QuecPython
HTTP 应用指导

LTE Standard 模块系列

版本：1.0.0

日期：2020-11-09

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，不得将其用于除本项目的实施与开展以外的任何其他目的。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，上海移远通信技术股份有限公司有权追究法律责任。

版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2020-11-09	Kingka/Kenney	文档创建
1.0.0	2020-11-09	Kingka/Kenney	临时版本

目录

图片索引

1 引言

HTTP 协议是用于客户端和服务器端通信的，主要通过客户端请求和服务器响应的切换完成通信过程。本文档主要介绍如何使用 QuecPython 类库 API 快速进行 HTTP 协议项目开发。

适用模块：

- EC100Y-CN（本文以该模块为例进行介绍）
- EC600S-CN

2 HTTP 协议基础

2.1. HTTP 协议

HTTP 协议是用于从万维网服务器传输超文本到本地浏览器的传送协议。基于 TCP 的应用层协议，它不关心数据传输的细节，HTTP（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，只有遵循统一的 HTTP 请求格式，服务器才能正确解析不同客户端发的请求，同样地，服务器遵循统一的响应格式，客户端才得以正确解析不同网站发过来的响应。

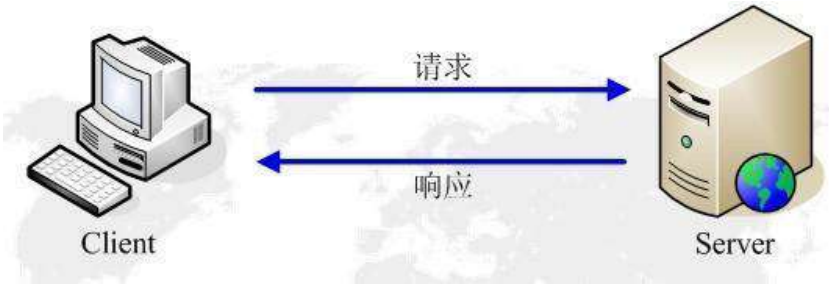

图 1：Client 与 Server 通信

2.2. HTTP 请求

HTTP 请求由请求行、请求头、空行、请求体组成。

图 2：HTTP 请求构成

2.2.1. 请求行

请求行由请求方式 + URL+ 协议版本组成。

- 常见的请求方法有 GET、POST、PUT、DELETE、HEAD；
- URL：客户端要获取的资源路径；
- 协议版本：客户端使用的 HTTP 协议版本号（目前使用的是 http1.1）。

2.2.2. 请求头

请求头是客户端向服务器发送请求的补充说明。

- host：请求地址；
- User-Agent：客户端使用的操作系统和浏览器的名称和版本；
- Content-Length：发送给 HTTP 服务器数据的长度；
- Content-Type：参数的数据类型；
- Cookie：将 cookie 的值发送给 HTTP 服务器；
- Accept-Charset：浏览器可接受的字符集；
- Accept-Language：浏览器可接受的语言；
- Accept：浏览器可接受的媒体类型。

2.2.3. 请求体

请求体携带请求参数。

- application/json：{"name":"value","name1":"value2”}；
- application/x-www-form-urlencoded： name1=value1&name2=value2；
- multipart/from-data：表格形式；
- text/xml；
- content-type：octets/stream。

2.3. HTTP 响应

HTTP 响应由状态行、响应头、空行、响应体组成。

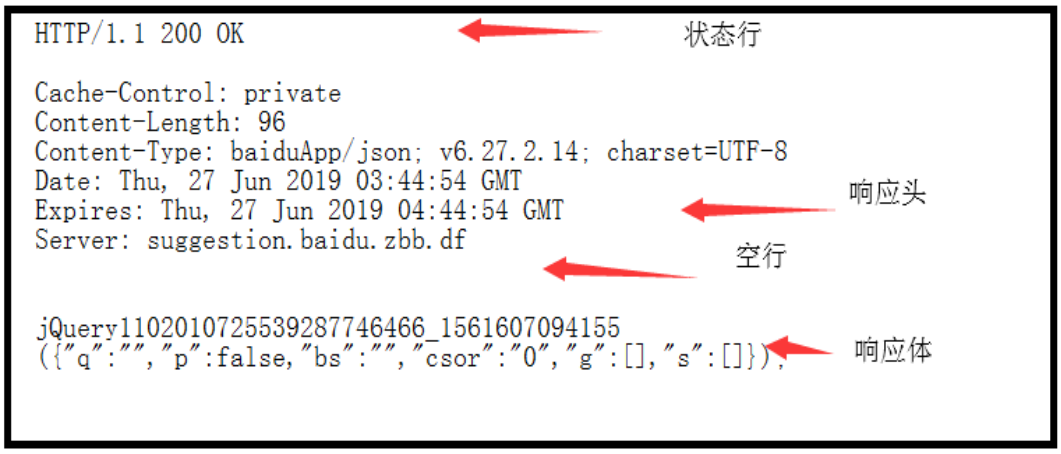

图 3：HTTP 响应构成

2.3.1. 状态行

状态行由 HTTP 版本号 + 响应状态码 + 状态说明组成。

响应状态码有 1XX、2XX、3XX、4XX、5XX。

- 1XX：提示信息 - 表示请求已被成功接收，继续处理；
- 2XX：成功 - 表示请求已被成功接收，理解，接受；
- 3XX：重定向 - 要完成请求必须进行更进一步的处理；
- 4XX：客户端错误 - 请求有语法错误或请求无法实现；
- 5XX：服务器端错误 - 服务器未能实现合法的请求响应头。

2.3.2. 响应头

响应头与请求头对应，是服务器对该响应的一些附加说明。

2.3.3. 响应体

为真正的响应数据，即为网页的 HTML 源代码。

2.4. URL

URL 是 WWW 的统一资源定位标志，就是指网络地址。

URL 格式：https://host:port/path?xxx=aaa&ooo=bbb

其中：

- http/https：这个是协议类型
- host：服务器的 IP 地址或者域名
- port：HTTP 服务器的端口，默认端口是 80
- path：访问资源的路径
- url 里面的？这个符号是个分割线，用来区分问号前面的是 path，问号后面的是参数
- url-params：问号后面的是请求参数，格式：xxx=aaa。多个参数用&符号连接

2.5. HTTP 协议请求方法

HTTP1.0 定义了三种请求方法：GET、POST 和 HEAD 方法。*HTTP1.1* 新增了五种请求方法：OPTIONS、PUT、DELETE、TRACE 和 CONNECT 方法。

- GET：请求指定的页面信息，并返回实体主体。
- POST：向指定资源提交数据进行处理请求，数据被包含在请求体中。
- HEAD：返回的响应中没有具体的内容，用于获取报头。
- OPTIONS：返回服务器针对特定资源所支持的 HTTP 请求方法，也可以利用向 web 服务器发送‘*’的请求来测试服务器的功能性
- PUT：向指定资源位置上传其最新内容
- DELETE：请求服务器删除 Request-URL 所标识的资源
- TRACE：回显服务器收到的请求，主要用于测试或诊断
- CONNECT：*HTTP1.1* 协议中预留给能够将连接改为管道方式的代理服务器。

3 HTTP 接口

3.1. request.get

该函数用于发送 GET 请求。

- **函数原型**

```
request.get(url, data, json, headers)
```

- **参数**

url：
网址，字符串类型。

data：
（可选参数）附加到请求的正文，json 字典类型，默认为 None。

json：
（可选参数）json 格式用于附加到请求的主体，默认为 None。

headers：
（可选参数）请求头，默认为 None。

- **返回值**

返回请求对象

3.2. request.post

该函数用于发送 POST 请求。

- **函数原型**

```
request.post(url, data, json, headers)
```

- **参数**

url：
网址，字符串类型。

data：
（可选参数）附加到请求的正文，json 字典类型，默认为 None。

json：
（可选参数）json 格式用于附加到请求的主体，默认为 None。

headers：
（可选参数）请求头，默认为 None。

- **返回值**

返回请求对象

3.3. request.put

该函数用于发送 PUT 请求。

- **函数原型**

```
request.put(url, data, json, headers)
```

- **参数**

url：
网址，字符串类型。

data：
（可选参数）附加到请求的正文，json 字典类型，默认为 None。

json：
（可选参数）json 格式用于附加到请求的主体，默认为 None。

headers：
（可选参数）请求头，默认为 None。

- **返回值**

返回请求对象

3.4. request.head

该函数用于发送 HEAD 请求。

- **函数原型**

```
request.head(url, data, json, headers)
```

- **参数**

url：
网址，字符串类型。

data：
（可选参数）附加到请求的正文，json 字典类型，默认为 None。

json：
（可选参数）json 格式用于附加到请求的主体，默认为 None。

headers：
（可选参数）请求头，默认为 None。

- **返回值**

返回请求对象

3.5. request.patch

该函数用于发送 PATCH 请求。

- **函数原型**

```
request.patch(url, data, json, headers)
```

- **参数**

url：
网址，字符串类型。

data：
（可选参数）附加到请求的正文，json 字典类型，默认为 None。

json：
（可选参数）json 格式用于附加到请求的主体，默认为 None。

headers：
（可选参数）请求头，默认为 None。

- **返回值**

返回请求对象

3.6. request.delete

该函数用于发送 DELETE 请求。

- **函数原型**

```
request.delete(url, data, json, headers)
```

- **参数**

url：
网址，字符串类型。

data：
（可选参数）附加到请求的正文，json 字典类型，默认为 None。

json：
（可选参数）json 格式用于附加到请求的主体，默认为 None。

headers：
（可选参数）请求头，默认为 None。

- **返回值**

返回请求对象

3.7. reponse 类方法说明

```
response =request.get(url)
```

方法	说明
response.content	返回响应的内容，以字节为单位
response.text	以文本方式返回响应的内容，编码为 unicode
response.json()	返回响应的 json 编码内容并转为 dict 类型
response.close()	关闭 socket

4 示例

将 QuecPython 开发板连接至电脑，接入后的操作方法详见《Quectel_QuecPython_基础操作说明》。

开发板连接至电脑后，创建 *test.py* 文件，导入 QuecPython 的 *request* 模块，分别创建 HTTP GET/PUT/POST/DELETE 等请求代码，编写完成后，将文件上传到开发板内并运行 *test.py* 文件，方法详见《Quectel_QuecPython_基础操作说明》。示例代码及运行结果详见如下章节。

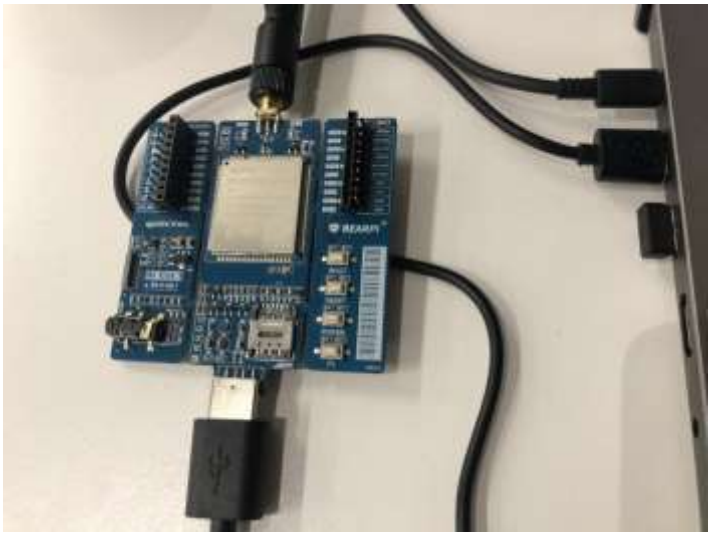

图 4：QuecPython 开发板与电脑连接

4.1. 请求 POST

- 示例代码

```
import request
import ujson

url = "http://httpbin.org/post"
data = {"key1": "value1", "key2": "value2", "key3": "value3"}

# POST 请求
response = request.post(url, data=ujson.dumps(data))
print(response.text)
```

- 代码运行结果

```
>>> import example
>>> example.exec('test.py')
{
  "args": {},
  "data": "{\"key3\": \"value3\", \"key1\": \"value1\", \"key2\": \"value2\"}",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "54",
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-5f8eaf92-1b8bbde8012cdddd47d7b15f"
  },
  "json": {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
  },
  "origin": "36.61.65.119",
  "url": "http://httpbin.org/post"
}

>>>
```

4.2. 请求 GET

- 示例代码

```
import request

url = "http://httpbin.org/get"

# GET 请求
response = request.get(url)
print(response.text)
```

- 代码运行结果

```
>>> import example
>>> example.exec('test.py')
{
  "args": {},
  "headers": {
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-5f8eb05d-4e742c1b2e51f1286b96c870"
  },
  "origin": "36.61.65.119",
  "url": "https://httpbin.org/get"
}
```

4.3. 请求 PUT

- 示例代码

```
import request

url = "http://httpbin.org/put"

# PUT 请求
response = request.put(url)
print(response.text)
```

- 代码运行结果

```
>>> import example
>>> example.exec('test.py')
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-5f8eb0ed-505e003f408841920b9a935d"
  },
  "json": null,
  "origin": "36.61.65.119",
  "url": "https://httpbin.org/put"
}
```

4.4. 请求 PATCH

- 示例代码

```
import request

url = "http://httpbin.org/patch"

# PATCH 请求
response = request.patch(url)
print(response.text)
```

- 代码运行结果

```
>>> import example
>>> example.exec('test.py')
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-5f8eb3d2-0e8ed2be20eeeff84a2f58cf"
  },
  "json": null,
  "origin": "36.61.65.119",
  "url": "https://httpbin.org/patch"
}
```

4.5. 请求 DELETE

- 示例代码

```
import request

url = "http://httpbin.org/delete"

# DELETE 请求
response = request.delete(url)
print(response.text)
```

- 代码运行结果

```
>>> import example
>>> example.exec('test.py')
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Host": "httpbin.org",
    "X-Amzn-Trace-Id": "Root=1-5f8eb44c-630673217781803043cb8c2c"
  },
  "json": null,
  "origin": "36.61.65.119",
  "url": "https://httpbin.org/delete"
}
```

4.6. 请求 HTTP 连接

- 示例代码

```
import request

url = "https://myssl.com"

# HTTPS 请求
response = request.get(url)
print(response.text)
```

- 代码运行结果

```
>>> import example
>>> example.exec('test.py')
<!doctype html>
<html class="no-js" lang="zh-CN">

<head>
  <meta charset="utf-8">

  <meta http-equiv="x-dns-prefetch-control" content="on">
  <link rel="dns-prefetch" href="//myssl.com">
  <link rel="dns-prefetch" href="//cdn.bootcss.com">
  <link rel="dns-prefetch" href="//zz.bdstatic.com">
  <link rel="dns-prefetch" href="//script.hotjar.com">
  <link rel="dns-prefetch" href="//static.hotjar.com">
  <link rel="dns-prefetch" href="//browser-update.org">
  <link rel="dns-prefetch" href="//www.google-analytics.com">
  <link rel="dns-prefetch" href="//hm.baidu.com">
```

5 附录 A 术语缩写

表 1：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序编程接口
HTTP	Hyper Text Transfer Protocol	超文本传输协议
SDK	Software Development Kit	软件开发工具包
TCP	Transmission Control Protocol	传输控制协议
URL	Uniform Resource Locator,	统一资源定位符
WWW	World Wide Web	万维网