XR871 HTTPC Developer Guide

Revision 1.0

Sep 11, 2017

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE INFORMATION FURNISHED BY XRADIO IS BELIEVED TO BE ACCURATE AND RELIABLE. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE. XRADIO DOES NOT ASSUME ANY RESPONSIBILITY AND LIABILITY FOR ITS USE. NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO. THIS DATASHEET NEITHER STATES NOR IMPLIES WARRANTY OF ANY KIND, INCLUDING FITNESS FOR ANY PARTICULAR APPLICATION.

THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Data	Summary of Changes
1.0	2017-9-11	Initial Version

Table 1-1 Revision History

Contents

Tables

Figures

1 模块概要

1.1 功能介绍

HTTPClient 模块提供了实现 http client 的应用接口，该模块基于 c 实现，可移植性高、支持 1.0./1.1、GET/POST、SSL、AUTH 等。xr871 SDK 中移植版本为 1.0，移植过程中裁剪了部分不需要的功能（代理），并对其提供的接口进行了再封装，使用户不用研究底层细节的实现，利用顶层的接口就可以实现客户端应用。（源码中 cmd_httpc.c 中提供了例程）

1.2 代码位置

module	file	location
HTTPClient	source	sdk/src/net/HTTPClient
	header	sdk/include/net/HTTPClient/
	demo	sdk/project/common/cmd/cmd_httpc.c

表 1-1 HTTPClient 文件结构

2 模块接口

module	function	detail
HTTPClient	HTTPC_open	int HTTPC_open(HTTPParameters *ClientParams) Purpose : 创建 httpc 会话句柄，并设置会话相关属性。 Param : ClientParams：会话用户参数指针 Uri：[in] 目标 uri HttpVerb：[in] cmd (GET/POST) UserName：[in] auth 用户名 Password：[in] auth 密码 AuthType：[in] auth 类型 pData：[in] post 数据 pLength：[in] post 数据长度 Flags：[in] 其他要求的 flag，如长连接 Returns : 返回状态 (0: ok)
	HTTPC_request	int HTTPC_request(HTTPParameters *ClientParams, HTTP_CLIENT_GET_HEADER Callback) Purpose : 发起连接请求。 Param : ClientParams: 会话用户参数指针 Callback: 添加用户自定义头部回调函数 该函数需要返回用户自定义的头部及头部值，格式需要满足”key1:valule1&key2:value2” Returns : 返回状态 (0: ok)
	HTTPC_get_request_info	int HTTPC_get_request_info(HTTPParameters *ClientParams, void *HttpClient) Purpose : 发起连接请求成功后，通过该函数可以获取连接参数。 Param : ClientParams: 会话用户参数指针 HttpClient: 获取的连接参数。该参数需要传入 HTTP_CLIENT 结构的指针。 Returns : 返回状态 (0: ok)
	HTTPC_write	int HTTPC_write(HTTPParameters *ClientParams, VOID *pBuffer, UINT32

		toWrite) Purpose： 写数据。 Param： ClientParams: 会话用户参数指针 pBuffer: 数据 buffer toWrite: 要写的字节数 Returns： 返回实际写的字节数
	HTTPC_read	int HTTPC_read(HTTPParameters *ClientParams, VOID *pBuffer, UINT32 toRead, UINT32 *recived) Purpose： 读数据。 Param： ClientParams: 会话用户参数指针 pBuffer: 数据 buffer toRead: 要读的字节数（不能大于 buffer 的长度） recived: 实际接收的数据的长度 Returns： 返回状态 (0: ok)
	HTTPC_close	int HTTPC_close(HTTPParameters *ClientParams) Purpose： 关闭当前连接，并释放相关资源。 Param： ClientParams: 会话用户参数指针 Returns： 返回状态 (0: ok)
	HTTPC_reset_session	int HTTPC_reset_session(HTTPParameters *ClientParams) Purpose： 重置上次会话的服务信息（例如：header 信息）。 Param： ClientParams: 会话用户参数指针 Returns： 返回状态 (0: ok)
	HTTPC_Register_user_certs	void HTTPC_Register_user_certs(HTTPC_USR_CERTS certs) Purpose： 安全访问下，使用该接口设置客户端证书参数。 Param： certs: 该回调用户返回证书，证书的格式要按照 tls 要求的客户端(security_client)执行。 Returns： 无

3 模块接口例程

本节提供上节介绍接口的示例，描述接口的使用方法及流程，文中代码皆为参考代码，不能直接运行，运行代码可直接参考 sdk 中源码文件 cmd_httpc.c 中包含了接口的使用。

3.1 客户端 GET 流程

第一步：安装客户端证书

```
//该步不是必须的，安全传输条件下，需要在发起请求之前设置证书
#ifdef HTTPC_CMD_REGISTER_USER_CALLBACK
    HTTPC_Register_user_certs(get_certs);
#endif
```

第二部：初始化并创建连接句柄

```
HTTPParameters *clientParams;
clientParams = malloc(sizeof(*clientParams));
clientParams->Uri = //赋值 uri
clientParams->HttpVerb = VerbGet;  //赋值方法

下面三行为认证相关的设置
//clientParams->UserName
//clientParams->Password
//clientParams->AuthType = AuthSchemaDigest

if (HTTPC_open(clientParams) != 0) {
       CMD_ERR("http open err..\n");
}
```

第三步：发起连接请求

```
if ((ret = HTTPC_request(clientParams, NULL)) != 0) {
       CMD_ERR("http request err..\n");
       goto relese;
}
```

第四步：获取请求反馈信息

```
if (HTTPC_get_request_info(clientParams, &httpClient) != 0){
```



```
        CMD_ERR("http get request info err..\n");
}
```

第五步：读写数据

```
if ((ret = HTTPC_read(clientParams, buf, toReadLength, (void *)&Received)) != 0)
{
        //CMD_DBG("get data,Received:%d\n",Received);
        if (ret == 1000) {
                ret = 0;
                CMD_DBG("The end..\n");
        } else
                CMD_ERR("Transfer err...ret:%d\n",ret);
        break;
} else {
        //CMD_DBG("get data,Received:%d\n",Received);
}
```

第六步：释放资源

```
HTTPC_close(clientParams);
```

3.2 客户端 POST 流程（非安全模式下）

第一步：初始化参数并创建句柄

```
HTTPParameters *clientParams;
clientParams = malloc(sizeof(*clientParams));
clientParams->Uri = //赋值 uri
clientParams->HttpVerb = VerbPost;

memcpy(buf, credentials, strlen(credentials));
clientParams->pData = buf; //post 的数据
clientParams->pLength = strlen(credentials); //post 数据长度

if ((ret = HTTPC_open(clientParams)) != 0) {
        CMD_ERR("http open err..\n");
        goto relese;
}
```

第二步：发起连接请求

```
if ((ret = HTTPC_request(clientParams, NULL)) != 0) {
        CMD_ERR("http request err..\n");
        goto relese;
}
```


第三步：获取连接请求反馈信息

```
if ((ret = HTTPC_get_request_info(clientParams, &httpClient)) != 0) {
       CMD_ERR("http get request info err..\n");
       goto relese;
}
```

第四步：读取数据

```
if ((ret = HTTPC_read(clientParams, buf, toReadLength, (void *)&Received)) != 0)
{
       //CMD_DBG("get data,Received:%d\n",Received);
       if (ret == 1000) {
              ret = 0;
              CMD_DBG("The end..\n");
       } else
              CMD_ERR("Transfer err...ret:%d\n",ret);
              break;
} else {
       //CMD_DBG("get data,Received:%d\n",Received);
}
```

第五步：释放资源并关闭连接

```
HTTPC_close(clientParams);
```


4 其他配置

1. 代码中配置支持 ssl

```
文件 Sdk/include/net/HTTPClient/API/HTTPClientWrapper.h

#define       HTTPC_SSL
```

2. 调试配置

```
Sdk/include/net/HTTPClient/API/debug.h

//#define HTTPCLIENT_DEBUG
```

