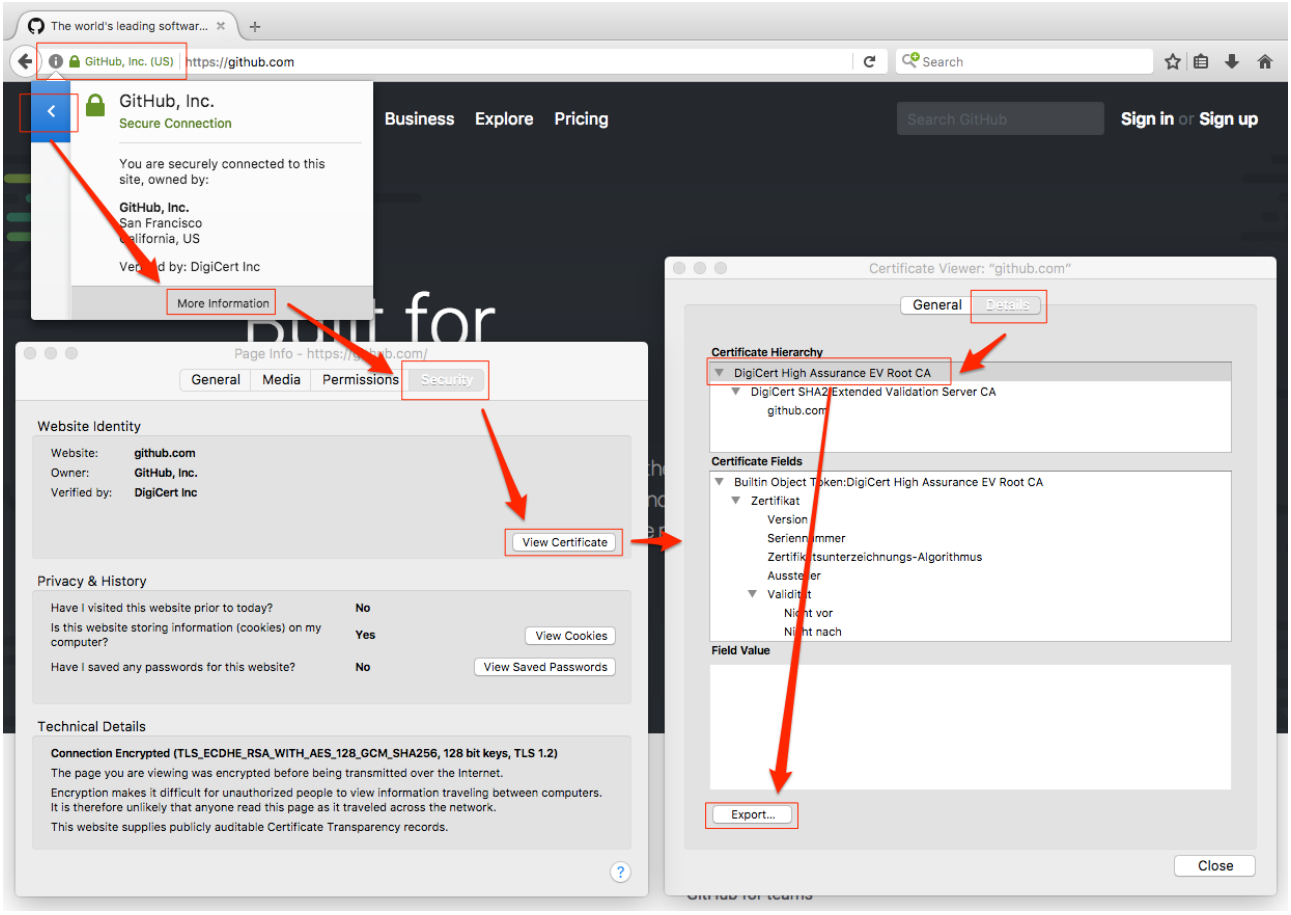
As mentioned above, the difference between HTTP and HTTPS is the authentication layer. The authentication is handled by digital certificates. To use HTTPS, the client needs to have the correct certificate for the requested website. The following steps will show an example on how to download, convert and include the binary of the GitHub[3] certificate, by using the Mozilla Firefox[4] web browser. The entire project will be based on an empty XdkApplicationTemplate, which can be found in the Welcome-Screen of the XDK-Workbench.

3.1 Getting the certificate

Open the GitHub website. In your navigation bar you should see a little green lock. Click it and follow the link in the first row of the appearing menu. Now click „More Information“. Navigate to the security tab and view the certificate information. In the details tab, you will see the certificate hierarchy. It is possible that the hierarchy might take a moment to appear.

For HTTPS usage, you have to download the root certificate, which is the highest one in the chain. Select the root certificate, which is the „DigiCert High Assurance EV Root CA“ certificate in our example and export it to a .cer or .der file, which are the binary data types for certificates.

Picture 2. Certificate information of a website in Firefox



[3] https://github.com/

[4] https://www.mozilla.org

Note: If you are having trouble with the authentication later on, it is most likely that you have downloaded the wrong certificate in this step. If you know the name of the certificate you can also download it immediately at the certificate authorities (CA) website. In the given example, the CA is DigiCert. You can find a list of all their root certificates on https://www.digicert.com/digicert-root-certificates.htm.

3.2 Converting the certificate to hex

The downloaded file will have the extension .cer oder .der, but for usage in the XDK application, the easiest way is to convert the file into a hexadecimal format, so it can be copied into a C-array.

For a C-array, the hexadecimal representation must be comma-separated and the numbers must have the leading `0x` notation.

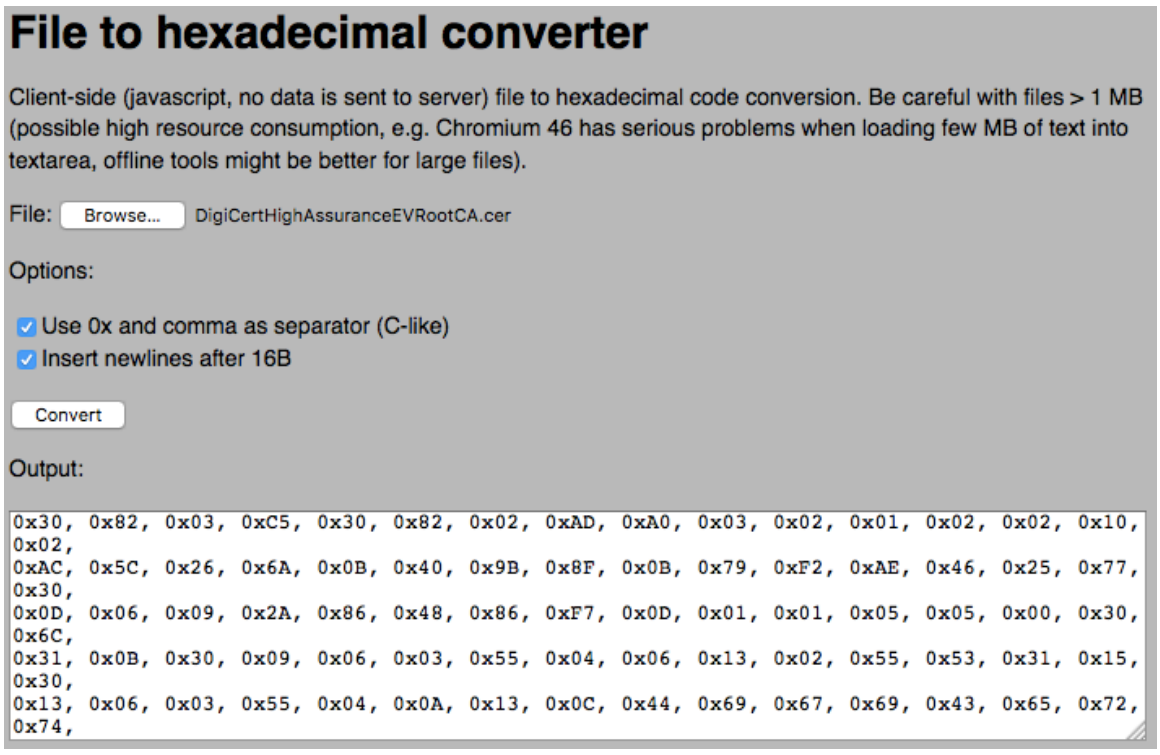
The conversion can be done in several ways. The easiest way to convert the certificate is to use the command line tool `xxd` that is by default available on Linux and Mac operating systems. The output of this command is directly usable C code.

Code 1. Convert Binary to Hex with xxd

```
xxd -i path/to/the/file.cer
```

If you don't have a Bash Shell, you can also use multiple different online tools or programs. An easy online tool is this File to hexadecimal converter[5] because you can directly upload a file and get a C-formatted response.

Picture 3. Converting the certificate to hex



[5] http://tomeko.net/online_tools/file_to_hex.php?lang=en

3.3 Storing the certificate

To keep the implementation code clean, the metadata of the certificate can be stored in the header file. Additionally, the hexadecimal version of the certificate the HTTPS API will require the length and the targets host name as well as a filename. Put the following code in your header file:

Code 2. Certificate data in the header file

```
#define HOST_NAME "github.com"
#define CA_FILE_NAME "digicert.der"

unsigned char digicert_root_crt[] = {
    // Place your hex values here.
    // In case you had trouble with converting the binary, see the actual hex
    // values of GitHub in the appendix!
};

int digicert_root_crt_len = 969;
```

If you are not sure if you have done the bin to hex conversion correctly, you can find the full code example including the correct hex values in the appendix.

Note: Depending on the purpose of the application, it might also be useful to store the certificate on an SD card and load it on run time. In this case please see the SdCardExample in the workbench.

4 Implementation

The main logic of this guide is based on the functions that are being provided by the Simple Link API. To access them include, the API at the beginning of your implementation file like in Code 3.

Code 3. Include Simple Link API

```
#include <simplelink.h>
```

4.1 Network Setup

To perform a network request, a client needs a working network connection to the server.
The easiest way to achieve this is to connect the XDK to the local network via its Wi-Fi interface. After setting up this link, the XDK can send requests to a locally hosted server, or if the access point is connected to the internet, to any server on the World Wide Web. However, establishing a Wi-Fi connection requires some configuration.

For just the minimal setup required to establish a Wi-Fi connection, a function similar to Code 5 can be used. If this doesn't work out of the box after providing the access point SSID and the password, or a special network setup is required, please refer to the XDK Wi-Fi guide available at xdk.io/guides for further instructions.

Note: The code snippets in this guide contain almost no error handling code in order to keep them short and simple. For real-world projects however, return code validation and error handling are substantial. Information on possible return and error codes can be found in the API documentation: http://xdk.bosch-connectivity.com/xdk_docs/html/modules.html.

Code 4. Additional includes for the network setup

```
#include "BCDS_WlanConnect.h"
#include "BCDS_NetworkConfig.h"
#include "PAL_initialize_ih.h"
#include "PAL_socketMonitor_ih.h"
```

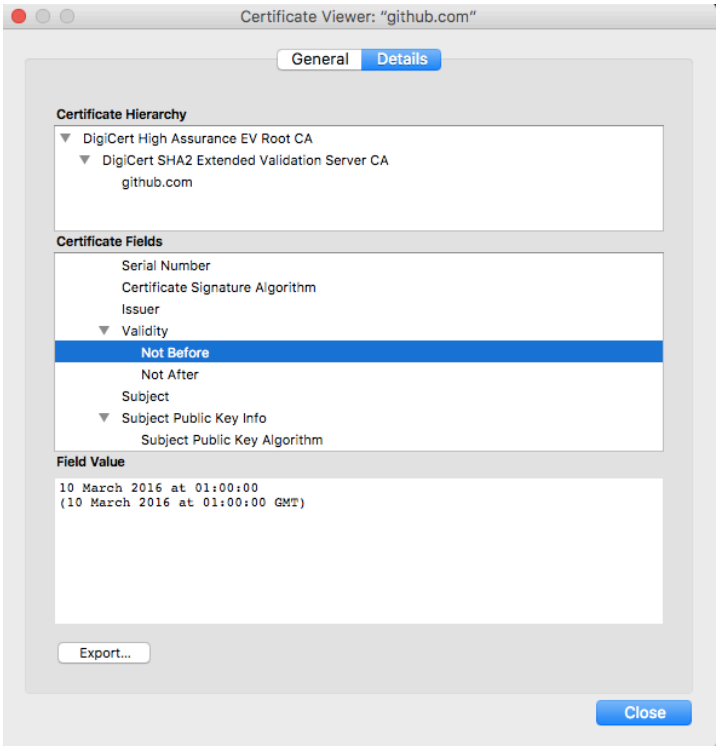
Code 5. Minimal Network setup

```
void networkSetup(void){
    WlanConnect_SSID_T connectSSID = (WlanConnect_SSID_T) "SSID";
    WlanConnect_PassPhrase_T connectPassPhrase = (WlanConnect_PassPhrase_T)
"PW";
    WlanConnect_Init();
    NetworkConfig_SetIpDhcp(0);
    WlanConnect_WPA(connectSSID, connectPassPhrase, NULL);

    PAL_initialize();
    PAL_socketMonitorInit();
}

void appInitSystem(xTimerHandle xTimer){
    BCDS_UNUSED(xTimer);

    // Consider to schedule the following instructions with a timer to avoid
    // problems. You could, for example, pack all the steps in one function
    // and call it within a timer callback:
    networkSetup();
}
```

4.2 Handling the certificate

Each certificate has a valid-from and a valid-until timestamp as can be seen in the screenshot below.

Picture 4. Certificate validities



Only in between these two dates will a HTTPS request with this certain certificate be successful. Since the XDK has no internal clock it is necessary to set a date and time. For this purpose, the actual date and time is of low importance, as long as the value is within the valid timeframe. After creating any `SlDateTime_t` value, the datetime can be written to the device by using the `sl_DevSet` function that belongs to the Device Interface[6] of the Simple Link API.

To use the certificate, it has to be written to the flash memory of the XDK by the usage of the functions `sl_FsOpen`, `sl_FsWrite` and `sl_FsClose`.

Note: The maximum size a file writer can handle is 1024 bytes. If your certificate has more than 1024 bytes you need to loop the writing.

The following code shows a simple function that implements the previously described steps. You can call it after your network setup call. As parameters, you can provide `CA_FILE_NAME`, `digicert_root_crt` and `digicert_root_crt_len` which you should have defined in your header file.

[6] For further information see http://xdk.bosch-connectivity.com/xdk_docs/html/group__device.html

Code 6. Flashing the certificate

```
// Flash the Certificate BEFORE attempting to connect.
// I.e. call this function in appInitSystem() before connectServerSecure from
// Code 8
void flashCertificate(char *fileName, _u8* data, _u32 length) {
    // For the purpose of readability this code has no error handling.
    // The simplelink API provides return codes of the type _i32 that can be
    // checked for the value SL_RET_CODE_OK

    // The datetime is required for certificate validation:
    SlDateTime_t dateTime;
    dateTime.sl_tm_day =   (_u32)12;
    dateTime.sl_tm_mon =   (_u32)3;
    dateTime.sl_tm_year =  (_u32)2017;
    dateTime.sl_tm_hour =  (_u32)0;
    dateTime.sl_tm_min =   (_u32)0;
    dateTime.sl_tm_sec =   (_u32)0;

    sl_DevSet(
        SL_DEVICE_GENERAL_CONFIGURATION,
        SL_DEVICE_GENERAL_CONFIGURATION_DATE_TIME,
        sizeof(SlDateTime_t),
        (_u8 *)(&dateTime)
    );

    // If a file with the same name already exists, call this first:
    // sl_FsDel((_u8*) CA_FILE_NAME, 0)

    // The file handle should not be 0 after a file was successfully created:
    _i32 fileHandle = 0;

    sl_FsOpen(
        (_u8*) fileName,
        FS_MODE_OPEN_CREATE(
            1024, _FS_FILE_PUBLIC_WRITE | _FS_FILE_PUBLIC_READ
        ),
        NULL,
        &fileHandle
    );

       // If the file is longer than 1024 bytes, you need to loop the writing.
       // "length" contains the length of the certificate
       // "writtenLength" contains the amount of actually written bytes.
       _i32 writtenLen = sl_FsWrite(fileHandle, 0, data, length);

       sl_FsClose(fileHandle, NULL, NULL, 0);
}
```

4.2 Establishing the connection

The connection via HTTPS is basically a normal TCP socket with some specific settings to enable the TLS certificate. For general information see the documentation of the Simple Link Socket Interface on http://xdk.bosch-connectivity.com/xdk_docs/html/group__socket.html.

In general, a socket can be created by calling `sl_Socket()`. To demand a secure socket, the value `SL_SEC_SOCKET` must be passed as the parameter for the protocol. The return value will be an Integer value identifying this socket. Bind it to a variable to use it in the following calls. To bind the certificate to this socket, you can use the `sl_SetSockOpt()` function. Now you can connect to the chosen host by calling `sl_Connect()`. After successfully connecting, you can perform your requests until you close your connection with `sl_Close()`.

Code 7. Connect with certificate

```
// Call this function after flashCertificate() in appInitSystem()
void connectServerSecure(void){
    // For the purpose of readability this code has no error handling.
    // The simplelink API provides return codes of the type _i32 that can be
    // checked for the value SL_RET_CODE_OK

    // Getting the IP address of HOST_NAME:
    Ip_Address_T destAddr = 0;
    PAL_getIpaddress((uint8_t*) HOST_NAME, &destAddr);

    // Creating a Socket (socketHandle ≦ 0 indicates an error):
    _i16 socketHandle = sl_Socket(SL_AF_INET, SL_SOCK_STREAM, SL_SEC_SOCKET);

    // Adding the certificate to the socket:
    sl_SetSockOpt(
        socketHandle,
        SL_SOL_SOCKET,
        SL_SO_SECURE_FILES_CA_FILE_NAME,
        CA_FILE_NAME,
        strlen(CA_FILE_NAME)
    );

    // Configuration the connection settings (IP, Port, etc.):
    SlSockAddrIn_t addr;
    addr.sin_family = SL_AF_INET;
    addr.sin_port = sl_Htons(443);
    addr.sin_addr.s_addr = destAddr;

    // Connecting:
    sl_Connect(socketHandle, ( SlSockAddr_t *)&addr, sizeof(SlSockAddrIn_t));

    // Implementation Placeholder
    // Do your request, etc. here

    // Closing:
    sl_Close(socketHandle);
}
```

Note: The secure method is set by default to `SL_SO_SEC_METHOD_SSLv3_TLSV1_2`. The snippet in Code 8 shows how this can be changed. You can find the other possible options in the socket API.

Code 8. Setting the secure method manually

```
// Use this before connecting:
SlSockSecureMethod secMethod;
secMethod.secureMethod = SL_SO_SEC_METHOD_SSLv3_TLSV1_2;
sl_SetSockOpt(
    socketHandle,
    SL_SOL_SOCKET,
    SL_SO_SECMETHOD,
    (_u8 *)&secMethod,
    sizeof(secMethod)
);
```

4.3 Performing a GET request

This chapter will help you create a minimal GET request which you can use for the implementation placeholder from Code 7. The following snippet will perform a request for a specific page of the given host and simply prints the result to the console. The socket API is providing two basic functions for this: `sl_Send()` and `sl_Recv()`.

You can copy the following function to your code and call it in your connectServerSecure() function as shown below.

Code 9. Minimal GET request

```
// GET request for https://github.com/about
sendGetRequest(socketHandle, HOST_NAME, "/about");
```

```
void sendGetRequest(_i16 socketHandle, char* host, char* path)
{
    char outBuf[1024];
    char inBuf[1024];

    sprintf(
        outBuf,
        "GET https://%s%s HTTP/1.1\r\nHost: %s\r\n\r\n", host, path, host
    );

    sl_Send(socketHandle, (const void *) outBuf, strlen(outBuf), 0);
    sl_Recv(socketHandle, inBuf, 1024, 0);
    printf("%s\r\n", inBuf);
}
```

Note: This is just an example with a fixed-length char array. Increase the buffer size in the example code if necessary or use a loop.

The request above will not process the returned information in any way, since it is just a demonstration of the API functions. For a more advanced function that has some string formatting logic, please see Code 10 in the appendix.

Appendix

I. GitHub certificate hex values

In case you had any trouble with getting or converting the GitHub Certificate, you can copy the following lines:

Code 10. GitHub certificate hex values

```
unsigned char digicert_root_crt[] = {
  0x30, 0x82, 0x03, 0xc5, 0x30, 0x82, 0x02, 0xad, 0xa0, 0x03, 0x02, 0x01,
  0x02, 0x02, 0x10, 0x02, 0xac, 0x5c, 0x26, 0x6a, 0x0b, 0x40, 0x9b, 0x8f,
  0x0b, 0x79, 0xf2, 0xae, 0x46, 0x25, 0x77, 0x30, 0x0d, 0x06, 0x09, 0x2a,
  0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05, 0x00, 0x30, 0x6c,
  0x31, 0x0b, 0x30, 0x09, 0x06, 0x03, 0x55, 0x04, 0x06, 0x13, 0x02, 0x55,
  0x53, 0x31, 0x15, 0x30, 0x13, 0x06, 0x03, 0x55, 0x04, 0x0a, 0x13, 0x0c,
  0x44, 0x69, 0x67, 0x69, 0x43, 0x65, 0x72, 0x74, 0x20, 0x49, 0x6e, 0x63,
  0x31, 0x19, 0x30, 0x17, 0x06, 0x03, 0x55, 0x04, 0x0b, 0x13, 0x10, 0x77,
  0x77, 0x77, 0x2e, 0x64, 0x69, 0x67, 0x69, 0x63, 0x65, 0x72, 0x74, 0x2e,
  0x63, 0x6f, 0x6d, 0x31, 0x2b, 0x30, 0x29, 0x06, 0x03, 0x55, 0x04, 0x03,
  0x13, 0x22, 0x44, 0x69, 0x67, 0x69, 0x43, 0x65, 0x72, 0x74, 0x20, 0x48,
  0x69, 0x67, 0x68, 0x20, 0x41, 0x73, 0x73, 0x75, 0x72, 0x61, 0x6e, 0x63,
  0x65, 0x20, 0x45, 0x56, 0x20, 0x52, 0x6f, 0x6f, 0x74, 0x20, 0x43, 0x41,
  0x30, 0x1e, 0x17, 0x0d, 0x30, 0x36, 0x31, 0x31, 0x31, 0x30, 0x30, 0x30,
  0x30, 0x30, 0x30, 0x30, 0x5a, 0x17, 0x0d, 0x33, 0x31, 0x31, 0x31, 0x31,
  0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x5a, 0x30, 0x6c, 0x31, 0x0b,
  0x30, 0x09, 0x06, 0x03, 0x55, 0x04, 0x06, 0x13, 0x02, 0x55, 0x53, 0x31,
  0x15, 0x30, 0x13, 0x06, 0x03, 0x55, 0x04, 0x0a, 0x13, 0x0c, 0x44, 0x69,
  0x67, 0x69, 0x43, 0x65, 0x72, 0x74, 0x20, 0x49, 0x6e, 0x63, 0x31, 0x19,
  0x30, 0x17, 0x06, 0x03, 0x55, 0x04, 0x0b, 0x13, 0x10, 0x77, 0x77, 0x77,
  0x2e, 0x64, 0x69, 0x67, 0x69, 0x63, 0x65, 0x72, 0x74, 0x2e, 0x63, 0x6f,
  0x6d, 0x31, 0x2b, 0x30, 0x29, 0x06, 0x03, 0x55, 0x04, 0x03, 0x13, 0x22,
  0x44, 0x69, 0x67, 0x69, 0x43, 0x65, 0x72, 0x74, 0x20, 0x48, 0x69, 0x67,
  0x68, 0x20, 0x41, 0x73, 0x73, 0x75, 0x72, 0x61, 0x6e, 0x63, 0x65, 0x20,
  0x45, 0x56, 0x20, 0x52, 0x6f, 0x6f, 0x74, 0x20, 0x43, 0x41, 0x30, 0x82,
  0x01, 0x22, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d,
  0x01, 0x01, 0x01, 0x05, 0x00, 0x03, 0x82, 0x01, 0x0f, 0x00, 0x30, 0x82,
  0x01, 0x0a, 0x02, 0x82, 0x01, 0x01, 0x00, 0xc6, 0xcc, 0xe5, 0x73, 0xe6,
  0xfb, 0xd4, 0xbb, 0xe5, 0x2d, 0x2d, 0x32, 0xa6, 0xdf, 0xe5, 0x81, 0x3f,
  0xc9, 0xcd, 0x25, 0x49, 0xb6, 0x71, 0x2a, 0xc3, 0xd5, 0x94, 0x34, 0x67,
  0xa2, 0x0a, 0x1c, 0xb0, 0x5f, 0x69, 0xa6, 0x40, 0xb1, 0xc4, 0xb7, 0xb2,
  0x8f, 0xd0, 0x98, 0xa4, 0xa9, 0x41, 0x59, 0x3a, 0xd3, 0xdc, 0x94, 0xd6,
  0x3c, 0xdb, 0x74, 0x38, 0xa4, 0x4a, 0xcc, 0x4d, 0x25, 0x82, 0xf7, 0x4a,
  0xa5, 0x53, 0x12, 0x38, 0xee, 0xf3, 0x49, 0x6d, 0x71, 0x91, 0x7e, 0x63,
  0xb6, 0xab, 0xa6, 0x5f, 0xc3, 0xa4, 0x84, 0xf8, 0x4f, 0x62, 0x51, 0xbe,
  0xf8, 0xc5, 0xec, 0xdb, 0x38, 0x92, 0xe3, 0x06, 0xe5, 0x08, 0x91, 0x0c,
  0xc4, 0x28, 0x41, 0x55, 0xfb, 0xcb, 0x5a, 0x89, 0x15, 0x7e, 0x71, 0xe8,
  0x35, 0xbf, 0x4d, 0x72, 0x09, 0x3d, 0xbe, 0x3a, 0x38, 0x50, 0x5b, 0x77,
  0x31, 0x1b, 0x8d, 0xb3, 0xc7, 0x24, 0x45, 0x9a, 0xa7, 0xac, 0x6d, 0x00,
  0x14, 0x5a, 0x04, 0xb7, 0xba, 0x13, 0xeb, 0x51, 0x0a, 0x98, 0x41, 0x41,
  0x22, 0x4e, 0x65, 0x61, 0x87, 0x81, 0x41, 0x50, 0xa6, 0x79, 0x5c, 0x89,
```

```
    0xde, 0x19, 0x4a, 0x57, 0xd5, 0x2e, 0xe6, 0x5d, 0x1c, 0x53, 0x2c, 0x7e,
    0x98, 0xcd, 0x1a, 0x06, 0x16, 0xa4, 0x68, 0x73, 0xd0, 0x34, 0x04, 0x13,
    0x5c, 0xa1, 0x71, 0xd3, 0x5a, 0x7c, 0x55, 0xdb, 0x5e, 0x64, 0xe1, 0x37,
    0x87, 0x30, 0x56, 0x04, 0xe5, 0x11, 0xb4, 0x29, 0x80, 0x12, 0xf1, 0x79,
    0x39, 0x88, 0xa2, 0x02, 0x11, 0x7c, 0x27, 0x66, 0xb7, 0x88, 0xb7, 0x78,
    0xf2, 0xca, 0x0a, 0xa8, 0x38, 0xab, 0x0a, 0x64, 0xc2, 0xbf, 0x66, 0x5d,
    0x95, 0x84, 0xc1, 0xa1, 0x25, 0x1e, 0x87, 0x5d, 0x1a, 0x50, 0x0b, 0x20,
    0x12, 0xcc, 0x41, 0xbb, 0x6e, 0x0b, 0x51, 0x38, 0xb8, 0x4b, 0xcb, 0x02,
    0x03, 0x01, 0x00, 0x01, 0xa3, 0x63, 0x30, 0x61, 0x30, 0x0e, 0x06, 0x03,
    0x55, 0x1d, 0x0f, 0x01, 0x01, 0xff, 0x04, 0x04, 0x03, 0x02, 0x01, 0x86,
    0x30, 0x0f, 0x06, 0x03, 0x55, 0x1d, 0x13, 0x01, 0x01, 0xff, 0x04, 0x05,
    0x30, 0x03, 0x01, 0x01, 0xff, 0x30, 0x1d, 0x06, 0x03, 0x55, 0x1d, 0x0e,
    0x04, 0x16, 0x04, 0x14, 0xb1, 0x3e, 0xc3, 0x69, 0x03, 0xf8, 0xbf, 0x47,
    0x01, 0xd4, 0x98, 0x26, 0x1a, 0x08, 0x02, 0xef, 0x63, 0x64, 0x2b, 0xc3,
    0x30, 0x1f, 0x06, 0x03, 0x55, 0x1d, 0x23, 0x04, 0x18, 0x30, 0x16, 0x80,
    0x14, 0xb1, 0x3e, 0xc3, 0x69, 0x03, 0xf8, 0xbf, 0x47, 0x01, 0xd4, 0x98,
    0x26, 0x1a, 0x08, 0x02, 0xef, 0x63, 0x64, 0x2b, 0xc3, 0x30, 0x0d, 0x06,
    0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05, 0x00,
    0x03, 0x82, 0x01, 0x01, 0x00, 0x1c, 0x1a, 0x06, 0x97, 0xdc, 0xd7, 0x9c,
    0x9f, 0x3c, 0x88, 0x66, 0x06, 0x08, 0x57, 0x21, 0xdb, 0x21, 0x47, 0xf8,
    0x2a, 0x67, 0xaa, 0xbf, 0x18, 0x32, 0x76, 0x40, 0x10, 0x57, 0xc1, 0x8a,
    0xf3, 0x7a, 0xd9, 0x11, 0x65, 0x8e, 0x35, 0xfa, 0x9e, 0xfc, 0x45, 0xb5,
    0x9e, 0xd9, 0x4c, 0x31, 0x4b, 0xb8, 0x91, 0xe8, 0x43, 0x2c, 0x8e, 0xb3,
    0x78, 0xce, 0xdb, 0xe3, 0x53, 0x79, 0x71, 0xd6, 0xe5, 0x21, 0x94, 0x01,
    0xda, 0x55, 0x87, 0x9a, 0x24, 0x64, 0xf6, 0x8a, 0x66, 0xcc, 0xde, 0x9c,
    0x37, 0xcd, 0xa8, 0x34, 0xb1, 0x69, 0x9b, 0x23, 0xc8, 0x9e, 0x78, 0x22,
    0x2b, 0x70, 0x43, 0xe3, 0x55, 0x47, 0x31, 0x61, 0x19, 0xef, 0x58, 0xc5,
    0x85, 0x2f, 0x4e, 0x30, 0xf6, 0xa0, 0x31, 0x16, 0x23, 0xc8, 0xe7, 0xe2,
    0x65, 0x16, 0x33, 0xcb, 0xbf, 0x1a, 0x1b, 0xa0, 0x3d, 0xf8, 0xca, 0x5e,
    0x8b, 0x31, 0x8b, 0x60, 0x08, 0x89, 0x2d, 0x0c, 0x06, 0x5c, 0x52, 0xb7,
    0xc4, 0xf9, 0x0a, 0x98, 0xd1, 0x15, 0x5f, 0x9f, 0x12, 0xbe, 0x7c, 0x36,
    0x63, 0x38, 0xbd, 0x44, 0xa4, 0x7f, 0xe4, 0x26, 0x2b, 0x0a, 0xc4, 0x97,
    0x69, 0x0d, 0xe9, 0x8c, 0xe2, 0xc0, 0x10, 0x57, 0xb8, 0xc8, 0x76, 0x12,
    0x91, 0x55, 0xf2, 0x48, 0x69, 0xd8, 0xbc, 0x2a, 0x02, 0x5b, 0x0f, 0x44,
    0xd4, 0x20, 0x31, 0xdb, 0xf4, 0xba, 0x70, 0x26, 0x5d, 0x90, 0x60, 0x9e,
    0xbc, 0x4b, 0x17, 0x09, 0x2f, 0xb4, 0xcb, 0x1e, 0x43, 0x68, 0xc9, 0x07,
    0x27, 0xc1, 0xd2, 0x5c, 0xf7, 0xea, 0x21, 0xb9, 0x68, 0x12, 0x9c, 0x3c,
    0x9c, 0xbf, 0x9e, 0xfc, 0x80, 0x5c, 0x9b, 0x63, 0xcd, 0xec, 0x47, 0xaa,
    0x25, 0x27, 0x67, 0xa0, 0x37, 0xf3, 0x00, 0x82, 0x7d, 0x54, 0xd7, 0xa9,
    0xf8, 0xe9, 0x2e, 0x13, 0xa3, 0x77, 0xe8, 0x1f, 0x4a
};
```

II. Formatting the HTTP response

The code below is an example of how the response could be formatted. Please note that this code only prints 10 lines and then breaks the loop, since it is just showing the general scheme. Feel free to increase the number up to the length you want.

Code 11. Format the output of the HTTP response

```
void sendGetRequest(_i16 socketHandle, char* host, char* path)
{
    char outBuf[1024];
    char inBuf[1024];

    _i16 bytesSent = 0;
    _i16 bytesReceived = 0;

    sprintf(
        outBuf,
        "GET https://%s%s HTTP/1.1\r\nHost: %s\r\n\r\n", host, path, host
    );

    printf("HTTP request:\r\n");
    printf("\r\n======\r\n");
    printf("%s", outBuf);
    printf("\r\n======\r\n");

    bytesSent = sl_Send(
        socketHandle, (const void *) outBuf, strlen(outBuf), 0
    );

    if( bytesSent <= 0 ) {
        printf("sl_Send failed: %i\r\n", bytesSent);
        return;
    }

    printf("HTTP response:\r\n");
    printf("\r\n======\r\n");

    int linesPrinted = 0;

    do {
        bytesReceived = sl_Recv(socketHandle, inBuf, 1024, 0);

        int lastStart = 0;
        int pos = 0;

        while (linesPrinted < 10 && pos < bytesReceived) {
            if (inBuf[pos] == '\n') {
                printf("%.*s", pos - lastStart - 1, inBuf + lastStart);
                printf("\r\n");
                lastStart = pos;
                linesPrinted++;
            }

            pos++;
        }

    } while(bytesReceived > 0);
```

```
        if (linesPrinted == 10) {
            printf("...\r\n");
        }

        printf("\r\n======\r\n");
    }
```