杭州海康威视数字技术股份有限公司

EHome 存储服务器组件集成

密级级别：[外部公开]
生效时间： 2018 11 月 7 日

目录

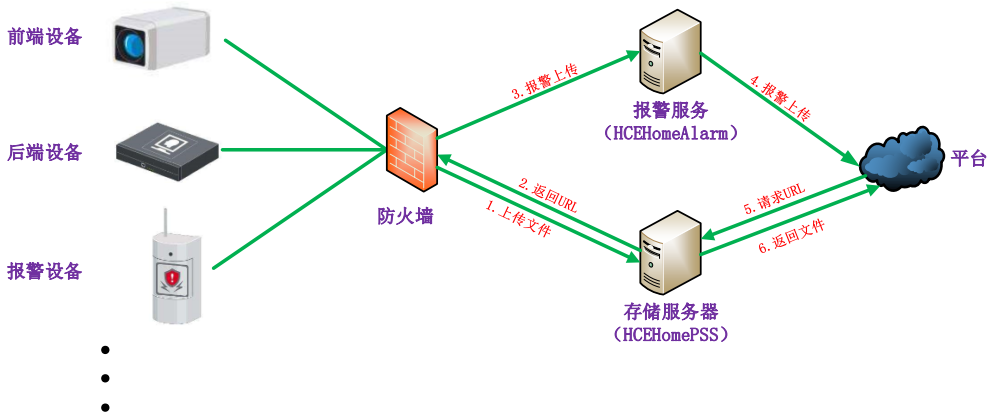
1. 概述

EHome 存储服务器组件是 EhomeSDK 库的一部分，存储服务器组件支持 EHome 协议的 Tomcat、VRB、KMS、云存储协议四种图片上传、下载协议。用户可通过集成存储服务器组件，实现 EHome 存储服务器和 EHome 存储上传客户端。

2. 业务场景

2.1 设备报警（带图片）上报

本场景发生在设备需要将带图片的报警，通过 EHomeSDK 上传至平台。此时，设备须先将图片上传至存储服务器，获取到图片 URL 或图片内容信息附在报警内容中（报警内容为 XML/Json 格式）,设备再将包含了图片信息的报警，通过 EHomeSDK 上传至平台。平台在接收到报警后，依据图片 URL 从存储服务器下载图片。该场景的网络拓扑图如下：

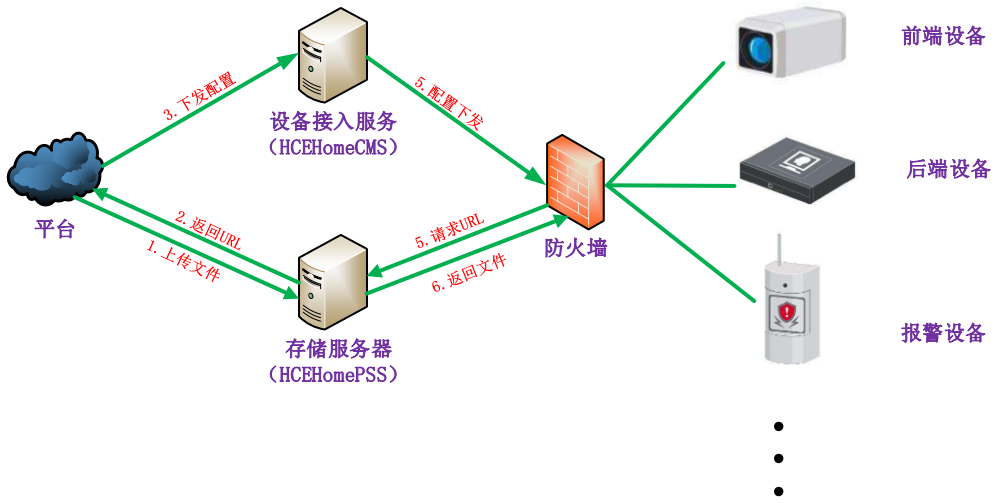


注：VRB、KMS、云存储协议 3 种协议中，设备上传图片至存储服务器，存储服务器返回图片 URL 给设备，Tomcat 协议中，存储服务器返回成功或失败的结果给设备，通过回调函数，将图片 URL 及图片信息传给集成应用程序，集成应用程序须将该信息传给平台。

2.2 平台下发图片至设备

本场景发生在平台需要将图片（如人脸图片），通过 EHomeSDK 下发至设备。此时，

平台须先将图片上传至存储服务器，获取到图片 URL 或图片内容信息附在配置参数中，平台再将包含了图片信息的配置参数，通过 EHomeSDK 下发给设备。设备在接收到配置信息后，依据图片 URL 从存储服务器下载图片。该场景的网络拓扑图如下：



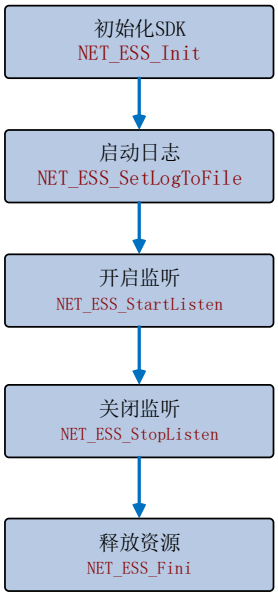
3. 存储服务器集成

3.1 功能介绍

本功能实现存储服务器的搭建，支持设备/平台通过 Tomcat、VRB、KMS、EHome5.。0 存储协议四种协议，完成图片上传和下载操作。

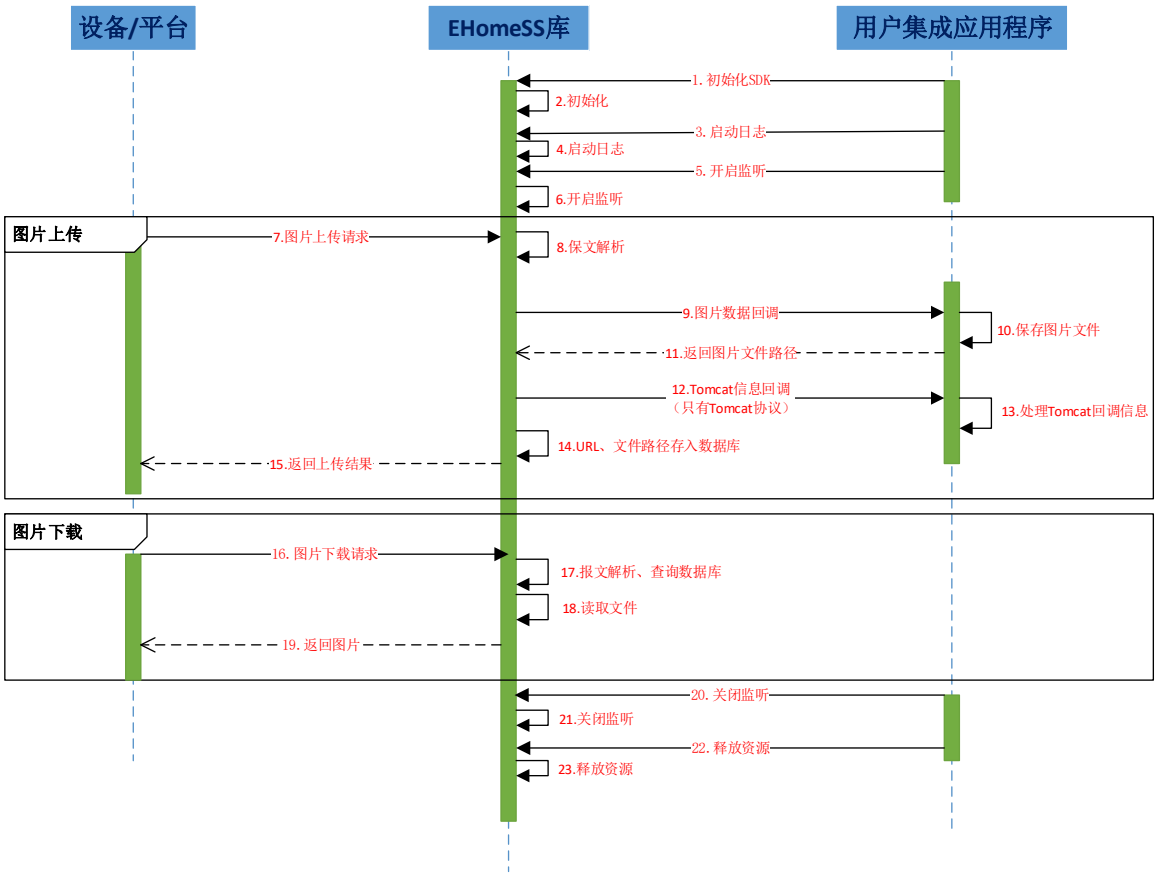
3.2 接口调用流程图

接口调用流程图如下：



3.3 时序图

存储服务器的运行时序图如下：



3.4 示例代码

```
#include <stdio.h>
#include <iostream>
#include "../incCn/HCEHomeSS.h"

#define SS_STORAGE_PATH "C:\\Picture" //文件保存路径
#define PIC_URI_LEN 128 //图片URI长度

//信息回调函数
int SS_Message_Callback(LONG iHandle, NET_EHOME_SS_MSG_TYPE enumType
	, void *pOutBuffer, DWORD dwOutLen, void *pInBuffer, DWORD dwInLen, void *pUser)
{
	if (enumType == NET_EHOME_SS_MSG_TOMCAT)
	{//Tomcat信息回调

		LPNET_EHOME_SS_TOMCAT_MSG pTomcatMsg = (LPNET_EHOME_SS_TOMCAT_MSG)pOutBuffer;
		char szPicUri[PIC_URI_LEN * 4] = { 0 };
		for (int i = 0; i < pTomcatMsg->dwPicNum; i++)
		{
			sprintf(szPicUri + i * PIC_URI_LEN, "%s"
				, pTomcatMsg->pPicURLs + (i * MAX_URL_LEN_SS));
		}
		char szUrlHead[PIC_URI_LEN] = { 0 };
		memcpy(szUrlHead, pTomcatMsg->szDevUri, 128);
		int picNum = pTomcatMsg->dwPicNum;
		char szMsg[5 * PIC_URI_LEN] = { 0 };
		sprintf(szMsg, "tomcat:uri[%s],picNum:[%d] ,picInfo:[%s]"
			, szUrlHead, picNum, szPicUri);
		printf(szMsg);

		FILE* pFile = fopen("C:/Picture/tomcatOutput.txt", "w+");
		if (pFile != NULL)
		{
			for (int i = 0; i < pTomcatMsg->dwPicNum; i++)
			{
				fwrite(szPicUri + i * PIC_URI_LEN, 1, strlen(szPicUri + i * PIC_URI_LEN), pFile);
				fwrite("\n", 1, 1, pFile);
			}
			fclose(pFile);
		}
	}
	else if (enumType == NET_EHOME_SS_MSG_KMS_USER_PWD)
	{
		*(BOOL*)pInBuffer = 1;
	}
	else if (enumType == NET_EHOME_SS_MSG_CLOUD_AK)
	{
		strncpy((char*)pInBuffer, "5e998dbbafb44ca783099afcdead40fa7A3Vf7Fh", dwInLen);
	}
	return TRUE;
}

//文件保存回调函数
int SS_Storage_Callback(LONG iHandle, const char* pFileName, void *pFileBuf, DWORD dwFileLen, char *pFilePath,
void *pUser)
{
	if (pFileName == NULL || pFileBuf == NULL || dwFileLen == 0)
	{
		return FALSE;
	}

	//判断文件路径是否存在,不存在创建文件夹
	if (!PathIsDirectory(SS_STORAGE_PATH))
	{
		if (!CreateDirectory(SS_STORAGE_PATH, NULL))
		{
			return FALSE;
		}
	}
```

```
    //文件保存
    char strFilePath[MAX_PATH] = { 0 };
    sprintf(strFilePath,    "%s\\%s", SS_STORAGE_PATH, pFileName);

    FILE* pFile = fopen(strFilePath, "wb+");
    if (pFile == NULL)
    {
        return FALSE;
    }

    uint dwWriteLen = fwrite(pFileBuf, 1, dwFileLen, pFile);

    fclose(pFile);
    if (dwWriteLen != dwFileLen)
    {
        return FALSE;
    }

    strncpy(pFilePath, strFilePath, 259);
    return TRUE;
}

int main()
{
    //初始化SDK
    NET_ESS_Init();
    //启动日志
    NET_ESS_SetLogToFile(3, "C:/EHomeSdkLog/", TRUE);

    //开启监听
    NET_EHOME_SS_LISTEN_PARAM struSSListenParam = { 0 };
    memcpy(struSSListenParam.struAddress.szIP, "10.8.97.60", strlen("10.8.97.60"));
    struSSListenParam.struAddress.wPort = 8080;
    memcpy(struSSListenParam.szKMS_UserName, "test", strlen("test"));
    memcpy(struSSListenParam.szKMS_Password, "12345", strlen("12345"));
    memcpy(struSSListenParam.szAccessKey, "test", strlen("test"));
    memcpy(struSSListenParam.szSecretKey, "12345", strlen("12345"));
    struSSListenParam.fnSSMsgCb = SS_Message_Callback;
    struSSListenParam.fnSStorageCb = SS_Storage_Callback;
    struSSListenParam.pUserData = NULL;
    long m_lSSHandle = NET_ESS_StartListen(&struSSListenParam);
    if (-1 == m_lSSHandle)
    {
        printf("NET_ESS_StartListen Failed, port：%d", struSSListenParam.struAddress.wPort);
    }
    else
    {
        printf("NET_ESS_StartListen succ port：%d", struSSListenParam.struAddress.wPort);
    }

    char cTmp = '\0';
    do
    {
        printf("Input q to exit!");
        cTmp = getchar();
    }while(cTmp != 'q');

    //关闭监听
    NET_ESS_StopListen(m_lSSHandle);
    //释放资源
    NET_ESS_Fini();

    return 0;
}
```

3.5 Demo 展示

图片服务器-上传、下载录像.mp4

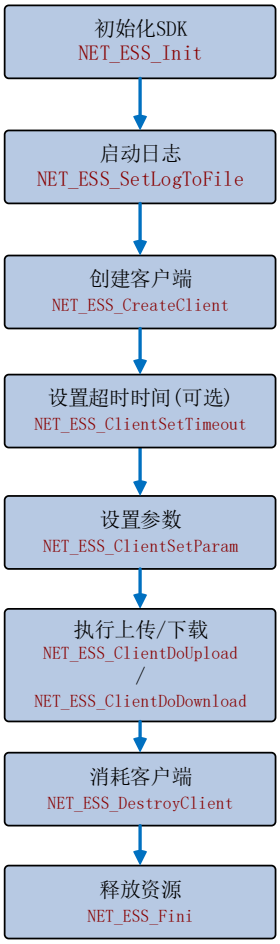
4. 文件上传、下载客户端集成

4.1 功能介绍

本功能实现向存储服务器上传、下载图片的客户端，支持平台通过 Tomcat、VRB、KMS、云存储协议四种协议，完成向图片服务器上传图片。

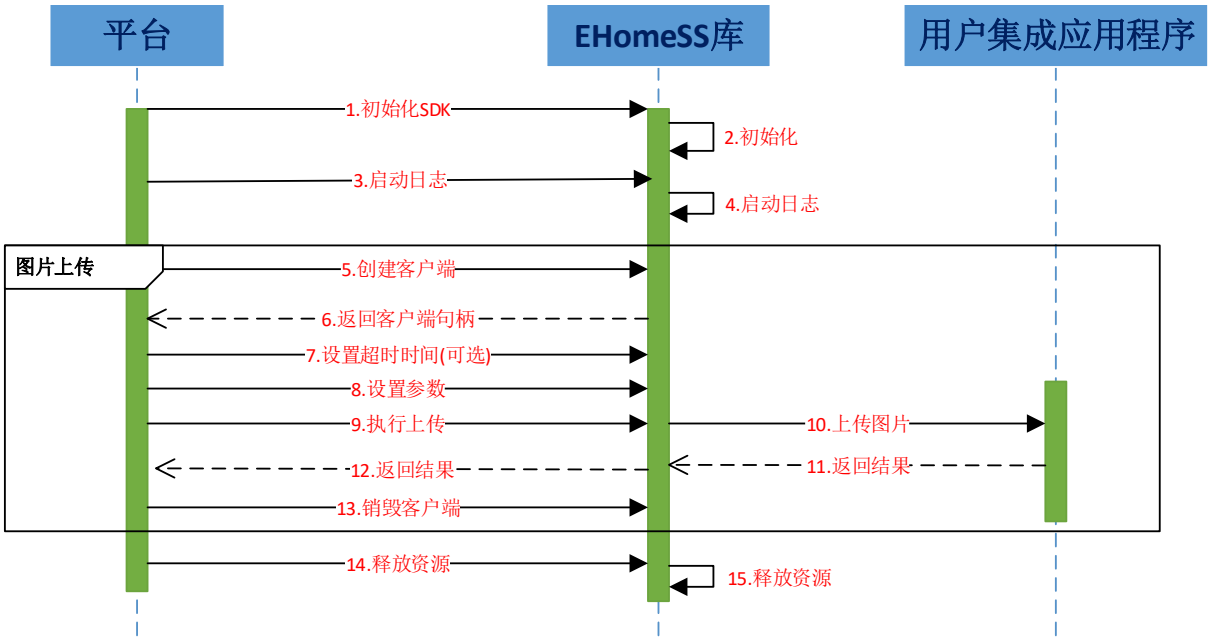
4.2 接口调用流程图

接口调用流程图如下：

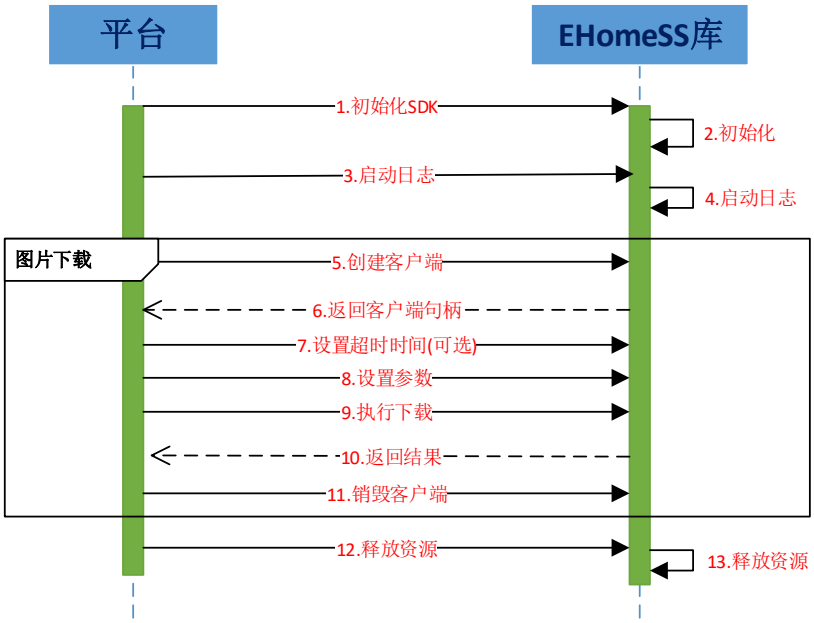


4.3 内部时序图

文件上传客户端的运行时序图如下：



图片下载客户端的运行时序图如下：



4.4 示例代码

```
#include <stdio.h>
#include <iostream>
#include "../incCn/HCEHomeSS.h"

int main()
```

```
{
    //初始化SDK
    NET_ESS_Init();
    //启动日志
    NET_ESS_SetLogToFile(3, "C:/EHomeSdkLog/", TRUE);

    NET_EHOME_SS_CLIENT_PARAM struClientParam;
    memset(&struClientParam, 0, sizeof(struClientParam));
    struClientParam.enumType = NET_EHOME_SS_CLIENT_TYPE_TOMCAT;
    memcpy(struClientParam.struAddress.szIP, "10.8.97.60", strlen("10.8.97.60"));
    struClientParam.struAddress.wPort = 8080;

    //Tomcat图片上传示例代码
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lssClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_FILE_PATH_PARAM_NAME, "C:\\Picture\\1.jpg");

        char szUrl[MAX_URL_LEN_SS] = { 0 };
        NET_ESS_ClientDoUpload(lSSClientHandle, szUrl, MAX_URL_LEN_SS - 1);

        NET_ESS_DestroyClient(lSSClientHandle);
    }

    //VRB图片上传示例代码
    struClientParam.enumType = NET_EHOME_SS_CLIENT_TYPE_VRB;
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lSSClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);

        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_FILE_PATH_PARAM_NAME, "C:\\Picture\\1.jpg");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_VRB_FILENAME_CODE, "filename=1.jpg&code=");

        NET_ESS_ClientDoUpload(lSSClientHandle, szUrl, MAX_URL_LEN_SS - 1);

        NET_ESS_DestroyClient(lSSClientHandle);
    }

    //Tomcat VRB图片下载示例代码
    struClientParam.enumType = NET_EHOME_SS_CLIENT_TYPE_VRB;
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lSSClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);

        byte* bFileContent = NULL;
        DWORD dwFileLen = 0;
        if (NET_ESS_ClientDoDownload(lSSClientHandle, szUrl, (void**)&bFileContent, dwFileLen))
        {
            FILE* pFile = fopen("C://Picture//vrb.jpg", "wb+");
            if (pFile != NULL)
            {
                fwrite(bFileContent, 1, dwFileLen, pFile);
                fclose(pFile);
            }
        }

        NET_ESS_DestroyClient(lSSClientHandle);
    }

    //KMS图片上传示例代码
    struClientParam.enumType = NET_EHOME_SS_CLIENT_TYPE_KMS;
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lSSClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);
```

```
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_FILE_PATH_PARAM_NAME, "C:\\Picture\\1.jpg");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_KMS_USER_NAME, "test");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_KMS_PASSWIRD, "12345");

        NET_ESS_ClientDoUpload(lSSClientHandle, szUrl, MAX_URL_LEN_SS - 1);

        DWORD dwErr = NET_ESS_GetLastError();

        NET_ESS_DestroyClient(lSSClientHandle);
    }

    //KMS图片下载示例代码
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lSSClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);

        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_KMS_USER_NAME, "test");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_KMS_PASSWIRD, "12345");

        byte* bFileContent = NULL;
        DWORD dwFileLen = 0;
        if (NET_ESS_ClientDoDownload(lSSClientHandle, szUrl, (void**)&bFileContent, dwFileLen))
        {
            FILE* pFile = fopen("C://Picture//kms.jpg", "wb+");
            if (pFile != NULL)
            {
                fwrite(bFileContent, 1, dwFileLen, pFile);
                fclose(pFile);
            }
        }

        DWORD dwErr = NET_ESS_GetLastError();

        NET_ESS_DestroyClient(lSSClientHandle);
    }

    //云存储协议图片上传示例代码
    struClientParam.enumType = NET_EHOME_SS_CLIENT_TYPE_CLOUD;
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lSSClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);

        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_FILE_PATH_PARAM_NAME, "C:\\Picture\\1.jpg");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_CLOUD_AK_NAME, "test");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_CLOUD_SK_NAME, "12345");

        NET_ESS_ClientDoUpload(lSSClientHandle, szUrl, MAX_URL_LEN_SS - 1);

        DWORD dwErr = NET_ESS_GetLastError();

        byte* bFileContent = NULL;

        NET_ESS_DestroyClient(lSSClientHandle);
    }

    //云存储协议图片下载示例代码
    struClientParam.enumType = NET_EHOME_SS_CLIENT_TYPE_CLOUD;
    lSSClientHandle = NET_ESS_CreateClient(&struClientParam);
    if (lSSClientHandle >= 0)
    {
        NET_ESS_ClientSetTimeout(lSSClientHandle, 60 * 1000, 60 * 1000);

        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_CLOUD_AK_NAME, "test");
        NET_ESS_ClientSetParam(lSSClientHandle, SS_CLIENT_CLOUD_SK_NAME, "12345");
```

```
            byte* bFileContent = NULL;
            DWORD dwFileLen = 0;
            if (NET_ESS_ClientDoDownload(lSSClientHandle, szUrl, (void**)&bFileContent, dwFileLen))
            {
                FILE* pFile = fopen("C://Picture//cloud.jpg", "wb+");
                if (pFile != NULL)
                {
                    fwrite(bFileContent, 1, dwFileLen, pFile);
                    fclose(pFile);
                }
            }

            NET_ESS_DestroyClient(lSSClientHandle);
        }

        //释放资源
        NET_ESS_Fini();

        return 0;
}
```

5. 附件

5.1 初始化

接口：

NET_DVR_API BOOL CALLBACK NET_ESS_Init()

5.2 资源释放

接口：

NET_DVR_API BOOL CALLBACK NET_ESS_Fini()

5.3 获取错误码

接口：

NET_DVR_API DWORD CALLBACK NET_ESS_GetLastError()

5.4 启动日志

日志等级：

ERROR_LEVEL = 1, DEBUG_LEVEL = 2, INFO_LEVEL = 3

接口：

NET_DVR_API BOOL CALLBACK NET_ESS_SetLogToFile(LONG iLogLevel, const char *strLogDir, BOOL bAutoDel)

5.5 获取版本信息

接口：

```
NET_DVR_API DWORD CALLBACK NET_ESS_GetBuildVersion()
```

5.6 开启监听

监听参数：

```
typedef struct tagNET_EHOME_SS_LISTEN_PARAM
{
    NET_EHOME_IPADDRESS struAddress;  //本地监听信息，IP为0.0.0.0的情况下，默认为本地地址，多个网卡的情况下，默认为从操作系统获取到的第一个
    char szKMS_UserName[MAX_KMS_USER_LEN];//KMS用户名
    char szKMS_Password[MAX_KMS_PWD_LEN];//KMS密码
    EHomeSSStorageCallBack fnSStorageCb; //存储服务器信息存储回调函数
    EHomeSSMsgCallBack      fnPSMsgCb; //存储服务器信息Tomcat回调函数
    char szAccessKey[MAX_CLOUD_AK_SK_LEN]; //云存储协议AK
    char szSecretKey[MAX_CLOUD_AK_SK_LEN]; //云存储协议SK
    void * pUserData;    //用户数据
    BYTE   byRes[64];

}NET_EHOME_SS_LISTEN_PARAM, *LPNET_EHOME_SS_LISTEN_PARAM;
```

信息回调函数：

```
enum NET_EHOME_SS_MSG_TYPE
{
    NET_EHOME_SS_MSG_TOMCAT = 1,          //Tomcat回调函数
    NET_EHOME_SS_MSG_KMS_USER_PWD,        //KMS用户名密码校验
    NET_EHOME_SS_MSG_CLOUD_AK             //云存储协议AK回调

}

typedef struct tagNET_EHOME_SS_TOMCAT_MSG
{
    char szDevUri[MAX_URL_LEN_SS];          //设备请求的URI
    DWORD dwPicNum;                             //图片数
    char* pPicURLs;                         //图片URL,每个URL MAX_URL_LEN_SS长度
    BYTE   byRes[64];

}NET_EHOME_SS_TOMCAT_MSG, *LPNET_EHOME_SS_TOMCAT_MSG

typedef BOOL(CALLBACK *EHomeSSMsgCallBack)(LONG iHandle, NET_EHOME_SS_MSG_TYPE enumType
    , void *pOutBuffer, DWORD dwOutLen, void *pInBuffer, DWORD dwInLen, void *pUser)
```

存储回调函数：

```
typedef BOOL(CALLBACK *EHomeSSStorageCallBack)(LONG iHandle, const char* pFileName, void *pFileBuf, DWORD dwFileLen, char *pFilePath, void *pUser);
```

接口：

```
NET_DVR_API LONG CALLBACK NET_ESS_StartListen(NET_EHOME_SS_LISTEN_PARAM* pSSListenParam)
```

5.7 关闭监听

接口：

```
NET_DVR_API BOOL    CALLBACK NET_ESS_StopListen(LONG lListenHandle)
```

5.8 设置初始化参数

初始化参数类型：

```
enum NET_EHOME_SS_INIT_CFG_TYPE
{
    NET_EHOME_SS_INIT_CFG_SDK_PATH = 1,                 //设置SS组件加载路径（仅Linux版本支持）
    NET_EHOME_SS_INIT_CFG_CLOUD_TIME_DIFF               //设置运存储的请求时间差值,默认15分钟,最小5分钟,
最大60分钟

}
```

接口：

```
NET_DVR_API BOOL NET_ESS_SetSDKInitCfg(NET_EHOME_SS_INIT_CFG_TYPE enumType, void* const lpInBuff)
```

5.9 创建图片上传下载客户端

图片上传下载客户端参数：

```
enum NET_EHOME_SS_CLIENT_TYPE
{
    NET_EHOME_SS_CLIENT_TYPE_TOMCAT = 1,          //Tomcat图片上传客户端
    NET_EHOME_SS_CLIENT_TYPE_VRB,                 //VRB图片上传客户端
    NET_EHOME_SS_CLIENT_TYPE_KMS,                 //KMS图片上传客户端
    NET_EHOME_SS_CLIENT_TYPE_CLOUD                //云存储协议客户端

};

typedef struct tagNET_EHOME_SS_CLIENT_PARAM
{
    NET_EHOME_SS_CLIENT_TYPE enumType; //文件上传客户端类型
    NET_EHOME_IPADDRESS struAddress;      //存储服务器地址
    BYTE    byRes[64];

}NET_EHOME_SS_CLIENT_PARAM, *LPNET_EHOME_SS_CLIENT_PARAM
```

接口：

```
NET_DVR_API LONG CALLBACK NET_ESS_CreateClient(NET_EHOME_SS_CLIENT_PARAM* pClientParam)
```

5.10 设置图片上传下载客户端超时时间

接口：

```
NET_DVR_API BOOL CALLBACK NET_ESS_ClientSetTimeout(LONG lHandle, DWORD dwSendTimeout, DWORD dwRecvTimeout)
```

5.11 设置图片上传下载客户端参数

特殊参数名称:

```
#define SS_CLIENT_FILE_PATH_PARAM_NAME "File-Path"  //图片文件路径
#define SS_CLIENT_VRB_FILENAME_CODE "Filename-Code"  //VRB协议的FilenameCode
#define SS_CLIENT_KMS_USER_NAME "KMS-Username"  //KMS图片服务器用户名
#define SS_CLIENT_KMS_PASSWIRD "KMS-Password"  //KMS图片服务器密码
#define SS_CLIENT_CLOUD_AK_NAME "Access-Key"  //云存储协议AcessKey
#define SS_CLIENT_CLOUD_SK_NAME "Secret-Key"  //云存储协议SecretKey
```

接口：

```
NET_DVR_API BOOL CALLBACK NET_ESS_ClientSetParam(LONG lHandle, const char* strParamName, const char* strParamVal)
```

5.12 图片上传下载客户端执行上传

接口：

```
NET_DVR_API BOOL CALLBACK NET_ESS_ClientDoUpload(LONG lHandle, char* strUrl, LONG dwUrlLen)
```

5.13 图片上传下载客户端执行下载

接口：

```
NET_DVR_API BOOL CALLBACK NET_ESS_ClientDoDownload(LONG lHandle, char* strUrl, void** pFileContent, DWORD& dwContentLen)
```

5.14 销毁客户端

接口：

```
NET_DVR_API BOOL CALLBACK NET_ESS_DestroyClient(LONG lHandle)
```

6. 修订记录

序	变更时间	版本	变更人	审批人	变更说明

号					
1	2018-11-08	V1.0	汪运 15	潘亚东	新建文档
2	2018-12-07	V1.1	汪运 15	潘亚东	1.新增 EHome5.0 存储协议内容
3	2018-12-08	V1.2	汪运 15	潘亚东	1.新增下载客户端内容
4	2019-01-02	V1.3	汪运 15	潘亚东	1.修改 Ehome5.0 存储协议为云存储协议