SCM1612
Wi-Fi 6 and BLE 5 Low-Power SoC

HTTP Client Development Guide

Revision 0.1
Date 2024-09-23

Contact Information
Senscomm Semiconductor (www.senscomm.com)
Room 303, International Building, West 2 Suzhou Avenue,
SIP, Suzhou, China
For sales or technical support, please send email to
info@senscomm.com

Version History

Version	Date	Description
0.1	2024-9-23	Initial draft

Table of Contents

1 Overview

This document serves as a guide for implementing applications that require running an HTTP client on the SCM1612 platform.

The SCM1612 SDK utilizes the esp_http_client module from ESP-IDF:

- **API Location:** lib/net/esp_http_client
- **Demo Location:** api/examples/protocols/http_client

2 Demo Configuration and Build

To run the HTTP client demo, follow these steps:

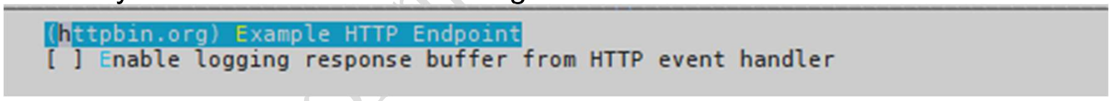
2.1 Set up Build Configuration

1. Select the HTTP client demo as a main application:
   $ make scm1612s_defconfig
   $ make menuconfig

2. Navigate to the following options in the menuconfig interface:
   - `Applications -> Protocols Demo`
   - Select `Protocols Demo -> HTTP Client Demo`
   - `Libraries/middleware -> net -> ESP HTTP Client`
   - Select `Enable HTTP Basic Authentication`
   - Select `Enable HTTP Digest Authentication`
3. Exit and save the configuration.

2.2 Set up Test HTTP Server.

1. Navigate to `Applications -> Example Configuration` in the menuconfig interface.
2. Modify the test HTTP server settings as needed.



```
(httpbin.org) Example HTTP Endpoint
[ ] Enable logging response buffer from HTTP event handler
```

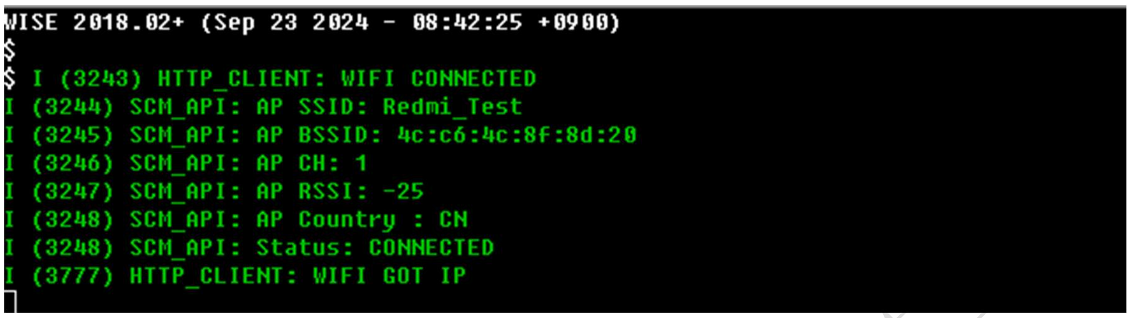
2.3 Set up Wi-Fi Parameters

1. Open the menuconfig interface:
   $ make menuconfig
2. Navigate to `Applications -> Common -> include WI-FI Configuration`
3. Enter the Wi-Fi parameters in `DEMO WI-FI Configuration`. Use the Help menu for each item if needed.
4. Exit and save the configuration.

2.4 Build wise-mcuboot.bin.

1. Build the project:
   $ make

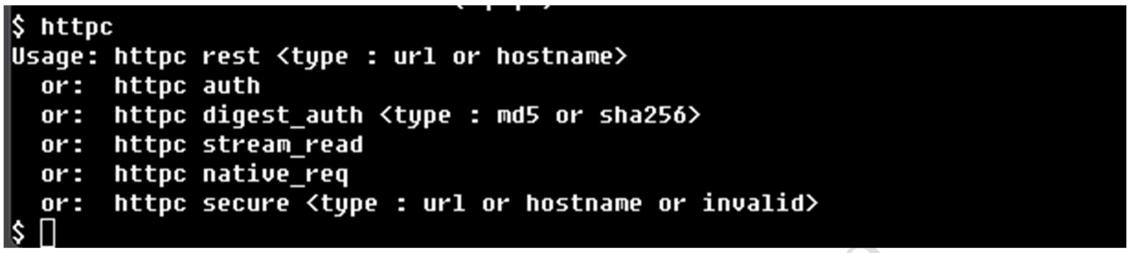
2. Refer to the `SDK_Getting_Started_Guide` to download the generated image and run it on an SCM1612 EVK.



```
WISE 2018.02+ (Sep 23 2024 - 08:42:25 +0900)
$
$ I (3243) HTTP_CLIENT: WIFI CONNECTED
I (3244) SCM_API: AP SSID: Redmi_Test
I (3245) SCM_API: AP BSSID: 4c:c6:4c:8f:8d:20
I (3246) SCM_API: AP CH: 1
I (3247) SCM_API: AP RSSI: -25
I (3248) SCM_API: AP Country : CN
I (3248) SCM_API: Status: CONNECTED
I (3777) HTTP_CLIENT: WIFI GOT IP
```

3 Using the HTTP Client

The HTTP client API provides functions for making various types of HTTP requests and handling authentication, streaming, and HTTPS connections.

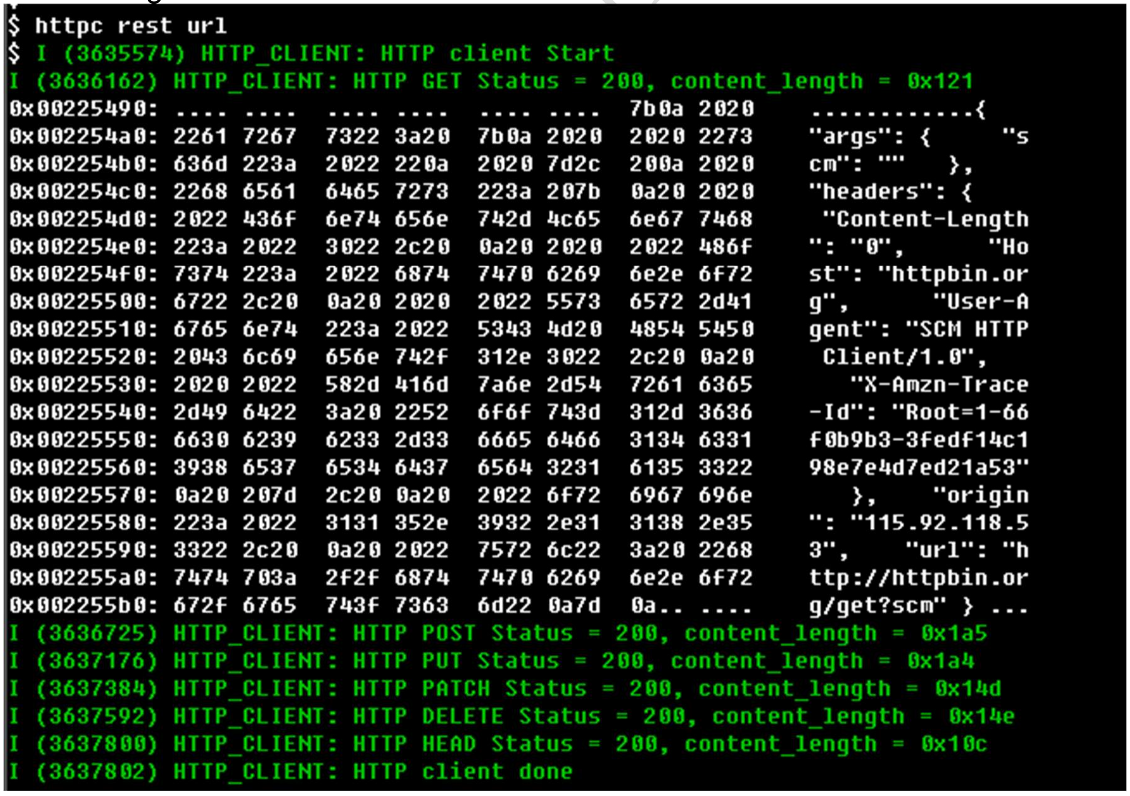


```
$ httpc
Usage: httpc rest <type : url or hostname>
  or:  httpc auth
  or:  httpc digest_auth <type : md5 or sha256>
  or:  httpc stream_read
  or:  httpc native_req
  or:  httpc secure <type : url or hostname or invalid>
$
```

3.1 Basic HTTP Request

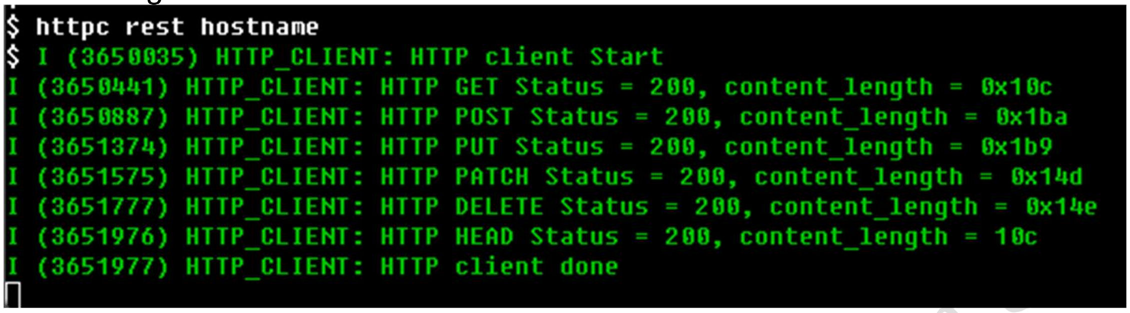
The esp_http_client API allows performing GET, POST, PUT, PATCH, and DELETE requests. Once a connection is established, you can make multiple requests before closing it. This section focuses on the common use cases of testing REST APIs with URLs or hostnames.

- Testing with URLs



```
$ httpc rest url
$ I (3635574) HTTP_CLIENT: HTTP client Start
I (3636162) HTTP_CLIENT: HTTP GET Status = 200, content_length = 0x121
0x00225490: .... ....  .... ....  .... ....  7b0a 2020    ............{
0x002254a0: 2261 7267  7322 3a20  7b0a 2020  2020 2273    "args": {     "s
0x002254b0: 636d 223a  2022 220a  2020 7d2c  200a 2020    cm": ""    },
0x002254c0: 2268 6561  6465 7273  223a 207b  0a20 2020    "headers": {
0x002254d0: 2022 436f  6e74 656e  742d 4c65  6e67 7468     "Content-Length
0x002254e0: 223a 2022  3022 2c20  0a20 2020  2022 486f    ": "0",       "Ho
0x002254f0: 7374 223a  2022 6874  7470 6269  6e2e 6f72    st": "httpbin.or
0x00225500: 6722 2c20  0a20 2020  2022 5573  6572 2d41    g",       "User-A
0x00225510: 6765 6e74  223a 2022  5343 4d20  4854 5450    gent": "SCM HTTP
0x00225520: 2043 6c69  656e 742f  312e 3022  2c20 0a20     Client/1.0",
0x00225530: 2020 2022  582d 416d  7a6e 2d54  7261 6365       "X-Amzn-Trace
0x00225540: 2d49 6422  3a20 2252  6f6f 743d  312d 3636    -Id": "Root=1-66
0x00225550: 6630 6239  6233 2d33  6665 6466  3134 6331    f0b9b3-3fedf14c1
0x00225560: 3938 6537  6534 6437  6564 3231  6135 3322    98e7e4d7ed21a53"
0x00225570: 0a20 207d  2c20 0a20  2022 6f72  6967 696e       },    "origin
0x00225580: 223a 2022  3131 352e  3932 2e31  3138 2e35    ": "115.92.118.5
0x00225590: 3322 2c20  0a20 2022  7572 6c22  3a20 2268    3",    "url": "h
0x002255a0: 7474 703a  2f2f 6874  7470 6269  6e2e 6f72    ttp://httpbin.or
0x002255b0: 672f 6765  743f 7363  6d22 0a7d  0a.. ....    g/get?scm" } ...
I (3636725) HTTP_CLIENT: HTTP POST Status = 200, content_length = 0x1a5
I (3637176) HTTP_CLIENT: HTTP PUT Status = 200, content_length = 0x1a4
I (3637384) HTTP_CLIENT: HTTP PATCH Status = 200, content_length = 0x14d
I (3637592) HTTP_CLIENT: HTTP DELETE Status = 200, content_length = 0x14e
I (3637800) HTTP_CLIENT: HTTP HEAD Status = 200, content_length = 0x10c
I (3637802) HTTP_CLIENT: HTTP client done
```

- Testing with Hostnames

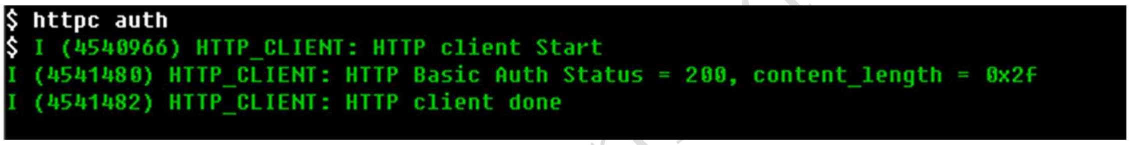


```
$ httpc rest hostname
$ I (3650035) HTTP_CLIENT: HTTP client Start
I (3650441) HTTP_CLIENT: HTTP GET Status = 200, content_length = 0x10c
I (3650887) HTTP_CLIENT: HTTP POST Status = 200, content_length = 0x1ba
I (3651374) HTTP_CLIENT: HTTP PUT Status = 200, content_length = 0x1b9
I (3651575) HTTP_CLIENT: HTTP PATCH Status = 200, content_length = 0x14d
I (3651777) HTTP_CLIENT: HTTP DELETE Status = 200, content_length = 0x14e
I (3651976) HTTP_CLIENT: HTTP HEAD Status = 200, content_length = 10c
I (3651977) HTTP_CLIENT: HTTP client done
```

3.2 HTTP Authentication

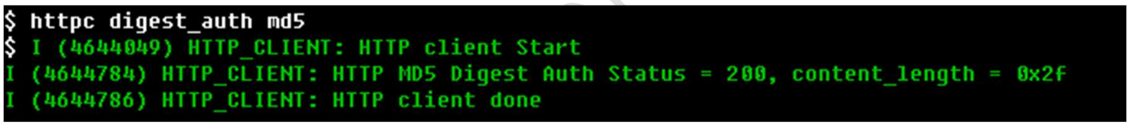
The HTTP client supports both Basic and Digest authentication. Digest authentication supports MD5 and SHA-256.

- Basic Authentication



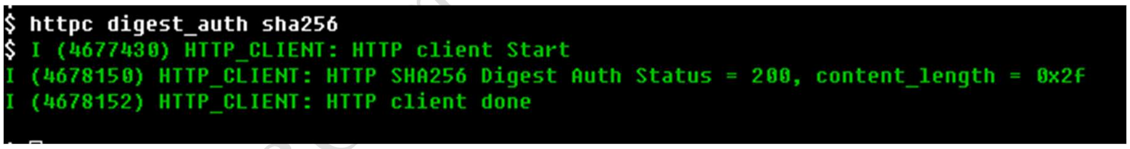
```
$ httpc auth
$ I (4540966) HTTP_CLIENT: HTTP client Start
I (4541480) HTTP_CLIENT: HTTP Basic Auth Status = 200, content_length = 0x2f
I (4541482) HTTP_CLIENT: HTTP client done
```

- MD5 Authentication



```
$ httpc digest_auth md5
$ I (4644049) HTTP_CLIENT: HTTP client Start
I (4644784) HTTP_CLIENT: HTTP MD5 Digest Auth Status = 200, content_length = 0x2f
I (4644786) HTTP_CLIENT: HTTP client done
```

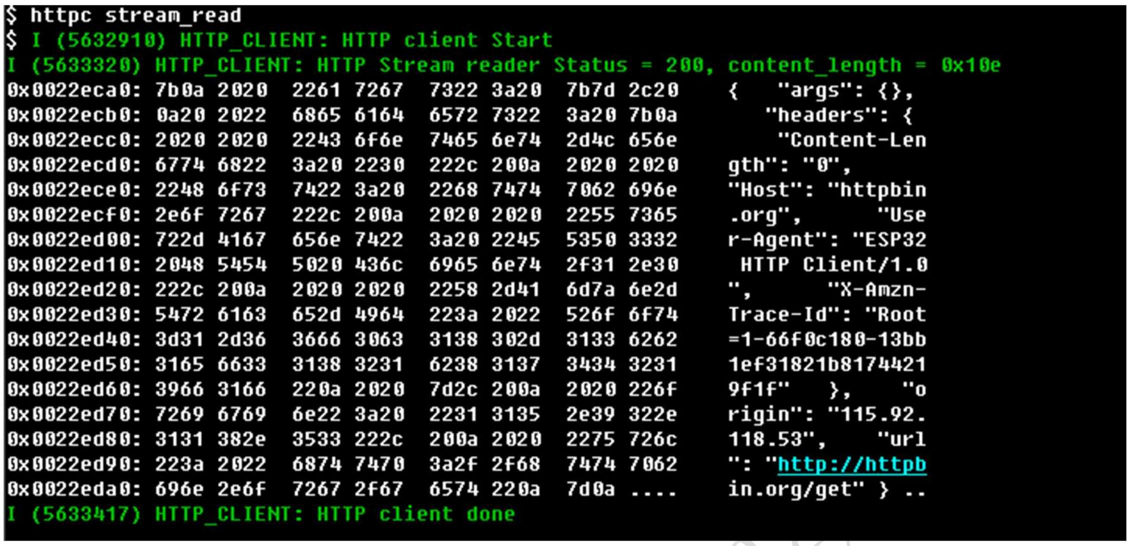
- SHA256 Authentication



```
$ httpc digest_auth sha256
$ I (4677430) HTTP_CLIENT: HTTP client Start
I (4678150) HTTP_CLIENT: HTTP SHA256 Digest Auth Status = 200, content_length = 0x2f
I (4678152) HTTP_CLIENT: HTTP client done
```

3.3 HTTP Streaming

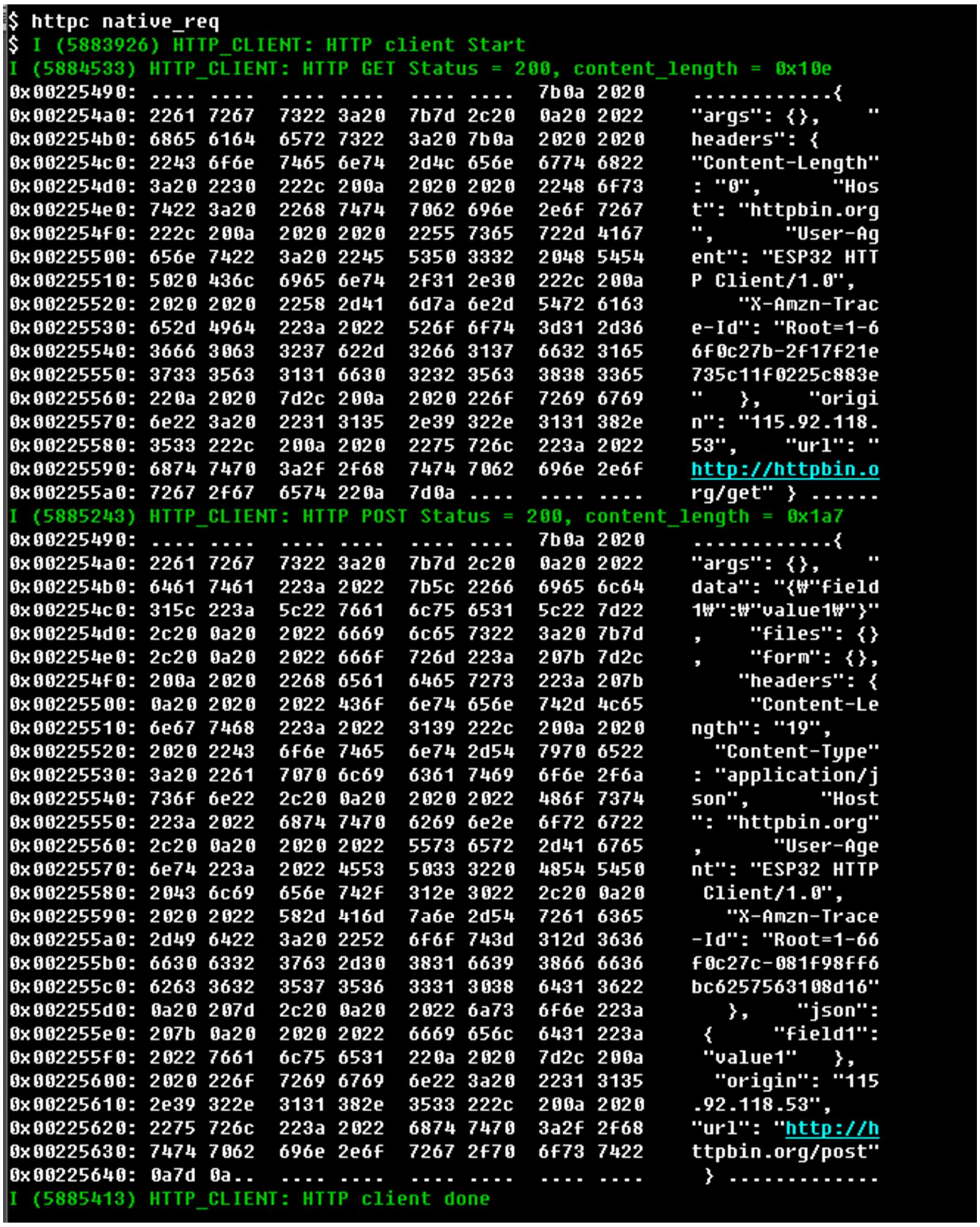
- For applications that require active control over data exchange (e.g., real-time data streams), you can use HTTP streaming. The application flow differs from typical requests.



```
$ httpc stream_read
$ I (5632910) HTTP_CLIENT: HTTP client Start
I (5633320) HTTP_CLIENT: HTTP Stream reader Status = 200, content_length = 0x10e
0x0022eca0: 7b0a 2020  2261 7267  7322 3a20  7b7d 2c20    {   "args": {},
0x0022ecb0: 0a20 2022  6865 6164  6572 7322  3a20 7b0a      "headers": {
0x0022ecc0: 2020 2020  2243 6f6e  7465 6e74  2d4c 656e       "Content-Len
0x0022ecd0: 6774 6822  3a20 2230  222c 200a  2020 2020    gth": "0",
0x0022ece0: 2248 6f73  7422 3a20  2268 7474  7062 696e    "Host": "httpbin
0x0022ecf0: 2e6f 7267  222c 200a  2020 2020  2255 7365    .org",      "Use
0x0022ed00: 722d 4167  656e 7422  3a20 2245  5350 3332    r-Agent": "ESP32
0x0022ed10: 2048 5454  5020 436c  6965 6e74  2f31 2e30     HTTP Client/1.0
0x0022ed20: 222c 200a  2020 2020  2258 2d41  6d7a 6e2d    ",      "X-Amzn-
0x0022ed30: 5472 6163  652d 4964  223a 2022  526f 6f74    Trace-Id": "Root
0x0022ed40: 3d31 2d36  3666 3063  3138 302d  3133 6262    =1-66f0c180-13bb
0x0022ed50: 3165 6633  3138 3231  6238 3137  3434 3231    1ef31821b8174421
0x0022ed60: 3966 3166  220a 2020  7d2c 200a  2020 226f    9f1f"   },    "o
0x0022ed70: 7269 6769  6e22 3a20  2231 3135  2e39 322e    rigin": "115.92.
0x0022ed80: 3131 382e  3533 222c  200a 2020  2275 726c    118.53",    "url
0x0022ed90: 223a 2022  6874 7470  3a2f 2f68  7474 7062    ": "http://httpb
0x0022eda0: 696e 2e6f  7267 2f67  6574 220a  7d0a ....    in.org/get" } ..
I (5633417) HTTP_CLIENT: HTTP client done
```

3.4 HTTP Native

The HTTP client provides low-level APIs for more fine-grained control over the HTTP connection.



```
$ httpc native_req
$ I (5883926) HTTP_CLIENT: HTTP client Start
I (5884533) HTTP_CLIENT: HTTP GET Status = 200, content_length = 0x10e
0x00225490: .... ....  .... ....  .... ....  7b0a 2020    ............{
0x002254a0: 2261 7267  7322 3a20  7b7d 2c20  0a20 2022    "args": {},    "
0x002254b0: 6865 6164  6572 7322  3a20 7b0a  2020 2020    headers": {
0x002254c0: 2243 6f6e  7465 6e74  2d4c 656e  6774 6822    "Content-Length"
0x002254d0: 3a20 2230  222c 200a  2020 2020  2248 6f73    : "0",       "Hos
0x002254e0: 7422 3a20  2268 7474  7062 696e  2e6f 7267    t": "httpbin.org
0x002254f0: 222c 200a  2020 2020  2255 7365  722d 4167    ",       "User-Ag
0x00225500: 656e 7422  3a20 2245  5350 3332  2048 5454    ent": "ESP32 HTT
0x00225510: 5020 436c  6965 6e74  2f31 2e30  222c 200a    P Client/1.0",
0x00225520: 2020 2020  2258 2d41  6d7a 6e2d  5472 6163        "X-Amzn-Trac
0x00225530: 652d 4964  223a 2022  526f 6f74  3d31 2d36    e-Id": "Root=1-6
0x00225540: 3666 3063  3237 622d  3266 3137  6632 3165    6f0c27b-2f17f21e
0x00225550: 3733 3563  3131 6630  3232 3563  3838 3365    735c11f0225c883e
0x00225560: 220a 2020  7d2c 200a  2020 226f  7269 6769    "   },    "origi
0x00225570: 6e22 3a20  2231 3135  2e39 322e  3131 382e    n": "115.92.118.
0x00225580: 3533 222c  200a 2020  2275 726c  223a 2022    53",    "url": "
0x00225590: 6874 7470  3a2f 2f68  7474 7062  696e 2e6f    http://httpbin.o
0x002255a0: 7267 2f67  6574 220a  7d0a ....  .... ....    rg/get" } ......
I (5885243) HTTP_CLIENT: HTTP POST Status = 200, content_length = 0x1a7
0x00225490: .... ....  .... ....  .... ....  7b0a 2020    ............{
0x002254a0: 2261 7267  7322 3a20  7b7d 2c20  0a20 2022    "args": {},    "
0x002254b0: 6461 7461  223a 2022  7b5c 2266  6965 6c64    data": "{₩"field
0x002254c0: 315c 223a  5c22 7661  6c75 6531  5c22 7d22    1₩":₩"value1₩"}"
0x002254d0: 2c20 0a20  2022 6669  6c65 7322  3a20 7b7d    ,    "files": {}
0x002254e0: 2c20 0a20  2022 666f  726d 223a  207b 7d2c    ,    "form": {},
0x002254f0: 200a 2020  2268 6561  6465 7273  223a 207b        "headers": {
0x00225500: 0a20 2020  2022 436f  6e74 656e  742d 4c65         "Content-Le
0x00225510: 6e67 7468  223a 2022  3139 222c  200a 2020    ngth": "19",
0x00225520: 2020 2243  6f6e 7465  6e74 2d54  7970 6522      "Content-Type"
0x00225530: 3a20 2261  7070 6c69  6361 7469  6f6e 2f6a    : "application/j
0x00225540: 736f 6e22  2c20 0a20  2020 2022  486f 7374    son",      "Host
0x00225550: 223a 2022  6874 7470  6269 6e2e  6f72 6722    ": "httpbin.org"
0x00225560: 2c20 0a20  2020 2022  5573 6572  2d41 6765    ,      "User-Age
0x00225570: 6e74 223a  2022 4553  5033 3220  4854 5450    nt": "ESP32 HTTP
0x00225580: 2043 6c69  656e 742f  312e 3022  2c20 0a20     Client/1.0",
0x00225590: 2020 2022  582d 416d  7a6e 2d54  7261 6365       "X-Amzn-Trace
0x002255a0: 2d49 6422  3a20 2252  6f6f 743d  312d 3636    -Id": "Root=1-66
0x002255b0: 6630 6332  3763 2d30  3831 6639  3866 6636    f0c27c-081f98ff6
0x002255c0: 6263 3632  3537 3536  3331 3038  6431 3622    bc6257563108d16"
0x002255d0: 0a20 207d  2c20 0a20  2022 6a73  6f6e 223a       },    "json":
0x002255e0: 207b 0a20  2020 2022  6669 656c  6431 223a     {      "field1":
0x002255f0: 2022 7661  6c75 6531  220a 2020  7d2c 200a     "value1"   },
0x00225600: 2020 226f  7269 6769  6e22 3a20  2231 3135      "origin": "115
0x00225610: 2e39 322e  3131 382e  3533 222c  200a 2020    .92.118.53",
0x00225620: 2275 726c  223a 2022  6874 7470  3a2f 2f68    "url": "http://h
0x00225630: 7474 7062  696e 2e6f  7267 2f70  6f73 7422    ttpbin.org/post"
0x00225640: 0a7d 0a..  .... ....  .... ....  .... ....     } ............
I (5885413) HTTP_CLIENT: HTTP client done
```

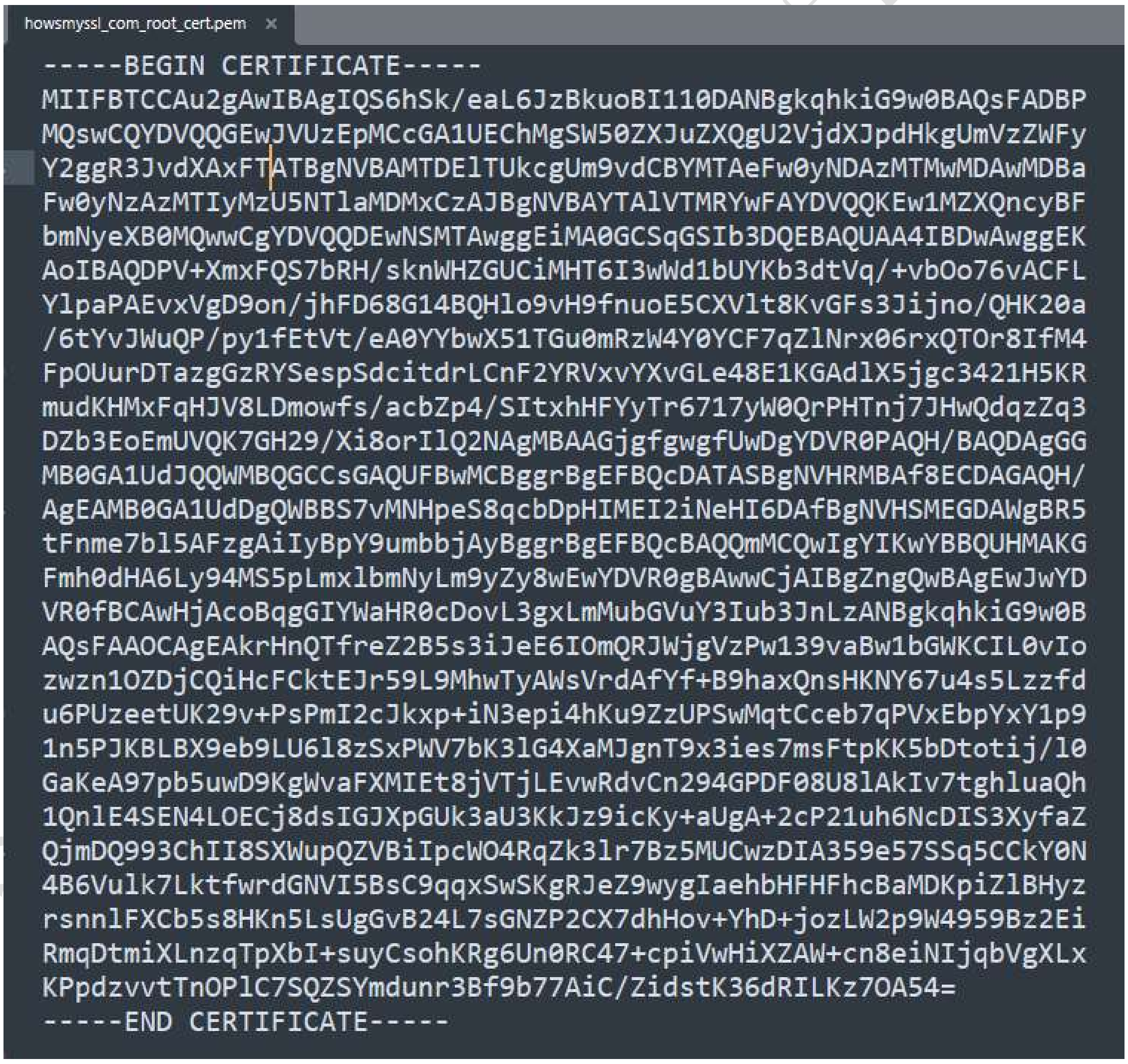
3.5 HTTPS Requests

The HTTP client supports SSL connections using mbed TLS. For demonstration, www.howsmyssl.com is a suitable test server.

3.5.1 Obtaining a Root CA Certificate

For HTTPS, a root CA certificate (PEM file) is required. This example demonstrates using openssl and save into the file howsmyssl_com_root_cert.pem:

```
openssl s_client -showcerts -connect www.howsmyssl.com:443 < /dev/null
```



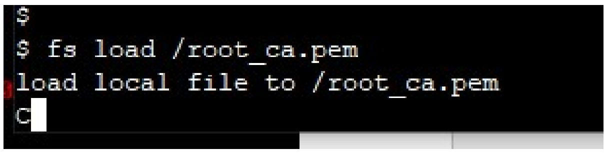
howsmyssl_com_root_cert.pem

```
-----BEGIN CERTIFICATE-----
MIIFBTCCAu2gAwIBAgIQS6hSk/eaL6JzBkuoBI110DANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJuZXQgU2VjdXJpdHkgUmVzZWFy
Y2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBYMTAeFw0yNDAzMTMwMDAwMDBa
Fw0yNzAzMTIyMzU5NTlaMDMxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBF
bmNyeXB0MQwwCgYDVQQDEwNSMTAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDPV+XmxFQS7bRH/sknWHZGUCiMHT6I3wWd1bUYKb3dtVq/+vbOo76vACFL
YlpaPAEvxVgD9on/jhFD68G14BQHlo9vH9fnuoE5CXVlt8KvGFs3Jijno/QHK20a
/6tYvJWuQP/py1fEtVt/eA0YYbwX51TGu0mRzW4Y0YCF7qZlNrx06rxQTOr8IfM4
FpOUurDTazgGzRYSespSdcitdrLCnF2YRVxvYXvGLe48E1KGAdlX5jgc3421H5KR
mudKHMxFqHJV8LDmowfs/acbZp4/SItxhHFYyTr6717yW0QrPHTnj7JHwQdqzZq3
DZb3EoEmUVQK7GH29/Xi8orIlQ2NAgMBAAGjgfgwgfUwDgYDVR0PAQH/BAQDAgGG
MB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDATASBgNVHRMBAf8ECDAGAQH/
AgEAMB0GA1UdDgQWBBS7vMNHpeS8qcbDpHIMEI2iNeHI6DAfBgNVHSMEGDAWgBR5
tFnme7bl5AFzgAiIyBpY9umbbjAyBggrBgEFBQcBAQQmMCQwIgYIKwYBBQUHMAKG
Fmh0dHA6Ly94MS5pLmxlbmNyLm9yZy8wEwYDVR0gBAwwCjAIBgZngQwBAgEwJwYD
VR0fBCAwHjAcoBqgGIYWaHR0cDovL3gxLmMubGVuY3Iub3JnLzANBgkqhkiG9w0B
AQsFAAOCAgEAkrHnQTfreZ2B5s3iJeE6IOmQRJWjgVzPw139vaBw1bGWKCIL0vIo
zwzn1OZDjCQiHcFCktEJr59L9MhwTyAWsVrdAfYf+B9haxQnsHKNY67u4s5Lzzfd
u6PUzeetUK29v+PsPmI2cJkxp+iN3epi4hKu9ZzUPSwMqtCceb7qPVxEbpYxY1p9
1n5PJKBLBX9eb9LU6l8zSxPWV7bK3lG4XaMJgnT9x3ies7msFtpKK5bDtotij/l0
GaKeA97pb5uwD9KgWvaFXMIEt8jVTjLEvwRdvCn294GPDF08U8lAkIv7tghluaQh
1QnlE4SEN4LOECj8dsIGJXpGUk3aU3KkJz9icKy+aUgA+2cP21uh6NcDIS3XyfaZ
QjmDQ993ChII8SXWupQZVBiIpcWO4RqZk3lr7Bz5MUCwzDIA359e57SSq5CCkY0N
4B6Vulk7LktfwrdGNVI5BsC9qqxSwSKgRJeZ9wygIaehbHFHFhcBaMDKpiZlBHyz
rsnnlFXCb5s8HKn5LsUgGvB24L7sGNZP2CX7dhHov+YhD+jozLW2p9W4959Bz2Ei
RmqDtmiXLnzqTpXbI+suyCsohKRg6Un0RC47+cpiVwHiXZAW+cn8eiNIjqbVgXLx
KPpdzvvtTnOPlC7SQZSYmdunr3Bf9b77AiC/ZidstK36dRILKz7OA54=
-----END CERTIFICATE-----
```

3.5.2 Upload the Root CA Certificate File

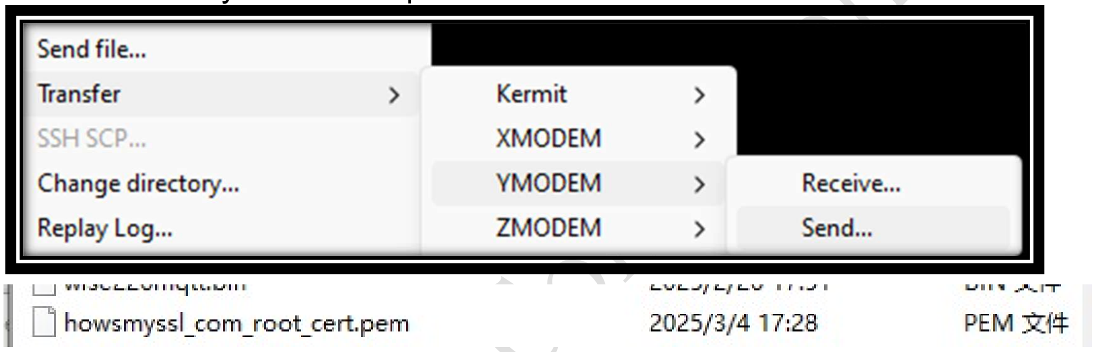
For the demo, the `fs load` command is used to upload certificate files. Follow the steps below to upload a file into WISE for demo purposes:

Use the fs load command and specify the file name under which you want to save the uploaded file

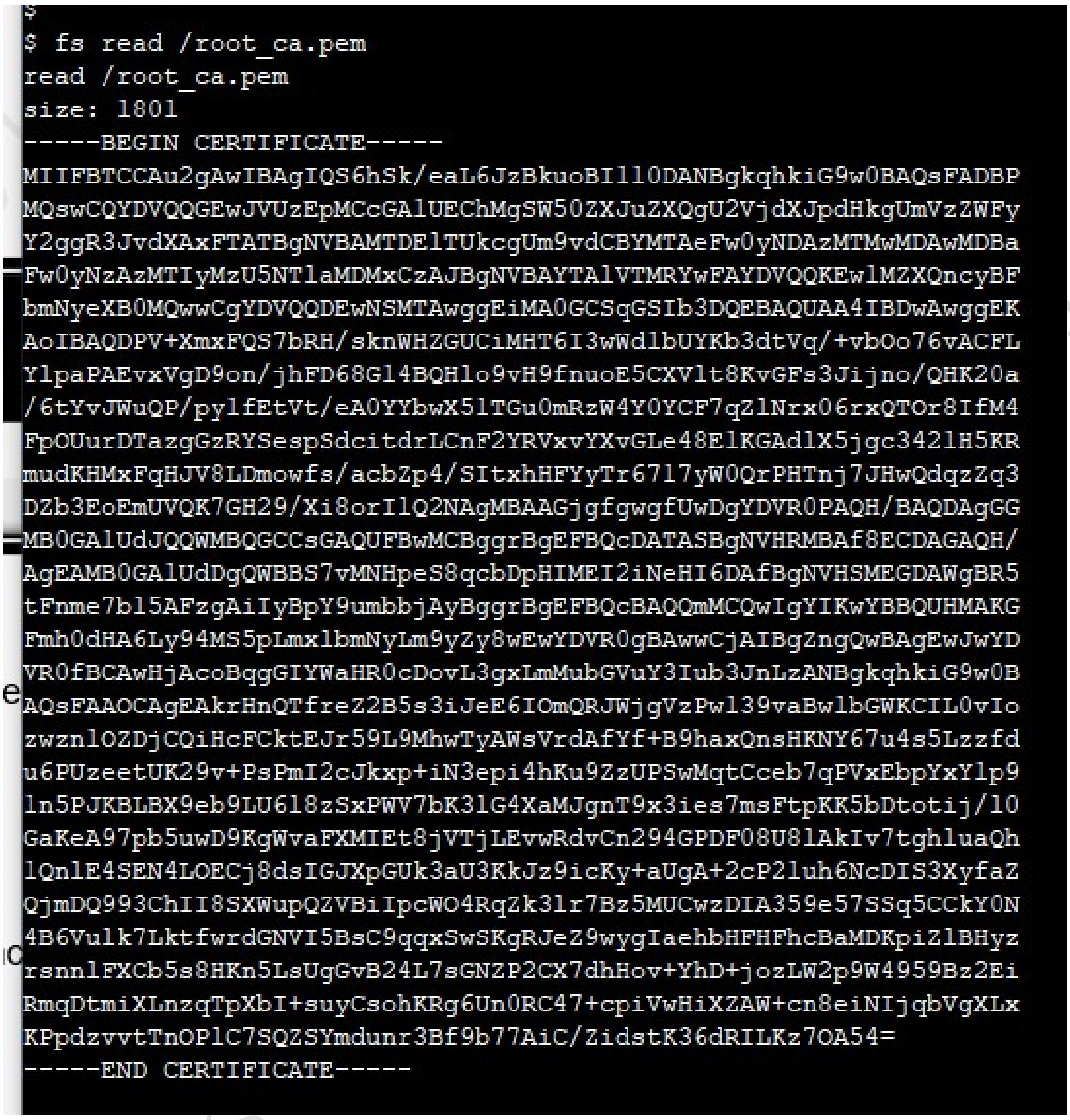


```
$
$ fs load /root_ca.pem
load local file to /root_ca.pem
C
```

Choose the file you want to upload from YMODEM.



Use the `fs read` command along with the file name to read the content of the specific file.

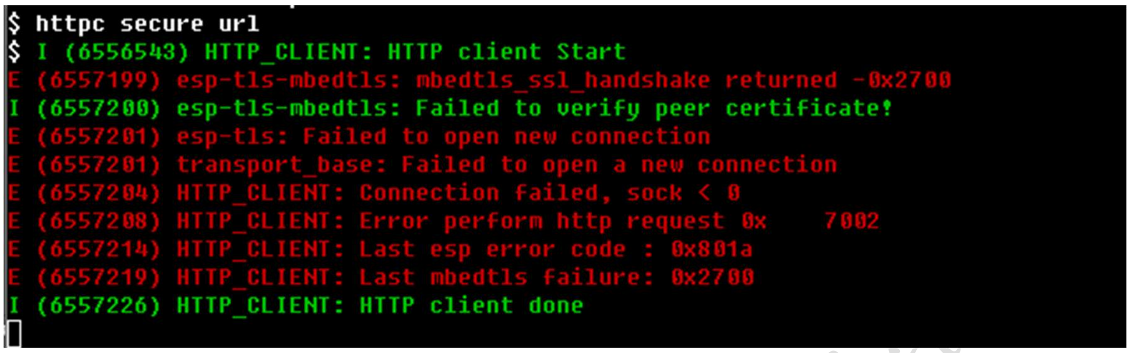


```
$
$ fs read /root_ca.pem
read /root_ca.pem
size: 1801
-----BEGIN CERTIFICATE-----
MIIFBTCCAu2gAwIBAgIQS6hSk/eaL6JzBkuoBIl10DANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJuZXQgU2VjdXJpdHkgUmVzZWFy
Y2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBYMTAeFw0yNDAzMTMwMDAwMDBa
Fw0yNzAzMTIyMzU5NTlaMDMxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBF
bmNyeXB0MQwwCgYDVQQDEwNSMTAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDPV+XmxFQS7bRH/sknWHZGUCiMHT6I3wWd1bUYKb3dtVq/+vbOo76vACFL
YlpaPAEvxVgD9on/jhFD68G14BQHlo9vH9fnuoE5CXVlt8KvGFs3Jijno/QHK20a
/6tYvJWuQP/py1fEtVt/eA0YYbwX51TGu0mRzW4Y0YCF7qZlNrx06rxQTOr8IfM4
FpOUurDTazgGzRYSespSdcitdrLCnF2YRVxvYXvGLe48E1KGAdlX5jgc3421H5KR
mudKHMxFqHJV8LDmowfs/acbZp4/SItxhHFYyTr6717yW0QrPHTnj7JHwQdqzZq3
DZb3EoEmUVQK7GH29/Xi8orIlQ2NAgMBAAGjgfgwgfUwDgYDVR0PAQH/BAQDAgGG
MB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDATASBgNVHRMBAf8ECDAGAQH/
AgEAMB0GA1UdDgQWBBS7vMNHpeS8qcbDpHIMEI2iNeHI6DAfBgNVHSMEGDAWgBR5
tFnme7bl5AFzgAiIyBpY9umbbjAyBggrBgEFBQcBAQQmMCQwIgYIKwYBBQUHMAKG
Fmh0dHA6Ly94MS5pLmxlbmNyLm9yZy8wEwYDVR0gBAwwCjAIBgZngQwBAgEwJwYD
VR0fBCAwHjAcoBqgGIYWaHR0cDovL3gxLmMubGVuY3Iub3JnLzANBgkqhkiG9w0B
AQsFAAOCAgEAkrHnQTfreZ2B5s3iJeE6IOmQRJWjgVzPw139vaBw1bGWKCIL0vIo
zwzn1OZDjCQiHcFCktEJr59L9MhwTyAWsVrdAfYf+B9haxQnsHKNY67u4s5Lzzfd
u6PUzeetUK29v+PsPmI2cJkxp+iN3epi4hKu9ZzUPSwMqtCceb7qPVxEbpYxY1p9
1n5PJKBLBX9eb9LU6l8zSxPWV7bK3lG4XaMJgnT9x3ies7msFtpKK5bDtotij/l0
GaKeA97pb5uwD9KgWvaFXMIEt8jVTjLEvwRdvCn294GPDF08U8lAkIv7tghluaQh
1QnlE4SEN4LOECj8dsIGJXpGUk3aU3KkJz9icKy+aUgA+2cP21uh6NcDIS3XyfaZ
QjmDQ993ChII8SXWupQZVBiIpcWO4RqZk3lr7Bz5MUCwzDIA359e57SSq5CCkY0N
4B6Vulk7LktfwrdGNVI5BsC9qqxSwSKgRJeZ9wygIaehbHFHFhcBaMDKpiZlBHyz
rsnnlFXCb5s8HKn5LsUgGvB24L7sGNZP2CX7dhHov+YhD+jozLW2p9W4959Bz2Ei
RmqDtmiXLnzqTpXbI+suyCsohKRg6Un0RC47+cpiVwHiXZAW+cn8eiNIjqbVgXLx
KPpdzvvtTnOPlC7SQZSYmdunr3Bf9b77AiC/ZidstK36dRILKz7OA54=
-----END CERTIFICATE-----
```

3.5.3 Time Synchronization for HTTPS

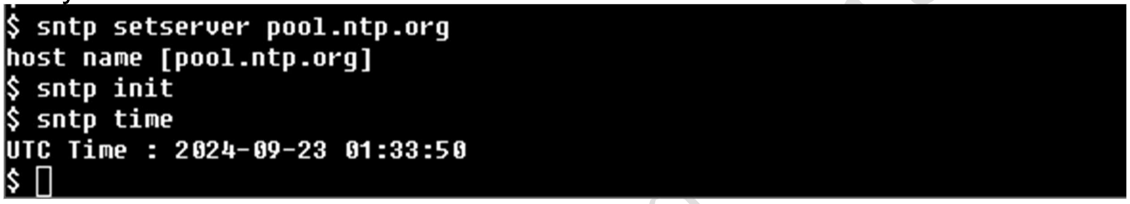
For HTTPS authentication verification, the device's time must be synchronized. The SCM1612 SDK supports STNP for time synchronization.

- If the time is not synchronized



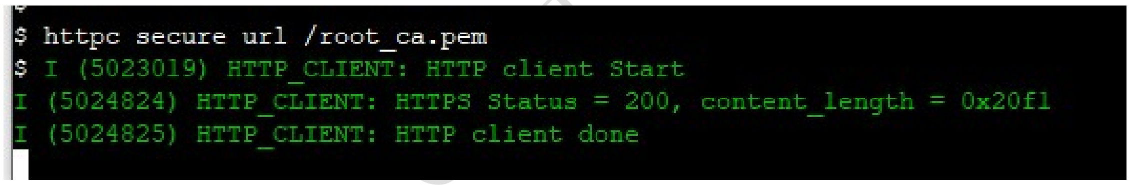
```
$ httpc secure url
$ I (6556543) HTTP_CLIENT: HTTP client Start
E (6557199) esp-tls-mbedtls: mbedtls_ssl_handshake returned -0x2700
I (6557200) esp-tls-mbedtls: Failed to verify peer certificate!
E (6557201) esp-tls: Failed to open new connection
E (6557201) transport_base: Failed to open a new connection
E (6557204) HTTP_CLIENT: Connection failed, sock < 0
E (6557208) HTTP_CLIENT: Error perform http request 0x    7002
E (6557214) HTTP_CLIENT: Last esp error code : 0x801a
E (6557219) HTTP_CLIENT: Last mbedtls failure: 0x2700
I (6557226) HTTP_CLIENT: HTTP client done
```

- Time synchronized



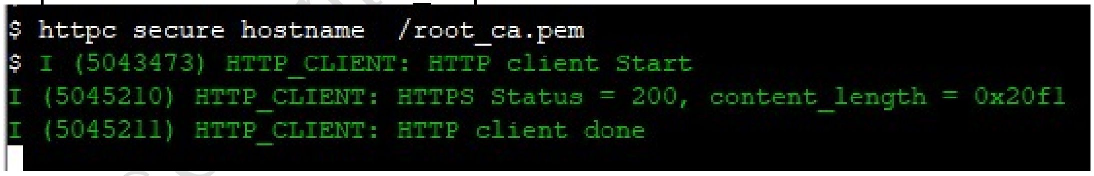
```
$ sntp setserver pool.ntp.org
host name [pool.ntp.org]
$ sntp init
$ sntp time
UTC Time : 2024-09-23 01:33:50
$ 
```

- After time synchronized, HTTPS request.
  httpc secure url /root_ca.pem



```
$ httpc secure url /root_ca.pem
$ I (5023019) HTTP_CLIENT: HTTP client Start
I (5024824) HTTP_CLIENT: HTTPS Status = 200, content_length = 0x20f1
I (5024825) HTTP_CLIENT: HTTP client done
```

httpc secure hostname /root_ca.pem



```
$ httpc secure hostname  /root_ca.pem
$ I (5043473) HTTP_CLIENT: HTTP client Start
I (5045210) HTTP_CLIENT: HTTPS Status = 200, content_length = 0x20f1
I (5045211) HTTP_CLIENT: HTTP client done
```