Hi3861 V100 / Hi3861L V100 HTTP

Development Guide

Issue 01

Date 2020-04-30

Trademarks and Permissions

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road,
Bantian, Longgang District,
Shenzhen 518129 P. R. China

Website: https://www.hisilicon.com/en/

Email: support@hisilicon.com

About This Document

Purpose

This document describes HTTP client development.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3861	V100
Hi3861L	V100

Intended Audience

The document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
⚠ DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.
⚠ WARNING	Indicates a hazard with a medium level of risk which, if not avoided, could result in death or serious injury.
⚠ CAUTION	Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury.

Symbol	Description
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
NOTE	Supplements the important information in the main text. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Change History

Issue	Date	Change Description
01	2020-04-30	This issue is the first official release.
00B01	2020-01-15	This issue is the first draft release.

Contents

1 Development Guidance

1.1 Overview

The HTTP sample uses the lightweight TCP/IP stack (lwIP) APIs to connect to specified IP addresses and obtain web pages.

NOTE

The HTTP implementation does not require new APIs and depends only on the APIs provided by lwIP. For details about related APIs, see the *Hi3861 V100/Hi3861L V100 lwIP Development Guide*.

1.2 Sample Code

The following is the sample code for the HTTP client to obtain the web page of a passed IP address.

```
#include "hi_stdlib.h"
#include "lwip/sockets.h"
#include "lwip/netdb.h"

#define HTTPC_DEMO_RECV_BUFSIZE 64
#define SOCK_TARGET_PORT  80
static const char *g_request = "GET / HTTP/1.1\r\n\
    Content-Type: application/x-www-form-urlencoded;charset=UTF-8\r\n\
    Host: baidu.com\r\n\
    Connection: close\r\n\
    \r\n";

unsigned int http_clienti_get(int argc, char* argv[])
{
    if ((argc != 1) || (argv == NULL)) {
        return 1;
    }
    struct sockaddr_in addr = {0};
    int s, r;
    char recv_buf[HTTPC_DEMO_RECV_BUFSIZE];
    addr.sin_family = AF_INET;
    addr.sin_port = PP_HTONS(SOCK_TARGET_PORT);
    addr.sin_addr.s_addr = inet_addr(argv[0]);
    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0) {
        return 1;
    }
    printf("... allocated socket");
    if (connect(s, (struct sockaddr*)&addr, sizeof(addr)) != 0) {
        printf("... socket connect failed errno=%d", errno);
        lwip_close(s);
        return 1;
    }
    printf("... connected");
    if (lwip_write(s, g_request, strlen(g_request)) < 0) {
        lwip_close(s);
        return 1;
    }
    printf("... socket send success");
    struct timeval receiving_timeout;
    /* 5S Timeout */
    receiving_timeout.tv_sec = 5;
    receiving_timeout.tv_usec = 0;
    if (setsockopt(s, SOL_SOCKET, SO_RCVTIMEO, &receiving_timeout, sizeof(receiving_timeout))
< 0) {
        printf("... failed to set socket receiving timeout");
        lwip_close(s);
        return 1;
    }
    printf("... set socket receiving timeout success");
    /* Read HTTP response */
    do {
        (void)memset_s(recv_buf, sizeof(recv_buf), 0, sizeof(recv_buf));
        r = lwip_read(s, recv_buf, sizeof(recv_buf) - 1);
        for (int i = 0; i < r; i++) {
            putchar(recv_buf[i]);
        }
    } while (r > 0);
    printf("... done reading from socket. Last read return=%d errno=%d\r\n", r, errno);
    lwip_close(s);
    return 0;
}
```