A76xx Series Open SDK_ FTP_应用指导

LTE 模组

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
技术支持邮箱：support@simcom.com
官网：www.simcom.com

名称：	A76xx Series Open SDK_FTP_应用指导
版本：	V1.00
类别：	应用文档
状态：	已发布

版权声明

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
邮箱：simcom@simcom.com
官网：www.simcom.com

了解更多资料，请点击以下链接：
http://cn.simcom.com/download/list-230-cn.html

技术支持，请点击以下链接：
http://cn.simcom.com/ask/index-cn.html 或发送邮件至 support@simcom.com

Version History

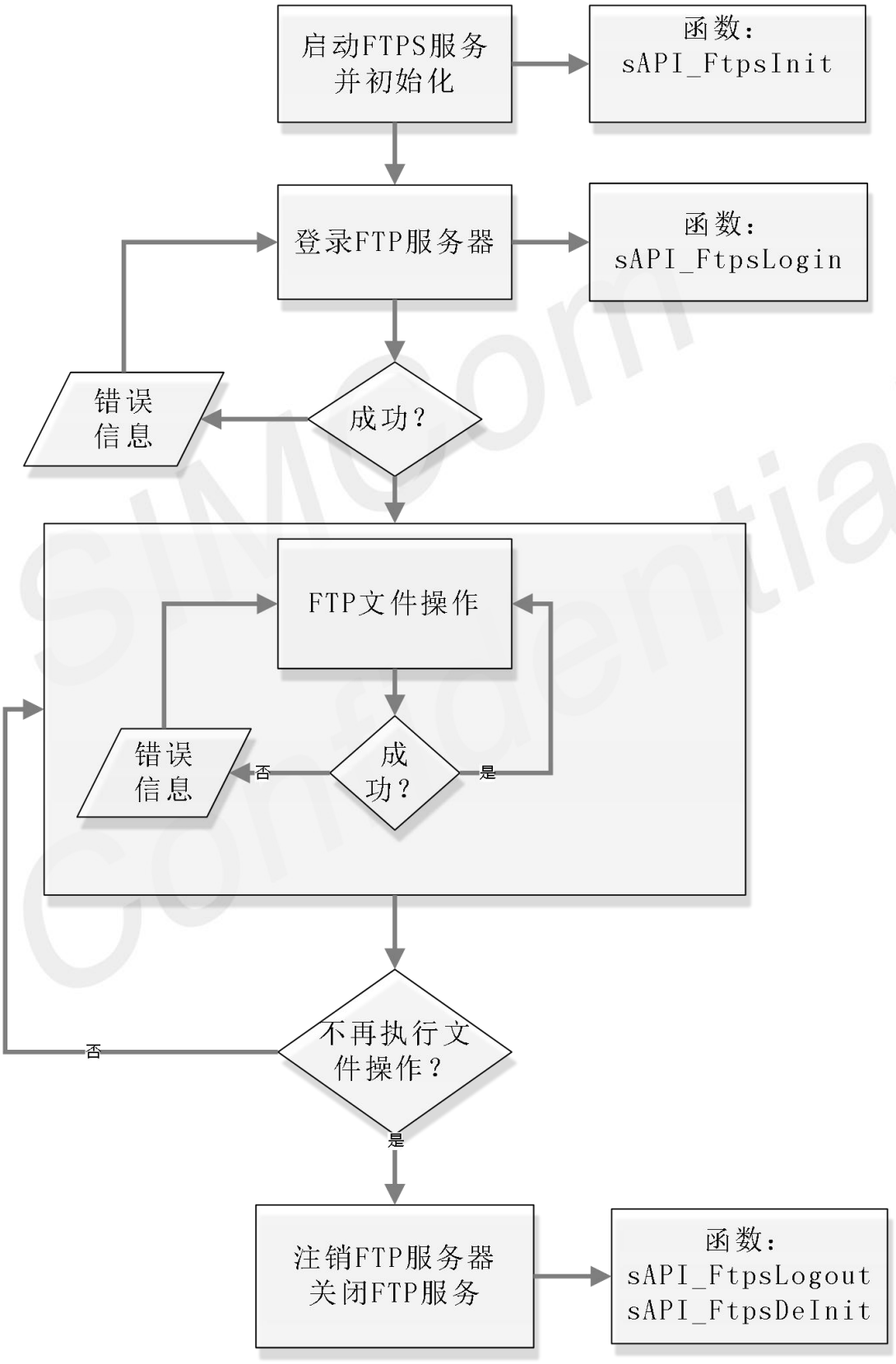
Version	Date	Owner	What is new
V1.00	2022-10-26		第一版

About this Document

本文档适用于 A1803S open 系列、A1603 open 系列、A1606 open 系列。

目录

1 使用流程



2API 介绍

本部分介绍了 FTP 相关 API。

2.1 sAPI_FtpsInit

该 API 用于启动 FTP(S)服务，通过激活 PDP 上下文的方式。你必须在任何其他 FTP(S) 相关操作之前执行 sAPI_FtpsInit。

接口:	void sAPI_FtpsInit(SC_PDP_ACTIVE_TYPE type, OSAMsgQRef msgQRef);
参数：	[in] type：用于 PDP 激活。 0：SC_FTPS_USB：用于 USB 口。 1：SC_FTPS_UART：用于 UART 口。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.2 sAPI_FtpsDeInit

该 API 用于停止 FTP(S)服务，当你不再使用 FTP(S)服务，使用此 API。

接口:	void sAPI_FtpsDeInit(SC_PDP_ACTIVE_TYPE type);
参数：	[in] type：用于 PDP 激活。 0：SC_FTPS_USB：用于 USB 口。 1：SC_FTPS_UART：用于 UART 口。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.3 sAPI_FtpsLogin

该 API 用于登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsLogin(SCftpsLoginMsg msg);
参数：	[in] msg: SCftpsLoginMsg:包含登录到 FTP(S)服务器所需的信息。包括服务器 IP 地址、用户名、密码、端口号、服务器类型。 IP address：主机地址，类型：String，最大长度 128。 Username：FTP (S)用户名，类型：String，最大长度 128。 Password：用户密码，类型：String，最大长度为 128。 Port：FTP(S)服务的主机监听端口，取值范围为 1 ~ 65535。 Server type：FTP (S)服务器类型，类型：Numeric，从 0 到 3，默认为 3。 0：FTP 服务器；

	1：带有 AUTH SSL 的显式 FTPS 服务器； 2：带有 AUTH TLS 显式 FTPS 服务器； 3：隐式 FTPS 服务器。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.4 sAPI_FtpsLogout

该 API 用于注销 FTP(S)服务器，在使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsLogout();
参数：	无。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。
NOTE:	当你想停止 FTP(S)服务，请使用 sAPI_FtpsLogout 注销 FTP(S)服务器，然后使用 sAPI_FtpsDeInit 停止 FTP，如果只使用 sAPI_FtpsDeInit，它将报告 ERROR。

2.5 sAPI_FtpsList

该 API 用于列出在 FTP(S) 服务器上指定目录中的项目，使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsList(char* dir);
参数：	[in] dir：要列出的目录，类型：String，最大长度为 112。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息 apiFtpsData：返回时，该结构包含数据的长度、内容以及标志是否结束。

2.6 sAPI_FtpsCreateDirectory

该 API 用于在 FTP(S) 服务器上创建一个新目录，请务必在创建目录之前成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsCreateDirectory(char* dir);
参数：	[in] dir： 要列出的目录，类型：String，最大长度为 112。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.7 sAPI_FtpsDeleteDirectroy

该 API 用于删除 FTP(S)服务器上的一个目录，删除目录前请确保已成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsDeleteDirectroy(char* dir);
参数：	[in] dir： 要删除的目录，类型：String，最大长度为 112。

返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.8 sAPI_FtpsChangeDirectory

该 API 用于改变 FTP(S)服务器上的当前目录。使用该 API 之前请确保已成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsChangeDirectory(char* dir);
参数：	[in] dir: 要更换的目录，类型：String，最大长度为 112。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.9 sAPI_FtpsGetCurrentDirectory

该 API 用于获取 FTP(S)服务器上的当前目录。使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsGetCurrentDirectory();
参数：	无。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。 char*：由 msgQRef 返回,目录名称。

2.10 sAPI_FtpsDeleteFile

该 API 用于删除 FTPS 服务器上的文件。使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsDeleteFile(char* fileName);
参数：	[in] fileName：要删除的文件的名称。类型：String，最大长度为 112。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.11 sAPI_FtpsDownloadFile

该 API 用于从 FTP(S) 服务器下载文件到模块，您可以选择保存下载文件的目录。使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsDownloadFile(char* fileName, SC_FTPS_FILE_LOCATION loc);
参数：	[in] fileNam：远程文件路径。类型：String，最大长度为 112。 [in] loc：保存下载文件的目录。类型：Numeric，范围为 1-2。 1: C: /(本地存储); 2: D: / (SD 卡);
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.12 sAPI_FtpsUploadFile

该 API 用于将文件上传到 FTP(S)服务器模块。通过设置 loc 参数，可以选择要上传文件的目录。使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsUploadFile(char* fileName, SIMCOM_FTPS_FILE_LOCATION loc, int startPos);
参数：	[in] fileName：远程文件路径。类型：String，最大长度为 112。 [in] loc：保存下载文件的目录。类型：Numeric，范围为 1-2。 1. C： /(本地存储)； 2. D： / (sd 卡)； [in] startPos： FTP“REST”命令的值，用于上次传输失败时中断传输。类型：Numeric，取值范围为 0 ~ 2147483647。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.13 sAPI_FtpsDownloadFileToBuffer

该 API 用于从 FTP(S)服务器获取文件到缓冲区。在使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsDownloadFileToBuffer(char* fileName, int startPos);
参数：	[in] fileName：远程文件路径。类型：String，最大长度为 112。 [in] startPos： FTP“REST”命令的值，用于上次传输失败时中断传输。类型：Numeric，取值范围为 0 ~ 2147483647。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.14 sAPI_FtpsGetFileSize

该 API 用于获取 FTP(S)服务器上的文件大小。在使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsGetFileSize(char* fileName);
参数：	[in] fileName：远程文件路径。类型：String，最大长度为 112。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.15 sAPI_FtpsTransferType

该 API 用于设置 FTP(S)服务器上的传输类型，在使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	SC_FTPS_RETURNCODE sAPI_FtpsTransferType(char* type);
参数：	[in] type：传输的类型：

	A：ASCII。 I：二进制。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.16 sAPI_FtpsGetTransferType

该 API 用于获取 FTP(S)服务器上的传输类型，在使用该 API 之前请确保已经成功登录 FTP(S)服务器。

接口:	void sAPI_FtpsGetTransferType();
参数：	无。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

2.17 sAPI_FtpsSslConfig

该 API 用于设置 FTPS 会话的 SSL 上下文 id。

接口:	void sAPI_FtpsSslConfig(int session, int sslId);
参数：	[in] session: 0 表示控制会话，1 表示数据会话。 [in] sslId: 0-9 期间的 SSL 上下文 ID。 请参阅《A76xx Series Open_SDK_SSL_应用指导》获取详细信息。
返回值:	SC_FTPS_ERROR_CODE：请参阅第 4 章查询错误码信息。

3 错误码信息

For more details, please refer to simcom_ftps.h.

<errcode>	Description
0	成功
1	SSL 警报
2	未知错误
3	FTP 忙碌
4	连接被服务器关闭
5	超时
6	传输失败
7	文件不存在或有其他错误
8	无效的参数
9	操作被服务器拒绝
10	网络错误
11	状态错误
12	解析服务器名称失败
13	创建 socket 出错
14	连接 socket 失败
15	关闭 socket 失败
16	SSL 会话关闭
17	文件错误，文件不存在或其他错误。
421	服务器响应连接超时，当收到错误码 421 时，您需要执行 AT+CFTPSLOGOU 注销服务器，然后再次 AT+CFTPSLOGIN 进行进一步操作。

4Demo

```
/**
  ******************************************************************************
  * @file       demo_ftps.c
  * @author    SIMCom OpenSDK Team
  * @brief     Source file of ftps function.
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2022 SIMCom Wireless.
  * All rights reserved.
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#ifdef FEATURE_SIMCOM_FTPS
#include "stdlib.h"
#include "string.h"
#include "stdio.h"
#include "simcom_os.h"
#include "simcom_ftps.h"
#include "simcom_common.h"
#include "simcom_debug.h"
#include "simcom_uart.h"

typedef enum{
    SC_FTPS_DEMO_INIT                   = 1,
    SC_FTPS_DEMO_LOGIN                    = 2,
```

```
    SC_FTPS_DEMO_LOGOUT                 = 3,
    SC_FTPS_DEMO_GETDIR                 = 4,
    SC_FTPS_DEMO_LIST                   = 5,
    SC_FTPS_DEMO_CHANGEDIR              = 6,
    SC_FTPS_DEMO_CREATDIR               = 7,
    SC_FTPS_DEMO_DELETEFILE             = 8,
    SC_FTPS_DEMO_DELETEDIR              = 9,
    SC_FTPS_DEMO_SIZE                   = 10,
    SC_FTPS_DEMO_DOWNLOADFILE           = 11,
    SC_FTPS_DEMO_DOWNLOADFILETOBUFFER   = 12,
    SC_FTPS_DEMO_UPLOADFILE             = 13,
    SC_FTPS_DEMO_DEINIT                 = 14,
    SC_FTPS_DEMO_SSLCONFIG              = 15,

    SC_FTPS_DEMO_MAX                    = 99
}SC_FTPS_DEMO_TYPE;


extern sMsgQRef simcomUI_msgq;
extern int ftpsTestTime;
extern int ftpsCurrentTime;
sMsgQRef ftpsUIResp_msgq;

extern void PrintfOptionMenu(char *options_list[], int array_size);
extern void PrintfResp(const char *format, ...);
extern SIM_MSG_T GetParamFromUart(void);
extern void FtpsTestDemoInit(void);


/**
  * @brief   FTPS demo operation
  * @param   void
  * @note
```

```
 * @retval void
 */
void FtpsDemo(void)
{
    enum appChannelType channeltype;
    SCapiFtpsData* ftpsData;
    char* dir = NULL;
    char* file = NULL;
    int* size = NULL;
    SC_STATUS status;
    SC_FTPS_RETURNCODE ret = SC_FTPS_RESULT_OK;

    SIM_MSG_T optionMsg ={0,0,0,NULL};
    UINT32 opt = 0;
    char *note = "\r\nPlease select an option to test from the items listed below.\r\n";
    char *options_list[] = {
        "1. Init",
        "2. Login",
        "3. Logout",
        "4. Print work directory",
        "5. List",
        "6. Change work directory",
        "7. Mkdir",
        "8. Delete file",
        "9. Delete folder",
        "10. Get file size",
        "11. Download file",
        "12. Download file to buffer",
        "13. Upload file",
        "14. DeInit",
        "15. Test",
        "99. back",
    };
```

```
    while(1)
    {
        SIM_MSG_T ftpsMsg = {0,0,0,NULL};
        PrintfResp(note);
        PrintfOptionMenu(options_list,sizeof(options_list)/sizeof(options_list[0]));
        sAPI_MsgQRecv(simcomUI_msgq,&optionMsg,SC_SUSPEND);
        if(SRV_UART != optionMsg.msg_id)
        {
            sAPI_Debug("%s,msg_id is error!!",__func__);
            break;
        }

        sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
        opt = atoi(optionMsg.arg3);
        free(optionMsg.arg3);

        switch(opt)
        {
            case SC_FTPS_DEMO_INIT:
            {
                if(NULL == ftpsUIResp_msgq)
                {
                    SC_STATUS status;
                    status = sAPI_MsgQCreate(&ftpsUIResp_msgq, "ftpsUIResp_msgq",
sizeof(SIM_MSG_T), 8, SC_FIFO);
                    if(SC_SUCCESS != status)
                    {
                        sAPI_Debug("msgQ create fail");
                        PrintfResp("\r\nFTP Init Fail!\r\n");
                        break;
                    }
                }
```

```
                sAPI_FtpsInit(SC_FTPS_USB,ftpsUIResp_msgq);

                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    sAPI_Debug("Init SUCCESS");
                    PrintfResp("\r\nFTP Init Successful!\r\n");
                    break;
                }
                else
                {
                    sAPI_Debug("Init FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                    PrintfResp("\r\nFTP Init Fail!\r\n");
                    break;
                }
            }

            case SC_FTPS_DEMO_LOGIN:
            {

                sAPI_Debug("FTPS_login");
                SCftpsLoginMsg msg = {0};
                msg.serverType = 0;

                PrintfResp("\r\nPlease input Ip.\r\n");
                optionMsg = GetParamFromUart();
                strcpy(msg.host,optionMsg.arg3);
                free(optionMsg.arg3);
```

```
                PrintfResp("\r\nPlease input Port.\r\n");
                optionMsg = GetParamFromUart();
                msg.port = atoi(optionMsg.arg3);
                free(optionMsg.arg3);



                PrintfResp("\r\nPlease input User.\r\n");
                optionMsg = GetParamFromUart();
                strcpy(msg.username,optionMsg.arg3);
                free(optionMsg.arg3);

                PrintfResp("\r\nPlease input Pass.\r\n");
                optionMsg = GetParamFromUart();
                strcpy(msg.password,optionMsg.arg3);
                free(optionMsg.arg3);

                sAPI_Debug("host = [%s],user = [%s],pass = [%s]",msg.host,msg.username,msg.password);

                ret = sAPI_FtpsLogin(msg);
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",
                    channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);
                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    sAPI_Debug("LOGIN SUCCESSFUL");
                    PrintfResp("\r\nFTP Login Successful!\r\n");
                    break;
                }
                else
                {
                    sAPI_Debug("LOGIN ERROR,ERROR CODE = [%d]",ftpsMsg.arg2);
```

```
                PrintfResp("\r\nFTP Login Fail!\r\n");
                break;
            }
        }

        case SC_FTPS_DEMO_LOGOUT:
        {
            sAPI_Debug("FTPS_logout");
            ret = sAPI_FtpsLogout();
            sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
            channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
            sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

            if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
            {
                sAPI_Debug("LOGOUT SUCCESSFUL");
                PrintfResp("\r\nFTP Logout Successful!\r\n");
                break;
            }
            else
            {
                sAPI_Debug("LOGOUT FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                PrintfResp("\r\nFTP Logout Fail!\r\n");
                break;
            }
        }

        case SC_FTPS_DEMO_GETDIR:
        {
            char path[256] = {0};
            sAPI_Debug("FTPS_getCurrentDirectory");
            ret = sAPI_FtpsGetCurrentDirectory();
```

```
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    dir = (char *)ftpsMsg.arg3;
                    sAPI_Debug("GET CURRENT DIRECTORY SUCCESS = [%s]",dir);
                    snprintf(path,sizeof(path),"\r\ncurrent path = %s\r\n",dir);
                    PrintfResp(path);
                    break;
                }
                else
                {
                    sAPI_Debug("GET CURRENT DIRECTORY FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                    PrintfResp("\r\nGet Directory Fail\r\n");
                    break;
                }

            }

            case SC_FTPS_DEMO_LIST:
            {

                sAPI_Debug("FTPS_list");
                PrintfResp("\r\nPlease input directory.\r\n");
                optionMsg = GetParamFromUart();
                dir = (char *)optionMsg.arg3;

                ret = sAPI_FtpsList(dir);
                if(ret != SC_FTPS_RESULT_OK)
```

```
                {
                    sAPI_Debug("LIST FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                    PrintfResp("\r\nFTP LIST Fail!\r\n");
                    break;
                }
                while(1)
                {
                    sAPI_Debug("COMING TO THE RECV");
                    ftpsMsg.arg3 = NULL;
                    status = sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                    sAPI_Debug("status = [%d]",status);
                    channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                    sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);
                    if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                    {
                        ftpsData = (SCapiFtpsData*)ftpsMsg.arg3;
                        sAPI_Debug("flag = [%d]",ftpsData->flag);
                        if(SC_DATA_COMPLETE == ftpsData->flag)
                        {
                            sAPI_Debug("get data complete!");
                            sAPI_Debug("flag = [%d],len = [%d]",ftpsData->flag,ftpsData->len);
                            free(ftpsData);
                            break;
                        }
                        else if(SC_DATA_RESUME == ftpsData->flag)
                        {
                            sAPI_Debug("get data successful!");
                            sAPI_Debug("flag = [%d],len = [%d]",ftpsData->flag,ftpsData->len);

                            //just for waiting uart ok now
                            sAPI_UartWrite(SC_UART,(UINT8*)ftpsData->data,ftpsData->len);
```

```
                    free(ftpsData->data);
                    free(ftpsData);
                }
                else
                {
                    sAPI_Debug("ERROR HAPPEN!");
                    break;
                }
            }
            else
            {
                sAPI_Debug("error code = [%d]",ftpsMsg.arg2);
                PrintfResp("\r\nFtp list fail!\r\n");
                break;
            }

        }
        free(dir);
        break;

    }

    case SC_FTPS_DEMO_CHANGEDIR:
    {

        PrintfResp("\r\nPlease input directory.\r\n");
        sAPI_Debug("sAPI_FtpsChangeDirectory");
        optionMsg = GetParamFromUart();
        sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
        dir = (char *)optionMsg.arg3;

        ret = sAPI_FtpsChangeDirectory(dir);
        sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
```

```
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    sAPI_Debug("Change Directory Successful");
                    PrintfResp("\r\nChange Directory Successful!\r\n");
                }
                else
                {
                    sAPI_Debug("Change Directory FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                    PrintfResp("\r\nChange Directory Fail!\r\n");
                }

                free(dir);
                break;

            }

            case SC_FTPS_DEMO_CREATDIR:
            {

                PrintfResp("\r\nPlease input directory.\r\n");
                sAPI_Debug("sAPI_FtpsCreateDirectory");
                optionMsg = GetParamFromUart();
                sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
                dir = (char *)optionMsg.arg3;

                ret = sAPI_FtpsCreateDirectory(dir);
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode =
```

```
[%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                    if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                    {
                        sAPI_Debug("Create Directory Successful");
                        PrintfResp("\r\nCreate Directory Successful!\r\n");
                    }
                    else
                    {
                        sAPI_Debug("Create Directory FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                        PrintfResp("\r\nCraete Directory Fail!\r\n");
                    }

                    free(dir);
                    break;

                }

                case SC_FTPS_DEMO_DELETEFILE:
                {

                    PrintfResp("\r\nPlease input file.\r\n");
                    sAPI_Debug("sAPI_FtpsDeleteFile");
                    optionMsg = GetParamFromUart();
                    sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
                    file = (char *)optionMsg.arg3;

                    ret = sAPI_FtpsDeleteFile(file);
                    sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                    channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                    sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode =
[%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);
```

```
                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    sAPI_Debug("Delete File Successful");
                    PrintfResp("\r\nDelete File Successful!\r\n");
                }
                else
                {
                    sAPI_Debug("Delete File Fail");
                    PrintfResp("\r\nDelete File Fail!\r\n");
                }

                free(file);
                break;

            }

            case SC_FTPS_DEMO_DELETEDIR:
            {

                PrintfResp("\r\nPlease input directory.\r\n");
                sAPI_Debug("sAPI_FtpsDeleteDirectory");
                optionMsg = GetParamFromUart();
                sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
                dir = (char *)optionMsg.arg3;

                ret = sAPI_FtpsDeleteDirectory(dir);
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode =
[%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
```

```
                    sAPI_Debug("Delete Directory Successful");
                    PrintfResp("\r\nDelete Directory Successful!\r\n");
                }
                else
                {
                    sAPI_Debug("Delete Directory Fail");
                    PrintfResp("\r\nDelete Directory Fail!\r\n");
                }

                free(dir);
                break;

            }

            case SC_FTPS_DEMO_SIZE:
            {

                char path[256] = {0};
                PrintfResp("\r\nPlease input directory.\r\n");
                sAPI_Debug("sAPI_FtpsGetFileSize");
                optionMsg = GetParamFromUart();
                sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
                file = (char *)optionMsg.arg3;

                ret = sAPI_FtpsGetFileSize(file);
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode =
[%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    size = (int *)ftpsMsg.arg3;
```

```
                    sAPI_Debug("GET SIZE SUCESS,SIZE = [%d]",*size);
                    snprintf(path,sizeof(path),"\r\nfile size = %d\r\n",*size);
                    PrintfResp(path);
                    free(size);
                }
                else
                {
                    sAPI_Debug("GET SIZE FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                    PrintfResp("\r\nGet File Size Fail!\r\n");
                }

                free(file);
                break;

            }

            case SC_FTPS_DEMO_DOWNLOADFILE:
            {

                PrintfResp("\r\nPlease input directory.\r\n");
                sAPI_Debug("sAPI_FtpsDownloadFile");
                optionMsg = GetParamFromUart();
                sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
                file = (char *)optionMsg.arg3;

                ret = sAPI_FtpsDownloadFile(file,SC_FTPS_FILE_FLASH);
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode =
[%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
```

```
            sAPI_Debug("DOWNLOAD FILE SUCCESSFUL");
            PrintfResp("\r\nDownload File Successful\r\n");
        }
        else
        {
            sAPI_Debug("DOWNLOAD FILE FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
            PrintfResp("\r\nDownload File Fail\r\n");
        }

        free(file);
        break;

    }

    case SC_FTPS_DEMO_DOWNLOADFILETOBUFFER:
    {

        PrintfResp("\r\nPlease input File name.\r\n");
        sAPI_Debug("sAPI_FtpsDownloadFileToBuffer");
        optionMsg = GetParamFromUart();
        sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
        file = (char *)optionMsg.arg3;

        ret = sAPI_FtpsDownloadFileToBuffer(file,0);
        if(ret != SC_FTPS_RESULT_OK)
        {
            sAPI_Debug("DownloadFileToBuffer FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
            PrintfResp("\r\nFTP DownloadFileToBuffer Fail!\r\n");
            break;
        }

        while(1)
        {
```

```
                    sAPI_Debug("COMING TO THE RECV");
                    ftpsMsg.arg3 = NULL;
                    status = sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                    sAPI_Debug("status = [%d]",status);
                    channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                    sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);
                    if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                    {
                        ftpsData = (SCapiFtpsData*)ftpsMsg.arg3;
                        sAPI_Debug("flag = [%d]",ftpsData->flag);
                        if(SC_DATA_COMPLETE == ftpsData->flag)
                        {
                            sAPI_Debug("get data complete!");
                            sAPI_Debug("flag = [%d],len = [%d]",ftpsData->flag,ftpsData->len);
                            free(ftpsData);
                            break;
                        }
                        else if(SC_DATA_RESUME == ftpsData->flag)
                        {
                            sAPI_Debug("get data successful!");
                            sAPI_Debug("flag = [%d],len = [%d]",ftpsData->flag,ftpsData->len);

                            //just for waiting uart ok now
                            sAPI_UartWrite(SC_UART,(UINT8*)ftpsData->data,ftpsData->len);

                            free(ftpsData->data);
                            free(ftpsData);
                        }
                        else
                        {
                            sAPI_Debug("ERROR HAPPEN!");
                            break;
```

```
                    }
                }
                else
                {
                    sAPI_Debug("error code = [%d]",ftpsMsg.arg2);
                    PrintfResp("FTP DOWNLOAD FILE TO BUFFER FAIL!");
                    break;
                }

            }

            free(file);
            break;

        }

        case SC_FTPS_DEMO_UPLOADFILE:
        {

            PrintfResp("\r\nPlease input directory.\r\n");
            sAPI_Debug("sAPI_FtpsUploadFile");
            optionMsg = GetParamFromUart();
            sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
            file = (char *)optionMsg.arg3;

            ret = sAPI_FtpsUploadFile(file,SC_FTPS_FILE_FLASH,0);
            sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
            channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
            sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode =
[%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

            if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
            {
```

```
            sAPI_Debug("UPLOAD FILE SUCCESSFUL");
            PrintfResp("\r\nUpload File Successful\r\n");
        }
        else
        {
            sAPI_Debug("UPLOAD FILE FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
            PrintfResp("\r\nUpload File Fail\r\n");
        }

        free(file);
        break;
    }

    case SC_FTPS_DEMO_DEINIT:
    {

        sAPI_Debug("FTPS_DeInit");
        sAPI_FtpsDeInit(SC_FTPS_USB);
        sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
        channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
        sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",
            channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

        if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
        {
            sAPI_Debug("DEINIT SUCCESS");
            PrintfResp("\r\nFTP DeInit Successful!\r\n");
            break;
        }
        else
        {
            sAPI_Debug("DEINIT FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
            PrintfResp("\r\nFTP DeInit Fail!\r\n");
```

```
            break;
        }
    }

    case SC_FTPS_DEMO_SSLCONFIG:
    {

        ret=sAPI_FtpsSslConfig ();
        sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
        if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
        {
            sAPI_Debug("Init SUCCESS");
            PrintfResp("\r\nSSLCONFIG Successful!\r\n");
            break;
        }
        else
        {
            sAPI_Debug("SSLCONFIG FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
            PrintfResp("\r\n SSLCONFIG Fail!\r\n");
            break;
        }
    }            }

    case SC_FTPS_DEMO_MAX:
    {
        sAPI_Debug("Return to the previous menu!");
        PrintfResp("\r\nReturn to the previous menu!\r\n");

        /*Logout first*/
        sAPI_Debug("FTPS_logout");
        sAPI_FtpsLogout();
        sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
        channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
```

```
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    sAPI_Debug("LOGOUT SUCCESSFUL");
                }
                else
                {
                    sAPI_Debug("LOGOUT FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                }

                memset(&ftpsMsg,0,sizeof(ftpsMsg));

                /*Then DeInit*/
                sAPI_Debug("FTPS_DeInit");
                sAPI_FtpsDeInit(SC_FTPS_USB);
                sAPI_MsgQRecv(ftpsUIResp_msgq,&ftpsMsg,SC_SUSPEND);
                channeltype = GET_SRV_FROM_MSG_ID(ftpsMsg.msg_id);
                sAPI_Debug("SRV = [%d],msgId = [%d],api = [%d],errCode = [%d]",channeltype,ftpsMsg.msg_id,ftpsMsg.arg1,ftpsMsg.arg2);

                if(SC_FTPS_RESULT_OK == ftpsMsg.arg2)
                {
                    sAPI_Debug("DEINIT SUCCESS");
                }
                else
                {
                    sAPI_Debug("DEINIT FAIL,ERRCODE = [%d]",ftpsMsg.arg2);
                }

                return;
            }
```

```
                default :
                    break;

            }
        }
}

#endif
```

5 变量定义

5.1 SC_PDP_ACTIVE_TYPE

变量名	SC_PDP_ACTIVE_TYPE
变量定义	typedef enum { SC_FTPS_USB, SC_FTPS_UART }SC_PDP_ACTIVE_TYPE;

5.2 SC_FTPS_FILE_LOCATION

变量名	SC_FTPS_FILE_LOCATION
变量定义	typedef enum{ SC_FTPS_FILE_FLASH = 1, SC_FTPS_FILE_SDCARD, SC_FTPS_FILE_EXT_FLASH_FS }SC_FTPS_FILE_LOCATION;

5.3 SC_FTPS_DATASOCKET_IP_TYPE

变量名	SC_FTPS_DATASOCKET_IP_TYPE
变量定义	typedef enum{ SC_PORT_RESPONSE_BY_SERVER, SC_SAME_AS_CONTROLSOCKET_IP }SC_FTPS_DATASOCKET_IP_TYPE;

5.4 SC_FTPS_API_TYPE

变量名	SC_FTPS_API_TYPE

变量定义	typedef enum{ SC_FTPS_INIT, SC_FTPS_DEINIT, SC_FTPS_LOGIN, SC_FTPS_LOGOUT, SC_FTPS_DOWNLOADFILE, SC_FTPS_DOWNLOADTOBUFFER, SC_FTPS_UPLOADFILE, SC_FTPS_DELETEFILE, SC_FTPS_CREATDIR, SC_FTPS_DELETEDIR, SC_FTPS_CHANGEDIR, SC_FTPS_GETDIR, SC_FTPS_LIST, SC_FTPS_SIZE, SC_FTPS_SETTYPE, SC_FTPS_GETTYPE, SC_FTPS_SSLCONFIG }SC_FTPS_API_TYPE;

5.5 SC_FTPS_DATA_FLAG

变量名	SC_FTPS_DATA_FLAG
变量定义	typedef enum{ SC_DATA_COMPLETE, SC_DATA_RESUME }SC_FTPS_DATA_FLAG;

5.6 SC_FTPS_DEMO_TYPE

变量名	SC_FTPS_DEMO_TYPE

变量定义	typedef enum{
	//初始化 FTP
	SC_FTPS_DEMO_INIT = 1,
	//登录 FTP
	SC_FTPS_DEMO_LOGIN = 2,
	//注销登录
	SC_FTPS_DEMO_LOGOUT = 3,
	//获取 FTPS 服务器上的当前目录
	SC_FTPS_DEMO_GETDIR = 4,
	//列出在 FTP(S) 服务器上指定目录中的项目

```
SC_FTPS_DEMO_LIST                       = 5,
//改变 FTP(S)服务器上的当前目录
SC_FTPS_DEMO_CHANGEDIR                  = 6,
//创建一个新目录
SC_FTPS_DEMO_CREATDIR                   = 7,
//删除 FTPS 服务器上的文件
SC_FTPS_DEMO_DELETEFILE                 = 8,
//删除 FTPS 服务器上的一个目录
SC_FTPS_DEMO_DELETEDIR                  = 9,
//获取 FTP(S)服务器上的文件大小
SC_FTPS_DEMO_SIZE                       = 10,

//从 FTP(S)  服务器下载文件到模块

SC_FTPS_DEMO_DOWNLOADFILE               = 11,

//从 FTP(S)服务器获取文件到缓冲区

SC_FTPS_DEMO_DOWNLOADFILETOBUFFER   = 12,
//将文件上传到 FTP(S)服务器模块
SC_FTPS_DEMO_UPLOADFILE                 = 13,
//停止 FTP(S)服务
SC_FTPS_DEMO_DEINIT                     = 14,
//设置 FTPS 会话的 SSL 上下文 id
SC_FTPS_DEMO_SSLCONFIG                  =15

    //退出 FTP 服务选项
    SC_FTPS_DEMO_MAX                        = 99
}SC_FTPS_DEMO_TYPE;
```