IPC SDK
Programming User Manual

(version 1.0.0.0)

1 Overview

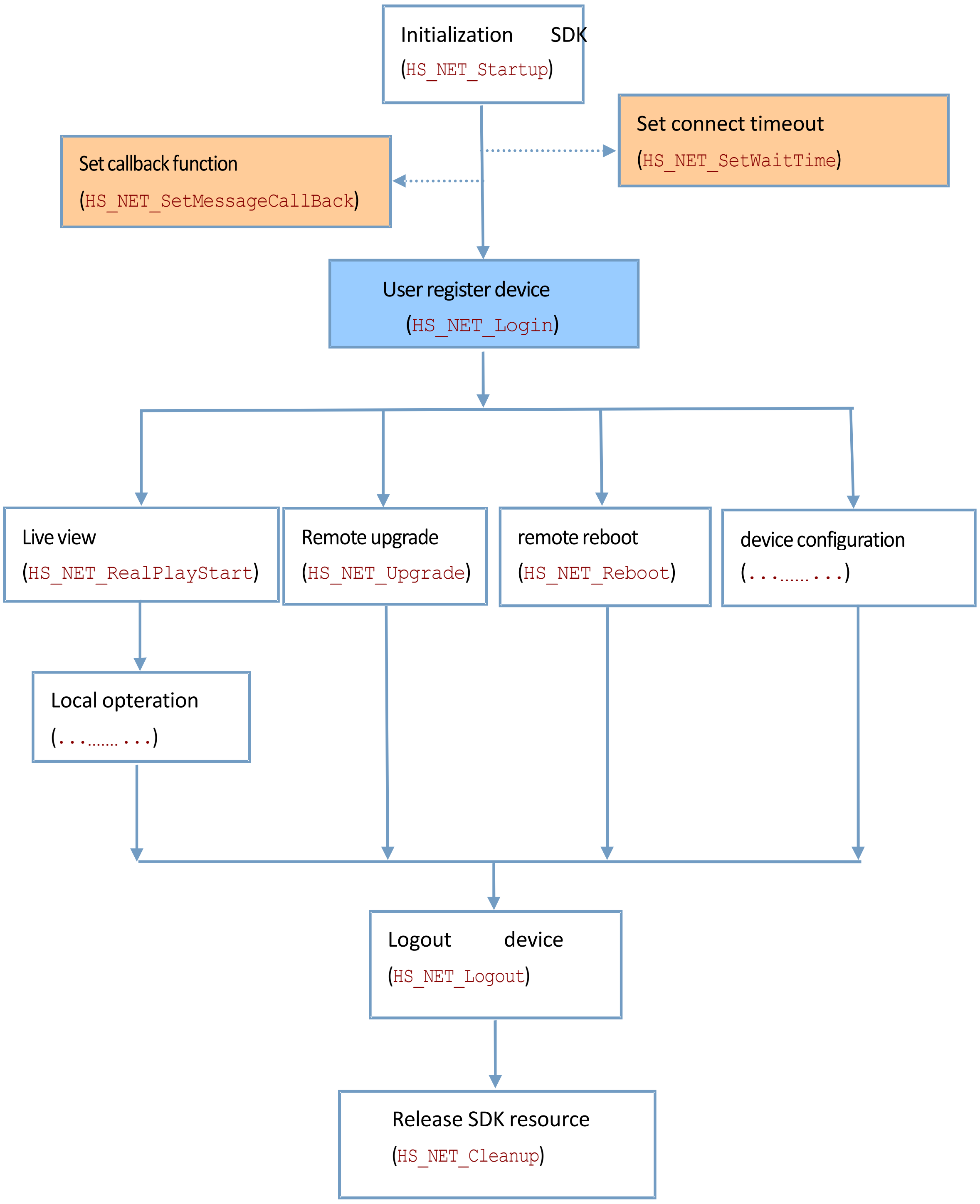
The device network SDK is developed based on private network communication protocol, and it is designed for the remote connection and configuration of IPC.

The functions supported by the SDK

1. Live view, Local operation, log query, decoding call back,etc.
2. Remote upgrade, remotely reboot, remotely shut down, and device configuration (system configuration, alarm configuration, users configuration), etc.

The SDK is used as client. The client connects to the device actively, and then does operation about the device,live view, device configuration and so on.

2 API Calling Procedure



Initialization SDK (HS_NET_Startup)
Set callback function (HS_NET_SetMessageCallBack)
Set connect timeout (HS_NET_SetWaitTime)
User register device (HS_NET_Login)
Live view (HS_NET_RealPlayStart)
Remote upgrade (HS_NET_Upgrade)
remote reboot (HS_NET_Reboot)
device configuration (. . .…… . . .)
Local opteration (. . .……. . . .)
Logout device (HS_NET_Logout)
Release SDK resource (HS_NET_Cleanup)

3 API Description

Remarks： This API is used to initialize SDK. Please call this API before calling any other API.

Parameters：None

Return：Return 0 on success, -1 on failure.

```
int
HS_NET_Startup();
```

Remarks： This API is used to release SDK resource. Please calling it before closing the program.

Parameters：None

Return：Return 0 on success, -1 on failure.

```
void
HS_NET_Cleanup();
```

Remarks：Default timeout of SDK to establish a connection is 3 seconds.

Parameters：

```
[in]  nWaitTime
    Timeout,unit: ms, value range: [500,50000], the
    actual max timeout time is different with
    different system connecting timeout
 [in]  nTryNum
    Connecting attempt times
[in]  nTryInterval
    The time interval of each connection
```

Return：Return 0 on success, -1 on failure.

```
void HS_NET_SetWaitTime(int nWaitTime, int nTryNum, int nTryInterval);
```

Remarks：cbMessageCallBack can't be set to NULL, or it will not receive message.

Parameters：

```
[in]  cbMessageCallBack
    Callback function to receive message
[in]  pUser
    User data
```

Return：None

```
typedef void(CALLBACK *fNetMessageCallBack)(int nLoginId, long msgType, void *pMsgData, void
*pUser);
void HS_NET_SetMessageCallBack(fNetMessageCallBack cbMessageCallBack, void *pUser);
```

Remarks：supports 128 users login at the same time

Parameters：

```
[in] pChaninfo
```

User name, password and so on.

Return： Return ID on success, -1 on failure.

```
int HS_NET_Login(HS_CLIENTINFO *pChaninfo);
```

Remarks：It is suggested to call this API to logout.

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login

Return：Return 0 on success, -1 on failure. int HS_NET_Logout(int nLoginId);

Remarks：The callback function of this API can be set to NULL, and it will not callback, if you want to play by yourself, please set this callback fun, and decode h264 to play.

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login [in]

pChaninfo

Live view parameter, channel is 0, main stream or sub stream, play handle [in]

dataFun

H264 data callback function

[in] pUser

User data

Return：Return ID on success, -1 on failure.

```
typedef void (CALLBACK* fNetAVDataCallBack)(int nRealPlayId, char* pBuf, long nSize,
HS_STREAM_INFO* pBufInfo, void *pUser);
int HS_NET_RealPlayStart(int nLoginId, HS_CLIENTINFO *pChaninfo, fNetAVDataCallBack
dataFun, void *pUser);
```

Remarks：This API is used to stop live view.

Parameters：

[in] nRealPlayId

Live view handle, the return value of HS_NET_RealPlayStart.

Return： Return 0 on success, -1 on failure.

```
int HS_NET_RealPlayStop(int nRealPlayId);
```

Remarks：This API is used to set playing handle or to change playing handle

Parameters：

[in] nRealPlayId

Live view handle, the return value of HS_NET_RealPlayStart. [in]

hWnd

Windows handle

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientSetWnd(int nRealPlayId, HWND hWnd);
```

Remarks： This API is used to fresh window

Parameters：

`[in] nRealPlayId`

`Live view handle, the return value of HS_NET_RealPlayStart.`

Return： Return 0 on success, -1 on failure.

```
int HS_NET_ClientRefreshWnd(int nRealPlayId);
```

Remarks：This API is used to get stream info.

Parameters：

`[in] nRealPlayId`

`Live view handle, the return value of HS_NET_RealPlayStart. [out] pBitRate`

`Bit rate`

`[out] pFrameRate`

`Frame rate`

`[out] pWidth`

`Width of video`

`[out] pHeight`

`Height of video`

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientGetStreamInfo(int nRealPlayId, unsigned long *pBitRate,  unsigned long
*pFrameRate, unsigned long *pWidth, unsigned long *pHeight);
```

Remarks：This API is used to open sound under exclusive sound card mode.

Parameters：

`[in] nRealPlayId`

`Live view handle, the return value of HS_NET_RealPlayStart.`

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientPlayAudioStart(int nRealPlayId);
```

Remarks：This API is used to close sound on monopolistic sound card mode.

Parameters：

`[in] nRealPlayId`

`Live view handle, the return value of HS_NET_RealPlayStart.`

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientPlayAudioStop(int nRealPlayId);
```

Remarks：This API is used to capture a frame and save to JPEG file.

Parameters：

`[in] nRealPlayId`

`Live view handle, the return value of HS_NET_RealPlayStart. [in] filename`

`URL of JPEG file`

[in] type

Only surpport :HS_NET_CAPTURE_JPEG

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientCapturePicture(int nRealPlayId, const char* filename, HsNetCaptureType type
= HS_NET_CAPTURE_JPEG);
```

Remarks：This API is used to start the manual record. If dwDurationSeconds vale is none-zero, will callback MSG_RECORD_PACKET_FINISH by endle of recording；or must to use HS_NET_ClientStopRecord to stop recording.

Parameters：

[in] nRealPlayId

Live view handle, the return value of HS_NET_RealPlayStart. [in]

filename

URL of record file

[in] dwDurationSeconds

Record time, 0 is to keep recording， or recording the this time， unit:second, default is 0.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientStartRecord(int nRealPlayId, const char* filename, DWORD
dwDurationSeconds=0);
```

Remarks：This API is used to stop recording.

Parameters：

[in] nRealPlayId

Live view handle, the return value of HS_NET_RealPlayStart.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientStopRecord(int nRealPlayId);
```

Remarks：This API is used to start another record. If dwDurationSeconds vale is none-zero, will callback MSG_RECORD_PACKET_FINISH by endle of recording；or must to use HS_NET_ClientStopRecord_Another to stop recording.

Parameters：

[in] nRealPlayId

Live view handle, the return value of HS_NET_RealPlayStart. [in]

filename

URL of record file

[in] dwDurationSeconds

Record time, 0 is to keep recording， or recording the this time， unit:second, default is 0.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientStartRecord_Another(int nRealPlayId, const char* filename, DWORD
dwDurationSeconds=0);
```

Remarks：This API is used to stop another recording.

Parameters：

[in] nRealPlayId

    Live view handle, the return value of HS_NET_RealPlayStart.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientStopRecord_Another(int nRealPlayId);
```

Remarks：This API is used to overlay characters or image on preview DC

Parameters：

[in] nRealPlayId

    Live view handle, the return value of HS_NET_RealPlayStart. [in]

dawFun

    Draw callback function.

[in] pUser

    User data.

Return：Return 0 on success, -1 on failure.

```
typedef void (CALLBACK* fNetDrawCallback)(int nRealPlayId,HDC hDC, void *pUser);
int HS_NET_ClientSetDrawCallback(int nRealPlayId, fNetDrawCallback dawFun, void *pUser);
```

Remarks：This API is used to callback yuv420 data

Parameters：

[in] nRealPlayId

    Live view handle, the return value of HS_NET_RealPlayStart. [in]

decfun

    Decode data callback function.

[in] pUser

    User data.

Return：Return 0 on success, -1 on failure.

```
typedef void (CALLBACK* fNetDecodeCallBack)(int nRealPlayId, char *py, char *pu, char *pv,
int ystride, int uvstride, HS_FRAME_INFO *pFrameInfo, void *pUser);
int HS_NET_ClientSetDecodeCallBack( int nRealPlayId, fNetDecodeCallBack decfun, void
*pUser);
```

Remarks：This API is used to cache frames. Default is 0.

Parameters：

[in] nRealPlayId

    Live view handle, the return value of HS_NET_RealPlayStart. [in]

frame

    Cache frames

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientSetRestoreFrame(int nRealPlayId,UINT frame);
```

Remarks：This API is used to digital larger image.

Parameters：

[in] nRealPlayId

Live view handle, the return value of HS_NET_RealPlayStart. [in]

pRect

The actual image of the designated area.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_ClientShowRect(int nRealPlayId, RECT *pRect);
```

Remarks：This API is used to get video display parameters.

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login

[in/out] imageprty

video display parameters.（User specified memory,the following interfaces are the same） Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetImageProperty(int nLoginId, HS_IMAGE_PROPERTY *imageprty);
```

Remarks：This API is used to set video display parameters.

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login [in]

pImageprty

video display parameters.（No changes to the parameters to obtain the return value ,he following interfaces are the same）

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetImageProperty(int nLoginId, HS_IMAGE_PROPERTY *pImageprty);
```

Remarks：This API is used to configure user information.

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login [in]

pUserInfo

User information.

[in] configType

The type of configure.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_UserConfig(int nLoginId, HS_USER_INFOR *pUserInfo, HS_USER_CONFIG_TYPE
configType);
```

Remarks：This API is used to query information of users；（when userInfo is NULL and bufNum is 0，the return vale is user numbers. Using this vale to malloc memory and call this API again. The following interfaces are the same）

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login

[in/out] userInfo

A pointer that number of HS_USER_INFOR memory to receive [in]

bufNum

The number of HS_USER_INFOR to receive

[in/out] nCount

actual number of HS_USER_INFOR is returned , when a query is greater than the number of bufNum, only to return bufNum

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetUserInfo(int nLoginId, HS_USER_INFOR userInfo[], int bufNum, int *nCount);
```

Remarks：This API is used to get the configuration parameters of motion detection. (currently only supports HS_AREA_DELIMIT_22_18 block mode; if data structures has channel fields, being filled 0.The following interfaces are the same)

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login

[in/out] pMDInfo

parameters of motion detection.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetMDInfo(int nLoginId, HS_MOTIONDETECTION_EX_PROPERTY* pMDInfo);
```

Remarks：This API is used to get the configuration parameters of motion detection. (Note: when motion detection enbaled, Only with alarm setting interface configuration enabled, will can be receive alarm messages. Alarm setting interface is the master switch)

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login [in]

pMDInfo

parameters of motion detection.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetMDInfo(int nLoginId, HS_MOTIONDETECTION_EX_PROPERTY* pMDInfo);
```

Remarks：This API is used to get parameters of video cover.(currently only supports up to three area)

Parameters：

[in] nLoginId

User ID, the return value of HS_NET_Login

[in/out] pVCInfo

parameters of video cover.

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetVideoCover(int nRealPlayId, HS_VIDEOCOVER_PROPERTY* pVCInfo);
```

Remarks：This API is used to set parameters of video cover.

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

pVCInfo

    parameters of video cover.

Return：Return 0 on success, -1 on failure.

`int HS_NET_SetVideoCover(int nRealPlayId, HS_VIDEOCOVER_PROPERTY* pVCInfo);`

Remarks：This API is used to get coding range of video .('@' Separator single stream and dual-stream ';' delimiter current stream encoding type and resolution ',' separator main stream and sub stream)

ex ： H264:1920x1080;H264:1280x960;H264:1280x720;H264:720x576 @H264:1920x1080,H264:720x576;H264:1920x1080,H264:720x480; H264:1280x720,H264:720x576;H264:1280x720,H264:352x288

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] encscope

    coding range of video.

Return：Return 0 on success, -1 on failure.

`int HS_NET_GetEncodingScope(int nLoginId, HS_ENCODING_SCOPE *encscope);`

Remarks：This API is used to set encoding parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

clientParam

    A pointer that number of HS_ENCODING_PROPERTY memory to receive [in]

bufNum

    The number of HS_ENCODING_PROPERTY to set

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetEncodingProperty(int nLoginId, HS_ENCODING_PROPERTY clientParam[], int
bufNum);
```

Remarks：This API is used to get encoding parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

nStreamType

    Stream type, filled with one of enum HS_STREAM_TYPE [in/out]

clientParam

    A pointer that number of HS_ENCODING_PROPERTY memory to receive [in]

bufNum

    The number of HS_ENCODING_PROPERTY to receive

[in/out] nCount

    actual number of HS_ENCODING_PROPERTY is returned , when a query is greater than the number of bufNum, nly to return bufNum

Return：Return 0 on success, -1 on failure.

```
int  HS_NET_GetEncodingProperty(int  nLoginId,  int  nStreamType,  HS_ENCODING_PROPERTY clientParam[], int bufNum, int *nCount);
```

Remarks：This API is used to set network parameter

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

pNetInfo

    Network parameter

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetNetParam(int nLoginId, HS_NET_PROPERTY *pNetInfo);
```

Remarks：This API is used to get network parameter

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] pNetInfo

    Network parameter

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetNetParam(int nLoginId, HS_NET_PROPERTY *pNetInfo);
```

Remarks：This API is used to set time parameter

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

timeInfo

    Time parameter

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetTimeParam(int nLoginId, HS_TIME_INFO *timeInfo);
```

Remarks：This API is used to get time parameter

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] timeInfo

    Time parameter

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetTimeParam(int nLoginId, HS_TIME_INFO *timeInfo);
```

Remarks：This API is used to device upgrades. (The progress and status messages via callback function)

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

filename

    Url of file

Return：Return 0 on success, -1 on failure.

```
int HS_NET_Upgrade(int nLoginId, const char* filename);
```

Remarks：This API is used to reboot the device.

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

Return：Return 0 on success, -1 on failure. int

HS_NET_Reboot(int nLoginId);

Remarks：This API is used to reset the device.

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

Return：Return 0 on success, -1 on failure. int

HS_NET_Reset(int nLoginId);

Remarks： This API is used to get log information.(only time filtering support, there must be time interval)

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

loggrepinfo

    Time filtering

[in/out] logeventinfo

    A pointer that number of HS_LOG_EVENT_INFO memory to receive [in]

bufNum

    Number of HS_LOG_EVENT_INFO to receive

[in/out] nCount

    actual number of HS_LOG_EVENT_INFO is returned , when a query is greater than the number of bufNum, only to return bufNum

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetLogFile(int nLoginId, HS_LOG_GREP_INFO *loggrepinfo, HS_LOG_EVENT_INFO
logeventinfo[], int bufNum, int *nCount);
```

Remarks：This API is used to get SMTP parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in/out] smtpprty

    SMTP parameters

Return: Return 0 on success, -1 on failure.

```
int HS_NET_GetSmtp(int nLoginId, HS_SMTP_PROPERTY *smtpprty);
```

Remarks: This API is used to set SMTP parameters

Parameters:

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

smtpprty

    SMTP parameters

Return: Return 0 on success, -1 on failure.

```
int HS_NET_SetSmtp(int nLoginId, HS_SMTP_PROPERTY *smtpprty);
```

Remarks: This API is used to get FTP parameters

Parameters:

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] ftpprty

    FTP parameters

Return: Return 0 on success, -1 on failure.

```
int HS_NET_GetFtp(int nLoginId, HS_FTP_PROPERTY *ftpprty);
```

Remarks: This API is used to set FTP parameters

Parameters:

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

ftpprty

    FTP parameters

Return: Return 0 on success, -1 on failure.

```
int HS_NET_SetFtp(int nLoginId, HS_FTP_PROPERTY *ftpprty);
```

Remarks: This API is used to get DDNS parameters

Parameters:

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] ddnsprty

    DDNS parameters

Return: Return 0 on success, -1 on failure.

```
int HS_NET_GetDDNS(int nLoginId, HS_DDNSSERVER_PROPERTY *ddnsprty);
```

Remarks: This API is used to set DDNS parameters

Parameters:

[in] nLoginId

```
    User ID, the return value of HS_NET_Login [in]
ddnsprty
    DDNS parameters
```

Return: Return 0 on success, -1 on failure.

```
int HS_NET_SetDDNS(int nLoginId, HS_DDNSSERVER_PROPERTY *ddnsprty);
```

Remarks: This API is used to get P2P parameters

Parameters:

```
[in]  nLoginId
    User  ID,  the  return  value  of  HS_NET_Login
[in/out] p2pprty
    P2P parameters
```

Return: Return 0 on success, -1 on failure.

```
int HS_NET_GetP2P(int nLoginId, HS_P2P_PROPERTY *p2pprty);
```

Remarks: This API is used to P2P parameters

Parameters:

```
[in]  nLoginId
    User ID, the return value of HS_NET_Login [in]
p2pprty
    P2P parameters
```

Return: Return 0 on success, -1 on failure.

```
int HS_NET_SetP2P(int nLoginId, HS_P2P_PROPERTY *p2pprty);
```

Remarks: This API is used to get P2P connecting status

Parameters:

```
[in]  nLoginId
    User  ID,  the  return  value  of  HS_NET_Login
[in/out] p2pinfo
    P2P connecting status parameter
```

Return: Return 0 on success, -1 on failure.

```
int HS_NET_GetP2PInfo(int nLoginId, HS_P2P_INFO *p2pinfo);
```

Remarks: This API is used to get OSD parameters

Parameters:

```
[in]  nLoginId
    User  ID,  the  return  value  of  HS_NET_Login
[in/out] displayprty
    OSD parameter
```

Return: Return 0 on success, -1 on failure.

```
int HS_NET_GetOSD(int nLoginId, HS_DISPLAY_PROPERTY *displayprty);
```

Remarks: This API is used to set OSD parameters

Parameters:

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

displayprty

    OSD parameters

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetOSD(int nLoginId, HS_DISPLAY_PROPERTY *displayprty);
```

Remarks：This API is used to get audio parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] adprty

    Audio parameters

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetAudio(int nLoginId, HS_AUDIO_PROPERTY *adprty);
```

Remarks：This API is used to set audio parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

adprty

    Audio parameters

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetAudio(int nLoginId, HS_AUDIO_PROPERTY *adprty);
```

Remarks：This API is used to get device information.

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] devinfo

    Device information

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetDeviceInfo(int nLoginId, HS_DEV_INFO *devinfo);
```

Remarks：This API is used to alter device name

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

devinfo

    Only can alter name field

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetDeviceInfo(int nLoginId, HS_DEV_INFO *devinfo);
```

Remarks：This API is used to get the scheduled reboot parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login

[in/out] sysreboot

    Scheduled reboot parameters

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetTimeReboot(int nLoginId, HS_SYSREBOOT_PROPERTY *sysreboot);
```

Remarks：This API is used to set the scheduled reboot parameters

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

sysreboot

    Scheduled reboot parameters

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetTimeReboot(int nLoginId, HS_SYSREBOOT_PROPERTY *sysreboot);
```

Remarks： This API is used to get protocol parameters.(currently only supports onvif whether to open authentication and HTTP port settings)

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

agrType

    Protocol type, one of enum HS_AGREEMENT_TYPE

[in/out] agreementPrty

    A pointer that number of HS_AGREEMENT_PROPERTY memory to receive [in]

bufNum

    Number of HS_AGREEMENT_PROPERTY to receive

[in/out] nCount

    actual number of HS_AGREEMENT_PROPERTY is returned , when a query is greater than the number of bufNum,   only to return bufNum

Return：Return 0 on success, -1 on failure.

```
int  HS_NET_GetAgreementProperty(int  nLoginId,  int  agrType,  HS_AGREEMENT_PROPERTY
agreementPrty[], int bufNum, int *nCount);
```

Remarks：This API is used to set protocol parameters.

Parameters：

[in] nLoginId

    User ID, the return value of HS_NET_Login [in]

agreementPrty

    Protocol parameter

[in] bufNum

    Number of HS_AGREEMENT_PROPERTY to set

Return：Return 0 on success, -1 on failure.

```
int  HS_NET_SetAgreementProperty(int  nLoginId,  HS_AGREEMENT_PROPERTY  agreementPrty[],  int
bufNum);
```

Remarks：This API is used to get support alarm range
Parameters：

```
[in]  nLoginId
     User  ID,  the  return  value  of  HS_NET_Login
[in/out] pAlarmScope
     Alram range
```

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetAlarmScopeV2(int nLoginId, HS_ALARM_SCOPE_V2 *pAlarmScope);
```

Remarks：This API is used to get alarm parameters.
Parameters：

```
[in]  nLoginId
     User  ID,  the  return  value  of  HS_NET_Login
[in/out] pAlarmPrty
     Alarm parameters.
```

Return：Return 0 on success, -1 on failure.

```
int HS_NET_GetAlarmPropertyV2(int nLoginId, HS_ALARM_PROPERTY_V2 *pAlarmPrty);
```

Remarks：This API is used to set alarm parameters.
Parameters：

```
[in]  nLoginId
     User ID, the return value of HS_NET_Login [in]
pAlarmPrty
     Alarm parameters.
```

Return：Return 0 on success, -1 on failure.

```
int HS_NET_SetAlarmPropertyV2(int nLoginId, HS_ALARM_PROPERTY_V2 *pAlarmPrty);
```

Remarks: This API is used to start get search device.
Parameters:

```
[in]  SearchDeviceCallback  -- Callback param
       pUser                -- this pointer
```

Return：Return 0 on success, -1 on failure.

```
typedef void (CALLBACK* fSearchDeviceCallback)(long msgType/*HS_SEARCH_TYPE*/, void
*pMsgData/*HS_STREAM_INFO*/, void *pUser);
int HS_NET_StartSearchDevice(fSearchDeviceCallback SearchDeviceCallback, void *pUser);
```

Remarks: This API is used to stop get search device.
Return：Return 0 on success, -1 on failure.

```
int HS_NET_StopSearchDevice();
```