SADP SDK

Developer Guide

Legal Information

About this Document

- This Document includes instructions for using and managing the Product. Pictures, charts, images and all other information hereinafter are for description and explanation only. Unless otherwise agreed, Hangzhou Hikvision Digital Technology Co., Ltd. or its affiliates (hereinafter referred to as "Hikvision") makes no warranties, express or implied.
- Please use this Document with the guidance and assistance of professionals trained in supporting the Product.

Acknowledgment of Intellectual Property Rights

- Hikvision owns the copyrights and/or patents related to the technology embodied in the Products described in this Document, which may include licenses obtained from third parties.
- Any part of the Document, including text, pictures, graphics, etc., belongs to Hikvision. No part of this Document may be excerpted, copied, translated, or modified in whole or in part by any means without written permission.
- HIKVISION and other Hikvision's trademarks and logos are the properties of Hikvision in various jurisdictions.
- Other trademarks and logos mentioned are the properties of their respective owners.

LEGAL DISCLAIMER

- TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS DOCUMENT AND THE PRODUCT DESCRIBED, WITH ITS HARDWARE, SOFTWARE AND FIRMWARE, ARE PROVIDED "AS IS" AND "WITH ALL FAULTS AND ERRORS". HIKVISION MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, MERCHANTABILITY, SATISFACTORY QUALITY, OR FITNESS FOR A PARTICULAR PURPOSE. THE USE OF THE PRODUCT BY YOU IS AT YOUR OWN RISK. IN NO EVENT WILL HIKVISION BE LIABLE TO YOU FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, OR INDIRECT DAMAGES, INCLUDING, AMONG OTHERS, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, OR LOSS OF DATA, CORRUPTION OF SYSTEMS, OR LOSS OF DOCUMENTATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY, OR OTHERWISE, IN CONNECTION WITH THE USE OF THE PRODUCT, EVEN IF HIKVISION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR LOSS.
- YOU ACKNOWLEDGE THAT THE NATURE OF THE INTERNET PROVIDES FOR INHERENT SECURITY RISKS, AND HIKVISION SHALL NOT TAKE ANY RESPONSIBILITIES FOR ABNORMAL OPERATION, PRIVACY LEAKAGE OR OTHER DAMAGES RESULTING FROM CYBER-ATTACK, HACKER ATTACK, VIRUS INFECTION, OR OTHER INTERNET SECURITY RISKS; HOWEVER, HIKVISION WILL PROVIDE TIMELY TECHNICAL SUPPORT IF REQUIRED.
- YOU AGREE TO USE THIS PRODUCT IN COMPLIANCE WITH ALL APPLICABLE LAWS, AND YOU ARE SOLELY RESPONSIBLE FOR ENSURING THAT YOUR USE CONFORMS TO THE APPLICABLE LAW.

ESPECIALLY, YOU ARE RESPONSIBLE, FOR USING THIS PRODUCT IN A MANNER THAT DOES NOT INFRINGE ON THE RIGHTS OF THIRD PARTIES, INCLUDING WITHOUT LIMITATION, RIGHTS OF PUBLICITY, INTELLECTUAL PROPERTY RIGHTS, OR DATA PROTECTION AND OTHER PRIVACY RIGHTS. YOU SHALL NOT USE THIS PRODUCT FOR ANY PROHIBITED END-USES, INCLUDING THE DEVELOPMENT OR PRODUCTION OF WEAPONS OF MASS DESTRUCTION, THE DEVELOPMENT OR PRODUCTION OF CHEMICAL OR BIOLOGICAL WEAPONS, ANY ACTIVITIES IN THE CONTEXT RELATED TO ANY NUCLEAR EXPLOSIVE OR UNSAFE NUCLEAR FUEL-CYCLE, OR IN SUPPORT OF HUMAN RIGHTS ABUSES.

- IN THE EVENT OF ANY CONFLICTS BETWEEN THIS DOCUMENT AND THE APPLICABLE LAW, THE LATTER PREVAILS.

Contents

Chapter 1 Overview

1.1 Introduction

SADP SDK allows you to search for the online Hikvision devices on the same LAN and get their information including the serial No., IP address, port No, etc. After the devices are found, SADP also supports activating devices, editing network parameters, resetting the password, and getting/configuring device parameters. SADP SDK supports all series of Hikvision products.

SDK includes all the files listed below. The SADP services depend on the driver files, thus you need to put sadp.dll and corresponding system driver files in the same directory level. Administration permission is required for the first installation.

Windows SDK	Sadp.h	Head file.
	Sadp.lib	LIB library file.
	Sadp.dll	DLL library file.
	libcrypto-1_1.dll libssl-1_1.dll	OpenSSL libraries.
Linux	libcrypto.so libssl.so	OpenSSL libraries.
Android	libcryptoPrivate.so libsslPrivate.so	OpenSSL libraries.
iOS	libcrypto.a libssl.a	OpenSSL libraries.
Mac	libcrypto.dylib libssl.dylib	OpenSSL libraries.

1.2 Update History

Summary of Changes in Version 3.1.0_Oct., 2023

1. Added a structure about device information ***SADP_SUBNET_DEVICE_INFO*** .
2. Added a structure about parameters of starting SADP service ***SADP_START_PARAM*** .
3. Added a structure about subnet information ***SADP_SUBNET_INFO*** .
4. Added a structure about resetting the password ***SADP_RESET_PARAM_V50*** .
5. Added a structure about device security code ***SADP_SAFE_CODE_V31*** .
6. Added a structure about the encrypted string ***SADP_ENCRYPT_STRING_V31*** .

7. Added a structure about GUID ***SADP_GUID_FILE_V31*** .
8. Added a structure about QR code information ***SADP_QR_CODES_V31*** .
9. Added an API for starting SADP services ***SADP_Start_V50*** .
10. Added an API for searching for specific subnet information ***SADP_InquirySpecificSubnet*** .
11. Added an API for resetting the password ***SADP_ResetPasswd_V50*** .
12. Added a callback function of getting online device information ***PSUBNET_DEVICE_FIND_CALLBACK*** .
13. Added 9 error codes: SADP_NO_PERMISSION (No permission: 1. For Win&Linux, no administrator permission for operating NIC. 2. For Android&IOS, no permission for multicast.), SADP_GET_EXCHANGE_CODE_ERROR (Failed to get the interchange code used for encryption.), SADP_CREATE_RSA_KEY_ERROR (Failed to generate RSA private and public key.), SADP_BASE64_ENCODE_ERROR (Encoding by BASE64 failed.), SADP_BASE64_DECODE_ERROR (Decoding by BASE64 failed.), SADP_AES_ENCRYPT_ERROR (Failed to encrypt the data via AES key.), SADP_PHONE_NOT_SET (Phone number is not set for verification when reset the password by scanning QR code.), SADP_NOENOUGH_BUF (Insufficient buffer size.), SADP_INVALID_SUBNET_IP (Invalid subnet range. The start IP address is greater than the end IP address, or the number of IP addresses exceeded 4096.).

Summary of Changes in Version 3.0.0_Aug., 2021

New document.

Chapter 2 Typical Applications

2.1 Search for Online Devices

SADP services allow you to search for the online Hikvision devices on the same LAN or in different network segments and get their information including the serial No., IP address, port No, etc.

Steps

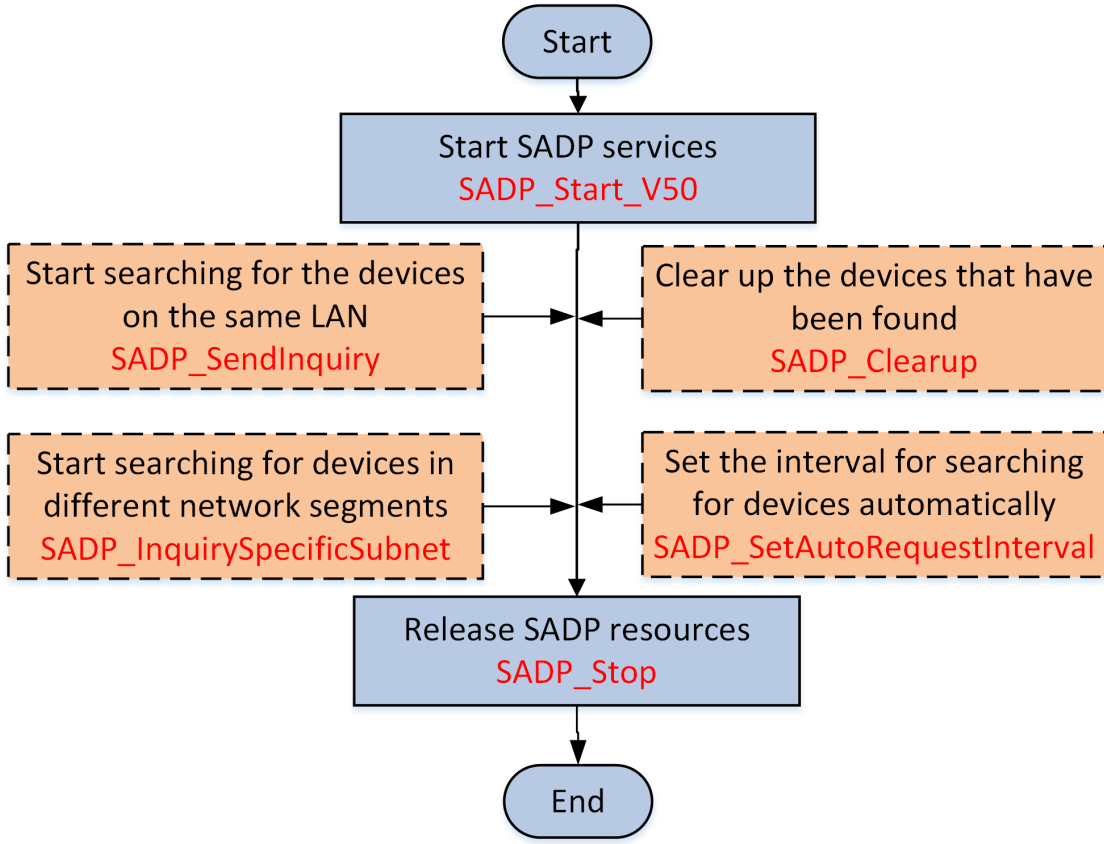

Figure 2-1 API Calling Flow of Searching for Online Devices

1. Call ***SADP_Start_V50*** to start SADP services.

   When a device is found, its information, including the serial No., IP address, port No., etc., will be called back via ***PDEVICE_FIND_CALLBACK_V40*** .

   Note

   When multiple devices are found, information of all devices will be called back (one device at a time).

2. **Optional:** Call ***SADP_InquirySpecificSubnet*** to start searching for devices in different network segments.

3. **Optional:** Call ***SADP_SetAutoRequestInterval*** to set the interval for automatically searching for devices.

Note

By default, the interval is 60 seconds.

4. **Optional:** Call ***SADP_SendInquiry*** to manually start searching for devices.
5. **Optional:** Call ***SADP_Clearup*** to clear up the devices that have been found.
6. Call ***SADP_Stop*** to stop device search and release SADP resources.

Example

Sample Code of Searching for Online Devices

```
#include <stdio.h>
#include <windows.h>
#include "Sadp.h"

//Device information callback
void CALLBACK SadpDataCallBack(const SADP_DEVICE_INFO_V40 *lpDeviceInfoV40, void* pUserData)
{
    printf("\r\n----------------------------------------\r\n");
    printf("      IP    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szIPv4Address); //Device IP address
    printf("     Mac    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szMAC);        //Device Mac address
    printf("SerialNO    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szSerialNO);    //Device Serial No.
    printf("  Result    %d\r\n", lpDeviceInfoV40->struSadpDeviceInfo.iResult);      //Type: 1. Device online, 2. Device
update, 3. Device Offline
    printf("\r\n----------------------------------------\r\n");
}

int main(void)
{
   //Enable SADP logs  3(print all logs)  "C:\\SadpLog"(log directory) false(save all log files)

   SADP_SetLogToFile(3, "C:\\SadpLog", false);

   //Enable SADP services
   int iRet = SADP_Start_V40(SadpDataCallBack);
   if (iRet == 0)
   {
      //Enabling SADP services failed. Get the error code.
      int iError = SADP_GetLastError();
      printf("SADP_Start_V40 Failed!  Err(%d)\r\n", iError);
   }

   //Set the interval for automatically searching for devices. 0-disable automatic search.

   SADP_SetAutoRequestInterval(10); //unit: second

   Sleep(20000);
   //Release SADP resources.
```

```
    SADP_Stop();
}
```

2.2 Activate a Device

You can activate a device after you find it via SADP.

Steps

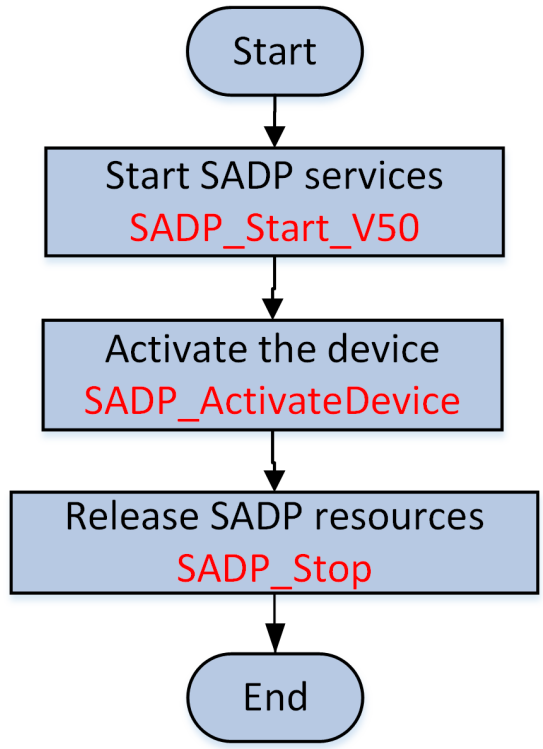

Figure 2-2 API Calling Flow of Activating a Device

1. Call ***SADP_Start_V50*** to start SADP services.

   Note

   Make sure you have found the online device via SADP before activating it. See details in ***Search for Online Devices*** .

2. Call ***SADP_ActivateDevice*** to activate the device.
3. Call ***SADP_Stop*** to release SADP resources.

Example

Sample Code of Activating Devices

```
#include <stdio.h>
#include <windows.h>
#include "Sadp.h"
```

```
//Device information callback
void CALLBACK SadpDataCallBack(const SADP_DEVICE_INFO_V40 *lpDeviceInfoV40, void* pUserData)
{
    printf("\r\n----------------------------------------\r\n");
    printf("     IP    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szIPv4Address); //Device IP address
    printf("    Mac    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szMAC);        //Device Mac address
    printf("SerialNO    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szSerialNO);    //Device Serial No.
    printf("  Result    %d\r\n", lpDeviceInfoV40->struSadpDeviceInfo.iResult);      //Type: 1. Device online, 2. Device
update, 3. Device Offline
    printf("\r\n----------------------------------------\r\n");
}
//Activate a device
int main(void)
{
   //Enable SADP logs  3(print all logs)  "C:\\SadpLog"(log directory) false(save all log files)
   SADP_SetLogToFile(3, "C:\\SadpLog", false);
   //Enable SADP services
   int iRet = SADP_Start_V40(SadpDataCallBack);
   if (iRet == 0)
   {
      //Enabling SADP services failed. Get the error code.
      int iError = SADP_GetLastError();
      printf("SADP_Start_V40 Failed!  Err(%d)\r\n", iError);
   }
   //Wait for a while. Make sure the device is found via SADP before activating it.
   Sleep(10000);
   //Activate the device: device serial No.; password
   iRet = SADP_ActivateDevice("DS-2CD2622FWD-IZS20180312BBWR681619114", "hik12345");
   if (iRet == 0)
   {
      //Failed. Get the error code.
      int iError = SADP_GetLastError();
      printf("SADP_ActivateDevice Failed!  Err(%d)\r\n", iError);
   }
   else
   {
      printf("SADP_ActivateDevice Succ!\r\n");
   }
  //Release SADP resources.
   SADP_Stop();
}
```

2.3 Edit Network Parameters

You can edit the device's network parameters after you find it via SADP.

Steps

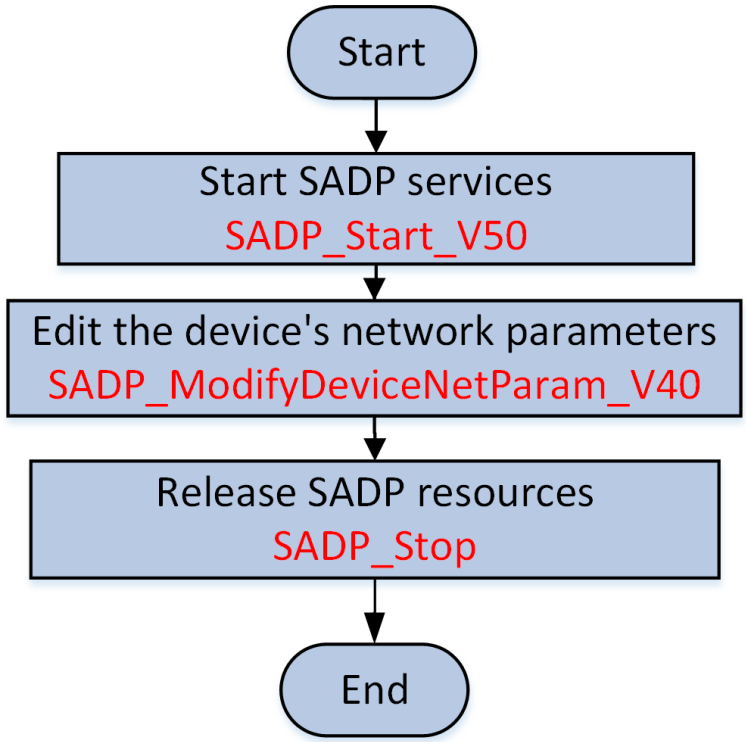

Figure 2-3 API Calling Flow of Editing Network Parameters

1. Call ***SADP_Start_V50*** to start SADP services.

   Note

   Make sure you have found the online device via SADP before editing its network parameters. See details in ***Search for Online Devices*** .

2. Call ***SADP_ModifyDeviceNetParam_V40*** to edit the device's network parameters.
3. Call ***SADP_Stop*** to release SADP resources.

Example

Sample Code of Editing Network Parameters

```
#include <stdio.h>
#include <windows.h>
#include "Sadp.h"
//Device information callback
void CALLBACK SadpDataCallBack(const SADP_DEVICE_INFO_V40 *lpDeviceInfoV40, void* pUserData)
{
    printf("\r\n----------------------------------------\r\n");
    printf("      IP    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szIPv4Address); //Device IP address
    printf("     Mac    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szMAC);        //Device Mac address
    printf("SerialNO    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szSerialNO);    //Device Serial No.
    printf("  Result    %d\r\n", lpDeviceInfoV40->struSadpDeviceInfo.iResult);      //Type: 1. Device online, 2. Device
update, 3. Device Offline
```

```
    printf("\r\n----------------------------------------\r\n");
}
//Edit network parameters
int main(void)
{
    //Enable SADP logs  3(print all logs)  "C:\\SadpLog"(log directory) false(save all log files)
    SADP_SetLogToFile(3, "C:\\SadpLog", false);
   //Enable SADP services
    int iRet = SADP_Start_V40(SadpDataCallBack);
    if (iRet == 0)
   {
       //Enabling SADP services failed. Get the error code.
       int iError = SADP_GetLastError();
       printf("SADP_Start_V40 Failed!  Err(%d)\r\n", iError);
    }
   //Wait for a while. Make sure the device is found via SADP before editing its network parameters.
    Sleep(10000);
    SADP_DEV_NET_PARAM struNetParam = { 0 };
    SADP_DEV_RET_NET_PARAM struDevRetNetParam = { 0 };
    strcpy(struNetParam.szIPv4Address, "192.168.1.64"); //ip
    strcpy(struNetParam.szIPv4SubNetMask, "255.255.255.0");
    strcpy(struNetParam.szIPv4Gateway, "192.168.1.1");
    strcpy(struNetParam.szIPv6Address, "::");
    strcpy(struNetParam.szIPv6Gateway, "::");
    struNetParam.wPort = 8000; //netsdk service port No.
    struNetParam.dwSDKOverTLSPort = 0;
    struNetParam.byDhcpEnable = 0; //Whether to enable DHCP
    struNetParam.byIPv6MaskLen = 64;
    struNetParam.wHttpPort = 80;
    //Edit network parameters: device Mac address; password; network parameter structure; returned network
parameter structure; size of returned network parameter structure
    iRet = SADP_ModifyDeviceNetParam_V40("a4-14-37-f9-e3-ee", "hik12345", &struNetParam,
&struDevRetNetParam, sizeof(struDevRetNetParam));
    if (iRet == 0)
    {
       //Failed. Get the error code.
       int iError = SADP_GetLastError();
       printf("SADP_ModifyDeviceNetParam_V40 Failed!  Err(%d)\r\n", iError);
       if (iError == SADP_LOCKED)
       {
          printf("Device has been locked. Time:%d minutes.", struDevRetNetParam.bySurplusLockTime);
       }
       else if (iError == SADP_PASSWORD_ERROR)
       {
          printf("Wrong password. Remaining attempts:%d times.", struDevRetNetParam.byRetryModifyTime);
       }
       else if (iError == SADP_NOT_ACTIVATED)
       {
          printf("Device has not been activated.");
       }
    }
    else
```

```
    {
        printf("SADP_ModifyDeviceNetParam_V40 Succ!\r\n");
    }
    //Release SADP resources.
    SADP_Stop();
}
```

2.4 Reset the Password

There are two ways to reset the device's password, by using the encrypted string or the device code. The former one is more recommended because it ensures higher level of security.

Steps

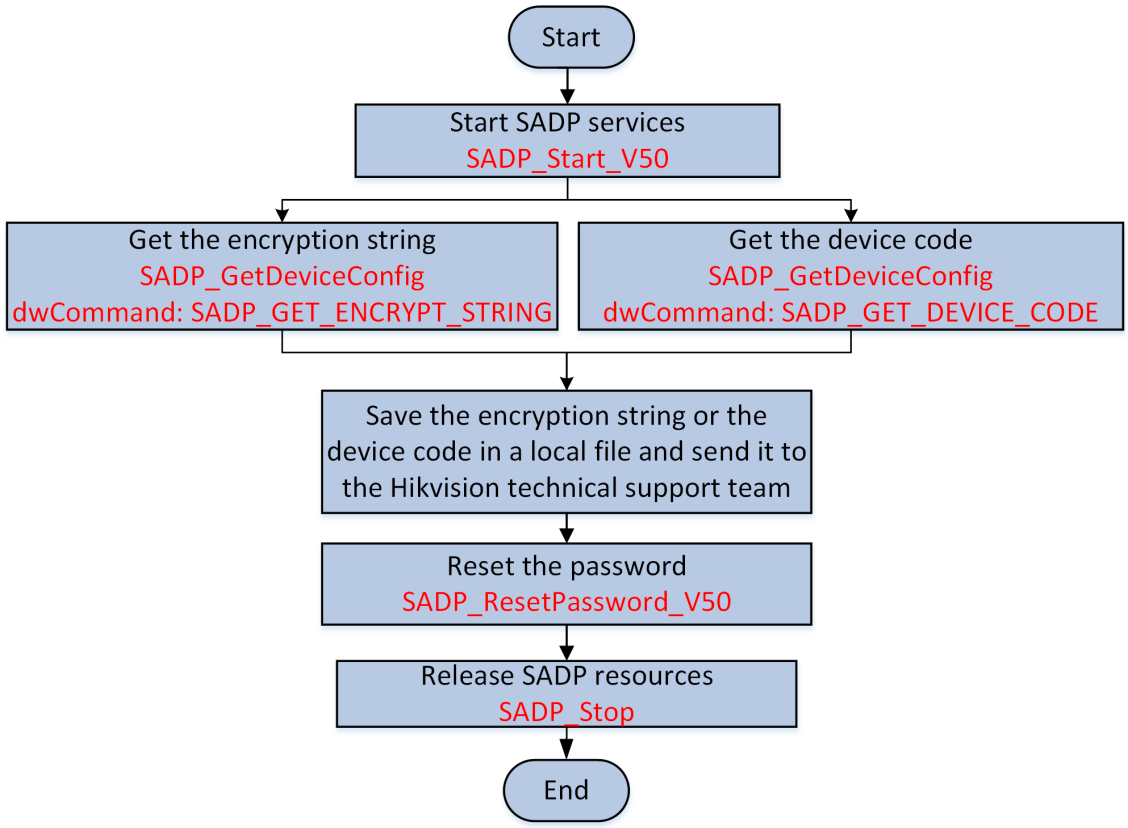

Figure 2-4 API Calling Flow of Resetting the Password

1. Call ***SADP_Start_V50*** to start SADP services.

   When a device is found, its information will be called back via ***PDEVICE_FIND_CALLBACK_V40*** .

Note

- If the value of **bySupport** & 0x01 in the structure ***SADP_DEVICE_INFO*** is 1, it indicates that resetting the password by encrypted string is supported.
- If the value of **bySupport** & 0x01 in the structure ***SADP_DEVICE_INFO*** is 1, it indicates that resetting the password by device code is supported.

2. Get the encrypted string or the device code according to the value of **bySupport** in the structure ***SADP_DEVICE_INFO*** .
   - If the value of **bySupport** & 0x01 in the structure ***SADP_DEVICE_INFO*** is 1, you can call ***SADP_GetDeviceConfig*** with the command (**dwCommand**) "SADP_GET_ENCRYPT_STRING" to get the encrypted string.

     Note

     The encrypted string will be invalid in 24 or 48 hours (depending on the device). You need to reset the password within this period.
   - If the value of **bySupport** & 0x01 in the structure ***SADP_DEVICE_INFO*** is 1, you can call ***SADP_GetDeviceConfig*** with the command (**dwCommand**) "SADP_GET_DEVICE_CODE" to get the device code.
3. Save the encrypted string or the device code in a local file (for example, Device.xml) and send the file to the Hikvision technical support team.

   A file for resetting the password will be returned. (The file will be invalid if you get a new encrypted string.)
4. Call ***SADP_ResetPasswd_V50*** to reset the password with the file for resetting the password.
5. Call ***SADP_Stop*** to release SADP resources.

Example

Sample Code of Resetting the Password

```
#include <stdio.h>
#include <windows.h>
#include "Sadp.h"
//Device information callback
void CALLBACK SadpDataCallBack(const SADP_DEVICE_INFO_V40 *lpDeviceInfoV40, void* pUserData)
{
    printf("\r\n----------------------------------------\r\n");
    printf("      IP    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szIPv4Address); //Device IP address
    printf("     Mac    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szMAC);        //Device Mac address
    printf("SerialNO    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szSerialNO);    //Device Serial No.
    printf("  Result    %d\r\n", lpDeviceInfoV40->struSadpDeviceInfo.iResult);      //Type: 1. Device online, 2. Device
update, 3. Device Offline
    printf("\r\n----------------------------------------\r\n");
}
//Reset the password
int main(void)
{
    //Enable SADP logs  3(print all logs)  "C:\\SadpLog"(log directory) false(save all log files)
    SADP_SetLogToFile(3, "C:\\SadpLog", false);
```

```
   //Enable SADP services
    int iRet = SADP_Start_V40(SadpDataCallBack);
    if (iRet == 0)
    {
       //Enabling SADP services failed. Get the error code.
       int iError = SADP_GetLastError();
       printf("SADP_Start_V40 Failed!  Err(%d)\r\n", iError);
    }
   //Wait for a while. Make sure the device is found via SADP before resetting its password.
    Sleep(10000);
   //Reset the password by using encrypted string.
    SADP_ENCRYPT_STRING struEncryptString;
    iRet = SADP_GetDeviceConfig("DS-2CD2622FWD-IZS20180312BBWR681619114", SADP_GET_ENCRYPT_STRING,
NULL, 0, &struEncryptString, sizeof(struEncryptString));
    if (iRet == 0)
    {
       int iError = SADP_GetLastError();
       char szTemp[100] = { 0 };
       if (iError == SADP_NOT_ACTIVATED)
       {
          printf("Device has not been activated.");
       }
       else if (iError == SADP_TIMEOUT)
       {
          printf("Timed out.");
       }
       else if (iError == SADP_DEVICE_DENY)
       {
          printf("Request is denied by the device.");
       }
       else
       {
          printf("Get device code failed, Error code%d", iError);
       }
      //Release SADP resources.
       SADP_Stop();
       return 0;
    }
    printf("Get device code succ, [%s]", struEncryptString.szEncryptString);
   //Save the encrypted string in a local file.
    FILE *pFile = fopen("C:\\Device.xml", "wb");
    if (NULL == pFile)
    {
       printf("Open File failed");
       //Release resources
       SADP_Stop();
       return 0;
    }
    int iWriteLen = fwrite(struEncryptString.szEncryptString, sizeof(BYTE), strlen(struEncryptString.szEncryptString),
pFile);
    if (0 == iWriteLen)
    {
```

```
        printf("Write 0 Byte Data");
        //Release resources
        SADP_Stop();
        return 0;
    }
    else
    {
        if (NULL != pFile)
        {
            fclose(pFile);
            pFile = NULL;
        }
    }
//---------------------------------------------------------------------------------------------------------------------------------
//After you get the file for resetting the password from the technical support team, you can continue the following operations.
//---------------------------------------------------------------------------------------------------------------------------------
    SADP_RESET_PARAM_V40 struResetParamV40 = { 0 };
    struResetParamV40.dwSize = sizeof(struResetParamV40);
    struResetParamV40.byResetType = 2; //Resetting the password by using the encrypted string
    strcpy(struResetParamV40.szPassword, "hik12345");
    strcpy(struResetParamV40.szAuthFile, "C:\\Encrypt_681619114_o2.xml"); //The file for resetting the password returned from the technical support team.
    SADP_RET_RESET_PARAM_V40 struRetResetParamV40 = { 0 };
    //Reset the password: device serial No.; structure about parameters for resetting the password; structure about returned information
    iRet = SADP_ResetPasswd_V40("DS-2CD2622FWD-IZS20180312BBWR681619114", &struResetParamV40, &struRetResetParamV40);
    if (iRet == 0)
    {
        //Failed. Get the error code.
        int iError = SADP_GetLastError();
        printf("SADP_ResetPasswd_V40 Failed!  Err(%d)\r\n", iError);
        if (iError == SADP_LOCKED)
        {
            printf("Device has been locked. Time:%d minutes.", struRetResetParamV40.bySurplusLockTime);
        }
        else if (iError == SADP_PASSWORD_ERROR)
        {
            printf("Wrong password. Remaining attempts:%d times.", struRetResetParamV40.byRetryGUIDTime);
        }
        else if (iError == SADP_NOT_ACTIVATED)
        {
            printf("Device has not been activated.");
        }
    }
    else
    {
        printf("SADP_ResetPasswd_V40 Succ!\r\n");
    }
    //Release SADP resources.
```

```
    SADP_Stop();
}
```

2.5 Get Device Parameters

You can get the device parameters after you find it via SADP.

Steps

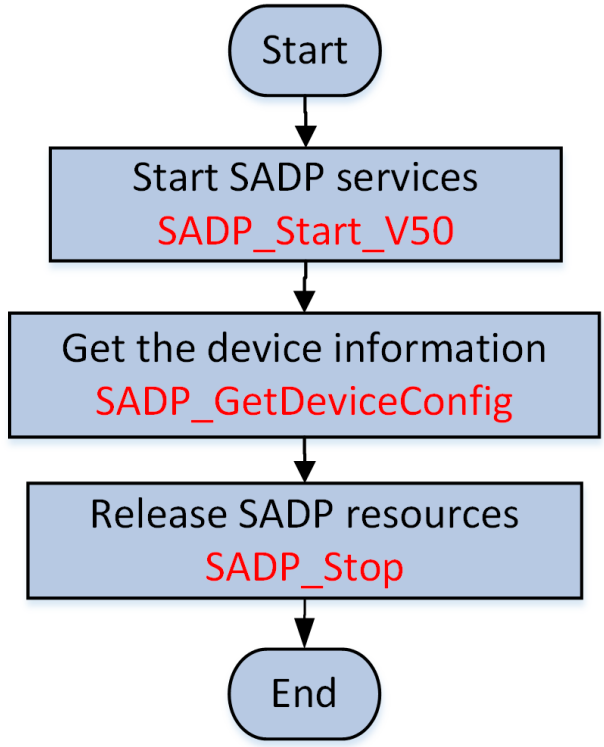

Figure 2-5 API Calling Flow of Getting Device Parameters

1. Call ***SADP_Start_V50*** to start SADP services.

   Note

   Make sure you have found the online device via SADP before getting its parameters. See details in ***Search for Online Devices*** .
2. Call ***SADP_GetDeviceConfig*** with different commands (**dwCommand**) to get the device information.
3. Call ***SADP_Stop*** to release SADP resources.

Example

Sample Code of Getting Device Parameters

```
#include <stdio.h>
```

```
#include <windows.h>
#include "Sadp.h"
//Device information callback
void CALLBACK SadpDataCallBack(const SADP_DEVICE_INFO_V40 *lpDeviceInfoV40, void* pUserData)
{
    printf("\r\n----------------------------------------\r\n");
    printf("     IP    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szIPv4Address); //Device IP address
    printf("     Mac    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szMAC);        //Device Mac address
    printf("SerialNO    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szSerialNO);    //Device Serial No.
    printf("  Result    %d\r\n", lpDeviceInfoV40->struSadpDeviceInfo.iResult);      //Type: 1. Device online, 2. Device
update, 3. Device Offline
    printf("\r\n----------------------------------------\r\n");
}
//Get device parameters (device code)
int main(void)
{
   //Enable SADP logs  3(print all logs)  "C:\\SadpLog"(log directory) false(save all log files)
   SADP_SetLogToFile(3, "C:\\SadpLog\\", false);
   //Enable SADP services
   int iRet = SADP_Start_V40(SadpDataCallBack);
   if (iRet == 0)
   {
      //Enabling SADP services failed. Get the error code.
      int iError = SADP_GetLastError();
      printf("SADP_Start_V40 Failed!  Err(%d)\r\n", iError);
   }
   //Wait for a while. Make sure the device is found via SADP before getting its parameters.
   Sleep(10000);
   //Get device parameters (here take getting device code as an example)
   SADP_SAFE_CODE struSafeCode = { 0 };
   iRet = SADP_GetDeviceConfig("DS-2CD2622FWD-IZS20180312BBWR681619114", SADP_GET_DEVICE_CODE, NULL,
0, &struSafeCode, sizeof(struSafeCode));
   if (iRet == 0)
   {
      int iError = SADP_GetLastError();
      char szTemp[100] = { 0 };
      if (iError == SADP_NOT_ACTIVATED)
      {
         printf("Device has not been activated.");
      }
      else if (iError == SADP_TIMEOUT)
      {
         printf("Timed out.");
      }
      else if (iError == SADP_DEVICE_DENY)
      {
         printf("Request is denied by the device.");
      }
      else
      {
         printf("Getting device code failed, Error code%d", iError);
      }
```

```
        //Release resources.
        SADP_Stop();
        return 0;
    }
    printf("Get device code succ, [%s]", struSafeCode.szDeviceCode);
    //Release SADP resources.
    SADP_Stop();
}
```

2.6 Configure Device Parameters

You can configure the device parameters after you find it via SADP.

Steps

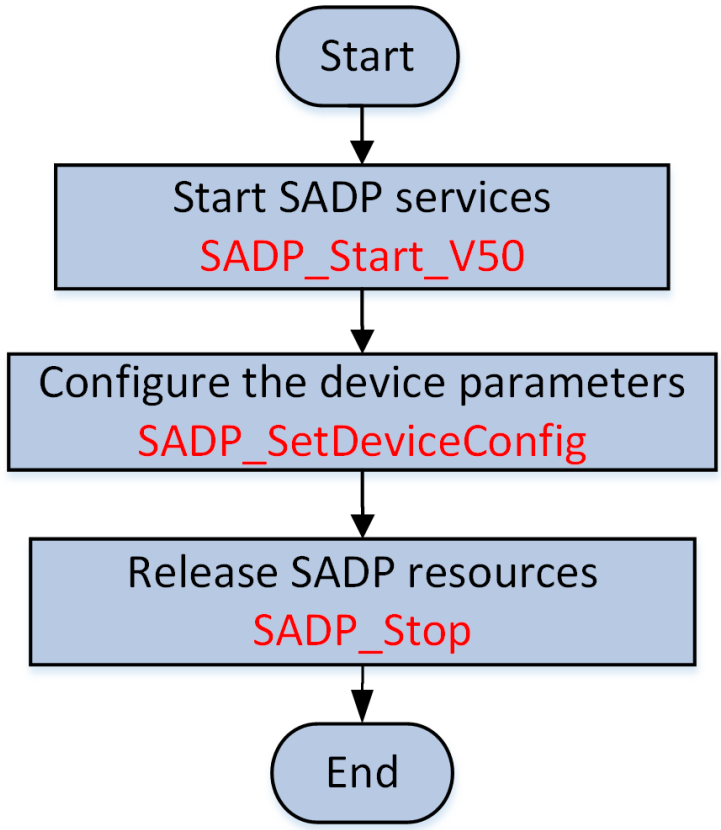

Figure 2-6 API Calling Flow of Configuring Device Parameters

1. Call ***SADP_Start_V50*** to start SADP services.

Note

Make sure you have found the online device via SADP before configuring its parameters. See details in ***Search for Online Devices*** .

2. Call ***SADP_SetDeviceConfig*** with different commands (**dwCommand**) to configure the device parameters.
3. Call ***SADP_Stop*** to release SADP resources.

Example

Sample Code of Configuring Device Parameters

```
#include <stdio.h>
#include <windows.h>
#include "Sadp.h"
//Device information callback
void CALLBACK SadpDataCallBack(const SADP_DEVICE_INFO_V40 *lpDeviceInfoV40, void* pUserData)
{
    printf("\r\n----------------------------------------\r\n");
    printf("      IP    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szIPv4Address); //Device IP address
    printf("     Mac    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szMAC);        //Device Mac address
    printf("SerialNO    %s\r\n", lpDeviceInfoV40->struSadpDeviceInfo.szSerialNO);    //Device Serial No.
    printf("  Result    %d\r\n", lpDeviceInfoV40->struSadpDeviceInfo.iResult);       //Type: 1. Device online, 2. Device
update, 3. Device Offline
    printf("\r\n----------------------------------------\r\n");
}
//Configure device parameters
int main(void)
{
    //Enable SADP logs  3(print all logs)  "C:\\SadpLog"(log directory) false(save all log files)
    SADP_SetLogToFile(3, "C:\\SadpLog\\", false);
   //Enable SADP services
    int iRet = SADP_Start_V40(SadpDataCallBack);
    if (iRet == 0)
    {
       //Enabling SADP services failed. Get the error code.
       int iError = SADP_GetLastError();
       printf("SADP_Start_V40 Failed!  Err(%d)\r\n", iError);
    }
  //Wait for a while. Make sure the device is found via SADP before configuring its parameters.
    Sleep(10000);
   //Configure device parameters (here take configuring EZVIZ cloud status as an example)
    SADP_HCPLATFORM_STATUS_INFO struStatus = { 0 };
    struStatus.dwSize = sizeof(SADP_HCPLATFORM_STATUS_INFO);
    struStatus.byEnableHCPlatform = 1; //Enable
    memcpy(struStatus.szPassword, "abcd1234", 16);
    SADP_DEV_LOCK_INFO struLockInfo = { 0 };
    iRet = SADP_SetDeviceConfig("DS-2CD7A47FWD-XZSG/JM20190807AACHD46539937",
SADP_SET_HCPLATFORM_STATUS, &struStatus, sizeof(struStatus), &struLockInfo, sizeof(struLockInfo));
    if (iRet == 0)
    {
       int iError = SADP_GetLastError();
       if (iError == SADP_TIMEOUT)
       {
          printf("Set HCPlatform status failed: Time out!");
       }
```

```
        else if (iError == SADP_DEVICE_DENY)
        {
            printf("Set HCPlatform status failed!");
        }
        else if (iError == SADP_ILLEGAL_VERIFICATION_CODE)
        {
            printf("Set HCPlatform status failed: Illegal Verification Code!");
        }
        else if (iError == SADP_LOCKED)
        {
            printf("Device Lock,   Lock %d minute.", struLockInfo.bySurplusLockTime);
        }
        else if (iError == SADP_NOT_ACTIVATED)
        {
            printf("dev has not Activate");
        }
        else if (iError == SADP_PASSWORD_ERROR)
        {
            printf("Password error,   %d tiem you can try.", struLockInfo.byRetryTime);
        }
        else
        {
            printf("Set HCPlatform status failed!,Error Number%d", iError);
        }
       //Release resources
        SADP_Stop();
        return 0;
    }
    printf("SADP_GetDeviceConfig SADP_SET_HCPLATFORM_STATUS success");
   //Release resources
    SADP_Stop();
}
```

Chapter 3 API Reference

3.1 SADP_GetSadpVersion

Get the SDK version information.

API Definition

```
unsigned int SADP_GetSadpVersion(
)
```

Return Value

The version No. is in the format of Va.b.c.d (value a: 31-24 bits, value b: 23-16 bits, value c: 15-8 bits, value d: 7-0 bits), e.g., if the return value is 0x3010103, then the version No. is V3.1.1.3.

3.2 SADP_SetLogToFile

Enable writing to log files.

API Definition

```
int SADP_SetLogToFile(
  int         nLogLevel,
  char const  *strLogDir,
  int         bAutoDel
)
```

Parameters

nLogLevel

[IN] Log level (the default value is 0) : 0-disable logs, 1-only output ERROR logs, 2-output ERROR and DEBUG logs, 3-output all information including ERROR, DEBUG, and INFO logs.

strLogDir

[IN] Directory path string, less than 256 characters. When the value is NULL, Windows default path is "C:/SadpLog/".

bAutoDel

[IN] Whether to delete the exceeded files, the default value is TRUE. The maximum number of files is 10.

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

3.3 SADP_Start_V50

Start SADP services.

API Definition

```
int SADP_Start_V50(
  SADP_START_PARAM  struStartParam
)
```

Parameters

struStartParam

[IN] Structure about starting SADP services. See details in ***SADP_START_PARAM*** .

Return Value

Return *1* for success, and return *0* for failure.

Remarks

1. Use the MAC address in the device information as an ID for the device because the serial No. may change.
2. If the IP address of the local PC is changed, you do not need to restart SADP services.

3.4 SADP_Stop

Stop device search and release SADP resources.

API Definition

```
int SADP_Stop(
  void
)
```

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

3.5 SADP_SendInquiry

Manually start searching for devices.

API Definition

```
int SADP_SendInquiry(
void)
```

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

Remarks

- By default, the online devices can be automatically refreshed every 60 seconds. By calling this API, you can refresh the online devices manually.
- After manually searching for devices, the information of newly found devices will be returned via the callback function configured in ***SADP_Start_V50*** .

3.6 SADP_SetAutoRequestInterval

Set the interval for automatically searching for devices.

API Definition

```
int SADP_SetAutoRequestInterval(
unsigned int   dwInterval
)
```

Parameters

dwInterval

[IN] Internal for automatically sending request for searching for devices, unit: second. By default, the value is 60. No automatic request will be sent when the value is 0.

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

3.7 SADP_ModifyDeviceNetParam_V40

Edit device network parameters, including device IP address, port No., etc.

API Definition

```
int SADP_ModifyDeviceNetParam_V40(
  const char          *sMAC,
  const char          *sPassword,
```

```
  const SADP_DEV_NET_PARAM     *lpNetParam,
  SADP_DEV_RET_NET_PARAM       *lpRetNetParam,
  unsigned int                 dwOutBuffSize
)
```

Parameters

sMAC

[IN] MAC address of the device.

sPassword

[IN] Password of the admin user.

lpNetParam

[IN] Network parameters to be edited. See details in ***SADP_DEV_NET_PARAM*** .

lpRetNetParam

[OUT] Returned network parameters. See details in ***SADP_DEV_RET_NET_PARAM*** .

dwOutBuffSize

[OUT] Output buffer size.

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

3.8 SADP_InquirySpecificSubnet

Search for devices in different subnets.

API Definition

```
int SADP_InquirySpecificSubnet(
  const SADP_SUBNET_INFO  *pSubnetInfo
)
```

Parameters

pSubnetInfo

[IN] Structure about subnet information. See details in ***SADP_SUBNET_INFO*** .

Return Value

Return *1* for success, and return *0* for failure.

Remarks

The same device may enter the callback multiple times, and the user needs to determine whether it is the same device via the serial No.

3.9 SADP_GetLastError

Get the error code.

API Definition

```
int SADP_GetLastError(
)
```

Return Value

The returned value is the error code. See details in ***Error Code*** .

3.10 SADP_Clearup

Clear up the devices that have been found.

API Definition

```
unsigned int SADP_Clearup(
void)
```

Return Value

Return *1* for success, and return *0* for failure.

3.11 SADP_GetDeviceConfig

Get the device information.

API Definition

```
int SADP_GetDeviceConfig(
  const char  *sDevSerialNO,
  DWORD      dwCommand,
  void       *lpInBuffer,
  DWORD      dwinBuffSize,
  void       *lpOutBuffer,
  DWORD      dwOutBuffSize
)
```

Parameters

sDevSerialNO

[IN] Device serial No.

dwCommand

[IN] Command.

lpInBuffer

[IN] Input parameter. Its content relates to **dwCommand**.

dwinBuffSize

[IN] Input buffer size.

lpOutBuffer

[OUT] Output buffer. Its content relates to **dwCommand**.

dwOutBuffSize

[OUT] Output buffer size.

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

Remarks

- Different functions are corresponding to different structure and command No., as shown below.

dwCommand	dwCommand description	lpInBuffer	lpOutBuffer	dwCommand value
SADP_GET_DEVICE_CODE	Get the device code.	NULL	***SADP_SAFE_CODE***	1
SADP_GET_ENCRYPT_STRING	Get the encrypted string.	NULL	***SADP_ENCRYPT_STRING***	2
SADP_GET_DEVICE_TYPE_UNLOCK_CODE	Get the device type unlock code.	NULL	***SADP_TYPE_UNLOCK_CODE***	3
SADP_GET_GUID	Get the GUID.	***SADP_GUID_FILE_COND***	***SADP_GUID_FILE***	5
SADP_GET_SECURITY_QUESTION	Get the security question.	NULL	***SADP_SECURITY_QUESTION_CFG***	6
SADP_GET_BIND_LIST	Get the linked device list.	NULL	***SADP_BIND_LIST***	12
SADP_GET_SELF_CHECK	Get the device self-inspection status.	NULL	***SADP_SELF_CHECK_STATE***	17

dwCommand	dwCommand description	lpInBuffer	lpOutBuffer	dwCommand value
SADP_DISK_LOCATE	Locate the bad disk.	NULL	NULL	18
SADP_GET_QR_CODES	Get the QR code.	NULL	***SADP_QR_CODES***	21
SADP_GET_PASSWORD_RESET_TYPE	Get the method of resetting the password.	NULL	***SADP_PASSWORD_RESET_TYPE_PARAM***	27
SADP_GET_PHONE_QR_CODES	Get the QR code for scanning to reset the password.	NULL	***SADP_PHONE_QR_CODES***	29
SADP_GET_DEVICE_CODE_V31	Get the device code (for devices of 3.1 and earlier versions)	NULL	***SADP_SAFE_CODE_V31***	30
SADP_GET_ENCRYPT_STRING_V31	Get the encrypted string (for devices of 3.1 and earlier versions)	NULL	***SADP_ENCRYPT_STRING_V31***	21
SADP_GET_GUID_V31	Get the GUID (for devices of 3.1 and earlier versions)	NULL	***SADP_GUID_FILE_V31***	32
SADP_GET_QR_CODES_V31	Get the QR code (for devices of 3.1 and earlier versions)	NULL	***SADP_QR_CODES_V31***	33

- The V31 command code is compatible with the protocol 3.0, that is, users can directly replace it with the V31 command code. The devices supporting multicast protocol 3.0 and 3.1 can be simultaneously accessed to.

3.12 SADP_SetDeviceConfig

Configure device parameters.

API Definition

```
int SADP_SetDeviceConfig(
  const char  *sDevSerialNO,
  DWORD       dwCommand,
  void        *lpInBuffer,
  DWORD       dwinBuffSize,
  void        *lpOutBuffer,
  DWORD       dwOutBuffSize
)
```

Parameters

sDevSerialNO

[IN] Device serial No.

dwCommand

[IN] Command.

lpInBuffer

[IN] Input parameter. Its content relates to **dwCommand**.

dwinBuffSize

[IN] Input buffer size.

dwOutBuffSize

[IN] Output buffer size.

lpOutBuffer

[OUT] Output buffer. Its content relates to **dwCommand**.

Return Value

Return *1* for success, and return *0* for failure.
If *0* is returned, you can call ***SADP_GetLastError*** to get the error code.

Remarks

Different functions are corresponding to different structure and command No., as shown below.

dwCommand	dwCommand description	lpInBuffer	lpOutBuffer	dwCommand value
SADP_SET_DEVICE_CUSTOM_TYPE	Set the custom model for a device.		NULL	4
SADP_SET_SECURITY_QUESTION	Set security questions.	***SADP_SECURITY_QUESTION_CFG***	***SADP_SECURITY_QUESTION***	7

dwCommand	dwCommand description	lpInBuffer	lpOutBuffer	dwCommand value
SADP_SET_HCPLATFORM_STATUS	Set HCPLATFORM status.	***SADP_HCPLATFORM_STATUS_INFO***	***SADP_DEV_LOCK_INFO***	8
SADP_SET_VERIFICATION_CODE	Set device verification code (verification code is used as the unique password for login, control, and live view.	***SADP_VERIFICATION_CODE_INFO***	***SADP_DEV_LOCK_INFO***	9
SADP_SET_BIND_LIST	Set linked device list.	***SADP_BIND_LIST***	***SADP_DEV_LOCK_INFO***	13
SADP_RESTORE_INACTIVE	Inactivate the device.	***SADP_INACTIVE_INFO***	***SADP_DEV_LOCK_INFO***	14
SADP_SET_WIFI_REGION	Set Wi-Fi area.	***SADP_WIFI_REGION_INFO***	***SADP_DEV_LOCK_INFO***	15
SADP_SET_CHANNEL_DEFAULT_PASSWORD	Set default password for the channel.	***SADP_CHANNEL_DEFAULT_PASSWORD***	***SADP_DEV_LOCK_INFO***	16
SADP_EHOME_ENABLE	Enable ISUP (EHome).	***SADP_EHOME_ENABLE_PARAM***	***SADP_DEV_LOCK_INFO***	19
SADP_SET_USER_MAILBOX	Set the user's email.	***SADP_USER_MAILBOX***	***SADP_DEV_LOCK_INFO***	20

3.13 SADP_ResetPasswd_V50

Reset device password.

API Definition

```
Int SADP_ResetPassword_V50(
  const char                    *sDevSerialNO,
  const SADP_RESET_PARAM_V50   *pResetParam,
 SADP_DEV_LOCK_INFO            *pLockInfo
)
```

Parameters

sDevSerialNO

[IN]Device serial No.

pResetParam

[IN] Structure about parameters of resetting password. See details in ***SADP_RESET_PARAM_V50*** .

pLockInfo

[OUT] Structure about device lock information. See details in ***SADP_DEV_LOCK_INFO*** .

Return Value

Return *1* for success, and return *0* for failure.

Remarks

The new API is backward compatible with the old API, that is, even if devices that do not support the multicast protocol 3.1, they can still use the new API to reset the password, and it will continue to interact with the device via protocol 3.0. The protocol 3.1 does not support byResetType when its value is 3 (reset password via QR code).

3.14 SADP_ActivateDevice

Activate the device.

API Definition

```
int SADP_ActivateDevice(
  const char   *sDevSerialNO,
  const char   *sCommand
)
```

Parameters

sDevSerialNO

Device serial No.

sCommand

Password.

Return Value

Return *1* for success, and return *0* for failure.

Chapter 4 Callback Function

4.1 PDEVICE_FIND_CALLBACK_V40

Callback function of getting online device information

Callback Function Definition

```
typedef void (CALLBACK *PDEVICE_FIND_CALLBACK_V40)(
  const SADP_DEVICE_INFO_V40   *lpDeviceInfo,
  void                         *pUserData
);
```

Parameters

lpDeviceInfo

[OUT] The information of the device that has been found. See details in ***SADP_DEVICE_INFO_V40*** .

pUserData

[OUT] User data pointer.

Related API

SADP_Start_V50

4.2 PSUBNET_DEVICE_FIND_CALLBACK

Callback function of getting device information in different subnets.

Callback Function Definition

```
typedef void (CALLBACK *PSUBNET_DEVICE_FIND_CALLBACK)(
  const SADP_SUBNET_DEVICE_INFO   *lpDeviceInfo,
  void                            *pUserData
);
```

Parameters

lpDeviceInfo

[OUT] The information of the device that has been found. See details in ***SADP_SUBNET_DEVICE_INFO*** .

pUserData

[OUT] User data pointer.

Related API

SADP_Start_V50

Appendix A. Data Structure

A.1 SADP_BIND_INFO

Structure about the Linked Device Information

Member	Data Type	Description
szSerialNO	char[]	The linked device serial No. The length is defined by the macro definition "SADP_MAX_SERIALNO_LEN" (64).
byiBind	unsigned char	Whether the device is linked: 0-no, 1-yes.
byRes	unsigned char[]	Reserved. The length is 127 bytes.

A.2 SADP_BIND_LIST

Structure about Linked Device List

Member	Data Type	Description
struBindInfo	***SADP_DEVICE_INFO*** []	The linked device information list. The number of elements in this array is defined by the macro definition "SADP_MAX_BIND_NUM" (32).
szPassword	char	Password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
byUnbindAll	unsigned char	Whether to unlink all devices: 0-no, 1-yes. This member is valid only when setting device parameters.
byRes	unsigned char[]	Reserved. The length is 127 bytes.

A.3 SADP_CHANNEL_DEFAULT_PASSWORD

Structure about The Channel Default Password

Member	Data Type	Description
szPassword	char[]	Device admin password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
szChannelDefaultPassword	char[]	Default password of the channel. It is used when activating NVR(s) or adding network camera(s). The length is defined by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.4

Structure about the Custom Device Type

Member	Data Type	Description
dwCodeSize	unsigned int	Structure size.
szDeviceTypeSecretKey	char[]	Device type secret key. The length is defined by the macro definition "MAX_UNLOCK_CODE_KEY" (256).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.5 SADP_DEV_LOCK_INFO

Custom Device Type Structure

Member	Data Type	Description
byRetryTime	unsigned char	Remaining attempts.
bySurplusLockTime	unsigned char	Remaining number of minutes for the user locking. The parameter is valid when the user is locked.
byRes	unsigned char[]	Reserved. The length is 126 bytes.

A.6 SADP_DEV_NET_PARAM

Network Parameters Structure

Member	Data Type	Description
szIPv4Address	char[]	IPv4 address. The length is 16 bytes.
szIPv4SubNetMask	char[]	IPv4 subnet mask. The length is 16 bytes.
szIPv4Gateway	char[]	IPv4 gateway. The length is 16 bytes.
szIPv6Address	char[]	IPv6 address. The length is 128 bytes.
szIPv6Gateway	char[]	IPv6 gateway. The length is 128 bytes.
wPort	unsigned short	Device network SDK service port No. (default value: 8000).
byIPv6MaskLen	unsigned char	IPv6 mask length
byDhcpEnable	unsigned char	Whether to enable DHCP: 0-disable, 1-enable.
wHttpPort	unsigned short	HTTP port

Member	Data Type	Description
dwSDKOverTLSPort	unsigned int	SDK Over TLS service port used in Device Network SDK.
byRes	unsigned char[]	Reserved. The length is 122 bytes.

A.7 SADP_DEV_RET_NET_PARAM

Structure about the Returned Network Parameters Information

Member	Data Type	Description
byRetryModifyTime	unsigned char	Remaining attempts to edit network parameters.
bySurplusLockTime	unsigned char	Remaining time, unit: minute. The parameter is valid when the user is locked.
byRes	unsigned char[]	Reserved. The length is 126 bytes.

A.8 SADP_DEVICE_INFO

Device Information Structure

Member	Data Type	Description
szSeries	char[]	Device series. The length is 12 bytes.
szSerialNO	char[]	Device serial No. The length is 48 bytes.
szMAC	char[]	Device MAC address. The length is 20 bytes.
szIPv4Address	char[]	Device IPv4 address. The length is 16 bytes.

Member	Data Type	Description
szIPv4SubnetMask	char[]	Device IPv4 subnet mask. The length is 16 bytes.
dwDeviceType	unsigned int	Device type. The specific value represents a device model.
dwPort	unsigned int	Device network SDK service port.
dwNumberOfEncoders	unsigned int	The number of device encoders (encoding channels). The value is 0 for the decoder.
dwNumberOfHardDisk	unsigned int	The number of device HDDs.
szDeviceSoftwareVersion	char[]	Device software version No. The length is 48 bytes.
szDSPVersion	char[]	Device DSP version No. The length is 48 bytes.
szBootTime	char[]	Launch time. The length is 48 bytes.
iResult	int	Information type, including device rebooting, device online, device offline, device update, etc. See details in Remarks.
szDevDesc	char[]	Device type description, corresponding to **dwDeviceType**. The length is 24 bytes.
szOEMinfo	char[]	OEM manufacturer information. The length is 24 bytes.
szIPv4Gateway	char[]	Device IPv4 gateway. The length is 16 bytes.
szIPv6Address	char[]	Device IPv6 address. The length is 46 bytes.
szIPv6Gateway	char[]	Device IPv6 gateway. The length is 46 bytes.

Member	Data Type	Description
byIPv6MaskLen	unsigned char	IPv6 subnet prefix length.
bySupport	unsigned char	Binary digits (bits): 0 (not support), 1 (support). bySupport & 0x01 (whether the device supports IPv6); bySupport & 0x02 (whether the device supports editing IPv6 parameters); bySupport & 0x04 (whether the device supports DHCP); bySupport & 0x08 (whether the device supports UDP multicast); bySupport & 0x10 (whether the device includes encryption node); bySupport & 0x20 (whether the device supports restoring default password); bySupport & 0x40 (whether the device supports resetting password); bySupport & 0x80 (whether the device supports synchronizing IPC password).
byDhcpEnabled	unsigned char	Whether to enable DHCP: 0-no, 1-yes.
byDeviceAbility	unsigned char	Device capabilities set. Whether the device supports functions of device type description, OEM manufacturer, IPv4 gateway, IPv6 address, IPv6 gateway, IPv6 subnet prefix, and DHCP: 0-no, 1-yes.
wHttpPort	unsigned short	HTTP port.

Member	Data Type	Description
wDigitalChannelNum	unsigned short	The number of digital channels.
szCmsIPv4	char[]	CMS server IPv4 address. The length is 16 bytes.
wCmsPort	unsigned short	CMS server listening port.
byOEMCode	unsigned char	OEM ID: 0-baseline device, 1-OEM device.
byActivated	unsigned char	Whether to activate device: 0-yes, 1-no.
szBaseDesc	char[]	Short model of the baseline device, and the model does not change with the customization. It is used for the model comparison of Hik-Connect device. The length is 24 bytes.
bySupport1	unsigned char	Binary digits (bits): 0 (not support), 1 (support). bySupport1 & 0x01 (whether the device supports resetting the password via method 2); bySupport1 & 0x02 (whether the device supports device locking); bySupport & 0x04 (whether the device supports importing GUID resetting password); bySupport & 0x08 (whether the device supports security question resetting password); bySupport & 0x10 (whether the device supports changing the logo for OEM); bySupport & 0x20 (whether the NVR supports linking with the front-end device);

Member	Data Type	Description
		bySupport & 0x40 (whether the device supports restoring inactivated status); bySupport & 0x80 (whether the device supportsWi-Fi enhancement mode).
byHCPlatform	unsigned char	Whether the device supports Hik-Connect: 0-reserved, 1-yes, 2-no.
byEnableHCPlatform	unsigned char	Whether to enable Hik-Connect: 0-reserved, 1-yes, 2-no.
byEZVIZCode	unsigned char	Device type: 0-baseline device, 1-Ezviz device.
dwDetailOEMCode	unsigned int	OEM code details. The OEM code contains three parts: customer No. (from 1 to 429496), menu style (2 digits), area No. (2 digits).
byModifyVerificationCode	unsigned char	Whether to edit the verification code: 0-reserved, 1-yes, 2-no.
byMaxBindNum	unsigned char	The maximum number of devices that can be linked with the NVR. This field is supported by NVRs only.
wOEMCommandPort	unsigned short	OEM command port, which is used to change the OEM logo.
bySupportWifiRegion	unsigned char	The list of Wi-Fi area supported by devices. Binary digits (bits): 0 (not support), 1 (support); bySupportWifiRegion & 0x01 (whether to support default, the default power is the same as that in North America);

Member	Data Type	Description
		bySupportWifiRegion & 0x02 (whether to support China); bySupportWifiRegion & 0x04 (whether to support North America); bySupportWifiRegion & 0x08 (whether to support Japan); bySupportWifiRegion & 0x10 (whether to support Europe); bySupportWifiRegion & 0x20 (whether to support world).
byEnableWifiEnhancement	unsigned char	Whether to enable the Wi-Fi enhancement mode: 0-no, 1-yes.
byWifiRegion	unsigned char	Device current area: 0-default, 1-China, 2-North America, 3-Japan, 4-Europe, 5-World.
bySupport2	unsigned char	Binary digits (bits): 1 (support), 0 (not support). bySupport2 & 0x01 (whether the device supports configuring the default password of a channel); bySupport2 & 0x02 (whether the device supports resetting the password through Email); bySupport2 & 0x04 (whether the inactivated device supports configuring SSID and password).

Remarks

Macro Definition	Value	Description
SADP_ADD	1	The device that has not appeared in SADP library list is online.
SADP_UPDATE	2	Online device information updating, including the changes of device IP, subnet mask, port, HDD, or the number of encoders.
SADP_DEC	3	Offline information of devices. A device sends the offline message or fails to be detected.
SADP_RESTART	4	The offline device appeared in the SADP library list will be online again.
SADP_UPDATEFAIL	5	Updating device failed.

A.9 SADP_DEVICE_INFO _V40

Device Information Structure

Member	Data Type	Description
struSadpDeviceInfo	***SADP_DEVICE_INFO***	Structure about the information of the device.
byLicensed	unsigned char	Whether the device is authorized: 0-reserved, 1-no, 2-yes.
bySystemMode	unsigned char	System mode: 0-reserved, 1-single control, 2-double control, 3-single control and cluster, 4-double control and cluster.

Member	Data Type	Description
byControllerType	unsigned char	Controller type: 0-reserved, 1-controller of type A, 2-controller of type B.
szEhmoeVersion	char[]	Ehmoe version No. The length is 16 bytes.
bySpecificDeviceType	unsigned char	Device type: 1-neutral, 2-baseline.
dwSDKOverTLSPort	unsigned int	The command port of SDK Over TLS.
bySecurityMode	unsigned char	Device security mode: 0-standard, 1-high-A, 2-high-B, 3-custom.
bySDKServerStatus	unsigned char	The SDK service status of the device: 0-disable, 1-enable.
bySDKOverTLSServerStatus	unsigned char	Service status of SDK over TLS: 0-disable, 1-enable.
szUserName	char[]	The user name (by default: admin) of the administrator. The length is defined by the macro definition "MAX_USERNAME_LEN + 1"(32 + 1).
szWifiMAC	char[]	MAC address of the Wi-Fi that the device is connected to. The length is 20 bytes.
byDataFromMulticast	unsigned char	The data source: 0-link protocol, 1-UDP multicast protocol.
bySupportEzvizUnbind	unsigned char	Whether the device supports unlinking Hik-Connect account: 0-not support, 1-support.
bySupportCodeEncrypt	unsigned char	Whether the device supports encrypting the resetting password via AES128_ECB: 0-no, 1-yes.

Member	Data Type	Description
bySupportPasswordResetType	unsigned char	Whether it supports getting the parameters of password resetting types: 0-not support, 1-support.
byEZVIZBindStatus	unsigned char	Whether the device is linked to a Hik-Connect account: 0-unknow, 1-linked, 2-not linked.
szPhysicalAccessVerification	unsigned char	Supported physical methods of adding devices: 1-AP distribution network transmission, 2-link via user token, 3-physical button, 4-link via scanning QR code (device token).
wHttpsPort	unsigned short	HTTPS port.
bySupportEzvizUserToken	unsigned char	Whether the device supports configuring Hik-Connect user token: 0-not support, 1-support.
szDevDescEx	char[]	Extension of the member **szDevDesc** in the SADP_DEVICE_INFO structure.
szSerialNOEx	char[]	Extension of the member **szSerialNO** in the SADP_DEVICE_INFO structure.
szManufacturer	char[]	Manufacturer information of the device.
bySupportResetPwByPhoneNo	unsigned char	Whether the device supports resetting the password by scanning the QR code: 0-not support, 1-support.
byRes	unsigned char	Reserved. The length is 183 bytes.

A.10 SADP_GUID_FILE_V31

GUID Structure

Member	Data Type	Description
dwGUIDSize	unsigned int	GUID length.
szGUID	char[]	GUID contents. The length is defined by the macro definition "MAX_GUID_LEN_V31" (512).
byRes	unsigned char	Reserved. The length is 256 bytes.

A.11 SADP_START_PARAM

Structure about Parameters of Starting SADP Service

Member	Data Type	Description
PDEVICE_FIND_CALLBACK_V40	***PDEVICE_FIND_CALLBACK_V40***	Required. Callback function of getting the information of online devices in the same subnet.
PSUBNET_DEVICE_FIND_CALLBACK	***PSUBNET_DEVICE_FIND_CALLBACK***	Callback function of getting the information of online devices in the different subnet. It can be set as NULL if the application does not need the function of getting this kind of information.
pUserData	void*	User data pointer.
byRes	unsigned char	Reserved. The length is 1024 bytes.

A.12 SADP_SAFE_CODE_V31

Structure about Parameters of Getting Device Security Code

Member	Data Type	Description
dwCodeSize	unsigned int	Structure size.
szDeviceCode	char[]	Device returned code. The length is defined by the macro definition "MAX_DEVICE_CODE_V31" (512).
byRes	unsigned char	Reserved. The length is 512 bytes.

A.13 SADP_SUBNET_INFO

Subnet Information Structure

Member	Data Type	Description
dwSize	unsigned int	Structure size.
byIPType	unsigned char	IP address type: 0-IPv4, 1-IPv6 (not supported).
byRes1	unsigned char	Reserved. The length is 3 bytes.
szStartSubnetIP	char[]	Start IP address of subnet. The length is 46 bytes.
szStopSubnetIP	char[]	End IP address of subnet. The length is 46 bytes.
byRes	unsigned char	Reserved. The length is 128 bytes.

A.14 SADP_ENCRYPT_STRING_V31

Structure about the Encrypted String

Member	Data Type	Description
dwEncryptStringSize	unsigned int	Encrypted string length.
szEncryptString	char[]	Encrypted string contents. he length is defined by the macro definition "MAX_ENCRYPT_CODE_V31" (1024).
byRes	unsigned char	Reserved. The length is 512 bytes.

A.15 SADP_SUBNET_DEVICE_INFO

Device Information Structure

Member	Data Type	Description
dwDeviceType	unsigned int	Device type. The specific value represents a device model.
szDevDesc	char[]	Device type description, corresponding to **dwDeviceType**. The length is 64 bytes.
szSerialNO	char[]	Device serial No. The length is 128 bytes.
szIPv4Address	char[]	Device IPv4 address. The length is 16 bytes.
szIPv4SubnetMask	char[]	Device IPv4 subnet mask. The length is 16 bytes.
szIPv4Gateway	char[]	IPv4 gateway. The length is 16 bytes.
szIPv6Address	char[]	IPv6 address. The length is 46 bytes.
szIPv6Gateway	char[]	IPv4 gateway. The length is 46 bytes.

Member	Data Type	Description
byIPv6MaskLen	unsigned char	IPv6 subnet prefix length.
bySupportIPv6	unsigned char	Whether the device supports IPv6: 0-not support, 1-support.
bySupportModifyIPv6	unsigned char	Whether the device supports editing IPv6: 0-not support, 1-support.
bySupportDhcp	unsigned char	Whether the device supports DHCP: 0-not support, 1-support.
byDhcpEnabled	unsigned char	Whether to enable DHCP: 0-no, 1-yes.
byRes	unsigned char	Reserved. The length is 1024 bytes.

A.16 SADP_EHOME_ENABLE_PARAM

Structure about ISUP Configuration

Member	Data Type	Description
dwSize	unsigned int	Structure size.
szDevID	char[]	Device ID. The length is defined by the macro definition "MAX_PASS_LEN" (16).
szEhomeKey	char[]	Ehome Key for registration. The length is defined by the macro definition "MAX_PASS_LEN" (16).
szPassword	char[]	Device password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 64 bytes.

A.17 SADP_ENCRYPT_STRING

Structure about the Encrypted String

Member	Data Type	Description
dwEncryptStringSize	unsigned int	Encrypted string length
szEncryptString	char[]	Encrypted string contents. The length is defined by the macro definition "MAX_ENCRYPT_CODE" (256)
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.18 SADP_GUID_FILE

GUID Structure

Member	Data Type	Description
dwGUIDSize	unsigned int	GUID length.
szGUID	char[]	GUID contents. The length is defined by the macro definition "MAX_GUID_LEN" (128).
byRetryGUIDTime	unsigned char	Remaining times of importing or exporting GUID.
bySurplusLockTime	unsigned char	Remaining number of minutes for user locking. The parameter is valid when the user is locked.
byRes	unsigned char[]	Reserved. The length is 254 bytes.

A.19 SADP_GUID_FILE_COND

Structure about Condition of Getting The GUID File

Member	Data Type	Description
szPassword	char[]	Password. The length is definied by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.20 SADP_HCPLATFORM_STATUS_INFO

Structure about the Hik-Connect Status Information

Member	Data Type	Description
dwSize	unsigned int	Structure size.
byEnableHCPlatform	unsigned char	Whether to enable Hik-Connect: 0-reserved, 1-yes, 2-no.
byRes	unsigned char[]	Reserved. The length is 3 bytes.
szPassword	char[]	Password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.21 SADP_INACTIVE_INFO

Structure about Parameters of Inactivating A Device

Member	Data Type	Description
szPassword	char[]	Password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.22 SADP_PASSWORD_RESET_TYPE_PARAM

Structure about Methods of Resetting the Password

Member	Data Type	Description
dwSize	unsigned int	Structure size.
byEnable	char[]	Whether the password resetting method is configured: 0 (no), 1 (one or multiple methods configured: GUID, security questions, security email, or HikConnect).
byGuidEnabled	unsigned char	Whether the GUID is exported: 0 (no), 1 (yes).
bySecurityQuestionEnabled	char[]	Whether the security questions are configured: 0 (no), 1 (yes).
bySecurityMailBoxEnabled	char[]	Whether the security email is configured: 0 (no), 1 (yes).
byHikConnectEnabled	char[]	Whether the device is bound to a HikConnect account: 0 (no), 1 (yes).
byRes1	char[]	Reserved. The length is 3 bytes.
byRes	char[]	Reserved. The length is 64 bytes.

A.23 SADP_PHONE_QR_CODES

Structure about Resetting the Password via Scanning QR Code

Member	Data Type	Description
dwSize	unsigned int	Structure size.
szDomainName	char[]	Domain name. The length is defined by the macro definition "MAX_QR_CODES" (256) .
szDevModel	char[]	Device model. The length is 32.
szQrCodes	char[]	QR code. The length is defined by the macro definition "MAX_QR_CODES_V31" (1024) .
byRes	unsigned char	Reserved. The length is 128 bytes.

A.24 SADP_QR_CODES

QR Code Information Structure

Member	Data Type	Description
dwCodeSize	unsigned int	Structure size.
dwMailBoxSize	unsigned int	Length of the reserved email address.
dwServiceMailBoxSize	unsigned int	Length of the service email address.
szQrCodes	char[]	QR code data. The length is defined by the macro definition "MAX_QR_CODES (256)".
szMailBoxAddr	char[]	Reserved email address. The length is defined by the macro definition "MAX_MAILBOX_LEN" (128).

Member	Data Type	Description
szServiceMailBoxAddr	char[]	Service email address. The length is defined by the macro definition "MAX_MAILBOX_LEN" (128).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.25 SADP_QR_CODES_V31

QR Code Information Structure

Member	Data Type	Description
dwCodeSize	unsigned int	Structure size.
dwMailBoxSize	unsigned int	Length of the reserved email address.
dwServiceMailBoxSize	unsigned int	Length of the service email address.
szQrCodes	char[]	QR code. The length is defined by the macro definition "MAX_QR_CODES_V31" (1024).
szMailBoxAddr	char[]	Reserved email address. The length is defined by the macro definition "MAX_MAILBOX_LEN" (128).
szServiceMailBoxAddr	char[]	Service email address. The length is defined by the macro definition "MAX_MAILBOX_LEN]" (128).
byRes	unsigned char	Reserved. The length is 256 bytes.

A.26 SADP_RESET_PARAM_V40

Structure about Parameters for Resetting The Password

Member	Data Type	Description
dwSize	unsigned int	Structure size.
byResetType	unsigned char	Password resetting type: 0-reserved, 1-restore to the default password via the device serial No. (obsolete), 2-importing and exporting files, 3-QR code, 4-GUID, 5-security question, 6-email.
byEnableSyncIPCPW	unsigned char	Whether to simultaneously enable the function of synchronizing the password of NVR with the password of IPC: 0-disable, 1-enable.
byRes2	unsigned char[]	Reserved. The length is 2 bytes.
szPassword	char[]	User password.
szCode	char[]	Special strings of the date that has been converted, or the string that has been encrypted via the encryption tool. It is valid when the value of byResetType is 2, 3, 6 and 7.
szAuthFile	char[]	The license file for resetting the password, it is valid when the value of byResetType is 2.
szGUID	char[]	GUID, it is valid when the value of byResetType is 4.
SADP_SECURITY_QUESTION_CFG	unsigned char	Security question configuration structure, it is valid when the value of byResetType is 5. See details in ***SADP_SECURITY_QUESTION_CFG*** .
byRes	unsigned char[]	Reserved. The length is 512 bytes.

A.27 SADP_RESET_PARAM_V50

Structure about Parameters for Resetting The Password

Member	Data Type	Description
dwSize	unsigned int	Structure size.
szPassword	char[]	User password. The length is defined by the macro definition "MAX_PASS_LEN_V31" (128).
szCode	char[]	Special strings of the date that has been converted, or the string that has been encrypted via the encryption tool. It is valid when the value of byResetType is 2, 3, 6 and 7. The length is defined by the macro definition "MAX_ ENCRYPT_CODE_V31" (1024).
szAuthFile	char[]	The license file for resetting the password, and it is valid when the value of byResetType is 2. The length is defined by the macro definition "MAX_FILE_ PATH_LEN" (260).
szGUID	char[]	GUID, and it is valid when the value of byResetType is 4. The length is defined by the macro definition "MAX_GUID_LEN_ V31" (512)。
SADP_SECURITY_QUESTION_ CFG	unsigned char	Structure about the security question configuration, and it is valid when the value of byResetType is 5. See details in ***SADP_SECURITY_QUESTION_CFG*** .
byResetType	unsigned char	Password resetting type: 0-reserved, 1-restore to the default password via the device

Member	Data Type	Description
		serial No. (obsolete), 2-importing and exporting files, 3-QR code, 4-GUID, 5-security question, 6- email, 7-scan QR code.
byEnableSyncIPCPW	unsigned char	Whether to simultaneously enable the function of synchronizing the password of NVR with the password of IPC: 0-disable, 1-enable.
byRes	unsigned char[]	Reserved. The length is 510 bytes.

A.28 SADP_SAFE_CODE

Structure about Parameters of Getting The Device Security Code

Member	Data Type	Description
dwCodeSize	DWORD	Device returned code length.
szDeviceCode	char[]	Device returned code. The length is defined by the macro definition "MAX_DEVICE_CODE" (128).
byRes	BYTE[]	Reserved. The length is 128 bytes.

A.29 SADP_SECURITY_QUESTION

Structure about Failing to Set the Security Question

Member	Data Type	Description
byRetryAnswerTime	unsigned char	Remaining attempts for setting the security question.
bySurplusLockTime	unsigned char	Remaining number of minutes for the user locking. The parameter is valid when the user is locked.
byRes	unsigned char[]	Reserved. The length is 254 bytes.

A.30 SADP_RET_RESET_PARAM_V40

Structure about the Returned Parameters for Resetting The Password

Member	Data Type	Description
byRetryGUIDTime	unsigned char	Remaining attempts to edit the returned parameters for resetting the password.
bySurplusLockTime	unsigned char	Remaining time, unit: minute. The parameter is valid when the user is locked.
bRetryTimeValid	unsigned char	Whether the filed byRetryGUIDTime is valid: 0-no, 1-yes.
bLockTimeValid	unsigned char	Whether the filed bySurplusLockTime is valid: 0-no, 1-yes.
byRes	unsigned char	Reserved. The length is 252 bytes.

A.31 SADP_SECURITY_QUESTION_CFG

Structure about the Security Question Configuration

Member	Data Type	Description
dwSize	unsigned int	Structure size.
struSecurityQuestion	***SADP_SINGLE_SECURITY_QUESTION_CFG*** []	Security question list. The length is defined by the macro definition "MAX_QUESTION_LIST_LEN" (32).
szPassword	char[]	Password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 512 bytes.

A.32 SADP_SELF_CHECK_STATE

Structure about the Device Self-Inspection Status

Member	Data Type	Description
dwSize	unsigned int	Structure size.
dwTotalDisk	int	The total number of disks, the default value is 1.
dwGoodDisk	int	The number of good disk, the default value is 1.
szCPU	char[]	CPU. The length is defined by the macro definition "MAX_CPU_LEN" (32).
szMemory	char[]	Memory. The length is defined by the macro definition "MAX_MEMORY_LEN" (32).
byProgress	unsigned char	Self-inspection progress, value range: [0,100].
byTemperatureState	unsigned char	Temperature status: 0-invalid, 1-normal, 2-exception.

Member	Data Type	Description
byFanState	unsigned char	Fan status: 0-invalid, 1-normal, 2-exception.
byPowerState	unsigned char	Power status: 0-invalid, 1-normal, 2-exception.
bySASConnectState	unsigned char	SAS connection status: 0-invalid, 1-connected, 2-disconnected.
byTotalNetworkPort	char[]	The total number of network interfaces, the default value is 1.
byConnectNetworkPort	char[]	The number of connected network interfaces, the default value is 1.
byRes	unsigned char[]	Reserved. The length is 129 bytes.

A.33 SADP_SINGLE_SECURITY_QUESTION_CFG

Structure about the Security Question

Member	Data Type	Description
dwSize	unsigned int	Structure size.
dwId	unsigned int	Security question No.
szAnswer	char[]	Answer (valid when it is set, invalid when it is gotten). The length is defined by the macro definition "MAX_ANSWER_LEN" (256).
byMark	unsigned char	Mark whether the question is set: 0-no, 1-yes
byRes	unsigned char[]	Reserved. The length is 127 bytes.

A.34 SADP_TYPE_UNLOCK_CODE

Structure about the Device Model Unlock Code

Member	Data Type	Description
dwCodeSize	unsigned int	Device model unlock code length
szDeviceTypeUnlockCode	char[]	Device model unlock code content. The length is defined by the macro definition "MAX_UNLOCK_CODE_RANDOM_LEN" (256).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.35 SADP_USER_MAILBOX

Email Configuration Structure

Member	Data Type	Description
dwSize	unsigned int	Structure size.
szPassword	char[]	Password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
szMailBoxAddr	char[]	Reserved email address for receiving security code returned by password server. The length is defined by the macro definition "MAX_MAILBOX_LEN" (128).
byRes	unsigned char[]	Reserved. The length is 127 bytes.

A.36 SADP_VERIFICATION_CODE_INFO

Structure about the Device Verification Code

Member	Data Type	Description
dwSize	unsigned int	Structure size.
szVerificationCode	char[]	Verification code, it is used as the only code for login, live view, and control. The length is defined by the macro definition "SADP_MAX_VERIFICATION_CODE_LEN" (12).
szPassword	char[]	Password. The length is defined by the macro definition "MAX_PASS_LEN" (16).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

A.37 SADP_WIFI_REGION_INFO

Structure about the Wi-Fi Area

Member	Data Type	Description
byMode	unsigned char	Mode: 0-reserved, 1-Wi-Fi area configuration mode, 2-Wi-Fi enhancement mode.
byWifiRegion	unsigned char	Wi-Fi area, it is valid when byMode is 1: 0-default, 1-China, 2-North America, 3-Japan, 4-Europe, 5-world.
byWifiEnhancementEnabled	unsigned char	Whether to enable Wi-Fi enhancement mode. It is valid when byMode is 2: 0-disable, 1-enable.
byRes	unsigned char	Reserved. The length is 128 bytes.

Member	Data Type	Description
szPassword	char[]	Password. The maximum length is defined by the macro definition"[MAX_PASS_LEN]" (16).
byRes	unsigned char[]	Reserved. The length is 128 bytes.

Appendix B. Appendixes

B.1 Error Code

Error Name	Error Code	Description
SADP_NOERROR	0	No error.
SADP_ALLOC_RESOURCE_ERROR	2001	Error occurred when allocating resources.
SADP_NOT_START_ERROR	2002	SADP not enabled.
SADP_NO_ADAPTER_ERROR	2003	No NIC.
SADP_GET_ADAPTER_FAIL_ERROR	2004	Getting NIC information failed.
SADP_PARAMETER_ERROR	2005	Parameter error.
SADP_OPEN_ADAPTER_FAIL_ERROR	2006	Opening NIC failed.
SADP_SEND_PACKET_FAIL_ERROR	2007	Sending data failed.
SADP_SYSTEM_CALL_ERROR	2008	Calling the system API failed.
SADP_DENY_OR_TIMEOUT_ERROR	2009	Device denied or timed out.
SADP_NPF_INSTALL_ERROR	2010	Installing NPF service failed.
SADP_TIMEOUT	2011	Timeout.
SADP_CREATE_SOCKET_ERROR	2012	Creating socket failed.
SADP_BIND_SOCKET_ERROR	2013	Binding socket failed.
SADP_JOIN_MULTI_CAST_ERROR	2014	Adding to multicast group failed.
SADP_NETWORK_SEND_ERROR	2015	Error occurred when sending data.
SADP_NETWORK_RECV_ERROR	2016	Error occurred when receiving data.
SADP_XML_PARSE_ERROR	2017	Error occurred when parsing multicast XML.
SADP_LOCKED	2018	Device locked.
SADP_NOT_ACTIVATED	2019	Device not activated.
SADP_RISK_PASSWORD	2020	Risky password.
SADP_HAS_ACTIVATED	2021	Device activated.
SADP_EMPTY_ENCRYPT_STRING	2022	The encrypted string is empty.
SADP_EXPORT_FILE_OVERDUE	2023	The exported file is expired.

Error Name	Error Code	Description
SADP_PASSWORD_ERROR	2024	Incorrect password.
SADP_LONG_SECURITY_ANSWER	2025	The answer of the security question is too long.
SADP_INVALID_GUID	2026	Invalid GUID.
SADP_ANSWER_ERROR	2027	Incorrect answer.
SADP_QUESTION_NUM_ERR	2028	The number of security questions is incorrect.
SADP_LOAD_WPCAP_FAIL	2030	Loading WPCAP failed.
SADP_ILLEGAL_VERIFICATION_CODE	2033	Invalid verification code.
SADP_BIND_ERROR_DEV	2034	Bind incorrect device(s).
SADP_EXTED_MAX_BIND_NUM	2035	Number exceeded limit.
SADP_MAILBOX_NOT_EXIST	2036	Email does not exist.
SADP_MAILBOX_NOT_SET	2038	Set the email address for resetting the password first.
SADP_INVALID_RESET_CODE	2039	Resetting password failed.
SADP_NO_PERMISSION	2040	No permission: 1. For Win&Linux, no administrator permission for operating NIC. 2. For Android&IOS, no permission for multicast.
SADP_GET_EXCHANGE_CODE_ERROR	2041	Failed to get the interchange code used for encryption.
SADP_CREATE_RSA_KEY_ERROR	2042	Failed to generate RSA private and public key.
SADP_BASE64_ENCODE_ERROR	2043	Encoding by BASE64 failed.
SADP_BASE64_DECODE_ERROR	2044	Decoding by BASE64 failed.
SADP_AES_ENCRYPT_ERROR	2045	Failed to encrypt the data via AES key.
SADP_PHONE_NOT_SET	2046	Phone number is not set for verification when reset the password by scanning QR code.
SADP_NOENOUGH_BUF	2047	Insufficient buffer size.
SADP_INVALID_SUBNET_IP	2048	Invalid subnet range. The start IP address is greater than the end IP address, or the number of IP addresses exceeded 4096.


See Far, Go Further