SCM1612
Wi-Fi 6 和 BLE 5 低功耗 SoC

HTTP 客户端 开发指南

文档版本 0.1
发布日期 2024-09-23

Senscomm Confidential

联系方式
速通半导体科技有限公司 (www.senscomm.com)
江苏省苏州市工业园区苏州大道西 2 号国际大厦 303 室
销售或技术支持，请发送电子邮件至
support@senscomm.com

版权所有

免责声明和注意事项

本文档仅按"现状"提供。速通半导体有限公司保留在无需另行通知的情况下对其或本文档中包含的任何规格进行更正、改进和其他变更的权利。

与使用本文档中的信息有关的一切责任，包括侵犯任何专有权利的责任，均不予承认。此处不授予任何明示或暗示、通过禁止或其他方式对任何知识产权的许可。

本文档中的所有第三方信息均按"现状"提供，不对其真实性和准确性提供任何保证。

本文档中提及的所有商标、商号和注册商标均为其各自所有者的财产，特此确认。

© 2024 速通半导体有限公司. 保留所有权利.

Senscomm Confidential

版本历史

版本	日期	描述
0.1	2024-09-23	初稿

Senscomm Confidential

目录

版本历史.....3
1 概述.....5
2 Demo 配置与构建.....6
2.1 设置构建配置.....6
2.2 设置测试 HTTP 服务器.....6
2.3 设置 Wi-Fi 参数.....6
2.4 构建 wise-mcuboot.bin.....7
2.5 Using the HTTP Client.....错误!未定义书签。
2.6 Using the HTTP Client.....错误!未定义书签。
3 使用 HTTP 客户端.....8
3.1 基本 HTTP 请求.....8
3.2 HTTP 身份认证.....9
3.3 HTTP 数据流传输.....10
3.4 HTTP 原生接口.....10
3.5 HTTPS 请求.....12
3.5.1 获取根 CA 证书.....12
3.5.2 HTTPS 时间同步.....14

Senscomm Confidential

1 概述

本文档旨在指导如何在 SCM1612 平台上实现运行 HTTP 客户端的应用程序。

SCM1612 SDK 使用了来自 ESP-IDF 的 esp_http_client 模块：

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

Senscomm Confidential

2 Demo 配置与构建

要运行 HTTP 客户端 Demo，请按照以下步骤进行操作：

2.1 设置构建配置

1. 选择 HTTP 客户端 Demo 作为主应用程序：
   $ make scm1612s_defconfig
   $ make menuconfig

2. 在 menuconfig 界面中，导航到以下选项：
   - `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. 退出并保存配置

2.2 设置测试 HTTP 服务器

1. 在 menuconfig 界面中导航至 Applications -> Example Configuration。
2. 根据需要修改测试 HTTP 服务器的设置。

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

2.3 设置 Wi-Fi 参数

1. 打开 menuconfig 界面：
   $ make menuconfig
2. 导航至 Applications -> Common -> include WI-FI Configuration。
3. 在 DEMO WI-FI Configuration 中输入 Wi-Fi 参数。如果需要，可以使用帮助菜单查看各项的详细说明。
4. 退出并保存配置。

Sensomm Confidential

2.4 构建 wise-mcuboot.bin

1. 构建项目：
   $ make
2. 请参考《SCM1612_SDK 入门指南.pdf》，将生成的镜像下载至 SCM1612 EVK，并运行它。

Senscomm Confi

3 使用 HTTP 客户端

HTTP 客户端 API 提供了多种类型的 HTTP 请求功能，并支持身份认证、数据流传输和 HTTPS 连接。

```
$ 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 基本 HTTP 请求

esp_http_client API 支持执行 GET、POST、PUT、PATCH 和 DELETE 请求。一旦建立连接后，可以在关闭连接前发起多次请求。本节重点介绍使用 URL 或主机名测试 REST API 的常见用例。

- 使用 URL 进行测试

Senscomm Confiden

```
$ 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
```

- 使用主机名进行测试

```
$ 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 身份认证

HTTP 客户端支持 Basic 和 Digest 两种身份认证方式。Digest 身份认证支持 MD5 和 SHA-256 算法。

- Basic 认证

```
$ 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 认证

```
$ 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 认证

```
$ 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 数据流传输

- 对于需要主动控制数据交换的应用场景（如实时数据流传输），可以使用 HTTP 流模式。应用的执行流程与常规请求有所不同。

```
$ 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 原生接口

HTTP 客户端提供了底层 API，可实现对 HTTP 连接的精细化控制。

版权所有

```
$ 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 请求

HTTP 客户端支持使用 mbed TLS 的 SSL 连接。推荐使用 www.howsmyssl.com 作为测试服务器进行演示。

3.5.1 获取根 CA 证书

对于 HTTPS 请求，需提供根 CA 证书（PEM 文件）。以下示例演示如何使用 openssl 获取根 CA 证书并保存到文件 howsmyssl_com_root_cert.pem：

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

howsmyssl_com_root_cert.pem

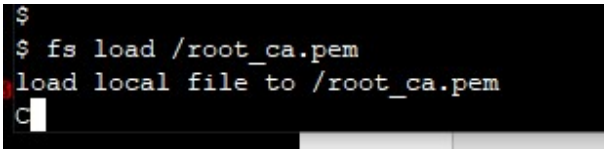
```
-----BEGIN CERTIFICATE-----
MIIFBTCCAu2gAwIBAgIQS6hSk/eaL6JzBkuoBI110DANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJuZXQgU2VjdXJpdHkgUmVzZWFy
Y2ggR3JvdXAxFTATBgNVBAMTDE1TUkcgUm9vdCBYMTAeFw0yNDAzMTMwMDAwMDBa
Fw0yNzAzMTIyMzU5NTlaMDMxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBF
bmNyeXB0MQwwCgYDVQQDEwNSMTAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
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-----
```


版权所有

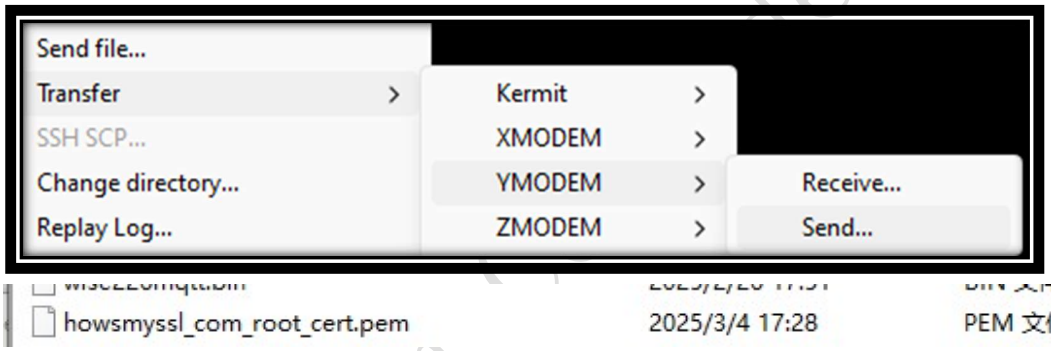
3.5.2 上传根 CA 证书文件

在此演示中，使用 fs load 命令上传证书文件。请按照以下步骤将文件上传到 WISE 进行演示：

使用 fs load 命令，并指定要保存上传文件的文件名。



从 YMODEM 选择您要上传的文件。



使用 fs read 命令并指定文件名，以读取该文件的内容。

Senscomm Confidential

```
$
$ fs read /root_ca.pem
read /root_ca.pem
size: 1801
-----BEGIN CERTIFICATE-----
MIIFBTCCAu2gAwIBAgIQS6hSk/eaL6JzBkuoBIll0DANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJuZXQgU2VjdXJpdHkgUmVzZWFy
Y2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBYMTAeFw0yNDAzMTMwMDAwMDBa
Fw0yNzAzMTIyMzU5NTlaMDMxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBF
bmNyeXB0MQwwCgYDVQQDEwNSMTAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDPV+XmxFQS7bRH/sknWHZGUCiMHT6I3wWdlbUYKb3dtVq/+vbOo76vACFL
YlpaPAEvxVgD9on/jhFD68G14BQHlo9vH9fnuoE5CXVlt8KvGFs3Jijno/QHK20a
/6tYvJWuQP/py1fEtVt/eA0YYbwX51TGu0mRzW4Y0YCF7qZlNrx06rxQTOr8IfM4
FpOUurDTazgGzRYSespSdcitdrLCnF2YRVxvYXvGLe48ElKGAdlX5jgc342lH5KR
mudKHMxFqHJV8LDmowfs/acbZp4/SItxhHFYyTr6717yW0QrPHTnj7JHwQdqzZq3
DZb3EoEmUVQK7GH29/Xi8orIlQ2NAgMBAAGjgfgwgfUwDgYDVR0PAQH/BAQDAgGG
MB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcDATASBgNVHRMBAf8ECDAGAQH/
AgEAMB0GA1UdDgQWBBS7vMNHpeS8qcbDpHIMEI2iNeHI6DAfBgNVHSMEGDAWgBR5
tFnme7bl5AFzgAiIyBpY9umbbjAyBggrBgEFBQcBAQQmMCQwIgYIKwYBBQUHMAKG
Fmh0dHA6Ly94MS5pLmxlbmNyLm9yZy8wEwYDVR0gBAwwCjAIBgZngQwBAgEwJwYD
VR0fBCAwHjAcoBqgGIYWaHR0cDovL3gxLmMubGVuY3Iub3JnLzANBgkqhkiG9w0B
AQsFAAOCAgEAkrHnQTfreZ2B5s3iJeE6IOmQRJWjgVzPwl39vaBwlbGWKCIL0vIo
zwznlOZDjCQiHcFCktEJr59L9MhwTyAWsVrdAfYf+B9haxQnsHKNY67u4s5Lzzfd
u6PUzeetUK29v+PsPmI2cJkxp+iN3epi4hKu9ZzUPSwMqtCceb7qPVxEbpYxYlp9
ln5PJKBLBX9eb9LU6l8zSxPWV7bK3lG4XaMJgnT9x3ies7msFtpKK5bDtotij/l0
GaKeA97pb5uwD9KgWvaFXMIEt8jVTjLEvwRdvCn294GPDF08U8lAkIv7tghluaQh
lQnlE4SEN4LOECj8dsIGJXpGUk3aU3KkJz9icKy+aUgA+2cP2luh6NcDIS3XyfaZ
QjmDQ993ChII8SXWupQZVBiIpcWO4RqZk3lr7Bz5MUCwzDIA359e57SSq5CCkY0N
4B6Vulk7LktfwrdGNVI5BsC9qqxSwSKgRJeZ9wygIaehbHFHFhcBaMDKpiZlBHyz
rsnnlFXCb5s8HKn5LsUgGvB24L7sGNZP2CX7dhHov+YhD+jozLW2p9W4959Bz2Ei
RmqDtmiXLnzqTpXbI+suyCsohKRg6Un0RC47+cpiVwHiXZAW+cn8eiNIjqbVgXLx
KPpdzvvtTnOPlC7SQZSYmdunr3Bf9b77AiC/ZidstK36dRILKz7OA54=
-----END CERTIFICATE-----
```

3.5.3 HTTPS 时间同步

为了进行 HTTPS 身份验证，设备时间必须同步。SCM1612 SDK 支持使用 STNP 进行时间同步。

- 如果时间未同步

版权所有

```
$ 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
```

- 时间同步完成

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

- 时间同步后，进行 HTTPS 请求

  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

Senscomm Confidential