360联运SDK集成说明

版 本：	**V2.0.0.1**
说 明：	新增客户端SDK3.0版本接入逻辑【见3.2】 1.此版本新增任务系统接口【见3.2.3】，支持用户通过完成任务获取权益功能 2.此版本更新了SDK初始化接口、支付接口【见3.2.1、3.2.2】【兼容原有见3.1中的初始化接口、支付接口，无需单独升级】
更新日期：	**2025.07.29**

一、联运SDK接入和上架流程

1. **请先申请开通SSP后台权限，待开通后在操作下一步**

> **Important**
>
> **请查看文档：** 软件管家联运ssp平台注册流程.pdf

2. **登录360联运平台https://openstore.360.cn/（初次使用需要申请开通权限），按照联运平台操作手册，在联运平台创建软件后获取以下字段值进行后续联运对接：**

> **Important**
>
> 测试应用请在创建时选择测试环境，软件名称加上测试字样，测试金额统一使用0.01元
>
> **请查看文档：** 360软件管家联运平台操作手册1.2.pdf

qid	**登录联运平台的360账号ID，建议使用公司公有账号注册，个人账号注册如出现人员调动会参数复杂的解换绑流程**
appid	**联运产品的ID，不同于softid，appid仅用于联运支付做校验**
appsecret	**生成签名用于校验，换取access_token需要**

2. **文档中心下载包含了最新版windows版SDK（开发环境C++）需集成到联运产品内， SDK支持的系统为win7及以上。**

> **Warning**
>
> 32位、64位的SDK文件请勿重命名

SDK文件下载地址：http://softdl.360tpcdn.com/pcrj/SDK_3.0.zip

3. 联调测试通过后，打包联运产品正式版安装包（需区分官方版本，避免互相覆盖），在联运平台进行信息补充后提交，审核通过后即可上架。

二、名词解释

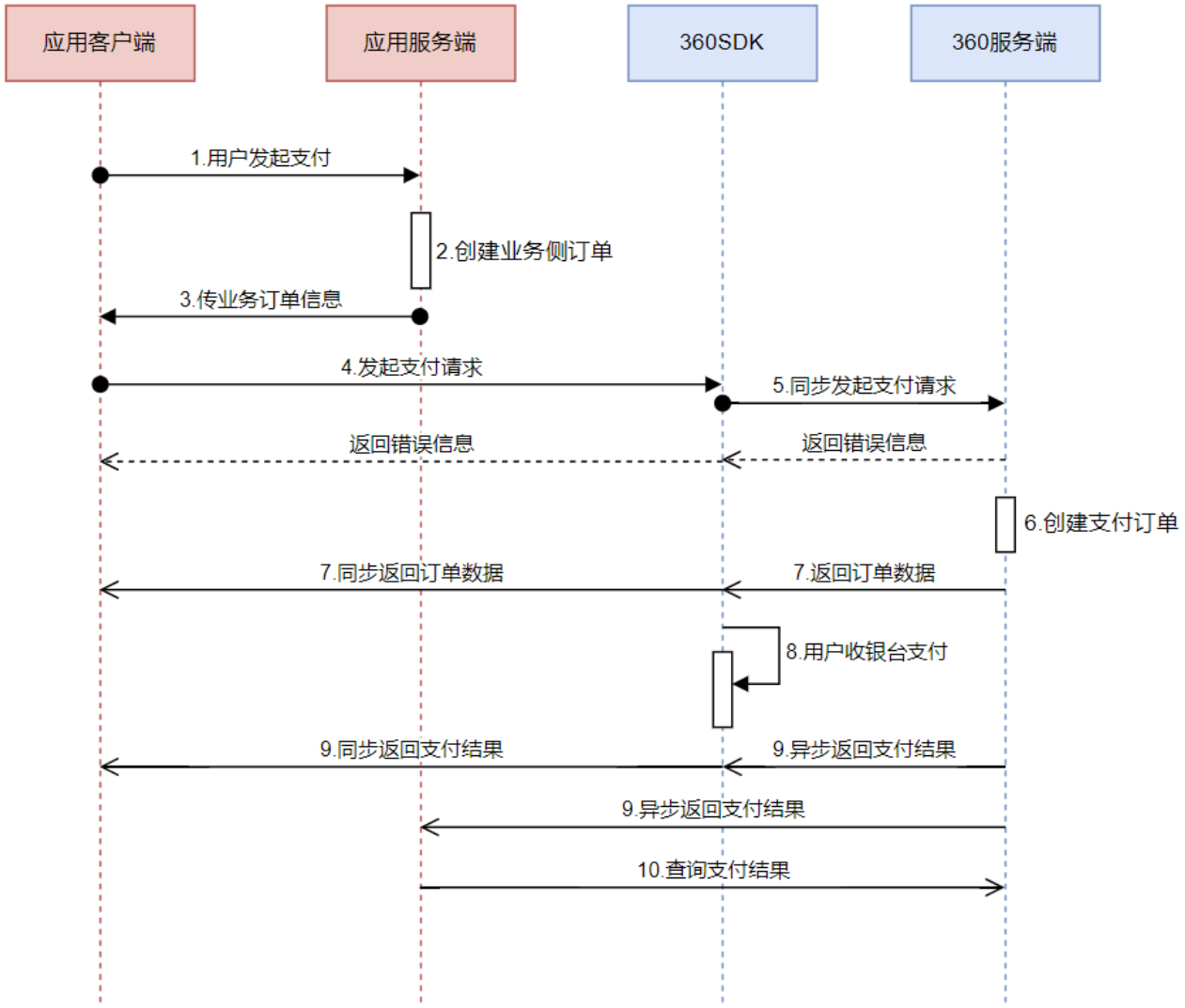
名词	解释说明
订阅SDK	由360提供，用于接入360的各种基础服务
厂商（接入方）	指接入该订阅SDK基础服务的开发者
QID	厂商的唯一标识。注：为安全性考虑，QID请勿硬编码在程序内，请厂商从自身云端获取
access_token	厂商调OPENAPI服务时，需传入此参数，类似于登录态，需通过appid和appsecret获取
appid（应用ID）	在联运开放平台厂商可自行创建应用后生成appid，相当于厂商应用的唯一标识。注：为安全性考虑，appid请勿硬编码在程序内，请厂商从自身云端获取
appsecret（应用秘钥）	在联运开放平台厂商可自行创建应用后生成appsecret，应用的私钥用作数据验证

三、客户端SDK接入说明

> Important
>
> 建议新接入开发者直接使用$SDK3.0$版本接入逻辑

发行包内容	解释说明
订阅SDK动态库	联运SDK 的主要功能载体，包含 360Lysdk.dll，360Base.dll、360NetBase.dll和360Util.dll、cacert.dat厂商应用发布时，需要带上这四个DLL 动态库以及一个DAT库，并确保这五个文件放在一个目录下

接入时序图



3.1.SDK 2.0版本接入逻辑

> Tip
>
> 此版本未接入任务系统，仅支持普通支付以及代扣续费支付逻辑
>
> 建议新接入开发者直接使用$SDK3.0$接入逻辑

3.1.1环境初始化

参数说明：SDK初始化所需环境信息，是一个结构体，包括以下字段：

字段名称	类型	含义	是否必填
dwSize	uint32	EnvInfo结构体大小	是
wszAppId	wstring(128)	appid	是

字段名称	类型	含义	是否必填
u64Qid	uint64	qid	是

3.1.1.1-订阅SDK初始化接口

SDK360_Init

> Important
>
> **说明：在打开软件界面时必须调用*sdk*初始化接口**

调用方法：

```
extern "C" int __stdcall SDK360_Init(const EnvInfo* pEnvInfo);

** 函数功能：SDK的初始化接口，调用支付接口之前必须先初始化
** pEnvInfo[in]：厂商的相关信息，详见EnvInfo结构体定义
```

3.1.1.2-订阅 SDK退出接口

SDK360_UnInit

> Important
>
> **说明：一般在接入方程序结束前调用。**

调用方法：

```
extern "C" int __stdcall SDK360_UnInit();

**函数功能：SDK的反初始化接口，不使用该模块时，调用该接口释放
```

3.1.2支付接口

OderRequest

参数说明：创建支付订单需要厂商提供的相关信息，是一个结构体，包括以下字段：

> Caution
>
> 高亮部分为连续支付新增字段

字段名称	类型	含义	是否必填
dwSize	uint32	OderRequest结构体大小	是
wszOrderId	wstring(40)	厂商订单id，厂商需要保证在同一 应用下订单id的唯一性	是

字段名称	类型	含义	是否必填
dwAmount	uint32	订单金额，单位“分”	是
tOrderCreateTime	__time64_t	厂商创建该订单的时间戳， 后期对账使用	是
wszUserId	wstring(40)	用户ID	是
wszProductId	wstring(40)	商品id，商品在厂商方的标识， 后期对账使用，由厂商提供	是
wszProductDescription	wstring(400)	商品描述，后期对账使用	是
dwDeductionOnOff	uint32	是否开启代购，1:开启; 0:不开启	否
dwDeductionType	uint32	代扣单位，0:按天; 1:按月	否
dwDeductionAmount	uint32	代扣金额，单位“分”	否
dwDeductionPeriod	uint32	代扣周期，与dwDeductionType搭配使用，间隔dwDeductionPeriod×dwDeductionType时间内代扣。如： dwDeductionType=0，dwDeductionPeriod=7，则每7天代扣一次； dwDeductionType=1，dwDeductionPeriod=3，则每3个月代扣一次	否
wszFirstDeductionTime	wstring(11)	首次代扣日期,YYYY-MM-DD格式	否

> **Important**
>
> 原SDK1.1中OderResponse2变更为现SDK2.0中得OderResponse

OderResponse

参数说明：表示创建订单返回的结果。包括以下字段：

字段名称	类型	含义	是否必填
dwSize	uint32	OderResponse结构体大小	厂商必填
dwTicket	uint32	SDK360_Pay或SDK360_AsyncPay接口 返回的值	SDK必填
nErrno	uint32	订单的生成状态，0:生成订单正确: 非0，生成订单错误	SDK必填

字段名称	类型	含义	是否必填
wszMsg	wstring(128)	订单生成状态的详细描述	SDK必填
wszQrCode	wstring(2048)	若订单生成成功，该变量保存订单 支付链接，用于生成二维码	SDK选填
dwAmount	uint32	若订单生成成功，该变量保存订单 金额，与请求金额一致	SDK选填
tExpireTime	__time64_t	若订单生成成功，该变量保存订单 失效的时间戳	SDK选填
dwLive	uint32	若订单生成成功，该变量保存订单 有效时长，单位“秒”。若担心客户 端本地时间不准确，可使用该字段 记录二维码有效期	SDK选填
wszTraceId	wstring(33)	服务端返回的追踪ID，方便后续订单有疑议时，服务端使用此ID最终订单完整链路	SDK选填

3.1.2.1-创建订单接口

说明：使用三方账号支付时调用

SDK360_Pay

函数功能：同步阻塞方式创建订单接口

调用方法：

```
extern "C" DWORD __stdcall SDK360_Pay(const OderRequest& fpOrderRequest, OderResponse& fpOrderResponse, SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack);

** fpOrderRequest[in]: 厂商提供的创建订单所需相关信息，详见OderRequest定义
** fnPayStatusCallBack[in]: 订单创建成功后，支付状态变更的通知回调接口,如不需要回调通知，请传NULL
** fpOrderResponse[out]: 返回的创建订单的详细信息，详见OderResponse定义
** @return [out]: 当前订单的dwTicket值，作为当前订单的唯一id，在OderResponse和回调函数中作为当前支付的关联值
```

SDK360_AsyncPay

函数功能：异步非阻塞方式创建订单接口

调用方法：

```
extern "C" DWORD __stdcall SDK360_AsyncPay(const OderRequest& fpOrderRequest,
SDK360_ORDERRESULT_CALLBACK fnOrderResultCallback, SDK360_PAYSTATUS_CALLBACK
fnPayStatusCallback);

** fpOrderRequest[in]: 厂商提供的创建订单所需相关信息，详见OderRequest定义
** fnOrderResultCallback[in]: 订单创建完成的接口回调函数,如不需要回调通知，请传NULL
** fnPayStatusCallBack[in]: 订单创建成功后，支付状态变更的通知回调接口
** @return [out]: 当前订单的dwTicket值，作为当前订单的唯一id，在OderResponse和回调函数中作为当前支付的关联值
```

3.1.2.2-取消订单接口

SDK360_CancelPay

函数功能：取消订单，SDK在等待用户支付时，会轮询订单状态，在用户支付或订单超时时，通过回调通知厂商。如不希望客户端回调通知，可以调用该接口

调用方法：

```
extern "C" int __stdcall SDK360_CancelPay(DWORD dwTicket);

** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
** @return [out]: 0:取消成功;其它值:取消失败
```

3.1.2.3-创建订单结果通知函数

SDK360_ORDERRESULT_CALLBACK

函数功能：创建订单的结果通知函数

回调方法：

```
typedef void(__stdcall *SDK360_ORDERRESULT_CALLBACK)(DWORD dwTicket, const
OderResponse& fpOderResponse)

** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
** fpOderResponse[in]: 创建订单的详细信息，详见OderResponse定义
```

3.1.2.4-订单支付状态通知回调函数

SDK360_PAYSTATUS_CALLBACK

函数功能：订单支付状通知

回调方法：

```
typedef void(__stdcall *SDK360_PAYSTATUS_CALLBACK)(DWORD dwTicket, int iOderStatus, int iPayChanel);

** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
** iOderStatus[in]: 订单状态 10 -待付款(初始状态) 20-付款完成（待通知厂商） 30-待厂商发权益（已通知厂商） 40-售后中（厂商发起退款） 50-交易完成（正常完成，厂商完成物品发放） 60-已取消（支付超时，过期等原因） 70-交易关闭（退款完成）
** iPayChanel [in]: 支付渠道，1-微信，2-支付宝
```

3.1.3代码样

> **Caution**
>
> // 代扣相关-下代码为连续支付使用
>
> //创建订单返回的结果-有代码变更
>
> //OderResponse2（SDK1.1中）变更为OderResponse（SDK2.0中）

```
// SDK360.h
#pragma once
#pragma pack(1)
// SDK初始化所需环境信息
struct EnvInfo
{
    DWORD dwSize; // EnvInfo结构体大小
    WCHAR wszAppId[128]; // appid ,厂商提供
    ULONGLONG u64Qid; // qid，厂商提供
};

// 创建支付订单需要厂商提供的相关信息
struct OderRequest
{
    DWORD dwSize;                           // 订单请求结构体大小
    WCHAR wszOrderId[40];                   // 厂商的订单id，厂商需要保证在同一应用下订单id的唯一性
    DWORD dwAmount;                         // 订单金额，单位“分”
    __time64_t tOrderCreateTime;            // 厂商创建该订单的时间戳，后期对账使用
    WCHAR wszUserId[40];                    // 用户id，厂商的用户唯一标识，后期对账使用
    WCHAR wszProductId[40];                 // 商品id，商品在厂商方的标识，后期对账使用
    WCHAR wszProductDescription[400];       // 商品描述，后期对账使用

    // 代扣相关
    DWORD dwDeductionOnOff;                  // 是否开启代购，1:开启；0:不开启
    DWORD dwDeductionType;                  // 代扣单位，0:按天；1:按月
    DWORD dwDeductionPeriod;                // 代扣周期，与dwDeductionType搭配使用，间隔dwDeductionPeriod×dwDeductionType时间内代扣；
    // 如dwDeductionType=0，dwDeductionPeriod=7，则每7天代扣一次；dwDeductionType=1，dwDeductionPeriod=3，则每3个月代扣一次
    DWORD dwDeductionAmount;                // 代扣金额，单位“分”
    WCHAR wszFirstDeductionTime[11];        // 首次代扣日期,YYYY-MM-DD格式
};
```

```
31
32  // 创建订单返回的结果。
33  struct OderResponse
34  {
35      DWORD dwSize;                               // 订单返回结构体大小
36      DWORD dwTicket;                             // SDK360_Pay或SDK360_AsyncPay接口返回的值
37      DWORD nErrno;                               // 订单的生成状态，0:生成订单正确: 非0，生成
    订单错误， todo，细化可能的错误码
38      WCHAR wszMsg[128];                          // 订单生成状态的详细描述
39      WCHAR wszQrCode[2048];                      // 若订单生成成功，该变量保存订单支付链接，用
    于生成二维码
40      DWORD dwAmount;                             // 若订单生成成功，该变量保存订单金额，与请求
    金额一致
41      __time64_t tExpireTime;                     // 若订单生成成功，该变量保存订单失效的时间戳
42      DWORD dwLive;                               // 若订单生成成功，该变量保存订单有效时长，单
    位“秒”。若担心客户端本地时间不准确，可使用该字段记录二维码有效期
43      WCHAR wszTraceId[33];                       // 若订单生成成功，该变量用来追踪服务端的订单
    链路
44  };
45
46  #pragma pack()
47  /*
48  ** 函数功能：订单支付状通知
49  ** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
50  ** iOderStatus[in]: 订单状态 10 -待付款(初始状态) 20-付款完成（待通知厂商） 30-待厂商发权
    益（已通知厂商） 40-售后中（厂商发起退款） 50-交易完成（正常完成，厂商完成物品发放） 60-已
    取消（支付超时，过期等原因） 70-交易关闭（退款完成）
51  ** iPayChanel [in]: 支付渠道，1-微信，2-支付宝
52  */
53
54  typedef void(__stdcall *SDK360_PAYSTATUS_CALLBACK)(DWORD dwTicket, int iOderStatus,
    int iPayChanel);
55
56  /*
57  ** 函数功能：创建订单的结果通知函数
58  ** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
59  ** fpOderResponse[in]: 创建订单的详细信息，详见OderResponse定义
60  */
61
62  typedef void(__stdcall *SDK360_ORDERRESULT_CALLBACK)(DWORD dwTicket, const
    OderResponse& fpOderResponse);
63
64  /*
65  ** 函数功能：SDK的初始化接口，调用支付接口之前必须先初始化
66  ** pEnvInfo[in]: 厂商的相关信息，详见EnvInfo结构体定义
67  ** @return [out]: 0:初始化成功;其它值:初始化失败
68  */
69
70  extern "C" int __stdcall SDK360_Init(const EnvInfo* pEnvInfo);
71
72  /*
73  函数功能：SDK的反初始化接口，不使用该模块时，调用该接口释放
```

```
*/

extern "C" int __stdcall SDK360_UnInit();

/*
** 函数功能：同步阻塞方式创建订单接口
** fpOrderRequest[in]: 厂商提供的创建订单所需相关信息，详见OderRequest定义
** fnPayStatusCallBack[in]: 订单创建成功后，支付状态变更的通知回调接口
** fpOrderResponse[out]: 返回的创建订单的详细信息，详见OderResponse定义
** @return [out]: 当前订单的dwTicket值，作为当前订单的唯一id，在OderResponse和回调函数中作为当前支付的关联值
*/

extern "C" DWORD __stdcall SDK360_Pay(const OderRequest& fpOrderRequest, OderResponse& fpOrderResponse, SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack);

/*
** 函数功能：异步非阻塞方式创建订单接口
** fpOrderRequest[in]: 厂商提供的创建订单所需相关信息，详见OderRequest定义
** fnOrderResultCallback[in]: 订单创建完成的接口回调函数
** fnPayStatusCallBack[in]: 订单创建成功后，支付状态变更的通知回调接口
** @return [out]: 当前订单的dwTicket值，作为当前订单的唯一id，在OderResponse和回调函数中作为当前支付的关联值
*/

extern "C" DWORD __stdcall SDK360_AsyncPay(const OderRequest& fpOrderRequest, SDK360_ORDERRESULT_CALLBACK fnOrderResultCallback, SDK360_PAYSTATUS_CALLBACK fnPayStatusCallback);

/*
** 函数功能：取消订单，sdk在等待用户支付时，底层会做轮询。取消订单可节省系统资源占用
** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
** @return [out]: 0:取消成功;其它值:取消失败
*/

extern "C" int __stdcall SDK360_CancelPay(DWORD dwTicket);

typedef int(__stdcall *FUN_SDK360_Init)(const EnvInfo* pEnvInfo);

typedef int(__stdcall *FUN_SDK360_UnInit)();

typedef DWORD(__stdcall *FUN_SDK360_Pay)(const OderRequest& fpOrderRequest, OderResponse& fpOrderResponse, SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack);

typedef DWORD(__stdcall *FUN_SDK360_AsyncPay)(const OderRequest& fpOrderRequest, SDK360_ORDERRESULT_CALLBACK fnOrderResultCallback, SDK360_PAYSTATUS_CALLBACK fnPayStatusCallback);

typedef int(__stdcall *FUN_SDK360_CancelPay)(DWORD dwTicket);

// testsdk.cpp
// testsdk.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。
```

```
//

#include <iostream>
#include <windows.h>
#include "SDK360.h"
#include "time.h"

using namespace std;

void __stdcall Sdk360PaystatusCallback(DWORD dwTicket, int iOderStatus, int
iPayChanel)
{
    cout << "Sdk360PaystatusCallback dwTicket:" << dwTicket << endl;
    cout << "Sdk360PaystatusCallback iOderStatus:" << iOderStatus << endl;
    cout << "Sdk360PaystatusCallback iPayChanel:" << iPayChanel << endl;
}

void __stdcall Sdk360OrderResultCallback(DWORD dwTicket, const OderResponse2&
pOderResponse)
{
    cout << "Sdk360OrderResultCallback dwTicket:" << dwTicket << endl;
    cout << "Sdk360OrderResultCallback pOderResponse nErrno:" << pOderResponse.nErrno
<< endl;
}

int main()
{
HMODULE hModule = LoadLibrary(L"./360Lysdk.dll");
    if (!hModule)
    {
        cout << "LoadLibrary Failed" << endl;
        return -1;
    }
    FUN_SDK360_Init SDK360_Init = (FUN_SDK360_Init)GetProcAddress(hModule,
"SDK360_Init");
    FUN_SDK360_UnInit SDK360_UnInit = (FUN_SDK360_UnInit)GetProcAddress(hModule,
"SDK360_UnInit");
    FUN_SDK360_Pay SDK360_Pay = (FUN_SDK360_Pay)GetProcAddress(hModule,
"SDK360_Pay");
    FUN_SDK360_AsyncPay SDK360_AsyncPay =
(FUN_SDK360_AsyncPay)GetProcAddress(hModule, "SDK360_AsyncPay");
    FUN_SDK360_CancelPay SDK360_CancelPay =
(FUN_SDK360_CancelPay)GetProcAddress(hModule, "SDK360_CancelPay");
    if (!SDK360_Init || !SDK360_UnInit || !SDK360_Pay || !SDK360_AsyncPay ||
!SDK360_CancelPay)
    {
        cout << "GetProcAddress Failed" << endl;
        return -1;
    }

    EnvInfo envInfo;
    envInfo.dwSize = sizeof(EnvInfo);
```

```
    wcscpy_s(envInfo.wszAppId, 128, L"abcde");
    envInfo.u64Qid = 2655065320;

    if (0 != SDK360_Init(&envInfo))
    {
        cout << "SDK360_Init Failed" << endl;
        return -1;
    }

    OderRequest orderRequest;
    memset(&orderRequest, 0, sizeof(OderRequest));
    orderRequest.dwSize = sizeof(OderRequest);
    wcscpy_s(orderRequest.wszOrderId, 40, L"iamoderid_19");
    orderRequest.dwAmount = 99999;
    orderRequest.tOrderCreateTime = time(NULL);
    wcscpy_s(orderRequest.wszUserId, 128, L"iamapquuuerid");
    wcscpy_s(orderRequest.wszProductId, 40, L"iamproductid___");
    wcscpy_s(orderRequest.wszProductDescription, 400, L"iamdescript");

    // 代扣相关
    orderRequest.dwDeductionOnOff = 1;
    orderRequest.dwDeductionType = 1;
    orderRequest.dwDeductionPeriod = 1;
    orderRequest.dwDeductionAmount = 100;
    wcscpy_s(orderRequest.wszFirstDeductionTime, 11, L"2025-01-15");

    OderResponse orderResponse;
    orderResponse.dwSize = sizeof(OderResponse)
    DWORD dwTicket = SDK360_Pay(orderRequest, orderResponse,
Sdk360PaystatusCallback);
    //DWORD dwTicket = SDK360_AsyncPay(orderRequest, Sdk360OrderResultCallback,
Sdk360PaystatusCallback);
    cout << "SDK360_AsyncPay return " << dwTicket << endl;
    if (orderResponse.nErrno == 0)
    {
      // todo 使用orderResponse中的wszQrCode 生成二维码，用户支付情况会在
LysdkPaystatusCallback回调函数中通知
    }

    //SDK360_CancelPay(dwTicket);
    Sleep(100000);
    SDK360_UnInit();

    return 0;
}
```

3.2.SDK 3.0版本接入逻辑

> Tip
>
> **此版本逻辑新增了任务系统，用户可以通过做任务的形式获取对应权益**
>
> **已接入开发者无需变更环境初始化、支付接口接入逻辑，直接接入3.2.3的任务中心即可**

3.2.1环境初始化

字段名称	类型	含义	是否必填
lpcwAppId	wstring(128)	appid	是
u64Qid	uint64	qid	是

3.2.1.1-订阅SDK初始化接口

SDK360_InitEx

> Important
>
> **说明：在打开软件界面时必须调用*sdk*初始化接口**

调用方法：

```
extern "C" int __stdcall SDK360_InitEx(LPCWSTR lpcwAppId, ULONGLONG u64Qid);

** 函数功能：SDK的初始化接口，调用支付接口之前必须先初始化
** @return [out]:   0:初始化成功;其它值:初始化失败,失败时，请检查360Util(64).dll, 360Base(64).dll, 360NetBase(64).dll 与sdk是否在同一目录
```

3.2.1.2-订阅SDK退出接口

SDK360_UnInit

> Important
>
> **说明：一般在接入方程序结束前调用。**

```
extern "C" int __stdcall SDK360_UnInit();

**函数功能：SDK的反初始化接口，不使用该模块时，调用该接口释放
```

3.2.2支付接口

> Important
>
> 此接口等同于3.1中的支付接口，已接入厂商无需变更

3.2.2.1-创建普通订单接口

SDK360_NormalPay

字段名称	类型	含义	是否必填
dwAmount	uint32	订单金额，单位“分”	是
lpcwFactoryOrderId	wstring(40)	厂商的订单id，厂商需要保证在同一应用下订单id的唯一性	是
tOrderCreateTime	__time64_t	厂商创建该订单的时间戳，后期对账使用，单位：秒	是
lpcwUserId	wstring(40)	用户id，厂商的用户唯一标识，后期对账使用	是
lpcwProductI	wstring(40)	商品id，商品在厂商方的标识，后期对账使用	是
lpcwProductDescription	wstring(400)	商品描述，后期对账使用	是
fnPayStatusCallBack	见3.2.2.4	订单创建成功后，当订单状态不是待支付(如20-付款完成 30-待厂商发权益 50-交易完成 60-已取消,支付超时)，调用该回调函数通知厂商，该回调函数可以为null，表示不需要回调通知	是
iSecondTimeOut	int32	接口的超时时间，单位：秒	是

调用方法：

```
extern "C" HANDLE __stdcall SDK360_NormalPay(
    DWORD dwAmount,
    LPCWSTR lpcwFactoryOrderId,
    __time64_t tOrderCreateTime,
    LPCWSTR lpcwUserId,
    LPCWSTR lpcwProductId,
    LPCWSTR lpcwProductDescription,
    SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack,
    int iSecondTimeOut)

 ** 函数功能：生成一个普通的支付订单
```

当接口请求正确时的返回值：

> **[!IMPORTANT]**
>
> **返回一个handle句柄，调用者需要通过基础辅助接口，先根据handle判断接口请求是否成功，在成功的情况下，获取hanlde承载的业务数据。业务数据通常为json格式的字符串。并在使用完数据后，释放句柄。**

SDK360_GetErrno(handle)返回：

```
0
```

SDK360_GetBody(handle)返回:

```
{
  "data" : {
   "expire_time" : "2025-07-28 18:11:31",      //订单的超时时间
   "order_amount" : 120,              // 订单金额
   "order_id" : 1,                //订单ID，回调时通过订单ID来区分是哪个订单的回调
   "qr_code" : http://feat-sm-union.buspay.qihoo.net/smUnion/api/trade/pay?trade_id=93871076&trade_code=194974451614100275 2&token=1949744516157779968&amount=120&et=1753697491&sign=3254d1b2fba3cc474f0f779a5e516bf5   // 支付二维码链接
  },
  "errmsg" : "",
  "errno" : 0    // 错误码，详见错误码说明
}
```

当接口请求不正确时的返回值：

SDK360_GetErrno(handle)返回：

```
详见3.2.5错误码
```

3.2.2.2-创建代扣续费订单接口

SDK360_PayWithDeductions

字段名称	类型	含义	是否必填
dwAmount	uint32	订单金额，单位“分”	是
lpcwFactoryOrderId	wstring(40)	厂商的订单id，厂商需要保证在同一应用下订单id的唯一性	是
tOrderCreateTime	__time64_t	厂商创建该订单的时间戳，后期对账使用，单位：秒	是
lpcwUserId	wstring(40)	用户id，厂商的用户唯一标识，后期对账使用	是
lpcwProductId	wstring(40)	商品id，商品在厂商方的标识，后期对账使用	是
lpcwProductDescription	wstring(400)	商品描述，后期对账使用	是
dwDeductionType	uint32	代扣单位，0:按天;1:按月	是

字段名称	类型	含义	是否必填
dwDeductionPeriod	uint32	代扣周期，与dwDeductionType搭配使用，间隔dwDeductionPeriod×dwDeductionType时间内代扣；如dwDeductionType=0，dwDeductionPeriod=7，则每7天代扣一次；dwDeductionType=1，dwDeductionPeriod=3，则每3个月代扣一次	是
dwDeductionAmount	uint32	代扣金额，单位“分”	是
lpcwFirstDeductionTime	wstring(11)	首次代扣日期,YYYY-MM-DD格式	是
fnPayStatusCallBack	见3.2.2.4	订单创建成功后，当订单状态不是待支付(如20-付款完成30-待厂商发权益50-交易完成60-已取消,支付超时)，调用该回调函数通知厂商，该回调函数可以为null，表示不需要回调通知	是
iSecondTimeOut	int32	接口的超时时间，单位：秒	是

调用方法：

```
extern "C" HANDLE __stdcall* SDK360_PayWithDeductions(
    DWORD dwAmount,
    LPCWSTR lpcwFactoryOrderId,
    __time64_t tOrderCreateTime,
    LPCWSTR lpcwUserId,
    LPCWSTR lpcwProductId,
    LPCWSTR lpcwProductDescription,
    DWORD dwDeductionType,
    DWORD dwDeductionPeriod,
    DWORD dwDeductionAmount,
    LPCWSTR lpcwFirstDeductionTime,
    SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack,
    int iSecondTimeOut);

 ** 函数功能：生成一个有代扣性质的支付订单
```

当接口请求正确时的返回值：

> **[!IMPORTANT]**
>
> **返回一个handle句柄，调用者需要通过基础辅助接口，先根据handle判断接口请求是否成功，在成功的情况下，获取hanlde承载的业务数据。业务数据通常为json格式的字符串。并在使用完数据后，释放句柄。**

SDK360_GetErrno(handle)返回：

```
0
```

SDK360_GetBody(handle)返回:

```
{
  "data" : {
   "expire_time" : "2025-07-28 18:11:31",     //订单的超时时间
   "order_amount" : 120,              // 订单金额
   "order_id" : 1,                //订单ID，回调时通过订单ID来区分是哪个订单的回调
   "qr_code" : http://feat-sm-union.buspay.qihoo.net/smUnion/api/trade/pay?trade_id=93871076&trade_code=194974451614100275 2&token=1949744516157779968&amount=120&et=1753697491&sign=3254d1b2fba3cc474f0f779a5e516bf5   // 支付二维码链接
  },
  "errmsg" : "",
  "errno" : 0    // 错误码，详见错误码说明
}
```

当接口请求不正确时的返回值：

SDK360_GetErrno(handle)返回：

```
详见3.2.5错误码
```

3.2.2.3-创建订单结果通知函数

SDK360_ORDERRESULT_CALLBACK

函数功能：创建订单的结果通知函数

回调方法：

```
typedef void(__stdcall *SDK360_ORDERRESULT_CALLBACK)(DWORD dwTicket, const OderResponse& fpOderResponse)

** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
** fpOderResponse[in]: 创建订单的详细信息，详见OderResponse定义
```

3.2.2.4-订单支付状态通知回调函数

SDK360_PAYSTATUS_CALLBACK

函数功能：订单支付状通知

回调方法：

```
typedef void(__stdcall *SDK360_PAYSTATUS_CALLBACK)(DWORD dwTicket, int iOderStatus, int iPayChanel);

** dwTicket[in]: SDK360_Pay或SDK360_AsyncPay接口返回的值
** iOderStatus[in]: 订单状态 10 -待付款(初始状态) 20-付款完成（待通知厂商） 30-待厂商发权益（已通知厂商） 40-售后中（厂商发起退款） 50-交易完成（正常完成，厂商完成物品发放） 60-已取消（支付超时，过期等原因） 70-交易关闭（退款完成）
** iPayChanel [in]: 支付渠道，1-微信，2-支付宝
```

3.2.3任务系统接口

> [!IMPORTANT]
>
> 所有任务接口都会返回一个handle句柄，调用者需要通过基础辅助接口，先根据handle判断接口请求是否成功，在成功的情况下，获取hanlde承载的业务数据。业务数据通常为json格式的字符串。并在使用完数据后，释放句柄。

3.2.3.1-获取任务列表

SDK360_GetTaskList

字段名称	类型	含义	是否必填
iSecondTimeOut	int32	接口的超时时间，单位：秒	是

调用方法：

```
extern "C" HANDLE __stdcall SDK360_GetTaskList(
    int iSecondTimeOut
);

** 函数功能：获取任务列表
```

当接口请求正确时的返回值：

SDK360_GetErrno(handle)返回：

```
0
```

SDK360_GetBody(handle)返回:

```
{
  "data":[
   {
     "task_amount" : 0,                        //任务金额
     "task_desc" : "任务3描述",                 //任务描述
     "task_icon" : "",                         //任务icon
     "task_id" : "task12",                     //任务id
     "task_name" : ""                          //任务名称
   },
   {
     "task_amount" : 0,
     "task_desc" : "任务3描述",
     "task_icon" : "",
     "task_id" : "task13",
     "task_name" : ""
   },
   {
     "task_amount" : 0,
     "task_desc" : "任务3描述",
```

```
      "task_icon" : "",
      "task_id" : "task14",
      "task_name" : ""
    },
    {
      "task_amount" : 0,
      "task_desc" : "任务3描述",
      "task_icon" : "",
      "task_id" : "task15",
      "task_name" : ""
    }
  ],
  "errmsg" : "",
  "errno" : 0    // 错误码，详见错误码说明
}
```

当接口请求不正确时的返回值：

SDK360_GetErrno(handle)返回：

```
详见3.2.5错误码
```

3.2.3.2-领取任务

SDK360_GenTaskOrder

字段名称	类型	含义	是否必填
lpcwTaskId	wstring(64)	任务ID，通过SDK360_GetTaskList接口获取到的任务ID	是
lpcwFactoryOrderId	wstring(40)	厂商的订单id，厂商需要保证在同一应用下订单id的唯一性	是
lpcwUserId	wstring(40)	用户id，厂商的用户唯一标识，后期对账使用	是
lpcwProductId	wstring(40)	商品id，商品在厂商方的标识，后期对账使用	是
lpcwProductDescription	wstring(400)	商品描述，后期对账使用	是
lpcwExt	wstring(2048)	预留字段	是
iSecondTimeOut	int32	接口的超时时间，单位：秒	是

调用方法：

```
extern "C" HANDLE __stdcall* SDK360_GenTaskOrder(
     LPCWSTR lpcwTaskId,
     LPCWSTR lpcwFactoryOrderId,
     LPCWSTR lpcwUserId,
     LPCWSTR lpcwProductId,
     LPCWSTR lpcwProductDescription,
     LPCWSTR lpcwExt,
     int iSecondTimeOut);

 ** 函数功能：领取一个任务
```

当接口请求正确时的返回值：

SDK360_GetErrno(handle)返回

```
0
```

SDK360_GetBody(handle)返回:

```
{
    "data" :{
    "order_amount": 100,   //订单金额，单位分
    "order_id": 1,      //订单id
    "task_info": {   //任务信息 仅当为 0元任务单时有此字段
        "task_type": 0,   //任务类型  0：url
        "task_content": "https://360.cn/?a=1"  //任务内容
        }
    },
    "errmsg" : "",
    "errno" : 0    // 错误码，详见错误码说明
}
```

当接口请求不正确时的返回值：

SDK360_GetErrno(handle)返回：

```
详见3.2.5错误码
```

3.2.3.3-获取领取任务的状态

SDK360_GetTaskOrderStatus

字段名称	类型	含义	是否必填
i64TaskOrderId	int64	领取任务的订单ID	是
lpcwExt	wstring(2048)	预留字段	是
iSecondTimeOut	int32	接口的超时时间，单位：秒	是

调用方法：

```
extern "C" HANDLE __stdcall* SDK360_GetTaskOrderStatus(__int64 i64TaskOrderId,  LPCWSTR lpcwExt, int iSecondTimeOut);

** 函数功能：获取已领取任务的状态
```

当接口请求正确时的返回值：

SDK360_GetErrno(handle)返回

```
0
```

SDK360_GetBody(handle)返回:

```
{
    "data":{
        "task_status": 0,   //订单状态 0:未完成(进行中) 1:领取成功 2:领取失败
        },
    "errmsg" : "",
    "errno" : 0    // 错误码，详见错误码说明
}
```

当接口请求不正确时的返回值：

SDK360_GetErrno(handle)返回：

```
详见3.2.5错误码
```

3.2.3.4-获取任务中心列表

SDK360_GetTaskCenterList

字段名称	类型	含义	是否必填
iSecondTimeOut	int32	接口的超时时间，单位：秒	是

调用方法：

```
extern "C" HANDLE __stdcall* SDK360_GetTaskCenterList(int iSecondTimeOut);

**函数功能：获取任务中心列表
```

当接口请求正确时的返回值：

SDK360_GetErrno(handle)返回

```
0
```

SDK360_GetBody(handle)返回:

```
{
```

```
  "data" : [
       {
        "order_id" : 2532,       //任务订单id
        "task_amount" : 100,     //任务金额
        "task_desc" : "这是描述",  //任务描述
        "task_id" : "task_id1",   //任务id
        "task_info" : {
             "task_content" : "https://wan.ludashi.com/special/task-center-360/index.html?appid=360ruanguan&mid2=796043764be000f06a1920c4a97eb729&task_id=1&ext=CxhzkheC0LJOGiHqfiEJTgDg0x4EiQaBiwb5HDyTiy8WItwZfhJmxHTiaf1lh-qNuXSfpC-gttjt8xPhJa33Tt2In6a__0Nlv6UgZu1rxoc",   //任务内容
             "task_type" : 0  //任务类型  0：url
         },
        "task_name" : "任务1",   //任务名称
        "task_status" : 1   //任务单状态 0:未完成(进行中) 1:已完成 2:任务失败
       }
    ],
    "errmsg" : "",
    "errno" : 0
}
```

当接口请求不正确时的返回值：

SDK360_GetErrno(handle)返回：

```
详见3.2.5错误码
```

3.2.4基础辅助接口

3.2.4.1-关闭句柄接口

SDK360_CloseHandle

字段名称	类型	含义	是否必填
handle	**void***	**调用业务接口返回的handle句柄**	**是**

调用方法：

```
extern "C" void __stdcall SDK360_CloseHandle(
    HANDLE handle
);

**函数功能：释放请求结果句柄
```

3.2.4.2-获取错误码

SDK360_GetErrno

字段名称	类型	含义	是否必填
handle	void*	调用业务接口返回的handle句柄	是

调用方法：

```
extern "C" int __stdcall SDK360_GetErrno(
    HANDLE handle
);

** 函数功能：获取请求的错误码
** @return  [out]:   0表示请求成功，非0表示请求过程出现错误
```

3.4.2.3-获取错误详细信息

SDK360_GetErrorMsg

字段名称	类型	含义	是否必填
handle	void*	调用业务接口返回的handle句柄	是

调用方法：

```
extern "C" LPCWSTR __stdcall SDK360_GetErrorMsg(
    HANDLE handle
);

** 函数功能：获取请求的错误信息
** @return [out]: 当SDK360_GetErrno返回为0时，该接口通常返回空字符串。当SDK360_GetErrno返回值不为0时，该接口返回errno的错误说明
```

3.2.4.4-获取http状态码

SDK360_GetHttpCode

字段名称	类型	含义	是否必填
handle	void*	调用业务接口返回的handle句柄	是

调用方法：

```
extern "C" int __stdcall SDK360_GetHttpCode(
    HANDLE handle
);

** 函数功能：获取sdk向服务端发送http请求时的http状态码，通常200表示正确。该接口调试时使用，生产环境只需判断SDK360_GetErrno是否返回0即可
** @return [out]: 当请求走到http请求阶段时，会返回http的状态码，否则返回0
```

3.2.4.5-获取服务端返回http响应头

SDK360_GetHeader

字段名称	类型	含义	是否必填
handle	**void***	**调用业务接口返回的handle句柄**	**是**

调用方法：

```
extern "C" LPCWSTR __stdcall SDK360_GetHeader(
    HANDLE handle
);

** 函数功能：获取请求到的http响应头
** @return [out]: 当接口请求正确时，返回http响应头
```

3.2.4.6-获取服务端返回的http响应头长度

SDK360_GetHeaderLength

字段名称	类型	含义	是否必填
handle	**void***	**调用业务接口返回的handle句柄**	**是**

调用方法：

```
extern "C" int __stdcall SDK360_GetHeaderLength(
    HANDLE handle
);

** 函数功能：获取请求到的http响应头长度
** @return [out]: 当接口请求正确时，返回http响应头长度，以字符为单位，不包含\0
```

3.2.4.7-获取本次请求的traceid

> **[!IMPORTANT]**
>
> **查找问题上下游来源信息，建议调用，否则问题无法追溯**

SDK360_GetTid

字段名称	类型	含义	是否必填
handle	void*	调用业务接口返回的handle句柄	是

调用方法：

```
extern "C" LPCWSTR __stdcall SDK360_GetTid(
    HANDLE handle
);


** 函数功能：获取本次请求的Traceid。Traceid为软管服务端的跟踪排查id，当对某次请求返回的结果有疑议时，需提供本次请求的Traceid
** @return [out]: 当接口请求完成时，获取本次请求的Traceid
```

3.2.4.8-获取业务数据

SDK360_GetBody

字段名称	类型	含义	是否必填
handle	void*	调用业务接口返回的handle句柄	是

调用方法：

```
extern "C" LPCWSTR __stdcall SDK360_GetBody(
    HANDLE handle
);

** 函数功能：获取请求到的结果
** @return [out]: 当接口请求正确时，返回http响应体
```

3.2.4.9-获取业务数据长度

SDK360_GetBodyLength

字段名称	类型	含义	是否必填
handle	void*	调用业务接口返回的handle句柄	是

调用方法：

```
extern "C" int __stdcall SDK360_GetBodyLength(
    HANDLE handle
);

** 函数功能：获取请求到的结果长度
** @return [out]: 当接口请求正确时，返回http响应体长度，以字符为单位，不包含\0
```

3.2.5错误码说明

code	含义
0	成功
1-99	查询CURL错误码，参考链接：libcurl - Error Codes
100	http状态码非200错误
10001	参数错误
10002	内部错误
10003	数据不存在
10005	请求体错误
10006	sign错误
10007	请求过期
10008	解密失败
10009	应用参数错误
10010	响应数据错误
10013	权限拒绝
10014	非法访问
10015	不支持的content_type
10016	参数类型错误
10017	不支持的接入模式
10018	触发限流错误
10020	qid错误
10021	厂商应用信息不匹配
104000	任务领取失败
104201	任务处于冷却期
104202	任务不存在
104203	任务尚未开始
104204	任务已经结束
104205	任务库存已经发放完毕
14206	任务未绑定，请先绑定任务

code	含义
14207	任务奖励已领取
14208	任务已失效
14209	任务未领取，请先领取任务
14210	操作太快，请稍后再试
14211	任务条件未满足，请继续完成游戏任务
14212	领取失败,请稍后重试

3.2.6代码样例

```
// SDK360.h
#pragma once
#include <Windows.h>
/*
** 函数功能：订单支付状通知
** dwTicket[in]:  LYSDK_Pay或LYSDK_AsyncPay接口返回的值
** iOderStatus[in]:  订单状态，订单状态 10 -待付款(初始状态) 20-付款完成（待通知厂商） 30-待厂商发权益    （已通知厂商） 40-售后中（厂商发起退款） 50-交易完成（正常完成，厂商完成物品发放） 60-已取消    （支付超时，过期等原因） 70-交易关闭（退款完成）
** iPayChanel [in]:  支付渠道，1-微信，2-支付宝
*/
typedef void(__stdcall *SDK360_PAYSTATUS_CALLBACK)(DWORD dwTicket, int iOderStatus, int iPayChanel);

/*
** 函数功能：SDK的初始化接口，调用支付接口之前必须先初始化
** lpcwAppId    [in]        appid ,厂商提供
** u64Qid     [in]        qid，厂商提供
** @return [out]:   0:初始化成功;其它值:初始化失败,失败时，请检查360Util(64).dll, 360Base(64).dll, 360NetBase(64).dll 与sdk是否在同一目录
*/
typedef int (__stdcall* FUN_SDK360_InitEx)(LPCWSTR lpcwAppId, ULONGLONG u64Qid);

/*
函数功能：SDK的反初始化接口，不使用该模块时，调用该接口释放
*/
typedef int (__stdcall* FUN_SDK360_UnInitEx)();

/*
** 函数功能:          获取任务列表
** iSecondTimeOut  [in]:    接口的超时时间
*/
typedef HANDLE (__stdcall* FUN_SDK360_GetTaskList)(int iSecondTimeOut);

/*
** 函数功能:          获取已领取任务的状态
```

```
** i64TaskOrderId  [in]:    领取任务的订单ID
** lpcwExt         [in]:    预留字段
** iSecondTimeOut  [in]:    接口的超时时间
*/
typedef HANDLE (__stdcall* FUN_SDK360_GetTaskOrderStatus)(__int64 i64TaskOrderId,
LPCWSTR lpcwExt, int iSecondTimeOut);

/*
** 函数功能:          获取任务中心列表
** iSecondTimeOut  [in]:    接口的超时时间
*/
typedef HANDLE (__stdcall* FUN_SDK360_GetTaskCenterList)(int iSecondTimeOut);

/*
** 函数功能:              领取一个任务
** lpcwTaskId        [in]:    任务ID，通过SDK360_GetTaskList接口获取到的任务ID
** lpcwFactoryOrderId    [in]:    厂商的订单id，厂商需要保证在同一应用下订单id的唯一性
** lpcwUserId        [in]:    用户id，厂商的用户唯一标识，后期对账使用
** lpcwProductId      [in]:    商品id，商品在厂商方的标识，后期对账使用
** lpcwProductDescription  [in]:    商品描述，后期对账使用
** lpcwExt         [in]:    预留字段
** iSecondTimeOut      [in]:    接口的超时时间
*/
typedef HANDLE (__stdcall* FUN_SDK360_GenTaskOrder)(
        LPCWSTR lpcwTaskId,
        LPCWSTR lpcwFactoryOrderId,
        LPCWSTR lpcwUserId,
        LPCWSTR lpcwProductId,
        LPCWSTR lpcwProductDescription,
        LPCWSTR lpcwExt,
        int iSecondTimeOut);

/*
** 函数功能:              生成一个普通的支付订单
** dwAmount         [in]:    订单金额，单位“分”
** lpcwFactoryOrderId    [in]:    厂商的订单id，厂商需要保证在同一应用下订单id的唯一性
** tOrderCreateTime     [in]:    厂商创建该订单的时间戳，后期对账使用，单位：秒
** lpcwUserId        [in]:    用户id，厂商的用户唯一标识，后期对账使用
** lpcwProductId      [in]:    商品id，商品在厂商方的标识，后期对账使用
** lpcwProductDescription  [in]:    商品描述，后期对账使用
** fnPayStatusCallBack   [in]:    订单创建成功后，当订单状态不是待支付(如20-付款完成 30-待厂商发权益 50-交易完成 60-已取消,支付超时),调用该回调函数通知厂商，该回调函数可以为null，表示不需要回调通知
** iSecondTimeOut      [in]:    接口的超时时间
*/
typedef HANDLE (__stdcall* FUN_SDK360_NormalPay)(
    DWORD dwAmount,
    LPCWSTR lpcwFactoryOrderId,
    __time64_t tOrderCreateTime,
    LPCWSTR lpcwUserId,
    LPCWSTR lpcwProductId,
    LPCWSTR lpcwProductDescription,
```

```
    SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack,
    int iSecondTimeOut);

/*
** 函数功能:              生成一个有代扣性质的支付订单
** dwAmount         [in]:    订单金额，单位“分”
** lpcwFactoryOrderId    [in]:    厂商的订单id，厂商需要保证在同一应用下订单id的唯一性
** tOrderCreateTime     [in]:    厂商创建该订单的时间戳，后期对账使用，单位：秒
** lpcwUserId        [in]:    用户id，厂商的用户唯一标识，后期对账使用
** lpcwProductId      [in]:    商品id，商品在厂商方的标识，后期对账使用
** lpcwProductDescription  [in]:    商品描述，后期对账使用
** dwDeductionType     [in]:    代扣单位，0:按天; 1:按月
** dwDeductionPeriod    [in]:    代扣周期，与dwDeductionType搭配使用，间隔dwDeductionPeriod×dwDeductionType时间内代扣;
**           如dwDeductionType=0，dwDeductionPeriod=7，则每7天代扣一次;dwDeductionType=1，dwDeductionPeriod=3，则每3个月代扣一次
** dwDeductionAmount    [in]:    代扣金额，单位“分”
** lpcwFirstDeductionTime  [in]:    首次代扣日期,YYYY-MM-DD格式
** fnPayStatusCallBack   [in]:    订单创建成功后，当订单状态不是待支付(如20-付款完成 30-待厂商发权益 50-交易完成 60-已取消,支付超时),调用该回调函数通知厂商，该回调函数可以为null，表示不需要回调通知
** iSecondTimeOut      [in]:    接口的超时时间
*/
typedef HANDLE (__stdcall* FUN_SDK360_PayWithDeductions)(
    DWORD dwAmount,
    LPCWSTR lpcwFactoryOrderId,
    __time64_t tOrderCreateTime,
    LPCWSTR lpcwUserId,
    LPCWSTR lpcwProductId,
    LPCWSTR lpcwProductDescription,
    DWORD dwDeductionType,
    DWORD dwDeductionPeriod,
    DWORD dwDeductionAmount,
    LPCWSTR lpcwFirstDeductionTime,
    SDK360_PAYSTATUS_CALLBACK fnPayStatusCallBack,
    int iSecondTimeOut);

/*
** 函数功能:          释放请求结果句柄
** handle      [in]:    调用业务接口返回的handle句柄
*/
typedef void (__stdcall* FUN_SDK360_CloseHandle)(
    HANDLE handle
);

/*
** 函数功能:          获取请求的错误码
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   0表示请求成功，非0表示请求过程出现错误
*/
typedef int (__stdcall* FUN_SDK360_GetErrno)(
    HANDLE handle
```

```
);

/*
** 函数功能:          获取请求的错误信息
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当SDK360_GetErrno返回为0时，该接口通常返回空字符串。当SDK360_GetErrno返回值不为0时，该接口返回errno的错误说明
*/
typedef LPCWSTR (__stdcall* FUN_SDK360_GetErrorMsg)(
    HANDLE handle
);

/*
** 函数功能:          获取sdk向服务端发送http请求时的http状态码，通常200表示正确。该接口调试时使用，生产环境只需判断SDK360_GetErrno是否返回0即可
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当请求走到http请求阶段时，会返回http的状态码，否则返回0
*/
typedef int (__stdcall* FUN_SDK360_GetHttpCode)(
    HANDLE handle
);

/*
** 函数功能:          获取请求到的http响应头
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当接口请求正确时，返回http响应头
*/
typedef LPCWSTR (__stdcall* FUN_SDK360_GetHeader)(
    HANDLE handle
);

/*
** 函数功能:          获取请求到的http响应头长度
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当接口请求正确时，返回http响应头长度，以字符为单位，不包含\0
*/
typedef int (__stdcall* FUN_SDK360_GetHeaderLength)(
    HANDLE handle
);

/*
** 函数功能:          获取本次请求的Traceid。Traceid为软管服务端的跟踪排查id，当对某次请求返回的结果有疑议时，需提供本次请求的Traceid
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当接口请求完成时，获取本次请求的Traceid
*/
typedef LPCWSTR (__stdcall* FUN_SDK360_GetTid)(
    HANDLE handle
);

/*
** 函数功能:          获取请求到的结果
```

```
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当接口请求正确时，返回http响应体
*/
typedef LPCWSTR (__stdcall* FUN_SDK360_GetBody)(
    HANDLE handle
);

/*
** 函数功能:          获取请求到的结果长度
** handle      [in]:    调用业务接口返回的handle句柄
** @return     [out]:   当接口请求正确时，返回http响应体长度，以字符为单位，不包含\0
*/
typedef int (__stdcall* FUN_SDK360_GetBodyLength)(
    HANDLE handle
);

/*
** 函数功能：设置网络请求的代理模式
** bFollowSystem[in]:     是否跟随系统代理配置， TRUE:跟随; FALSE:手动配置代理
** lpcwProxy[in]:       代理服务的详细配置，[http://]ip:port;[[https://]ip:port]格式。当bFollowSystem为TRUE时，忽略此参数。
** @return [out]:       TRUE: 设置成功; FALSE: 设置失败
*/
typedef BOOL (__stdcall* FUN_SDK360_SetProxy)(BOOL bFollowSystem, LPCWSTR lpcwProxy);

// TestLySdk.cpp
#include "SDK360.h"
#include <iostream>
#include <windows.h>
#include <assert.h>
using namespace std;

FUN_SDK360_InitEx SDK360_InitEx = NULL;
FUN_SDK360_UnInitEx SDK360_UnInitEx = NULL;
FUN_SDK360_GetTaskList SDK360_GetTaskList = NULL;
FUN_SDK360_GetTaskOrderStatus SDK360_GetTaskOrderStatus = NULL;
FUN_SDK360_GetTaskCenterList SDK360_GetTaskCenterList = NULL;
FUN_SDK360_GenTaskOrder SDK360_GenTaskOrder = NULL;
FUN_SDK360_NormalPay SDK360_NormalPay = NULL;
FUN_SDK360_PayWithDeductions SDK360_PayWithDeductions = NULL;

FUN_SDK360_CloseHandle SDK360_CloseHandle = NULL;
FUN_SDK360_GetErrno SDK360_GetErrno = NULL;
FUN_SDK360_GetErrorMsg SDK360_GetErrorMsg = NULL;
FUN_SDK360_GetTid SDK360_GetTid = NULL;
FUN_SDK360_GetHttpCode SDK360_GetHttpCode = NULL;
FUN_SDK360_GetHeader SDK360_GetHeader = NULL;
FUN_SDK360_GetHeaderLength SDK360_GetHeaderLength = NULL;
FUN_SDK360_GetBody SDK360_GetBody = NULL;
FUN_SDK360_GetBodyLength SDK360_GetBodyLength = NULL;

```

```
void __stdcall SDK360_PAYSTATUS_callback(DWORD dwOrderId, int iOrderStatus, int iPayChannel)
{
    wcout << L"SDK360_PAYSTATUS_callback Called, dwOrderId:" << dwOrderId << L" OrderStatus:" << iOrderStatus << L" PayChannel:" << iPayChannel << endl;
    return;
}

void PrintResponse(HANDLE handle)
{
    if (!handle)
    {
        wcout << L"Request return NULL" << endl;
        return;
    }
    int iErrno = SDK360_GetErrno(handle);

    if (iErrno != 0)
    {
        // 如果errno大于1000，一般是参数错误，需要在调试阶段解决
        wcout << L"Request return errno:" << iErrno << L"; Msg:" << SDK360_GetErrorMsg(handle) << endl;

        // 如果Tid 不为空，反馈给软管服务端
        wcout << L"Response Tid:" << SDK360_GetTid(handle) << endl;

        // 如果httpcode不为200，反馈给软管服务端
        wcout << L"Response http code:" << SDK360_GetHttpCode(handle) << endl;

        // 查看返回的Header，看看有没有什么错误端倪
        wcout << L"Response Header:" << SDK360_GetHeader(handle) << endl;
    }
    else
    {
        int iBodyLength = SDK360_GetBodyLength(handle);
        LPCWSTR lpcwBody = SDK360_GetBody(handle);
        assert(lpcwBody);
        wcout << L"Response Body Size:" << iBodyLength << endl;
        wcout << L"Response Body:" << lpcwBody << endl;
    }
}

int main()
{
    HMODULE hModule = LoadLibrary(L"./360LySdk.dll");
    if (!hModule)
    {
        wcout << L"LoadLibrary Failed" << endl;
        return -1;
    }

    SDK360_InitEx = (FUN_SDK360_InitEx)GetProcAddress(hModule, "SDK360_InitEx");
```

```
278     SDK360_UnInitEx = (FUN_SDK360_UnInitEx)GetProcAddress(hModule,
    "SDK360_UnInitEx");
279     SDK360_GetTaskList = (FUN_SDK360_GetTaskList)GetProcAddress(hModule,
    "SDK360_GetTaskList");
280     SDK360_GetTaskOrderStatus =
    (FUN_SDK360_GetTaskOrderStatus)GetProcAddress(hModule, "SDK360_GetTaskOrderStatus");
281     SDK360_GetTaskCenterList = (FUN_SDK360_GetTaskCenterList)GetProcAddress(hModule,
    "SDK360_GetTaskCenterList");
282     SDK360_GenTaskOrder = (FUN_SDK360_GenTaskOrder)GetProcAddress(hModule,
    "SDK360_GenTaskOrder");
283     SDK360_NormalPay = (FUN_SDK360_NormalPay)GetProcAddress(hModule,
    "SDK360_NormalPay");
284     SDK360_PayWithDeductions = (FUN_SDK360_PayWithDeductions)GetProcAddress(hModule,
    "SDK360_PayWithDeductions");
285
286     SDK360_CloseHandle = (FUN_SDK360_CloseHandle)GetProcAddress(hModule,
    "SDK360_CloseHandle");
287     SDK360_GetErrno = (FUN_SDK360_GetErrno)GetProcAddress(hModule,
    "SDK360_GetErrno");
288     SDK360_GetErrorMsg = (FUN_SDK360_GetErrorMsg)GetProcAddress(hModule,
    "SDK360_GetErrorMsg");
289     SDK360_GetTid = (FUN_SDK360_GetTid)GetProcAddress(hModule, "SDK360_GetTid");
290     SDK360_GetHttpCode = (FUN_SDK360_GetHttpCode)GetProcAddress(hModule,
    "SDK360_GetHttpCode");
291     SDK360_GetHeader = (FUN_SDK360_GetHeader)GetProcAddress(hModule,
    "SDK360_GetHeader");
292     SDK360_GetHeaderLength = (FUN_SDK360_GetHeaderLength)GetProcAddress(hModule,
    "SDK360_GetHeaderLength");
293     SDK360_GetBody = (FUN_SDK360_GetBody)GetProcAddress(hModule, "SDK360_GetBody");
294     SDK360_GetBodyLength = (FUN_SDK360_GetBodyLength)GetProcAddress(hModule,
    "SDK360_GetBodyLength");
295
296     if (!SDK360_InitEx || !SDK360_UnInitEx || !SDK360_GetTaskList ||
    !SDK360_GetTaskOrderStatus || !SDK360_GetTaskCenterList || !SDK360_GenTaskOrder ||
    !SDK360_NormalPay || !SDK360_PayWithDeductions
297         || !SDK360_CloseHandle || !SDK360_GetErrno || !SDK360_GetErrorMsg ||
    !SDK360_GetHttpCode || !SDK360_GetHeader || !SDK360_GetHeaderLength ||
    !SDK360_GetBody || !SDK360_GetBodyLength)
298     {
299         wcout << L"GetProcAddress Failed" << endl;
300         return -1;
301     }
302     if (SDK360_InitEx(L"yuObot2Y", 2655065320))
303     {
304         wcout << L"SDK360_InitEx Failed, Check 360Base(64).dll 360NetBase(64).dll
    360Util(64).dll is Exist??" << endl;
305         return -1;
306     }
307     HANDLE handle = NULL;
308
309     // 创建一个支付订单
310     handle = SDK360_NormalPay(
```

```
        120,   // 1.20 yuan
        L"useroderid_121",
        time(NULL),
        L"userid_001",
        L"lpcwProductId_001",
        L"lpcwProductDescription_001",
        SDK360_PAYSTATUS_callback,
        60);
    wcout << L"SDK360_NormalPay Called" << endl;
    PrintResponse(handle);
    SDK360_CloseHandle(handle);

    // 创建一个支持代扣的订单，从2025年01月15日开始，每月扣费一次，每次扣费1元
    handle = SDK360_PayWithDeductions(
        120,   // 1.20 yuan
        L"useroderid_122",
        time(NULL),
        L"userid_001",
        L"lpcwProductId_002",
        L"lpcwProductDescription_002",
        1,
        1,
        100,
        L"2025-01-15",
        NULL,
        60);
    wcout << L"SDK360_PayWithDeductions Called" << endl;
    PrintResponse(handle);
    SDK360_CloseHandle(handle);


    // 以下是任务相关接口

    // 获取任务列表
    handle = SDK360_GetTaskList(60);
    wcout << L"SDK360_GetTaskList Called" << endl;
    PrintResponse(handle);
    SDK360_CloseHandle(handle);

    // 领取一个任务
    handle = SDK360_GenTaskOrder(
        L"task12",
        L"useroderid_122",
        L"userid_002",
        L"lpcwProductId_002",
        L"lpcwProductDescription_002",
        L"",
        60);
    wcout << L"SDK360_GenTaskOrder Called" << endl;
    PrintResponse(handle);
    SDK360_CloseHandle(handle);

```

```
    // 获取一个领取任务的状态信息
    handle = SDK360_GetTaskOrderStatus(
        654321,   //SDK360_GenTaskOrder returned order_id
        L"",
        60);
    wcout << L"SDK360_GetTaskOrderStatus Called" << endl;
    PrintResponse(handle);
    SDK360_CloseHandle(handle);

    // 获取用户的任务中心列表
    handle = SDK360_GetTaskCenterList(60);
    wcout << L"SDK360_GetTaskCenterList Called" << endl;
    PrintResponse(handle);
    SDK360_CloseHandle(handle);
    SDK360_UnInitEx();
    return 0;
};
```

四、服务端接口说明

4.1.OPENAPI

接口域名：api.openstore.360.cn

> **[!IMPORTANT]**
>
> **请保存好，在调用返回数据时的Header头信息内的Header-Tid的值，以便出现问题时，快速定位原因（如不提供，则无法跟进处理）**

4.1.1接口请求sign生成规则

参数名	必选	类型	说明
sign	是	string	1）参数排序（升序），参数=参数值”的模式用“&” 字符拼接。 2）md5（a=1&b=2$appsecret$）小写，appsecret由360提供，只有请求授权接口非必填其他接口都为必填

> **[!IMPORTANT]**
>
> **-参数排序（升序），参数=参数值”的模式用“&” 字符拼接。**

> -md5（a=1&b=2$appsecret$）小写，appsecret由360提供
>
> —注意：
>
> 厂商服务端通过回调接口回传数据验证签名时，参数值为空的不能参与$sign$计算
>
> 还请动态解析参数，不进行硬编码

1）如下是PHP生成sign的代码，供厂商参考：

```
function createSign($paramArr,$securityKey)  {
    echo 'base param:'.json_encode($paramArr).PHP_EOL;
if(isset($paramArr['sign'])){
         unset($paramArr['sign']);
    }
    //排序
    ksort($paramArr);
    $calcString = '';
    foreach ($paramArr as $key => $value) {
if($value === ""){
continue;
}
        $calcString .= $key . '=' . $value . '&';
    }
    $step1 = rtrim($calcString,'&').$securityKey;

//echo $step1."----------------------------------------------------------------------ori sign".PHP_EOL;
echo "before md5 str: ".$step1.PHP_EOL;
    $sign = md5($step1);
    return $sign;
}

```

2）如下是Golang生成sign的代码，供厂商参考：

```
func StringMd5(s string) string {
    m := md5.New()
    m.Write([]byte(s))
    md5Str := hex.EncodeToString(m.Sum(nil))
    return md5Str
}

func UniversalBuildSign(paramMap map[string]string, salt string) (afterStr string, beforeStr string) {
    var keys []string
    if _, ok := paramMap["sign"]; ok {
        delete(paramMap, "sign")
    }
    for k := range paramMap {
        if paramMap[k] == "" {
```

```
            continue
        }
        keys = append(keys, k)
    }
    sort.Strings(keys)
    paramStr := ""
    for _, k := range keys {
        paramStr += k + "=" + paramMap[k] + "&"
    }

    paramStr = strings.TrimRight(paramStr, "&") + salt
    md5Str := StringMd5(paramStr)
    return md5Str, paramStr
}
```

> [!CAUTION]
>
> paramArr—为业务请求参数
>
> securityKey-为360提供的appsecret

4.1.2业务接口说明：

4.1.2.1-access_token接口

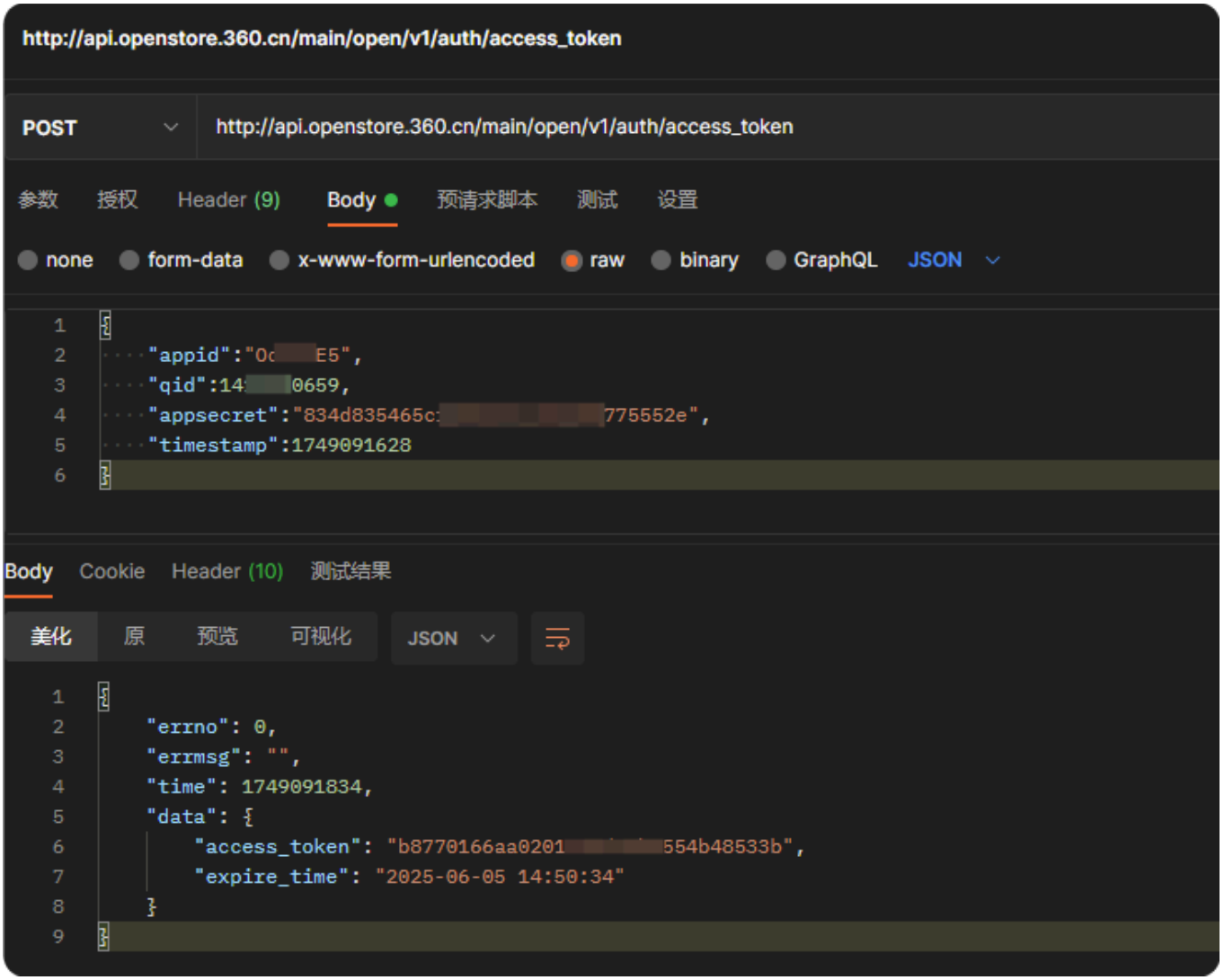
接口地址：/main/open/v1/auth/access_token【POST】【Content-Type=application/json】【应用调用频次限制：3 次/秒】

> [!CAUTION]
>
> 新申请token老的token会失效

参数名	必选	类型	说明
appid	是	string	appid由360提供，同上
timestamp	是	int64	当前时间戳（单位秒）
qid	是	int64	由360提供，同上
appsecret	是	string	appsecret由360提供，只有请求授权接口必填，其他接口都为非必填

```
{
    "errno": 0,
    "errmsg": "",
    "data": {
            "access_token": "xxxxxxx"
            "expire_time": "2023-12-02 21:20:00"
        },
    "time": 0,
}
```

参考：



4.1.2.2-订单退款申请接口

> **Important**
>
> 通过任务系统完成的订单不允许退款

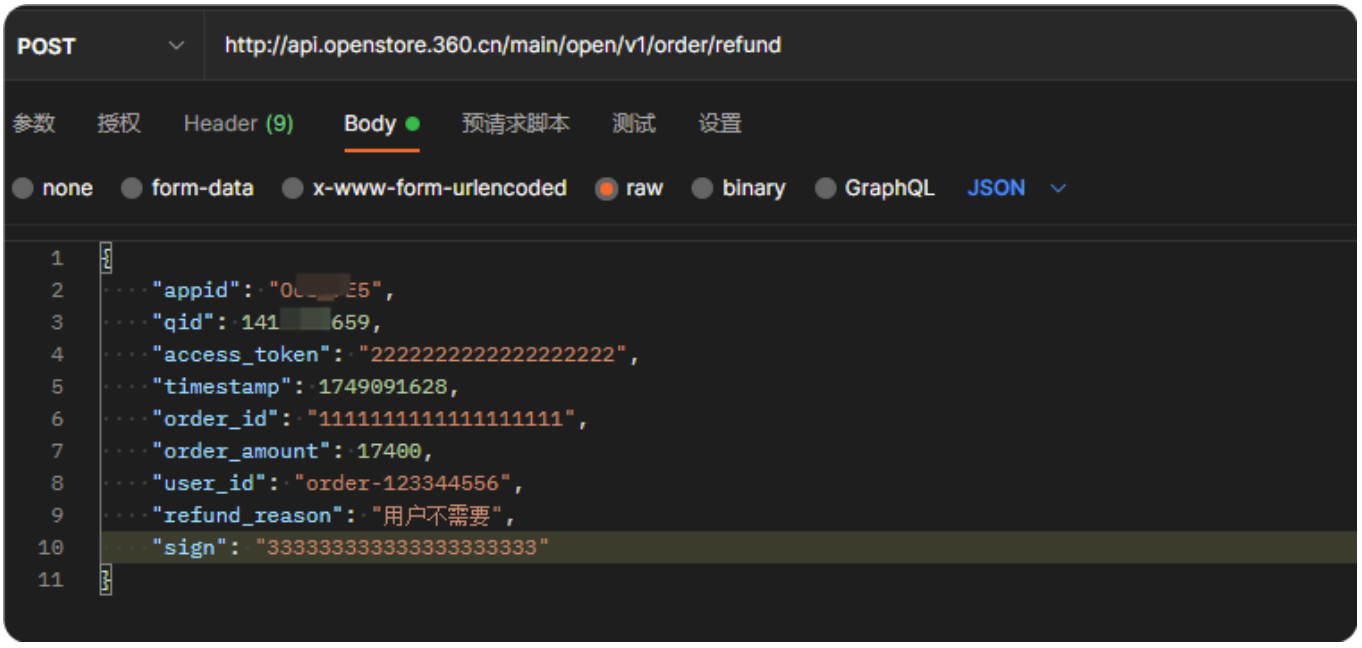
接口地址：/main/open/v1/order/refund【POST】【Content-Type=application/json】【应用调用频次限制：3 次/秒】

参数名	必选	类型	说明
appid	是	string	appid由360提供，同上
timestamp	是	int64	当前时间戳（单位秒）
qid	是	int64	由360提供，同上
access_token	是	string	请求授权接口可得到，只有请求授权接口非必填，其他接口都为必填。生效时间：3小时

参数名	必选	类型	说明
sign	是	string	1）参数排序（升序），参数=参数值”的模式用“&” 字符拼接。 2）md5（a=1&b=2$appsecret$）小写，appsecret由360提供，只有请求授权接口非必填其他接口都为必填
order_id	是	string	厂商调用sdk传入的订单id
order_amount	是	int64	订单金额，单位为分
user_id	是	string	用户id，厂商的用户唯一标识
refund_reason	是	string	申请退款说明，长度200个字符

```
{
    "errno": 0,
    "errmsg": "",
    "data": {},
    "time": 1701658768
}
```

参考：



Sign值md5计算方式：按字段顺序升序，&符链接，最后拼接appsecret值，以上图为例：

```
**【按结构体的字段顺序】**
appid=值&qid=值&access_token=值&timestamp=值&order_id=值&order_amount=值&user_id=值&refund_reason=值appsecret
```

4.1.2.3-订单查询接口

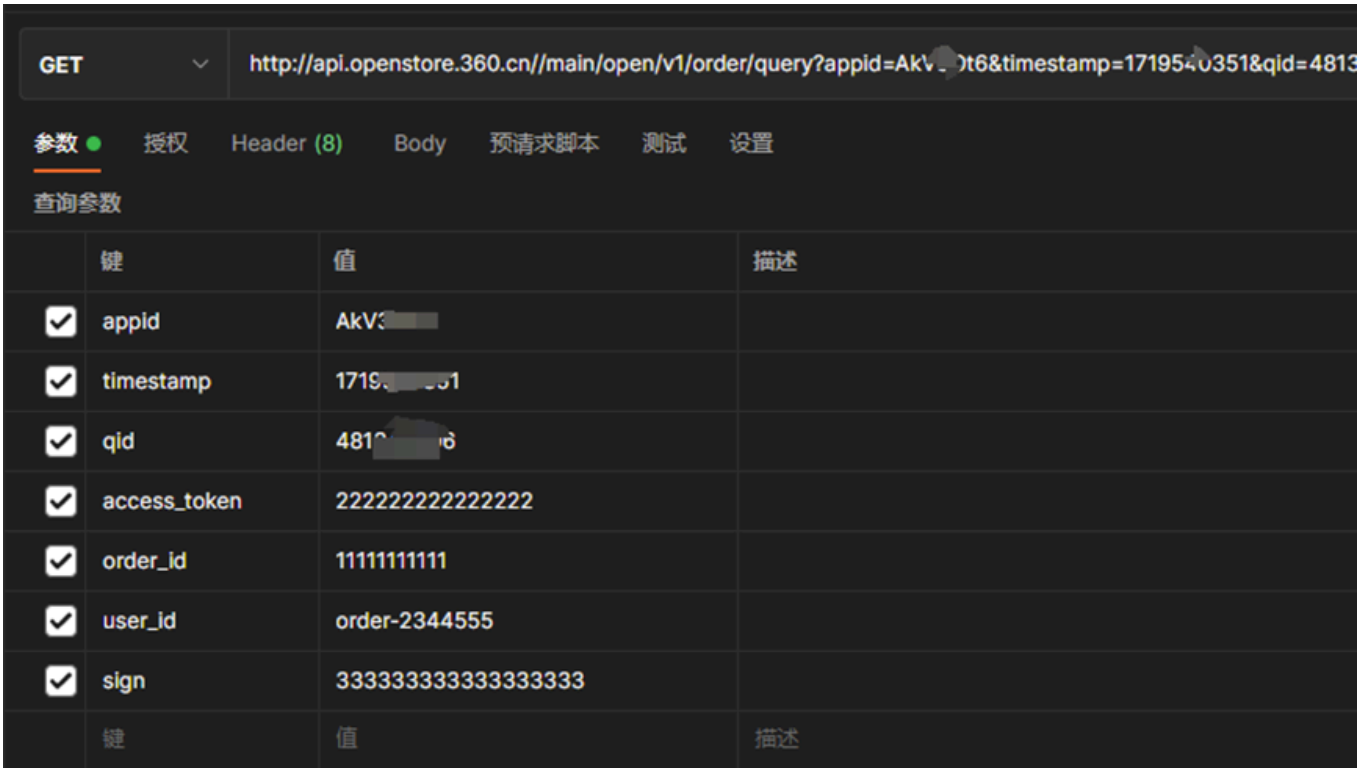
接口地址：/main/open/v1/order/query【GET】【应用调用频次限制：5 次/秒】

> [!WARNING]
>
> 请厂商不要通过客户端轮询此接口，SDK本身提供了订单变更状态的通知

参数名	必选	类型	说明
appid	是	string	appid由360提供，同上
timestamp	是	int64	当前时间戳（单位秒）
qid	是	int64	由360提供，同上
access_token	是	string	请求授权接口可得到，只有请求授权接口非必填，其他接口都为必填。生效时间：3小时
sign	是	string	1）参数排序（升序），参数=参数值”的模式用“&” 字符拼接。 2）md5（a=1&b=2$appsecret$）小写，appsecret由360提供，只有请求授权接口非必填其他接口都为必填
order_id	是	string	厂商调用sdk传入的订单id
user_id	是	string	用户id，厂商的用户唯一标识

```
{
    "errno": 0,
    "errmsg": "",
    "data": {
        "mfr_order_id": "xxxx", //厂商订单编号
        "mfr_order_amount": 1000, //厂商订单金额 单位为分
        "mfr_create_time": "20060102150405", //厂商订单创建时间
        "mfr_product_id": "xxx", //厂商商品id
        "mfr_product_name": "uuuuu", //厂商商品名称
        "order_status": 10, //订单状态 10 -待付款(初始状态) 20-付款完成（待通知厂商） 30-待厂商发权益（已通知厂商） 40-售后中（厂商发起退款） 50-交易完成（正常完成，厂商完成物品发放） 60-已取消（支付超时，过期等原因） 70-交易关闭（退款完成）
        "pay_chanel": 1, //支付渠道 1-微信，2-支付宝
        "order_code": "xxx-uuuu-oooo-pppp" //360订单编号
        "bank_trade_code": "20201007220014973114185605O2", //银行流水号
        "order_pay_time": "2024-01-03 15:57:53", //付款时间
        "order_refund_time": "2024-01-03 16:04:06" //退款时间
    },
    "time": 1701658768
}
```

参考：



Sign值md5计算方式：按字段顺序升序，&符链接，最后拼接appsecret值，以上图为例：

```
**【按结构体的字段顺序】**
access_token=值&appid=值&order_id=值&qid=值&timestamp=值&user_id=值appsecret
```

> [!NOTE]
>
> 错误码说明

code	说明	code	说明
0	成功	100001	应用信息不存在
10001	参数错误	100002	应用信息已下线
10002	服务内部错误	100003	厂商信息不存在
10006	sign错误	100004	厂商信息已下线
10007	请求过期	100005	订单不存在
10012	access_token错误	100006	应用分类不存在
10014	非法访问	100008	认证信息错误
10015	不支持的content_type	100009	订单状态有误，不允许进行此操作
10016	参数类型错误		

4.1.2.4-厂商订单推送

软件管家订单服务推送至厂商订单服务（仅订单支付成功或者退款成功后进行推送，其它状态变更不予推送，推送地址由厂商提供，订单支付成功回调地址和退款回调地址可360平台配置）【如推送不成功，会补推30次，每秒1次，如30次后仍失败，请通过查询订单接口进行兜底处理。】

> [!CAUTION]
>
> 1）请收到推送参数时，厂商需自行校验所有推送的数据是否一致，如不一致请勿发送权益
>
> 2）接口请求sign生成规则：参数排序（升序），参数=参数值"的模式用"&" 字符拼接。2、md5（a=1&b=2*appsecret*）小写，appsecret由360提供
>
> 3）请求方式：post + application/json
>
> 4）请注意：callback_type新增支持连续支付回调类型

请求参数：

参数名	必选	类型	说明
app_id	是	string	软件id
qid	是	int64	厂商id
mfr_order_id	否	string	厂商订单id
mfr_order_amount	否	int64	厂商订单金额，单位分
mfr_product_id	否	string	厂商产品id
mfr_product_name	否	string	厂商产品名称
order_code	否	string	软管订单编号
order_status	否	int64	10 -待付款(初始状态) 20-付款完成（待通知厂商、任务完成） 30-待厂商发权益（已通知厂商、任务完成） 40-售后中（厂商发起退款） 50-交易完成（正常完成，厂商完成物品发放、任务完成） 60-已取消（支付超时，过期等原因） 70-交易关闭（退款完成） 注：【20、30、50均需要视为支付成功状态，用于订单查询时判断】
pay_chanel	否	int64	支付渠道 1-微信，2-支付宝 ，order_status＞10且≠60时有此字段
bank_trade_code	否	string	银行交易码 ，支付渠道 1-微信，2-支付宝 ，order_status＞10且≠60时有此字段
trans_time	否	string	支付时间或者退款时间【签约、解约逻辑中不使用，默认传0001-01-01 00:00:00】
sign	是	string	sign生成规则由软件管家提供，通请求openapi的算法保持一致

参数名	必选	类型	说明
callback_type	是	int64	回调类型 1 订单状态变更回调（普通下单扣款，退款） 回调类型 2 表示代扣推送回调（自动扣款），需要厂商创建订单，order_extra这时候返回的是顶级父产品的订单号，厂家通过这个来说明是那个用户那个产品的代扣单子，商家收到这个回调，需要自己通过子的mfr_order_id创建新订单 'order_extra': '{"mfr_order_id":"xxxx"}'，仅使用mfr_order_id即可，如有其他字段无需使用 回调类型 3 表示的是签约推送（签约、取消签约信息），order_extra这时候返回的是顶级父产品的订单号 'order_extra': '{"mfr_order_id":"xxxx","agreement_number":"xxxx","auto_pay_status":1}'，需要全部使用
order_extra	否	string	根据callback_type状态返回{json}【为json字符串】
mfr_order_id	否	string	订单号
agreement_number	否	string	签约号
auto_pay_status	否	int64	签约状态: 1-开通签约 2-取消签约
timestamp	是	int64	当前时间戳，单位秒

callback_type的响应

> [!CAUTION]
>
> 用户权益发完毕，按此格式响应推送订单，360-Server收到此响应后，order_status状态码将由30变更为50

```
{
    "code": 200, //成功后code设置为200，失败设置为非200
    "data": "xxx", //如果有附加信息可以通过此字段传递
    "message": "success" //code码描述，200=success
}
```

callback_type=1推送参数示例

> [!CAUTION]
>
> 普通支付、退款推送，需要下发商品权益

```
{
    "app_id": "xxxx",
    "bank_trade_code": "xxxx",
    "callback_type": 1,
    "mfr_order_amount": 9900,
    "mfr_order_id": "xxxxx",
    "mfr_product_id": "xxxxxxx",
    "mfr_product_name": "ccccccc",
    "order_code": "xxxxxxxx",
    "order_extra":"", //为空
    "order_status": 30,
    "pay_channel": 1,
```

```
    "qid": 11111111,
    "sign": "c03bdfaf65e4209c4aebe7ce2f255840",
    "timestamp": 1709266143,
    "trans_time": "2024-03-01 12:09:01"
}
```

callback_type=2时推送参数示例

> **[!CAUTION]**
>
> **续费订单支付、退款推送，需要下发商品权益**
>
> **order_extra内仅使用mfr_order_id即可，如有其他字段无需使用**
>
> **例如下面示例中order_extra内的agreement_number、auto_pay_status字段为系统自动生成 无含义 可以忽略**

```
{
    'app_id': 'xxxx',
    'bank_trade_code': 'xxxx',
    'callback_type': 2,
    'mfr_order_amount': 9900,
    'mfr_order_id': 'xxxx',
    'mfr_product_id': 'xxxx',
    'mfr_product_name': 'xxxx',
    'order_code': 'xxxxxxxx',
    'order_extra':'{"mfr_order_id":"XXXXX","agreement_number":"","auto_pay_status":0}',
    'order_status': 20,
    'pay_channel': 2,
    'qid': 11111111,
    'sign': 'c03bdfaf65e4209c4aebe7ce2f255840',
    'timestamp': 1709266143,
    'trans_time': '2024-03-01 12:09:01'
}
```

callback_type=3时推送参数示例

> **[!IMPORTANT]**
>
> **仅做为签约、取消签约状态通知，无需下发或取消商品权益**

> **[!CAUTION]**
>
> **签约、取消签约推送**

```
{
    'app_id': 'xxxx',
    'bank_trade_code': 'xxxx',
    'callback_type': 3,
    'mfr_order_amount': 9900,
    'mfr_order_id': 'xxxx',
    'mfr_product_id': 'xxxx',
```

```
    'mfr_product_name': 'xxxx',
    'order_code': 'xxxx',
    'order_extra':
'{"mfr_order_id":"xxxx","agreement_number":"xxxx","auto_pay_status":1}',
    'order_status': 50,
    'pay_channel': 2,
    'qid': 11111111,
    'sign': 'c03bdfaf65e4209c4aebe7ce2f255840',
    'timestamp': 1709266143,
    'trans_time': '2024-03-01 12:09:01'
}
```

> [!CAUTION]
>
> 取消签约推送

```
{
    'app_id': 'xxxx',
    'bank_trade_code': 'xxxx',
    'callback_type': 3,
    'mfr_order_amount': 9900,
    'mfr_order_id': 'xxxx',
    'mfr_product_id': 'xxxx',
    'mfr_product_name': 'xxxx',
    'order_code': 'xxxx',
    'order_extra':
'{"mfr_order_id":"xxxx","agreement_number":"xxxx","auto_pay_status":2}',
    'order_status': 50,
    'pay_channel': 2,
    'qid': 11111111,
    'sign': 'c03bdfaf65e4209c4aebe7ce2f255840',
    'timestamp': 1709266143,
    'trans_time': '2024-03-01 12:09:01'
}
```

4.1.2.5-发票开具接口

> [!IMPORTANT]
>
> **请注意：当前发现邮件方式给到用户发票时，可能存在被拦截的情况，建议在产品内，给用户提供发票下载入口**
>
> Connect-Type：json格式

1）普票

普票开票接口地址：/main/gateway/v1/order/invoicing【POST】【应用调用频次限制：3 次/秒】

参数名	必选	类型	说明
order_id	是	string	厂商调用sdk传入的订单id，即mfr_order_id
invoice_title	是	string	发票title

参数名	必选	类型	说明
user_email	是	string	用户邮箱
tax_register_no	否	string	纳税人识别号（长度为15-20位,由数字和大写字母组成）
address	否	string	地址
phone	否	string	电话
bank_name	否	string	银行名称
bank_account	否	string	银行账号
remarks	否	string	备注

响应说明

```
{
    "data": {
        "download_url": "http://www.fapiao.com/dzfp-web/pdf/download?request=jiPK8jSMZRBqWVEcgXwgqhXqqq1ZQxB3Pn9Sy.jHSlwlOoihulTYA4PIWpvxcBFkr5U5ddQJ-3zzrqgGUeoNtQ__%5EdfJCjBeagG", // PDF 下载地址
        "invoice_code": "012002300311", // 发票代码
        "invoice_no": "28563912", // 发票号码
        "receipt_url": "https://www.fapiao.com/fpt-wechat/wxaddcard.do?code=nTzLuTSySkO%2FCSFLqHZkf6iWFTRFJ%2BLpebViBhG22gORPuHDsLp1jWc9cAo%2B1HiJosxUv6RUCXPH%OAeIP6X%2BsH4g%3D%3D", // 收票地址，用该地址生成二维码，供用 户扫码获取发票
        "success_time": "2024-01-29 14:07:37", // 开票日期
        "verify_code": "04088381497270657211" // 校验码
    },
    "errmsg": "",
    "errno": 0,
    "time": 1706508810
}
```

普票-红冲接口：/main/gateway/v1/invoice/cancel【POST】【应用调用频次限制：3 次/秒】

参数名	必选	类型	说明
appid	是	string	
qid	是	int64	
order_id	是	string	

响应说明：

```
{
    'code':0,
    'msg':'操作成功',
    'data':{}
}
```

2）专票

> [!IMPORTANT]
>
> 因为专票申请后需要人工审核后，才能正常开具或者作废发票，一般时效性为5个工作日内，申请后可以通过专票查询接口定期查询
>
> 不支持开具个人title的专票
>
> Connect-Type：json格式

专票开票接口：/main/gateway/v1/invoice/dospecial【POST】【应用调用频次限制：3 次/秒】

参数名	必选	类型	说明
order_id	是	String	厂商调用sdk传入的订单id
invoice_title	是	string	发票title
user_email	是	string	用户邮箱
tax_register_no	是	string	纳税人识别号（长度为15-20位,由数字和大写字母组成）
address	是	string	地址
phone	是	string	电话
bank_name	是	string	银行名称
bank_account	是	string	银行账号
custom_type	是	string	商户类型，1企业
remarks	否	string	备注

响应说明

```
{
    "data": {
        "source_id": "1809172892785221632"
    },
    "errmsg": "",
    "errno": 0,
    "time": 1720175405
}
```

备注：返回专票信息建议保存，红冲时需要invoice_num、source_id等

专票查询接口：/main/gateway/v1/invoice/queryspecial【POST】【应用调用频次限制：3 次/秒】

参数名	必选	类型	说明
request_type	是	string	请求类型 1-开票查询 2-红冲查询
source_id	是	string	开票接口返回的source_id

响应说明

```
{
    "data": {
        "download_url": "https://dev.fapiao.com:19443/dzfp-web/pdf/download?request=dkiZcYM-ZbQmTT3hgQAfxkccIleVDG4sPSzcj3UfW4qilGfaU5uYQzNRij99Bt2mAOgVSaaOBbqvVDZER1MGYw__%5EbCGbhdIEda", // PDF 下载地址
        "invoice_code": "", // 发票代码
        "invoice_num": "24442666000661505582", // 发票号码
        "receipt_url": "http://testwx.fapiao.com/fpt-wechat/wxaddcard.do?code=i2SZwhurDtvaLVgYBn%2BmCcYX8tiscRUbbsS%2BBAWRzKsBsjmcdj6iQ%2Bq2JIF%2FflJsZqrKrJOnp6Fj%2BZ6p5%2FE7Ww%3D%3D",// 收票地址，用该地址生成二维码，供用 户扫码获取发票
        "status": "SUCCESS_END",
        "success_time": "2024-06-20 08:55:41"
    },
    "errmsg": "",
    "errno": 0,
    "time": 1720179357
}
```

专票红冲接口：/main/gateway/v1/invoice/cancelspecial【POST】【应用调用频次限制：3 次/秒】

参数名	必选	类型	说明
category	是	string	红冲类别 1：销方红冲
invoice_num	是	string	发票号码，销方红冲必填
red_reason	是	string	红冲原因： INVOICE_MISTAKE，销方红冲必填
source_id	是	string	蓝票申请单来源id
order_id	是	String	厂商调用sdk传入的订单id

响应说明

```
{
    "data": {},
    "errmsg": "",
    "errno": 0,
    "time": 1720179357
}
```

五、文件格式类产品行为规范

> [!IMPORTANT]
>
> 请严格按照此规范说明执行，我们会不定期抽查，确认存在问题后，我们会对此产品下架，同时做删白处理

行为规范文档	文件格式关联产品行为规范.pdf

六、用户客诉处理

> [!IMPORTANT]
>
> 还请及时处理相关客诉问题

客诉处理要求文档	联运软件客诉处理规范1.0.pdf

七、QAQ

问	答
1.为什么常出现多次推送	1）确保接口响应时间在10s以内，否则会出现重复推送的情况且IP有多个，如响应时间超过10s，建议做异步处理 2）回调格式不符合要求，请检查代码逻辑 3）不可用，请检查接口
2.SDK_Pay会弹出收银台窗口吗?	不会，只是支付地址，需要厂商来生成支付二维码
3.必须在启动软件的时候初始化么?	是的，还请在软件启动时初始化SDk
4.产品上有什么要求么?	1）软件名称以软著名称为主，我们建议加上360专版或联运版 2）联运版与管版，互不相互覆盖安装、升级
5.如果用户有发票诉求怎么办?	1）请通过发票接口申请开票【可以放置到产品内部，或CP方运营后台】 2）可以在软件管家-个人中心-申请发票，输入银行流水号后开具发票 3）如果用户对已开具发票申请红冲，请联系接口人
6.如果用户要求退款怎么办?	由厂商发起退款。如用户反馈到360侧，会引导用户去联系软件客服处理。
7.用户客诉谁来处理?	为方便用户得到更专业的答复，如用户反馈到360侧，会引导用户去联系软件客服处理。

问	答
8.如果需要出口IP做加白，怎么办?	请联系接口人获取
9.为什么收不到推送	1）请检查地址是否正确 2）如果是https，请确认证书是否为权威机构颁发，如果不是，请升级证书或替换为http
10.在连续支付业务中，为什么不能签约	请检查您旗下的产品给openapi回传的产品product id是不是有重复

推广组新增投放业务字段影响的接口

以下接口将在 2024.12.10 上线，openAPI 用户需要于 2024.12.10 前完成开发及同步上线，以免平台上线后 openAPI 客户未更新，影响推广组的增改操作；

目录

1、获取用户投放业务列表（新增）

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/group/getIndustryTags

Headers 请求参数

字段	类型	必须	描述
apiKey	String	Yes	注册时分配到的 api key，该参数作为 HTTP HEAD 字段传递。
accessToken	String	Yes	身份认证通过后平台分配的临时 token，作为操作对应广告账户的凭证，10 小时有效。参数获取方式见接口 clientLogin

请求示例

```
curl -X POST \

--header 'apiKey:APIKEY' \

--header 'accessToken:ACCESSTOKEN' \

'https://api.e.360.cn/dianjing/group/getIndustryTags'
```

返回结果

```
{

  "data": {
```

```
"value": 11,
"label": "模拟一级行业 1",
"children": [
  {
    "value": 21,
    "label": "模拟二级行业 1.1",
    "children": [
      {
        "value": 31,
        "label": "模拟三级行业 1.1",
        "children": [
          {
            "value": 41,
            "label": "模拟四级行业 1.1"
          }
        ]
      }
    ]
  },
  {
    "value": 22,
    "label": "模拟二级行业 1.2",
    "children": [
      {
```

```
          "value": 32,
          "label": "模拟三级行业 1.2",
          "children": [
            {
              "value": 42,
              "label": "模拟四级行业 1.2"
            }
          ]
        }
      ]
    }
  ]
 }
}
```


返回结果说明

字段	类型	是否必须	描述
data	object	Yes	返回数据
data.value	number	Yes	一级投放业务编号
data.label	string	Yes	一级投放业务名称
data.children	array	No	数组，包含多个子投放业务的 value 和 label
data.children.value	number	Yes	子投放业务编号
data.children.label	string	Yes	子投放业务名称
data.children.children	array	No	子投放业务的子级数组，包含更深层级的行业数据

2、添加推广组

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/group/add

Headers 请求参数

字段	类型	必须	描述
apiKey	String	Yes	注册时分配到的 api key，该参数作为 HTTP HEAD 字段传递。
accessToken	String	Yes	身份认证通过后平台分配的临时 token，作为操作对应广告账户的凭证，10 小时有效。参数获取方式见接口 clientLogin

Body 请求参数

字段	类型	必须	描述
campaignId	int	Yes	推广计划 id
name	String	Yes	推广组名称
price	double	Yes	推广组出价
negativeWords	Json	No	匹配方式为短语匹配或者精确匹配的否定关键词（短语不得超过 200 个，精确不得超过 400 个）,使用 JSON 格式：其中 phrase 代表短语 exact 代表精确 {"phrase":["短语匹配 1"],"exact":["精确 1","精确 2"]}
status	String	No	启用/暂停状态，enable-启用，pause-暂停，默认启用

industryTags（新增）	Json	No	推广组投放业务，用户投放业务列表不为空时必传。使用 JSON 格式，包含以下字段：
			• path: 数组类型，表示投放业务路径编码，从一级到四级业务编码依次排列
			• literal: 字符串类型，当叶子节点为自定义行业时必传。必须以"其他-"为前缀
			
			示例 1 - 标准业务路径:
			{
			"industryTags": {
			"path": [11,21,31,41]
			}
			}
			
			示例 2 - 自定义行业:
			{
			"industryTags": {
			"path": [14,24,34],
			"literal": "其他-模拟游戏行业"

字段	类型	必须	描述
			} }

请求示例

```
curl -X POST \

--header 'apiKey:APIKEY' \

--header 'accessToken:ACCESSTOKEN' \

--data 'campaignId=3924245063&name=推广组_auto_exex&price=100&negativeWords={"phrase":["
短语匹配 1"],"exact":["精确 1","精确 2"]}&status=pause' \

'https://api.e.360.cn/dianjing/group/add'
```

返回结果

```
{

    "id":1810329949

}
```

返回结果说明

字段	类型	描述
id	int	新增的推广组 id

3、批量添加推广组

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/group/batchAdd

Headers 请求参数

字段	类型	必须	描述
apiKey	String	Yes	注册时分配到的 api key，该参数作为 HTTP HEAD 字段传递。
accessToken	String	Yes	身份认证通过后平台分配的临时 token，作为操作对应广告账户的凭证，10 小时有效。参数获取方式见接口 clientLogin

Body 请求参数

字段	类型	必须	描述
groups	String	Yes	JSON 格式的推广组对象数组，单次请求提交数量最多 100 个。

groups 请求示例

```
[
    {
        "campaignId": "1306991174",
        "name": "测试组 1",
        "price": "2",
        "status": "pause",
        "negativeWords": {
            "phrase": ["up 短语匹配 1"],
            "exact": ["up 精确 1","精确 2"]
        }
    }
]
```

groups 字段说明

字段	类型	必须	描述
campaignId	int	Yes	推广计划 id
name	String	Yes	推广组名称
price	double	Yes	推广组出价
negativeWords	Json	No	匹配方式为短语匹配或者精确匹配的否定关键词（短语不得超过 200 个，精确不得超过 400 个）,使用 JSON 格式：其中 phrase 代表短语 exact 代表精确 {"phrase":["短语匹配 1"],"exact":["精确 1","精确 2"]}
industryTags（新增）	Json	No	推广组投放业务，用户投放业务列表不为空时必传。使用 JSON 格式，包含以下字段： • path: 数组类型，表示投放业务路径编码，从一级到四级业务编码依次排列 • literal: 字符串类型，当叶子节点为自定义行业时必传。必须以"其他-"为前缀

字段	类型	必须	描述

			示例 1 - 标准业务路径:
			{
			"industryTags": {
			"path": [11,21,31,41]
			}
			}
			
			示例 2 - 自定义行业:
			{
			"industryTags": {
			"path": [14,24,34],
			"literal": "其他-模拟游戏行业"
			}
			}

status	String	No	启用/暂停状态，enable-启用，pause-暂停，默认启用
adType	String	No	组类型，1-动态凤舞组，2-普通组，默认 2 普通组
packageId	String	No	商品包 id，adType=1 商品组时为必填项
monitorCode	String	No	自定义监控代码，只有商品组可以添加。1.必须以半角?或&开头，自定义名称={监控代码}，用“&“连接所有需要监控信息例 如：?q={device}&w={keywordid}&e={planid} 2.现支持的通配符参数有：{keywordid} {device} {planid}

请求示例

```
curl -X POST \

--header 'apiKey:APIKEY' \

--header 'accessToken:ACCESSTOKEN' \

--data 'groups=[{"campaignId":"1306991174","name":"测试组
1","price":"2","status":"pause","negativeWords":{"phrase":["up 短语匹配 1"],"exact":["up
精确 1","精确 2"]}},{"campaignId":"1306991174","name":"测试组
2","price":"2","status":"pause","negativeWords":{"phrase":["up 短语匹配 1"],"exact":["up
精确 1","精确 2"]}}]' \

'https://api.e.360.cn/dianjing/group/batchAdd
```

返回结果

```
{
```

```
    "groupIdList":[1810333277],

    "failures":[]

}
```

返回结果说明

字段	类型	描述
groupIdList	Int	新增的推广组 id 数组，每个 id 和 groups 参数中的顺序一致,如果某个推广组新建失败，对应的 id 为 0。

4、更新推广组

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/group/update

Headers 请求参数

字段	类型	必须	描述
apiKey	String	Yes	注册时分配到的 api key，该参数作为 HTTP HEAD 字段传递。
accessToken	String	Yes	身份认证通过后平台分配的临时 token，作为操作对应广告账户的凭证，10 小时有效。参数获取方式见接口 clientLogin

Body 请求参数

字段	类型	必须	描述
id	int	Yes	推广组 id
name	String	No	推广组名称
price	double	No	推广组出价
negativeWords	Json	No	匹配方式为短语匹配或者精确匹配的否定关键词（短语不得超过 200 个，精确不得超过 400 个）,使用 JSON 格式：其中 phrase 代表短语 exact 代表精确 {"phrase":["短语匹配 1"],"exact":["精确 1","精确 2"]}，并会完全替换原有的否词和精确否词。
industryTags（新增）	Json	No	推广组投放业务，用户投放业务列表不为空时必传。使用 JSON 格式，包含以下字段：

字段	类型	必须	描述

			• path: 数组类型，表示投放业务路径编码，从一级到四级业务编码依次排列
			• literal: 字符串类型，当叶子节点为自定义行业时必传。必须以"其他-"为前缀
			
			示例 1 - 标准业务路径:
			{
			"industryTags": {
			"path": [11,21,31,41]
			}
			}
			
			示例 2 - 自定义行业:
			{
			"industryTags": {
			"path": [14,24,34],
			"literal": "其他-模拟游戏行业"
			}
			}

status	String	No	推广组状态，enable-启用，pause-暂停

请求示例

```
curl -X POST \

--header 'apiKey:APIKEY' \

--header 'accessToken:ACCESSTOKEN' \

--data
'format=JSON&id=1810329949&status=pause&name=update_auto_exec&price=30&negativeWords={"
phrase":["up 短语匹配 1"],"exact":["up 精确 1","精确 2"]}' \

'https://api.e.360.cn/dianjing/group/update'
```

返回结果

```
{
    "id":"1810329949"
}
```

5、批量更新推广组

根据推广组 id 批量更新相应的推广组

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/group/batchUpdate

Headers 请求参数

字段	类型	必须	描述
apiKey	String	Yes	注册时分配到的 api key，该参数作为 HTTP HEAD 字段传递。
accessToken	String	Yes	身份认证通过后平台分配的临时 token，作为操作对应广告账户的凭证，10 小时有效。参数获取方式见接口 clientLogin

Body 请求参数

字段	类型	必须	描述
groups	string	yes	json 格式的推广计划数组，最多支持 100 个

groups 示例

```
[{
    "id": 1810329949,
    "name": "改成一个奇怪的名字 1",
    "status": "pause",
    "negativeWords": {
        "phrase": ["up 短语匹配 1"],
        "exact": ["up 精确 1", "精确 2"] //会完全替换原有的否词和精确否词。
    }
}]
```

groups 字段说明

字段	类型	必须	描述
id	int	Yes	推广组 id
name	String	No	推广组名称
price	double	No	推广组出价
negativeWords	Json	No	匹配方式为短语匹配或者精确匹配的否定关键词（短语不得超过 200 个，精确不得超过 400 个）,使用 JSON 格式：

字段	类型	必须	描述
			其中 phrase 代表短语 exact 代表精确 {"phrase":["短语匹配 1"],"exact":["精确 1","精确 2"]}，并会完全替换原有的否词和精确否词。

industryTags（必填）	Json	No	推广组投放业务，用户投放业务列表不为空时必传。使用 JSON 格式，包含以下字段：
			• path: 数组类型，表示投放业务路径编码，从一级到四级业务编码依次排列
			• literal: 字符串类型，当叶子节点为自定义行业时必传。必须以"其他-"为前缀
			
			示例 1 - 标准业务路径:
			{
			"industryTags": {
			"path": [11,21,31,41]
			}
			}
			
			示例 2 - 自定义行业:
			{
			"industryTags": {
			"path": [14,24,34],
			"literal": "其他-模拟游戏行业"
			}
			}

status	String	No	推广组状态，enable-启用，pause-暂停

请求示例

```
curl -X POST \

--header 'apiKey:APIKEY' \

--header 'accessToken:ACCESSTOKEN' \

--data 'groups=[{"id": 1810329949,"name": "改成一个奇怪的名字 1","status":
"pause","negativeWords": {"phrase":["up 短语匹配 1"],"exact":["up 精确 1","精确 2"]}}]' \

'https://api.e.360.cn/dianjing/group/batchUpdate'
```

返回结果

```
{
    "affectedRecords":[1810329949],
    "failures":[]
```

```
}
```

6、批量获取推广组信息

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/group/getInfoByIdList

Headers 请求参数

字段	类型	必须	描述
apiKey	String	Yes	注册时分配到的 api key，该参数作为 HTTP HEAD 字段传递。
accessToken	String	Yes	身份认证通过后平台分配的临时 token，作为操作对应广告账户的凭证，10 小时有效。参数获取方式见接口 clientLogin

Body 请求参数

字段	类型	必须	描述
idList	string	yes	json 格式的 id 数组，如[1000001,20000001,3000001],去重后数量限制 1000

请求示例

```
curl -X POST \

--header 'apiKey:APIKEY' \

--header 'accessToken:ACCESSTOKEN' \

--data 'idList=[1810329949]' \

'https://api.e.360.cn/dianjing/group/getInfoByIdList'
```

返回结果

```
{

    "groupList": [

        {

            "id": 1810329949,

            "status": "pause",

            "price": "30",
```

```
            "negativeWords": "up 短语匹配 1",
            "exactNegativeWords": "up 精确 1,精确 2",
            "campaignId": 3924245063,
            "name": "改成一个奇怪的名字 1",
            "addTime": "2019-03-05 13:28:11",
            "industryTags": "{\"path\":[1,15,70]}",
            "industryTagsTitle": "机械设备>农林机械>渔业机械>其他-行业 111",
            "updateTime": "2019-03-05 13:53:27"
        }
    ]
}
```


返回结果说明

字段	类型	描述
id	string	推广组的 id
campaignId	string	广告计划的 id
name	string	推广组的名称
price	string	推广组默认出价
negativeWords	string	短语匹配的否定关键词
industryTags（新增）	string	投放行业详情
industryTagsTitle（新增）	string	投放全行业标题
exactNegativeWords	string	精确匹配的否定关键词
status	string	推广组状态值，enable 表示启用，pause 表示暂停
addTime	string	推广组创建时间
updateTime	string	推广组最后更新时间

7、整账户下载接口

获取推广计划的完整数据

异步获取推广计划的完整数据，生成下载文件

HTTP 请求方式

POST

请求地址

https://api.e.360.cn/dianjing/account/getAllObjects

请求参数不变，下载的文件表格最后增加一列：推广组投放业务的字段

AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI
建词质量	移动质量度	物料类型	关键词移动	创意移动质	创意移动链	移动搜索出	电话直呼	风舞内容	错误信息	推广组投放业务									
		推广组								{"industryTags":{"path":[1,15,70]},"industryTagsTitle":"机械设备>农林机械>种植机械"}									
		推广组								{"industryTags":{"path":[1,15,71]},"industryTagsTitle":"机械设备>农林机械>其他>其他-自定义投放业务"}									

推广组投放业务列内容格式：

{"industryTags":{"path":[1,15,70]},"industryTagsTitle":"
机械设备>农林机械>种植机械"}

或者

{"industryTags":{"path":[1,15,71]},"industryTagsTitle":"
机械设备>农林机械>其他>其他-自定义投放业务"}