

DLFrame 接口说明 V1.0

Document Number:		Document Version:	1.0
Owner:	Yongchuan.wan	Date:	2021-07-21
Document Type:			
NOTE:	<p>ALL MATERIALS INCLUDED HEREIN ARE COPYRIGHTED AND CONFIDENTIAL UNLESS OTHERWISE INDICATED. The information is intended only for the person or entity to which it is addressed and may contain confidential and/or privileged material. Any review, retransmission, dissemination, or other use of or taking of any action in reliance upon this information by persons or entities other than the intended recipient is prohibited.</p> <p>This document is subject to change without notice. Please verify that your company has the most recent specification.</p> <p>Copyright © 2019 UNISOC Communications Inc.</p>		

修订记录

版本	日期	作者	说明
V1.0	2021-07-21		草稿

目录

DLFrame 接口说明 V1.0	1
修订记录.....	2
目录.....	3
1. 目的.....	4
1.1. 缩略词说明.....	4
1.2. 参考文档.....	4
2. DLFrame 场景流程说明	4
2.1. 自动下载流程（包含下载端口自动监听）	5
2.2. 半自动下载流程（下载端口需要调用者自己指定.....	6
3. DLFrame 库 c 风格调用方式	7
3.1. 接口定义.....	7
3.2. 上下文实例.....	7
3.2.1. 创建实例.....	7
3.3. 下载过程接口.....	8
3.3.1. 初始化.....	8
3.3.2. 注册回调函数.....	9
3.3.3. 加载 Pac 文件	10
3.3.4. 启动下载流程.....	11
3.3.5. 停止下载流程.....	12
3.4. OnMessage 回调消息说明.....	16
3.4.1. 回调函数.....	16
4. 命令行使用方式.....	18

1. 目的

本文针对 Download 工具 DLFrame 下载的接口进行说明，适用于所有 PC（包括 Linux，Windows 平台）上位机程序开发人员和对外合作开发人员。

1.1. 缩略词说明

名称	全称	定义

1.2. 参考文档

2. DLFrame 场景流程说明

DLFrame 库支持全自动下载和半自动下载

全自动下载，即启动流程调用后，DLFrame 库会自动监听 PC 上连接的正处于下载模式的手机；

半自动下载，适用于二次开发的用户，不需要底层对端口进行监听，而又他们自己实现端口监听，自己指定端口进行下载

2.1. 自动下载流程（包含下载端口自动监听）

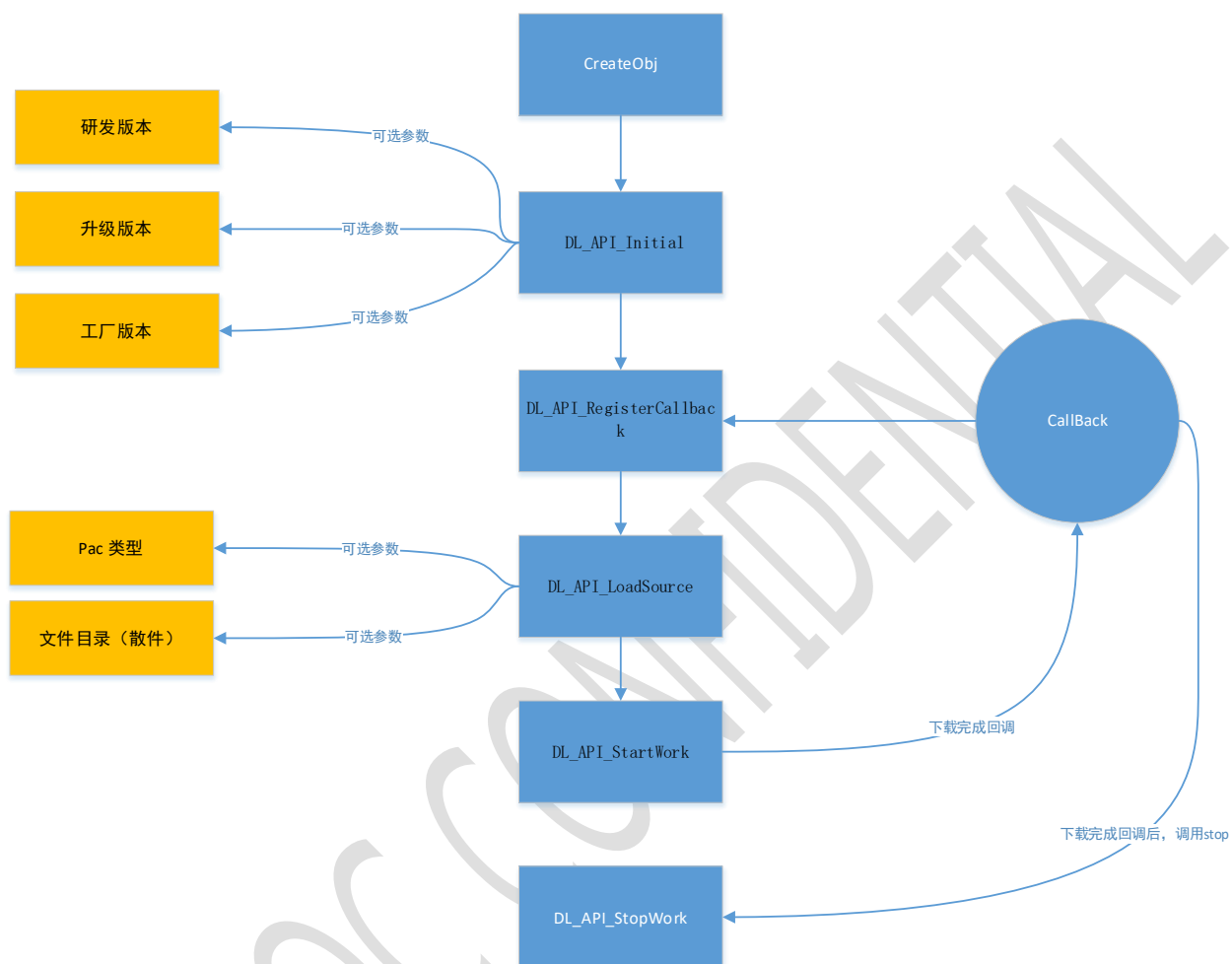
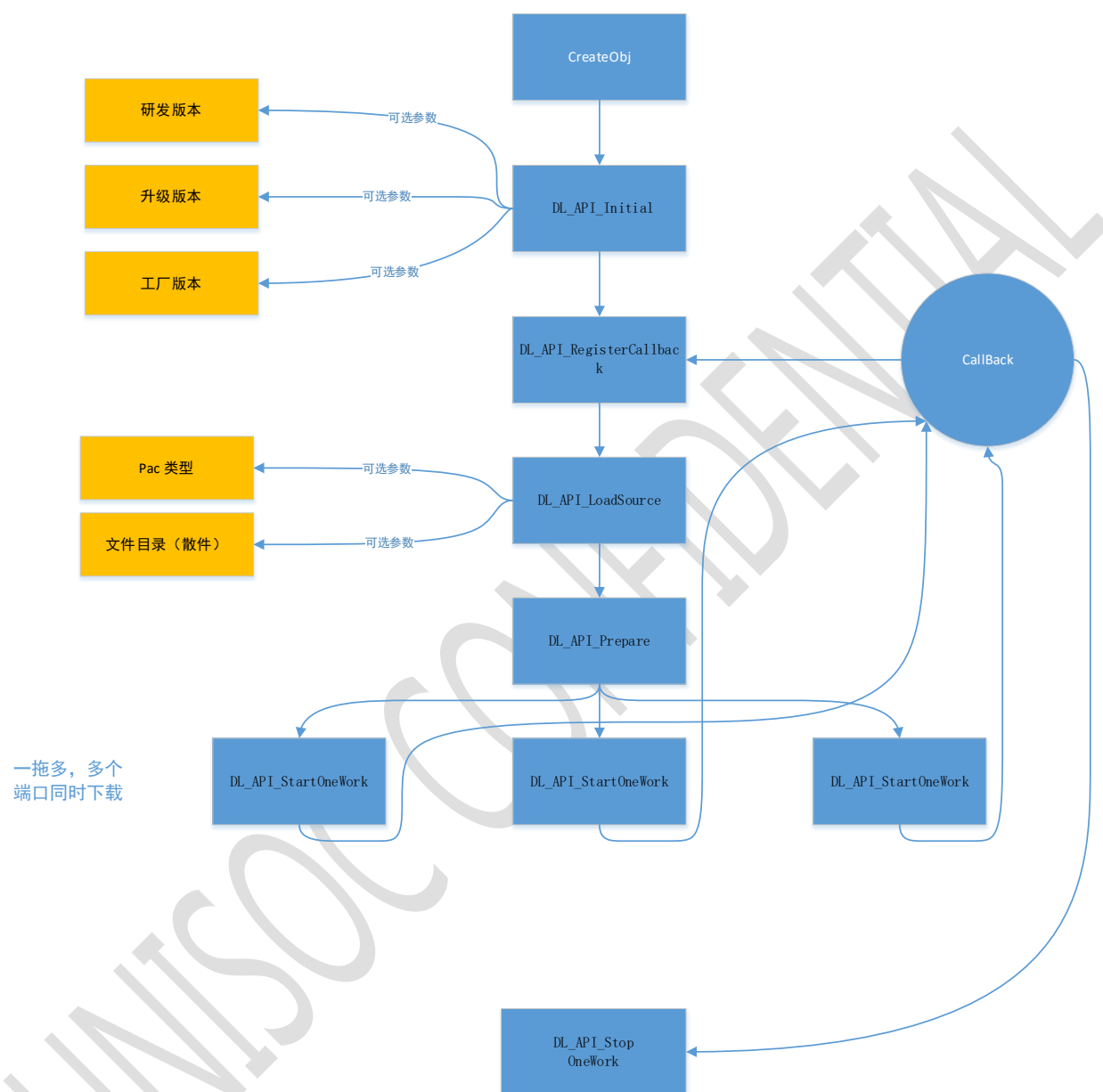


图 1 自动找端口下载流程，该过程不需要动态库用户再度实现下载端口监控过程

2.2. 半自动下载流程（下载端口需要调用者自己指定）



当前调用方式需要在调用端自己给下载过程指定下载端口，DLFrame 不会自动找端口，如果当前手机没有处于下载模式，也不会在下次出下载模式的情况下自动启动下载，需要动态库用户自己调用接口下载

3. DLFrame 库 c 风格调用方式

3.1. 接口定义

```
extern "C" DLFRAME_API UniHandle CreatedLObj();
extern "C" DLFRAME_API bool DL_API_Initial(UniHandle hHandle, VERSION_TYPE version);
extern "C" DLFRAME_API void DL_API_RegisterCallback(UniHandle hHandle, UI_OnMessage lpFunc);
extern "C" DLFRAME_API bool DL_API_LoadSource(UniHandle hHandle, DownloadSource nSourceType,
LPCTSTR szPac);
extern "C" DLFRAME_API void DL_API_Prepare(UniHandle hHandle);
extern "C" DLFRAME_API bool DL_API_StartOneWork(UniHandle hHandle, int port);
extern "C" DLFRAME_API bool DL_API_StopOneWork(UniHandle hHandle, int port);
extern "C" DLFRAME_API void DL_API_Release(UniHandle hHandle);
extern "C" DLFRAME_API bool DL_API_StartWork(UniHandle hHandle, int port);
extern "C" DLFRAME_API bool DL_API_StopWork(UniHandle hHandle);
```

3.2. 上下文实例

DLFrame 库提供了 CreateObj() 接口用于生成 void 指针类型上下文实例，释放直接调用 delete

3.2.1. 创建释放实例

【语法】

```
UniHandle * CreatedLObj();
```

【说明】

该接口必须在 DLL 加载完成之后或者静态调用，添加 lib 库依赖之后调用

【参数】

无

【返回值】

UniHandle 的指针，该类型是当前下载过程的上下文实例，贯穿整个下载流程，所有的下载过程要通过该对象表明当前下载会话处。

【sample code】

```
void* m_pDLApi = CreatedLObj();
if (m_pDLApi == NULL)
{
    _tprintf(_T("Failed to Create DLObject."));
    return FALSE;
}
```

```
}
```

释放对象【samplecode】

```
if (m_pDLApi)
{
    DL_API_Release(DL_API_Release);
    m_pDLApi = NULL;
}
```

3.3. 下载过程接口

3.3.1. 初始化

【语法】

```
bool DL_API_Initial (UniHandle hHandle , VERSION_TYPE version) = 0;
```

【说明】

完成初始化工作，加载配置文件（配置文件为调用该库 exe 命名，举例：**Demo1.exe** 调用了该 dll，那么就调用 **Demo1.ini**，由于初始配置较多，请将动态库同级目录下 **ResearchDownload.ini** 内容拷贝到对应 ini 文件(**Demo1.ini**)中），指定版本用途

【参数】

Parameter name	In/Out	Explain
hHandle	IN	上下文
version	IN	版本类型

```
enum VERSION_TYPE {
    RESEARCH_VERSION, //研发用，功能自选
    FACTORY_VERSION, //擦除校准数据
    UPGRADE_VERSION, //保留校准数据
};
```

【返回值】

false : 不成功

true : 成功

【sample code】

```
void* m_pDLApi = CreatedLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
```



```

    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

```

3.3.2. 注册回调函数

【语法】

```
void DL_API_RegisterCallback (UniHandle hHandle , IUIInterface* pCallBack)
```

【说明】

注册回调函数，用于上报所有 UI 所需数据，如 pac 包解包进度，开始下载的哪个分区，以及下载进度，下载状态，以及相应报错信息

【参数】

Parameter name	In/Out	Explain
hHandle	IN	上下文
pCallBack	IN	IUIInterface 实例指针

```
typedef void(*IUI_OnMessage) (UINT msgID, UINT wParam, void* lParam);
```

【返回值】

无

【sample code】

```

void OnMessage(UINT msgID, UINT wParam, void* lParam)
{
    ...
}

.....

void* m_pDLApi = CreateDLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);

```

3.3.3. 加载 Pac 文件

【语法】

bool DL_API_LoadSource (UniHandle hHandle, DownloadSource nSourceType, LPCTSTR szPac)

【说明】

加载 pac 文件，并解析 pac 文件，完成下载前的文件准备

【参数】

Parameter name	In/Out	Explain
hHandle	IN	上下文
nSourceType	IN	Img 路径类型 (DownloadSource_Folder 包含目录散件，各个分区文件存储在该目录路径下，szPac 参数为散件所在目录；类型 DownloadSource_Pac 时，szPac 表示 pac 文件路径)
szPac	IN	.pac 文件路径

【返回值】

True：成功

False：不成功

【sample code】

```
void* m_pDLApi = CreateDLobj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);
If (!DL_API_LoadPacket(m_ pDLApi , DownloadSource_Pac, L"E:\\img\\1.pac"))
{
    _tprintf(_T("Load pac file failed."));
    return FALSE;
}
```

```
typedef enum DownloadSource
{
    DownloadSource_Pac,
    DownloadSource_Folder
};
```

3.3.4. 启动下载流程

【语法】

```
bool DL_API_StartWork(UniHandle hHandle, int port) = 0;
```

【说明】

启动下载流程，执行之后，如果检查到 pc 连接的设备进入下载模式，即开始该设备下载过程

【参数】

Parameter name	In/Out	Explain
UniHandle	IN	上下文
port	IN	指定使用固定端口进行下载，为 0 时，表示自动检测端口，不指定固定端口

注意：如果 port 参数为 0，在 windows 下表示自动检测下载端口，如果有手机连上，就会马上启动线程开始下载，在 Linux 下，表示 usbttty0

【返回值】

True：成功

False：不成功

【sample code】

```
void* m_pDLApi = CreatedLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);
If (!DL_API_LoadPacket(m_pDLApi , DownloadSource_Pac, L"E:\\img\\1.pac"))
{
    _tprintf(_T("Load pac file failed."));
    return FALSE;
}

If (!DL_API_StartWork(m_pDLApi , 0))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}
```

3.3.5. 停止下载流程

【语法】

```
bool DL_API_StopWork(m_pDLApi) ;
```

【说明】

停止当前实例所触发的所有下载过程

【参数】

无

【返回值】

True： 成功

False： 不成功

【sample code】

```
void* m_pDLApi = CreateDLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);
If(!DL_API_LoadPacket(m_ pDLApi , DownloadSource_Pac,L"E:\\img\\1.pac"))
{
    _tprintf(_T("Load pac file failed."));
    return FALSE;
}
If(!DL_API_StartWork(m_pDLApi ,0))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}
```

等待下载结束后执行(MSG_ID_DL_ON_END) ，使用 setEvent 推出等待

```
g_hStopEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

WaitForSingleObject(m_hStopEvent, INFINITE);
ResetEvent(m_hStopEvent);
if(!DL_API_StopWork(m_pDLApi))
{
    _tprintf(_T("stop work failed."));
    return FALSE;
}
```

3.3.6. 半自动下载流程 Prepare

【语法】

```
void DL_API_Prepare(UniHandle hHandle);
```

【说明】

该方法特用半自动下载流程，在加载完 **pac** 包后，先使用该方法，然后再执行启动方法

【参数】

传入上下文

【返回值】

无

【sample code】

```
void* m_pDLApi = CreatedLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);
If (!DL_API_LoadPacket(m_ pDLApi , DownloadSource_Pac,L"E:\\img\\1.pac"))
{
    _tprintf(_T("Load pac file failed."));
    return FALSE;
}
DL_API_Prepare(m_pDLApi);
If (!DL_API_StartOneWork(m_pDLApi , 17))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}
```

3.3.7. 半自动下载流程下载启动

【语法】

```
bool DL_API_StartOneWork(UniHandle hHandle, int port);
```

【说明】

该方法在下载手机已经处于下载模式，使用接口的人已经获取到手机端口后，直接启动下载，可以支持一拖多下载

如多路下载，需要在每个 **DL_API_StartOneWork** 两边上锁

【参数】

- 1 传入上下文
- 2 下载端口号

【返回值】

是否启动成功

【sample code】

```
void* m_pDLApi = CreatedLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);
If(!DL_API_LoadPacket(m_ pDLApi , DownloadSource_Pac,L"E:\\img\\l.pac"))
{
    _tprintf(_T("Load pac file failed."));
    return FALSE;
}

DL_API_Prepare(m_pDLApi);
Xxx.lock()
If(!DL_API_StartOneWork(m_pDLApi ,17))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}

xxx.unlock()
Xxx.lock()
If(!DL_API_StartOneWork(m_pDLApi ,27))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}

xxx.unlock()
```

3.3.8. 半自动下载流程下载停止下载

【语法】

```
bool DL_API_StopOneWork(UniHandle hHandle, int port);
```

【说明】

该方法在下载手机已经处于下载模式，使用接口的人已经获取到手机端口后，直接启动下载，可以支持一拖多下载

如多路下载，需要在每个 **DL_API_StartOneWork** 两边上锁

【参数】

- 1 传入上下文
- 2 下载端口号

【返回值】

是否启动成功

【sample code】

```
void* m_pDLApi = CreateDLObj();
if (!DL_API_Initial (pDLApi, RESEARCH_VERSION))
{
    _tprintf(_T("Failed to Research Version initial."));
    return FALSE;
}

DL_API_RegisterCallback (m_pDLApi, OnMessage);
If (!DL_API_LoadPacket(m_ pDLApi , DownloadSource_Pac,L"E:\\img\\1.pac"))
{
    _tprintf(_T("Load pac file failed."));
    return FALSE;
}
DL_API_Prepare(m_pDLApi);
xxx.lock()
If (!DL_API_StartOneWork(m_pDLApi , 17))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}
xxx.unlock()
xxx.lock()
If (!DL_API_StartOneWork(m_pDLApi , 27))
{
    _tprintf(_T("start work failed"));
    return FALSE;
}
xxx.unlock()
```

等待下载结束后执行 (MSG_ID_DL_ON_END) ，使用 setEvent 推出等待

```

g_hStopEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

WaitForSingleObject(m_hStopEvent, INFINITE);
ResetEvent(m_hStopEvent);
xxx.lock()

if(!DL_API_StopOneWork(m_pDLApi, port))
{
    _tprintf(_T("stop work failed. "));
    return FALSE;
}
xxx.unlock

```

3.4. OnMessage 回调消息说明

3.4.1. 回调函数

【语法】

```
typedef void(*IUI_OnMessage)(UINT msgID, UINT wParam, void* lParam);
```

【说明】

回调函数返回下层上报的通道

【参数】

Parameter name	In/Out	Explain
msgID	IN	区分不同类型消息
wParam	IN	根据 msgID 确定，不同的 msgID 含义不同
lParam	IN	回调上报的数据，根据 msgID 确定，不同的 msgID， 数据结构不同

【消息类型说明】

msgID (Value)	wParam (value)	lParam	定义头文件
WM_BINPAC_PROG_MSG (pac 包解包过程反馈消息)	PROG_BEGIN (解包开始)	无	SecBinPackApi.h
	PROG_PROCEED (解包进度)	解包进度值（数据转换 *(DWORD*) lParam;）	

	PROG_END (解包结束)	无	
MSG_ID_DL_NEW_PORT (检测到有新的下载端)	下载端口号	无	IDownloadApi.h
MSG_ID_DL_PAC_TITLE (上报 pac 包信息头中对应的产品名称)	无	字符串名称 (数据转换 (LPCTSTR) lParam)	
MSG_ID_DL_MSG_ERROR 返回错误消息, 该部分	无	字符串	
MSG_ID_DL_FILES	无	PDL_FILE_INFO (包含了 pac 包所有文件信息)	
BM_CHECK_BAUDRATE 下载状态 (检测波特	端口号	无	Global.h
BM_CONNECT 连接手机	端口号	无	
BM_ERASE_FLASH	端口号	无	
BM_DOWNLOAD	端口号	当前下载分区数据最大值	
BM_DOWNLOAD_PROCESS 下载进度	端口号	下载进度值	
BM_READ_FLASH	端口号	读取分区 size	
BM_READ_FLASH_PROCESS	端口号	读取 flash 进度值	
BM_RESET 正在执行下载问自动开	端口号	无	
BM_SET_FIRST_MODE	端口号	无	
BM_READ_CHIPTYPE (状态) 读取芯片类型	端口号	无	
BM_READ_NVITEM (读取 NV)	端口号	无	
BM_CHANGE_BAUD (更改 端口波特率)	端口号	无	
BM_BEGIN (下载开始)	端口号	无	
BM_FILE_BEGIN (单个分 区开始下载, 每一个分	端口号	返回当前分区文件大小本 次下载大小, (__int64	
BM_END (下载结束)	端口号	BMOBJ, 结果信息 _BMOBJ * pbj = (_BMOBJ *) lpParam;	
BM_FILE_TITLE	端口号	当前开始下载文件的名称	

MSG_ID_DL_ON_END	端口号	结果信息 (_DL_RESULT_INFO_T_)	
------------------	-----	------------------------------	--

结果信息结构

```
typedef struct _DL_RESULT_INFO_T_
{
    char szSN[X_SN_LEN + 1] = { 0 };
    char szSN2[X_SN_LEN + 1] = { 0 };
    char szIMEI[X_SN_LEN + 1] = { 0 };
    char szChipUID[X_SN_LEN + 1] = { 0 };
    char szFlashUID[MAX_PATH] = { 0 };
    int bSucce = 0;
    int nSeconds = 0;
    TCHAR szErrorMsg[_MAX_PATH * 2];
    void* tSoftSim;
    char szTestResultGUID[50];
    _DL_RESULT_INFO_T_()
    {
        Clear();
    }
    void Clear()
    {
        memset(this, 0, sizeof(_DL_RESULT_INFO_T_));
    }
}DL_RESULT_INFO_T, * PDL_RESULT_INFO_T;
```

4. 命令行使用方式

目前命令行只支持全自动下载，半自动下载过程一般是客户做二次开发使用

DLoader version is R1.20.2201,build on 202005025.

usage: CmdDloader <command> [<option>]; e.g. sudo ./CmdDloader -pac ./test.pac

commands and option:

-pac <pac-path> *download pac file path
[-version <integer>] 0:ResearchDownload, 1:FactoryDownload, 2:UpgradeDownload.
Default is 0.
[-port <integer>] download port, default is auto obtain.
[-baud <integer>] set baud-rate for serial port, default is 115200.
[-nvbk <0|1>] backup nv option, 0: not backup, 1: backup. Default is 1.
[-filebk <0|1>] backup need backup partition, 0: not backup, 1: backup. Default is 1.
[-phasecheck] flash phasecheck,default randomly generated SN.

[-SN <sn>] set with sn string.
[-KeepCharge] support usb power supply.
[-PowerOff] power off device after download
[-reset] reset device to normal after download.

例子: CmdDloader.exe -pac e:/1.pac -port 1

-pac <路径> 该选项为必选项想, 指定 pac 包路径
-port <数字> 指定下载串口, 为可选项
-version <0/1/2> 为可选项, 默认版本为 0 表示 researchdownload 配置相关的功能
其他的命令为可选项
-baud <波特率> 设置波特率
-Poweroff 下载完成后退出下载模式
-reset 下载完成后重启手机
-SN <设备序列号> 设置序列号