UNISOC DLFramework 集成指导说明

文档版本 V1.1

发布日期 2022-12-09

紫光展锐（上海）科技有限公司

紫光展锐科技有限公司

前 言

概述

Download 是紫光展锐发布的 Windows 系统下载工具，包括：

- FactoryDownload：擦除校准参数；不备份手机相关数据；用于单板生产阶段；
- UpgradeDownload：默认会对需要备份的分区进行备份；用于整机组装及售后维修；
- ResearchDownload：下载配置灵活；备份配置灵活；研发端专用工具；

Download 工具正式发布的版本通过严苛测试和认证，可直接导入产线量产。

客户也可通过 DLFramework 提供的 API 集成到自研的产线测试系统中，本文档主要描述了将 DLFramework 集成到产线自研测试系统的方法。

读者对象

本文档主要适用于量产测试系统开发人员，必须具备以下经验和技能：

- 熟悉 Download 工具的操作和使用；
- 熟悉 C++编程语言。

缩略语

缩略语	英文全名	中文解释
DL	Download	下载
DLFW	Download Framework	下载工具的业务流程模块
BM	Boot Mode	启动模式
AAL	Application Abstraction Layer	应用抽象层
CAL	Control Abstraction Layer	控制抽象层
HAL	Hardware Abstraction Layer	硬件抽象层

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
说明	用于突出重要/关键信息、 补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

修改记录

文档版本	发布日期	作者	修改说明
V1.0	2022-08-10	UNISOC	第一次正式发布。
V1.1	2022-12-09	UNISOC	增加第三章“命令行快速集成”。

目 录

图目录

表目录

1 概述

1.1 什么是 Download 工具

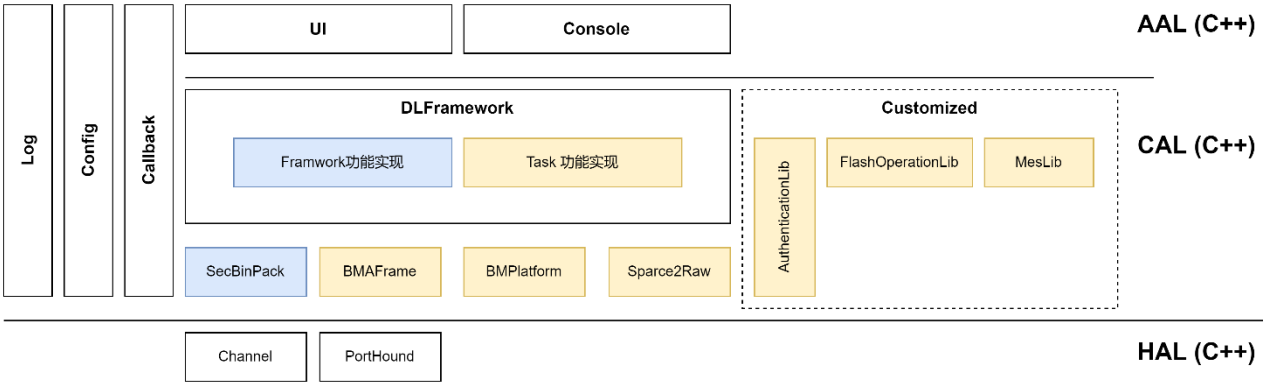
Download 工具的用途是通过串口或者 USB 将手机软件写进硬件模块中。

它的优点是可以同时对多个模块进行下载，以提高效率，而且操作简单，只需将模块正确连接到端口并切换到正确的模式，程序将自动检测到并开始下载过程；整个流程是自动化的，一个模块下载完成后，手工更换模块后，程序将自动检测到新的模块，重新开始下载。

1.2 系统架构

模块化设计是 Download 工具中最主要特点，其设计框图如下图所示：

图1-1 模块框图



说明

- CAL 模块底色说明：蓝色为 Framework 业务实现部分，黄色为 Task 业务实现部分；
- 以上架构是基于 Download_R27.XX.XXXX 及以后版本；

整个系统主要由三层构成，分别为 HAL、CAL 和 AAL。

1.2.1 AAL

AAL 层是 Download UI 实现层，开发语言采用 C++ ，基于 MFC，支持如下的基本功能：

- 配置文件（INI）可视化编辑;
- 测试主界面，支持一拖多并行下载，用户可直观的查看 Port、Step、Status、Progress、Time、Rate 等信息。

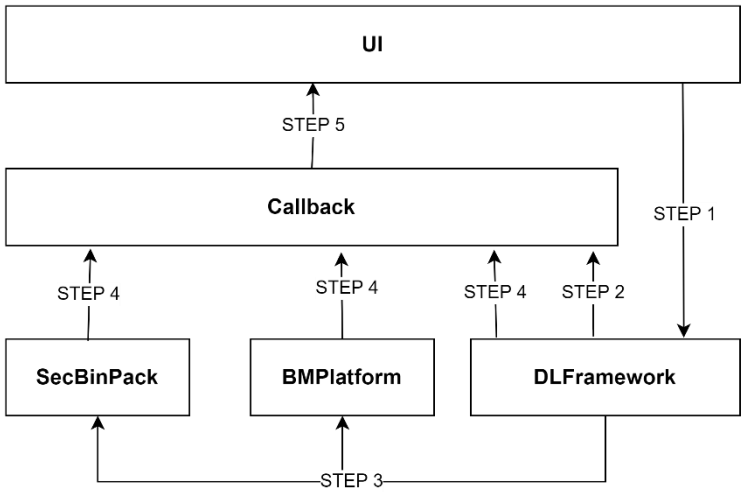
1.2.2 CAL

CAL 层是整个系统中最核心的一层，是测试具体过程的实现，包含 DLFramework、BMAFrame 和 BMPlatform 等主要模块，其中：

- SecBinPack：Package 解压和生成 Package；
- Sparce2Raw：文件格式转化；
- BMAFrame：下载流程执行等功能；
- BMPlatform：下载功能的实现等功能；
- DLFramework： 下载流程的操作等功能；
- AuthenticationLib：客制化模块；下载工具授权；
- FlashOperationLib：客制化模块；Flash 操作：读取、写入、检查、擦除；
- MesLib：客制化模块；MES 连接、流程检查、投板、上传测试结果等；

CAL 层的所有模块均使用 C++语言开发；AAL 层通过 DLFramework 模块接口调用 CAL 层；CAL 层重要信息通过回调功能模块（Callback.dll）上报给 AAL 层 UI：

图1-2 AAL 和 CAL 数据交互



说明	
STEP 1	注册Callback
STEP 2	创建Callback Object
STEP 3	传递Callback Object到不同模块
STEP 4	调用Callback API
STEP 5	传递Data到UI层

1.2.3 HAL

驱动控制层，主要由手机通讯（Channel）和设备监控（PortHound）组成；

- Channel：基于通讯协议（USB、UART）实现的通路；
- PortHound：串口检测，串口获取等；

1.2.4 其他

- Config：参数配置功能模块；配置参数（Download.ini、BinPack.ini、BMError.ini、BMFileType.ini、BMTimeout.ini、DLFramework.ini）操作；
- iSpLog：日志功能模块；创建日志文件、打印测试日志；
- Callback：回调功能模块；CAL 库通过回调功能模块把信息传递到 AAL 层；

1.3 目录结构

表1-1 Download 工具目录结构

-Download_RXX.XX.XXXX			
	-Source		源码目录
	-Doc		文档目录
	-Bin		工具目录
	FactoryDownload.exe		
	ResearchDownload.exe		
	UpgradeDownload.exe		
	-System		依赖信息目录（隐藏文件）
	-Setting		常用配置文件目录
		FactoryDownload.ini	
		ResearchDownload.ini	
		UpgradeDownload.ini	
	-App		依赖库目录
		DLFramework.dll	
		BMAFrame9.dll	
		BMPlatform9.dll	
		BMFileType.ini	
		BMAConfig.xml	
		…	
		-Customized	客制化目录
		-UnisocMES	展锐 MES 目录
	-Backup		备份文件目录
	-ImageFiles		PAC 包解压目录
	-Log		日志目录

1.4 集成方案

Download 工具目前提供 2 种集成方案：API 和命令行；

API：按照一定流程调用 DLFramework API，可以实现整个下载的功能；

命令行：按照命令行说明，执行配置的命令，可以实现部分下载的功能；

图1-3 API

DLFramework API		
DLFW_Startup	创建System资源;	Framework
DLFW_Cleanup	释放System资源	Framework
DLFW_ReloadSettings	重新加载配置; 从INI中读取	Framework
DLFW_SyncParameters	同步Framework参数到Task	Framework
DLFW_LoadPacket	加载Packet	Framework
DLFW_LoadFiles	散件加载	Framework
DLFW_DoPacket	生成Packet	Framework
DLFW_DeviceMoniter	启动和关闭USB检测功能	Framework
DLFW_SetProperty	设置参数到System资源	Framework
DLFW_GetProperty	从System中获取参数	Framework
DLFW_CreateTask	创建Task资源	Task
DLFW_FreeTask	释放Task资源	Task
DLFW_RunTask	执行下载任务	Task
DLFW_StopTask	停止下载任务	Task
DLFW_TaskSetProperty	设置参数到Task资源	Task
DLFW_TaskGetProperty	从Task中获取参数	Task

说明

API 的详细说明请参考 4.1 章;

图1-4 命令行

命令行	
-Pac <PacFilePath-String>	Download packet file path; Arguments must be configured.
-Version <Version-Number>	0:ResearchDownload; 1:FactoryDownload; 2:UpgradeDownload; Default is 0:ResearchDownload
-Port <Port-Number>	Download port; Default is 0(auto find the available device);
-Policy <Policy-Number>	Load download packet file policy; Refer to the definition in BinPack.ini; Default is 0;
-Baudrate <Baudrate-Number>	Baudrate for serial port; Default is 115200;
-SkipFileID <FileID-String>	FileID which want to skip download, split with ','; Default is null;
-Timeout <Timeout-Number>	Timeout(uint:ms) from execute to start downloading; Default is 60000ms;
-WriteSN1 <SN1-String>	Set with SN1 string; Default is null;
-WriteSN2 <SN2-String>	Set with SN2 string; Default is null;
-MutiFileID <FileID-String>	FileID for Multi-project co-software; Default is null;
-MutiProjectID <ProjectID-String>	ProjectID for Multi-project co-software; Default is null;
-MutiProjectBase <ProjectBase-String>	ProjectBase for Multi-project co-software; Default is null;
-ReadNV <NVFile-String>	Read the NV according to the configured NV file; Default is null;
-SendFlag	Enable the SendFlag function;
-WriteEfuse	Enable the write efuse function;
-WriteTimeStamp	Write Time Stamp;
-PowerOff	Power off device after download
-Reset	Reset device to normal after download;
-EZMode	Only output status and results in the console window;
-DDDP	Disable detail download process printing; Default is false;
-Help	Help information; Default is false;

说明

命令行参数的详细说明请参考第 3.2 章；

2 API 快速集成

本章节通过一个命令行（Console Window）实例描述如何基于 DLFramework 快速集成，其中涉及的接口定义和参数说明可以参考第四章。

2.1 环境准备

2.1.1 开发环境

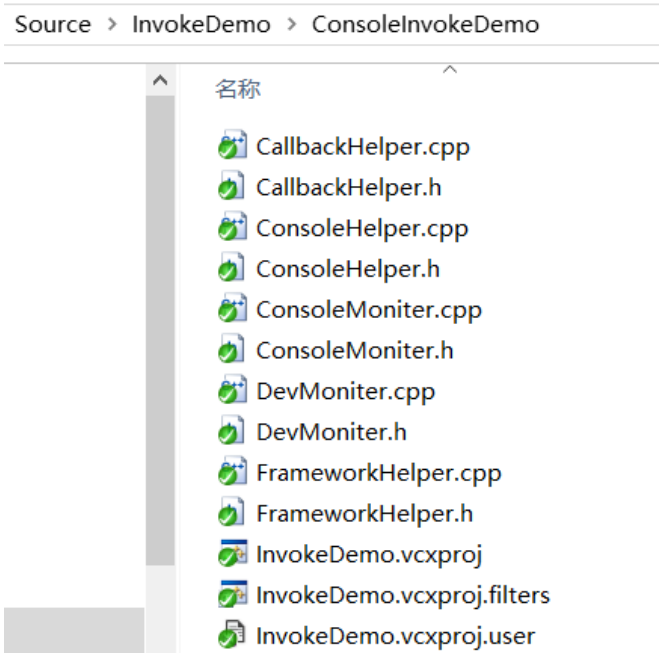
IDE: Microsoft Visual Studio C++ 2019，版本推荐： V10.0.17763.0

2.1.2 开发包

Download 发布版本包。

- 运行库文件和配置文件：发布包 Bin\App 目录
- .h, .lib 和公共类：Source\Common 目录
- 配置文件：Bin\Setting 和 Bin\App
- 本示例源码：Source\InvokeDemo\ConsoleInvokeDemo

图2-1 示例源码路径



2.2 集成步骤

2.2.1 步骤 1 创建 Win32 Console Application

按照 Visual Studio Wizard 提示创建 Win32 Console Application，并设置工程的编译和 Link 环境路径到 Bin 路径下。

图2-2 创建 Win32 Console Application

2.2.2 步骤 2 实现 CDLFWDriver

ConsoleInvokeDemo 采用动态加载的方式调用 DLFramework。客户集成时，建议采用动态加载方式调用 DLFramework，可以实现展锐下载功能模块和配置文件与客户测试系统其它功能模块隔离，从而便于后续对展锐下载工具的管理和维护。客户通过动态加载方式集成时，可以直接基于 DLFWDrvDef.h、DLFWDriver.h、DLFWDriver.cpp，不需要在进行封装。

图2-3 定义函数指针类型（DLFWDrvDef.h）

DLFWDrvDef.h

ConsoleInvokeDemo (全局范围)

```
#pragma once
#include "ISpLogExport.h"
#include "DLFrameworkExport.h"

typedef SPRESULT( SP_API* PDLFW_Startup )( DL_TYPE_ENUM eDLType, LPCSPCALLBACK_PARAM lpCallBack, LPVOID pLogUtil );
typedef void ( SP_API* PDLFW_Cleanup )( void );

typedef SPRESULT( SP_API* PDLFW_ReloadSettings )( void );
typedef SPRESULT( SP_API* PDLFW_SyncParameters )( void );
typedef SPRESULT( SP_API* PDLFW_LoadPacket )( LPCWSTR lpszPacketPath, LPCWSTR lpszImageFiles, uint32 u32Timeout );
typedef SPRESULT( SP_API* PDLFW_LoadFiles )( LPCWSTR lpszImagesDir, uint32 u32Timeout );
typedef SPRESULT( SP_API* PDLFW_DoPacket )( LPCDLFW_DOPACKET_PARAMS lpParams );
typedef SPRESULT( SP_API* PDLFW_DeviceMoniter )( BOOL bStart );

typedef SP_HANDLE( SP_API* PDLFW_CreateTask )( LPCSPCALLBACK_PARAM lpCallBack, LPVOID pLogUtil, BOOL bUart );
typedef void( SP_API* PDLFW_FreeTask )( SP_HANDLE hTask );
typedef SPRESULT( SP_API* PDLFW_RunTask )( SP_HANDLE hTask, DWORD dwPort );
typedef SPRESULT( SP_API* PDLFW_StopTask )( SP_HANDLE hTask );

typedef SPRESULT( SP_API* PDLFW_SetProperty )( INT nProperty, INT nFlags, LPCVOID lpValue );
typedef SPRESULT( SP_API* PDLFW_GetProperty )( INT nProperty, INT nFlags, LPVOID lpValue );
typedef SPRESULT( SP_API* PDLFW_TaskSetProperty )( SP_HANDLE hTask, INT nProperty, INT nFlags, LPCVOID lpValue );
typedef SPRESULT( SP_API* PDLFW_TaskGetProperty )( SP_HANDLE hTask, INT nProperty, INT nFlags, LPVOID lpValue );

typedef BOOL( SP_API* PCreateISpLogObject )( ISpLog** lppObject );
```


图2-4 定义 DLFW_DRIVER_T 结构体（DLFWDrvDef.h）

```
typedef struct _tagDLFW_DRIVER_T
{
    PDLFW_Startup pDLFW_Startup;
    PDLFW_Cleanup pDLFW_Cleanup;

    PDLFW_ReloadSettings pDLFW_ReloadSettings;
    PDLFW_SyncParameters pDLFW_SyncParameters;
    PDLFW_LoadPacket pDLFW_LoadPacket;
    PDLFW_DoPacket pDLFW_DoPacket;
    PDLFW_LoadFiles pDLFW_LoadFiles;
    PDLFW_DeviceMoniter pDLFW_DeviceMoniter;

    PDLFW_CreateTask pDLFW_CreateTask;
    PDLFW_FreeTask pDLFW_FreeTask;
    PDLFW_RunTask pDLFW_RunTask;
    PDLFW_StopTask pDLFW_StopTask;

    PDLFW_SetProperty pDLFW_SetProperty;
    PDLFW_GetProperty pDLFW_GetProperty;

    PDLFW_TaskSetProperty pDLFW_TaskSetProperty;
    PDLFW_TaskGetProperty pDLFW_TaskGetProperty;

    PCreateISpLogObject pCreateISpLogObject;

    tagDLFW_DRIVER_T( void )
```

图2-5 CDLFWDriver 类定义

```
DLFWDriver.h
ConsoleInvokeDemo                                   CDLFWDriver
 1   #pragma once
 2   #include "DLFWDrvDef.h"
 3
 4   class CDLFWDriver sealed
 5   {
 6   public:
 7       CDLFWDriver( void );
 8       ~CDLFWDriver( void );
 9
10       BOOL Startup( string_t& strPath );
11       void Cleanup( void );
12
13       SPRESULT DLFW_Startup( DL_TYPE_ENUM eDLType, LPCSPCALLBACK_PARAM lpCallBack, LPVOID pLogUtil );
14       void     DLFW_Cleanup( void );
15
16       SPRESULT DLFW_ReloadSettings();
17       SPRESULT DLFW_SyncParameters();
18       SPRESULT DLFW_LoadPacket( LPCWSTR lpszPacketPath, LPCWSTR lpszImageFiles, uint32 u32Timeout = INFINITE );
19       SPRESULT DLFW_LoadFiles(LPCWSTR lpszImagesDir, uint32 u32Timeout = INFINITE);
20       SPRESULT DLFW_DoPacket( LPCDLFW_DOPACKET_PARAMS lpParams );
21       SPRESULT DLFW_DeviceMoniter( BOOL bStart = TRUE );
22
23       SP_HANDLE DLFW_CreateTask( LPCSPCALLBACK_PARAM lpCallBack, LPVOID pLogUtil, BOOL bUart = FALSE );
24       void      DLFW_FreeTask( SP_HANDLE hTask );
25       SPRESULT DLFW_RunTask( SP_HANDLE hTask, DWORD dwPort );
26       SPRESULT DLFW_StopTask( SP_HANDLE hTask );
27       SPRESULT DLFW_SetProperty( INT nProperty, INT nFlags, LPCVOID lpValue );
28       SPRESULT DLFW_GetProperty( INT nProperty, INT nFlags, LPVOID lpValue );
29       SPRESULT DLFW_TaskSetProperty( SP_HANDLE hTask, INT nProperty, INT nFlags, LPCVOID lpValue );
30       SPRESULT DLFW_TaskGetProperty( SP_HANDLE hTask, INT nProperty, INT nFlags, LPVOID lpValue );
31
32       BOOL CreateISpLogObject( ISpLog** lppObject );
33
34   private:
35       HMODULE m_hDLL;
36       HMODULE m_hLog;
37       SPHANDLE m_hHandle;
38       DLFW_DRIVER_T m_drv;
39   };
```


图2-6 示例 CDLFWDriver::DLFW_Startup 实现

```
SPRESULT CDLFWDriver::DLFW_Startup( DL_TYPE_ENUM eDLType, LPCSPCALLBACK_PARAM lpCallBack, LPVOID pLogUtil )
{
    if ( NULL != m_drv.pDLFW_Startup )
    {
        return m_drv.pDLFW_Startup( eDLType, lpCallBack, pLogUtil );
    }
    else
    {
        assert( 0 );
        return SP_E_POINTER;
    }
}
```

2.2.3 步骤 2 实现 CFrameworkHelper

添加 CDLFWDriver 接口调用类，命名为 CFrameworkHelper，并加入工程中，如下图所示：

图2-7 CFrameworkHelper

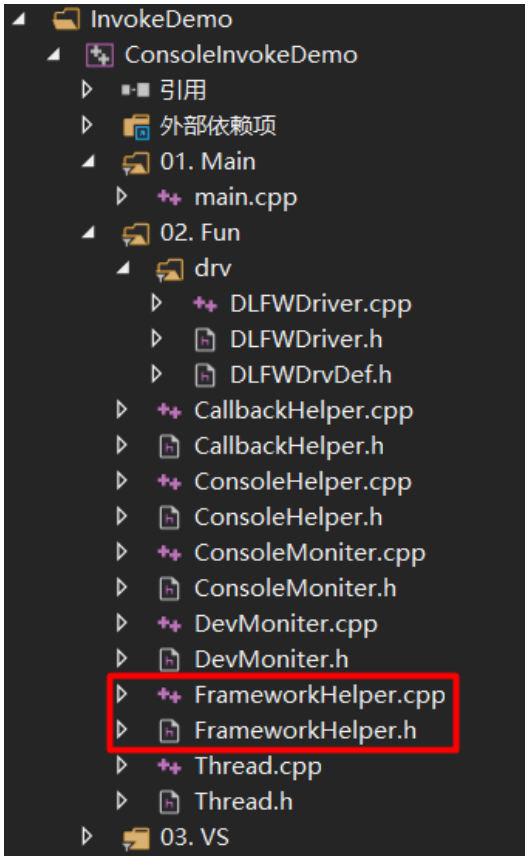


图2-8 CFrameworkHelper 类定义

FrameworkHelper.h

ConsoleInvokeDemo (全局范围)

```
1   #pragma once
2   #include "Tr.h"
3   #include "DLFWDriver.h"
4   #include "CallbackHelper.h"
5
6   ////////////////////////////////////////////////////////////////////////
7   class CFrameworkHelper
8   {
9   public:
10      CFrameworkHelper( void );
11      virtual ~CFrameworkHelper( void );
12
13  public:
14      std::wstring GetAppPath( void );
15
16      SPRESULT Startup( UINT nLv );
17      void Cleanup();
18
19      SPRESULT LoadPacket();
20      SPRESULT SyncParameters();
21      SPRESULT ReloadSettings();
22      SPRESULT GetProperty( INT nProperty, INT nFlags, LPVOID lpValue );
23
24      SPRESULT CreatTask( UINT nLv );
25      SPRESULT RunTask();
26      SPRESULT StopTask();
27      void FreeTask();
28
29  private:
30      ISpLog* m_pTaskLog = NULL;
31      ISpLog* m_pFramLog = NULL;
32      SP_HANDLE m_hTask = NULL;
33      CCallbackHelper m_fnCallHelper;
34      CDLFWDriver m_fnDLFWDriver;
35  };
```


图2-9 CFrameworkHelper 的方式实现

```
SPRESULT CFrameworkHelper::Startup( UINT nLv )
{
    CPrintf print( L"DLFW_Startup" );

    if ( !m_fnDLFWDriver.Startup( GetAppPath() ) )
    {
        return SP_E_FAIL;
    }

    m_fnDLFWDriver.CreateISpLogObject( &m_pFramLog );
    if ( NULL == m_pFramLog )
    {
        std::cout << "CreateISpLogObject Fail." << endl;
        return SP_E_LOAD_LIBRARY;
    }

    OpenArgs_T arg;
    arg.nLogLevel = nLv;
    strcpy_s( arg.szModule, "Frame" );
    wcscpy_s( arg.Local.szLogFile, L"MyFrame.Log" );
    m_pFramLog->Open( ( LPCVOID )&arg );

    SPCALLBACK_PARAM cb;
    cb.pFunc = MyCallBack;
    cb.pParam = &m_fnCallHelper;

    CHKRESULT( m_fnDLFWDriver.DLFW_Startup( FACTORY_TYPE, &cb, m_pFramLog ) );
    CHKRESULT( m_fnDLFWDriver.DLFW_DeviceMoniter( TRUE ) );

    return SP_OK;
}
```

2.2.3.1 Startup/Cleanup

- Startup：首先调用 **DLFW_Startup** 接口初始化 DLFramework 内部资源。
- Cleanup：调用 **DLFW_Cleanup** 接口释放 DLFramework 内部资源。

2.2.3.2 LoadPacket/LoadFiles

- LoadPacket：调用 **DLFW_LoadPacket** 接口加载下载的 Packet，DLFramework 库将根据传入的 Packet 路径解析 Packet。
- LoadFiles：调用 **DLFW_LoadFiles** 接口加载散件文件，DLFramework 库将根据传入的散件路径加载下载的所有文件。

2.2.3.3 CreateTask/FreeTask

- CreateTask：调用 **DLFW_CreateTask** 接口创建一个任务。任务创建过程中 DLFramework 根据 INI 配置和 Packet 的配置，分配资源，完成内部初始化。
- FreeTask：调用 **DLFW_FreeTask** 接口释放创建的任务对象以及内部资源。

2.2.3.4 ReloadSettings/SyncParameters

- ReloadSettings：调用 **DLFW_ReloadSettings** 接口，重新读取 ini 配置文件的参数。
- SyncParameters：调用 **DLFW_SyncParameters** 接口，同步 Framework 的参数到 Task 中。

2.2.3.5 GetProperty/SetProperty

- GetProperty：调用 **DLFW_GetProperty** 接口，从 Framework 内部资源获取信息。
- SetProperty：调用 **DLFW_SetProperty** 接口，把外部信息设置到 Framework 内部资源。

2.2.3.6 Run/Stop

- Run：调用 **DLFW_RunTask** 接口执行下载。
- Stop：调用 **DLFW_StopTask** 接口停止下载。

2.2.4 步骤 3 实现 CCallbackHelper

如图 2.8 AAL 和 CAL 层数据交付描述，DLFramework 以及底层库通过回调将数据上报给 UI。

常用的回调数据的类型定义如下：

图2-10 常用回调类型定义

Callback类型说明		
CALLBACK_LOAD_PACKET_PROGRESS	加载Packet的进度	Framework
CALLBACK_PACKET_INFO	Packet的信息；名称、版本、大小	Framework
CALLBACK_DL_STEP_DESCRIPTION	下载Image信息；File ID、File Type	Framework
CALLBACK_DL_IMAGE_BEGIN	下载Image开始；包含Image大小	Task
CALLBACK_DL_END	下载结束；包含下载结果等信息	Task
CALLBACK_STRING_INFO	下载字符串信息传递；SN、IMEI、错误信息等	Task
CALLBACK_STEP_INFO	下载步骤;	Task
CALLBACK_PROGRESS_INFO	下载Image进度	Task

图2-11 各个回调数据对应的处理函数：

```
class CCallbackHelper
{
public:
    CCallbackHelper( void );
    virtual ~CCallbackHelper( void );

public:
    BOOL HandleCallbackData( LPCSPCALLBACK_DATA lpcbData );

public:
    int m_nPort = 0;

private:
    BOOL __cbHandleLoadPacketProgress( LPCCALLBACKDATA_LOAD_PACKET_PROGRESS lpcbData );
    BOOL __cbHandlePacketInfo( LPCCALLBACKDATA_PACKET_INFO lpcbData );
    BOOL __cbHandleDlImageBegin( LPCCALLBACKDATA_DL_IMAGE_SIZE lpcbData );
    BOOL __cbHandleDlStepDescription( LPCCALLBACKDATA_DL_STEP_DESCRIPTION lpcbData );
    BOOL __cbHandleProgressInfo( LPCCALLBACKDATA_PROGRESS_INFO lpcbData );
    BOOL __cbHandleStepInfo( LPCCALLBACKDATA_STEP_INFO lpcbData );
    BOOL __cbHandleDlEnd( LPCCALLBACKDATA_DL_END lpcbData );
    BOOL __cbHandleStringInfo( LPCCALLBACKDATA_STRING_INFO lpcbData );

private:
    CConsoleHelper m_fnConsole;
    std::vector<string_t> m_vFileID;
    string_t m_strStep = _T( "" );
    uint32 m_u32Step = 0;
    double m_dImageSize = 0.0;
    uint32 m_u32StepStartTime = 0;
    uint32 m_u32StartTime = 0;
};
```

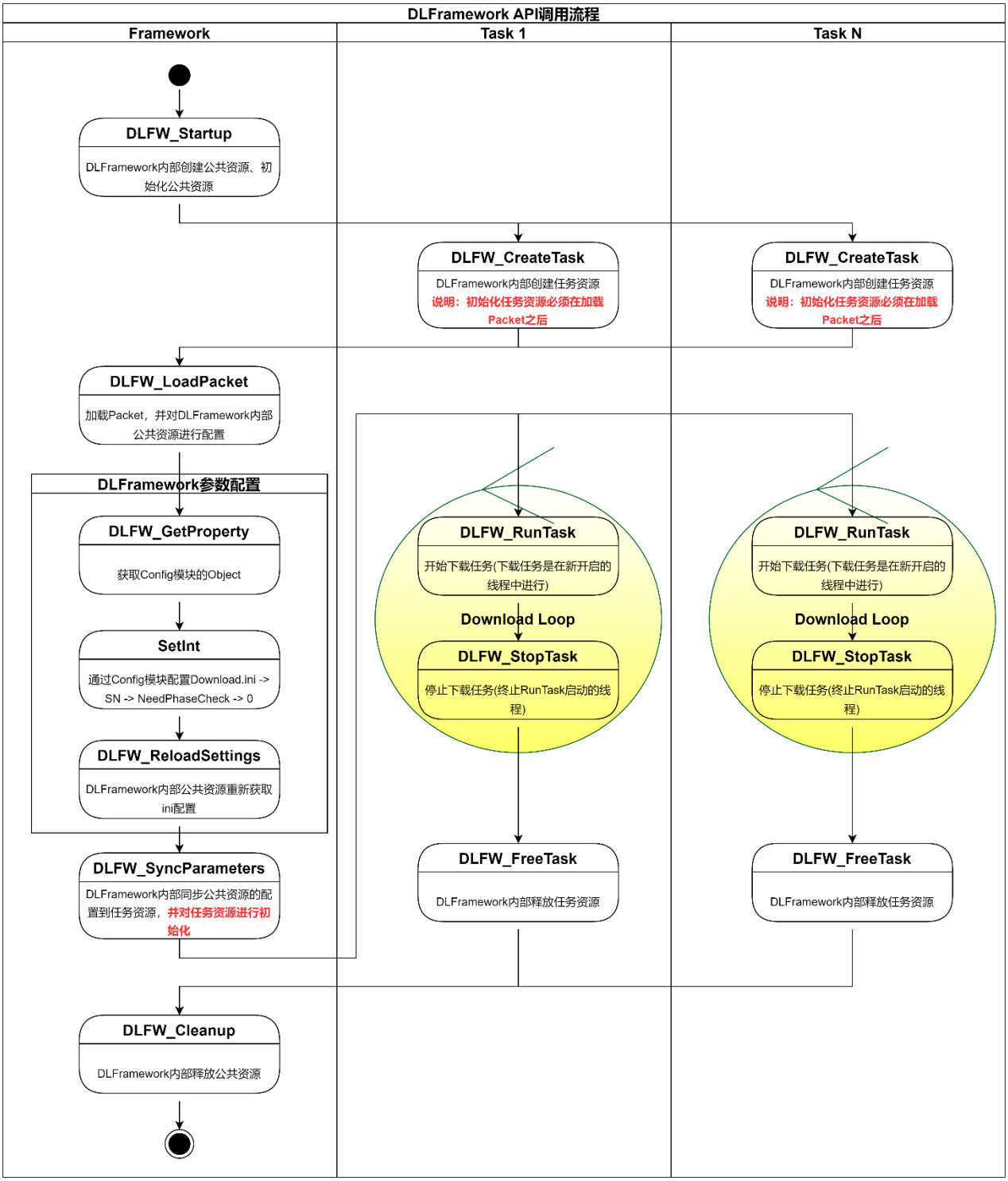
说明

测试任务中通过调用回调函数来实现数据上报，因此回调函数的实现要尽量简化，避免大延迟、复杂处理和 UI 呈现等降低测试执行效率，严重的情况可能会干扰测试时序引起测试失败。

2.2.5 步骤 3 修改 main.cpp

2.2.5.1 下载流程图

图2-12 下载流程图



2.2.5.2 下载流程源码实现

```
#include "stdafx.h"
#include "FrameworkHelper.h"
#include "DevMoniter.h"
#include "ConsoleMoniter.h"
#include <iostream>
#include "IConfig.h"

CFrameworkHelper g_fnFrameWork;
CDevMoniter g_fnDevMoniter;
CConsoleMoniter g_fnConsoleMoniter;
IConfig* g_pCfgObject = NULL;

SPRESULT Download()
{
    std::wstring strIniPath = g_fnFrameWork.GetAppPath() + L"App\\DLFramework.ini";
    UINT nLv = GetPrivateProfileIntW( _T( "Log" ), _T( "Level" ), SPLOGLV_ERROR,
strIniPath.c_str() );

    SPRESULT spRlt = SP_OK;
    /// DLFramework内部创建公共资源、初始化公共资源
    spRlt = g_fnFrameWork.Startup( nLv );
    if ( SP_OK != spRlt )
    {
        goto _END_CLEANUP;
    }
    /// DLFramework内部创建任务资源
    spRlt = g_fnFrameWork.CreatTask( nLv );
    if ( SP_OK != spRlt )
    {
        goto _END_FREE;
    }
    /// 加载Packet，并对DLFramework内部公共资源进行配置
    spRlt = g_fnFrameWork.LoadPacket();
    if ( SP_OK != spRlt )
    {
        goto _END_FREE;
    }
    /// 获取Config模块的Object
    g_fnFrameWork.GetProperty( DLFW_ATTR_CONFIG_OBJECT, 0, &g_pCfgObject );
    if ( SP_OK != spRlt || NULL == g_pCfgObject )
    {
```



```
    goto _END_FREE;
}
/// 通过Config模块配置Download.ini -> SN -> NeedPhaseCheck -> 0
spRlt = g_pCfgObject->SetInt( CFGTYPE_DL, _T( "SN" ), _T( "NeedPhaseCheck" ), 0 );
if (SP_OK != spRlt)
{
    goto _END_FREE;
}
/// DLFramework内部公共资源重新获取ini配置
spRlt = g_fnFrameWork.ReloadSettings();
if ( SP_OK != spRlt )
{
    goto _END_FREE;
}
/// DLFramework内部同步公共资源的配置到任务资源
spRlt = g_fnFrameWork.SyncParameters();
if ( SP_OK != spRlt )
{
    goto _END_FREE;
}
/// 开辟线程监测Console是否有"Download Start"
/// 有"Download Start"，进行下载
/// 120S没有"Download Start"，退出下载
g_fnDevMoniter.StartMoniter( TRUE );
g_fnDevMoniter.StartMoniter( FALSE, 120000 );
if ( g_fnDevMoniter.GetRunResult() )
{
    /// 开始下载任务(下载任务是在新开启的线程中进行)
    spRlt = g_fnFrameWork.RunTask();
    if ( SP_OK != spRlt )
    {
        goto _END_STOP;
    }
    /// 开辟线程监测Console是否有"PASS"和"FAIL"，确认下载是否完成
    g_fnConsoleMoniter( KEY_PASS );
    g_fnConsoleMoniter( KEY_FAIL );
    g_fnConsoleMoniter.StartMoniter( TRUE );
    g_fnConsoleMoniter.StartMoniter( FALSE );
_END_STOP:
    /// 停止下载任务(终止RunTask启动的线程)
    g_fnFrameWork.StopTask();
}
```



```
_END_FREE:
    /// DLFramework内部释放任务资源
    g_fnFrameWork.FreeTask();

_END_CLEANUP:
    /// DLFramework内部释放公共资源
    g_fnFrameWork.Cleanup();

    return SP_OK;
}

int _tmain( int argc, _TCHAR* argv[] )
{
    Download();
    getchar();
    return 0;
}
```

2.2.6 步骤 4 编译运行

选择 Win32 Debug 编译，在 Bin 目录下将会生成 ConsoleInvokeDemo.exe，运行效果如下：

图2-13 Demo 运行效果 - LoadPacket

```
[ ENTER ] DLFW_Startup( )
[ LEAVE ] DLFW_Startup( )
[ ENTER ] DLFW_CreateTask( )
[ LEAVE ] DLFW_CreateTask( )
[ ENTER ] DLFW_LoadPacket( D:\06.UnisocPackages\SharkL3(9863A)\s9863a1h10_1_go_32b_2g-user-gms_SHARKL3_R11.pac )
Start loading packet...
Loading packet progress is  99%
End loading packet
PACKET: s9863a1h10_1; Version: MocorDroidR; Packet Size: 2437.91MB; LoadTime = 2022/08/20 22:28
[ LEAVE ] DLFW_LoadPacket( )
[ ENTER ] DLFW_ReloadSettings( )
[ LEAVE ] DLFW_ReloadSettings( )
[ ENTER ] DLFW_SyncParameters( )
No.0   FileID: FDL                          ; FileType: FDL                          ;
No.1   FileID: FDL2                         ; FileType: NAND_FDL                     ;
No.2   FileID: _REPARTITION_                ; FileType: FORCE_REPARTITION2           ;
No.3   FileID: NV_LTE                       ; FileType: CODE2                        ;
No.4   FileID: ProdNV                       ; FileType: CODE2                        ;
No.5   FileID: EraseUBOOT                   ; FileType: EraseFlash2                  ;
No.6   FileID: EraseUBOOTLOG                ; FileType: EraseFlash2                  ;
No.7   FileID: SPLLoader                    ; FileType: UBOOT_LOADER2                ;
No.8   FileID: VBMETA                       ; FileType: CODE2                        ;
No.9   FileID: VBMETA_SYSTEM                ; FileType: CODE2                        ;
No.10  FileID: VBMETA_SYSTEM_EXT            ; FileType: CODE2                        ;
No.11  FileID: VBMETA_VENDOR                ; FileType: CODE2                        ;
No.12  FileID: VBMETA_PRODUCT               ; FileType: CODE2                        ;
No.13  FileID: Modem_LTE                    ; FileType: CODE2                        ;
No.14  FileID: Modem_LTE_DELTANV            ; FileType: CODE2                        ;
No.15  FileID: DSP_LTE_LTE                  ; FileType: CODE2                        ;
No.16  FileID: DSP_LTE_GGE                  ; FileType: CODE2                        ;
No.17  FileID: DFS                          ; FileType: CODE2                        ;
No.18  FileID: GPS_GL                       ; FileType: CODE2                        ;
No.19  FileID: GPS_BD                       ; FileType: CODE2                        ;
No.20  FileID: Modem_WCN                    ; FileType: CODE2                        ;
No.21  FileID: BOOT                         ; FileType: CODE2                        ;
No.22  FileID: DTBO                         ; FileType: CODE2                        ;
No.23  FileID: Super                        ; FileType: YAFFS_IMG2                   ;
No.24  FileID: Cache                        ; FileType: YAFFS_IMG2                   ;
No.25  FileID: UserData                     ; FileType: YAFFS_IMG2                   ;
No.26  FileID: BootLogo                     ; FileType: CODE2                        ;
No.27  FileID: Fastboot_Logo                ; FileType: CODE2                        ;
No.28  FileID: Socko                        ; FileType: YAFFS_IMG2                   ;
No.29  FileID: Odmko                        ; FileType: YAFFS_IMG2                   ;
No.30  FileID: FLASH_LTE                    ; FileType: EraseFlash2                  ;
No.31  FileID: EraseMisc                    ; FileType: EraseFlash2                  ;
No.32  FileID: EraseSysdumpdb               ; FileType: EraseFlash2                  ;
No.33  FileID: Trustos                      ; FileType: CODE2                        ;
No.34  FileID: Teecfg                       ; FileType: CODE2                        ;
No.35  FileID: SML                          ; FileType: CODE2                        ;
No.36  FileID: Persist                      ; FileType: YAFFS_IMG2                   ;
No.37  FileID: EraseMetadata                ; FileType: EraseFlash2                  ;
No.38  FileID: UBOOTLoader                  ; FileType: UBOOT_LOADER2                ;
```


图2-14 Demo 运行效果 - RunTask

```
[ LEAVE ] DLFW_SyncParameters( )
Please connect the DUT in 111s
Download Start
Download Start
Download Start
[ ENTER ] DLFW_RunTask( Port: 3 )
[ LEAVE ] DLFW_RunTask( )
Start Downloading...
No.0   Downloading image FDL                    progress is 100%; Size: 0.050M; Time:1266ms
No.1   Downloading image FDL2                   progress is 100%; Size: 0.731M; Time:1563ms
No.3   Downloading image NV_LTE                 progress is 100%; Size: 0.900M; Time:47ms
No.4   Downloading image ProdNV                 progress is 100%; Size: 0.043M; Time:16ms
No.7   Downloading image SPLLoader              progress is 100%; Size: 0.053M; Time:0ms
No.8   Downloading image VBMETA                 progress is 100%; Size: 1.000M; Time:31ms
No.9   Downloading image VBMETA_SYSTEM          progress is 100%; Size: 0.004M; Time:16ms
No.10  Downloading image VBMETA_SYSTEM_EXT      progress is 100%; Size: 0.002M; Time:16ms
No.11  Downloading image VBMETA_VENDOR          progress is 100%; Size: 0.004M; Time:16ms
No.12  Downloading image VBMETA_PRODUCT         progress is 100%; Size: 0.002M; Time:15ms
No.13  Downloading image Modem_LTE              progress is 100%; Size: 25.000M; Time:625ms
No.14  Downloading image Modem_LTE_DELTANV      progress is 100%; Size: 0.024M; Time:15ms
No.15  Downloading image DSP_LTE_LTE            progress is 100%; Size: 20.000M; Time:500ms
No.16  Downloading image DSP_LTE_GGE            progress is 100%; Size: 10.000M; Time:265ms
No.17  Downloading image DFS                    progress is 100%; Size: 1.000M; Time:31ms
No.18  Downloading image GPS_GL                 progress is 100%; Size: 0.570M; Time:16ms
No.19  Downloading image GPS_BD                 progress is 100%; Size: 0.570M; Time:16ms
No.20  Downloading image Modem_WCN              progress is 100%; Size: 3.000M; Time:78ms
No.21  Downloading image BOOT                   progress is 100%; Size: 64.000M; Time:1578ms
No.22  Downloading image DTBO                   progress is 100%; Size: 8.000M; Time:203ms
No.23  Downloading image Super                  progress is 100%; Size: 2630.000M; Time:79016ms
No.24  Downloading image Cache                  progress is 100%; Size: 0.043M; Time:16ms
No.25  Downloading image UserData               progress is 100%; Size: 2.157M; Time:703ms
No.26  Downloading image BootLogo               progress is 100%; Size: 2.637M; Time:78ms
No.27  Downloading image Fastboot_Logo          progress is 100%; Size: 2.637M; Time:94ms
No.28  Downloading image Socko                  progress is 100%; Size: 31.297M; Time:781ms
No.29  Downloading image Odmko                  progress is 100%; Size: 0.449M; Time:15ms
No.33  Downloading image Trustos                progress is 100%; Size: 1.475M; Time:31ms
No.34  Downloading image Teecfg                 progress is 100%; Size: 0.004M; Time:16ms
No.35  Downloading image SML                    progress is 100%; Size: 0.064M; Time:16ms
No.36  Downloading image Persist                progress is 100%; Size: 2.000M; Time:62ms
No.38  Downloading image UBOOTLoader            progress is 100%; Size: 0.731M; Time:32ms
Elapsed Time: 115.73s.
Download Successful.
[ ENTER ] DLFW_StopTask( )
[ LEAVE ] DLFW_StopTask( )
[ ENTER ] DLFW_FreeTask( )
[ LEAVE ] DLFW_FreeTask( )
[ ENTER ] DLFW_Cleanup( )
[ LEAVE ] DLFW_Cleanup( )
```

----结束

3 命令行快速集成

命令行下载的实现也是基于 DLFramework 的 API，通过执行配置的命令，实现部分下载功能。本章介绍如何通过 DLFramework 的 API 集成命令行工具，并介绍如何使用命令行工具。

3.1 实现方式

命令行下载的实现是基于 DLFramework 的 API，其实现与 API 快速集成相似。详细步骤可参考第二章。

- 创建控制台应用，设置工程的输出目录、附加包含目录和附加库目录。
- 添加 Source\Common\Drv 中的 DLFWDriver.cpp、DLFWDriver.h 和 DLFWDrvDef.h。
- 添加 Source\UI\CmdDloader 中的相关文件。
- 修改 Download.cpp（下载流程）或 CmdHelper.cpp（命令行参数）。

说明

如需功能扩展可在 Download.cpp 和 CmdHelper.cpp 基础上增加新功能。

图3-1 Source\UI\CmdDloader 源码

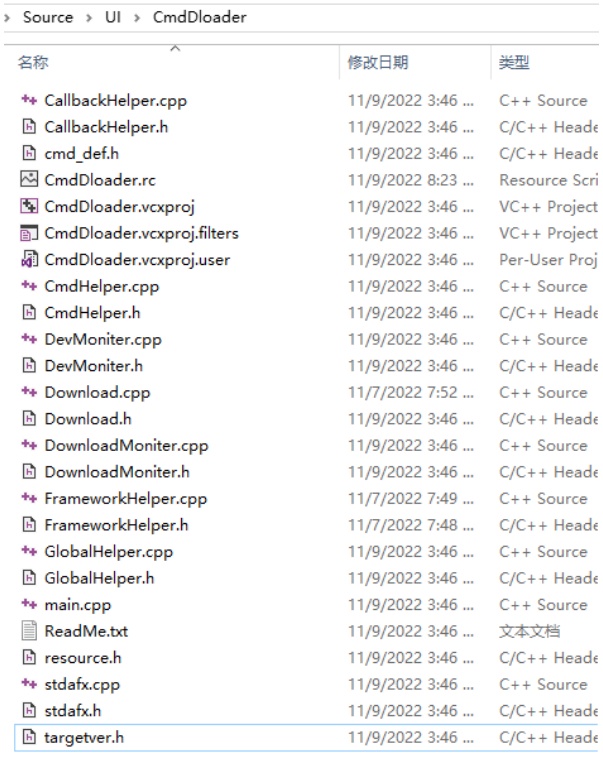
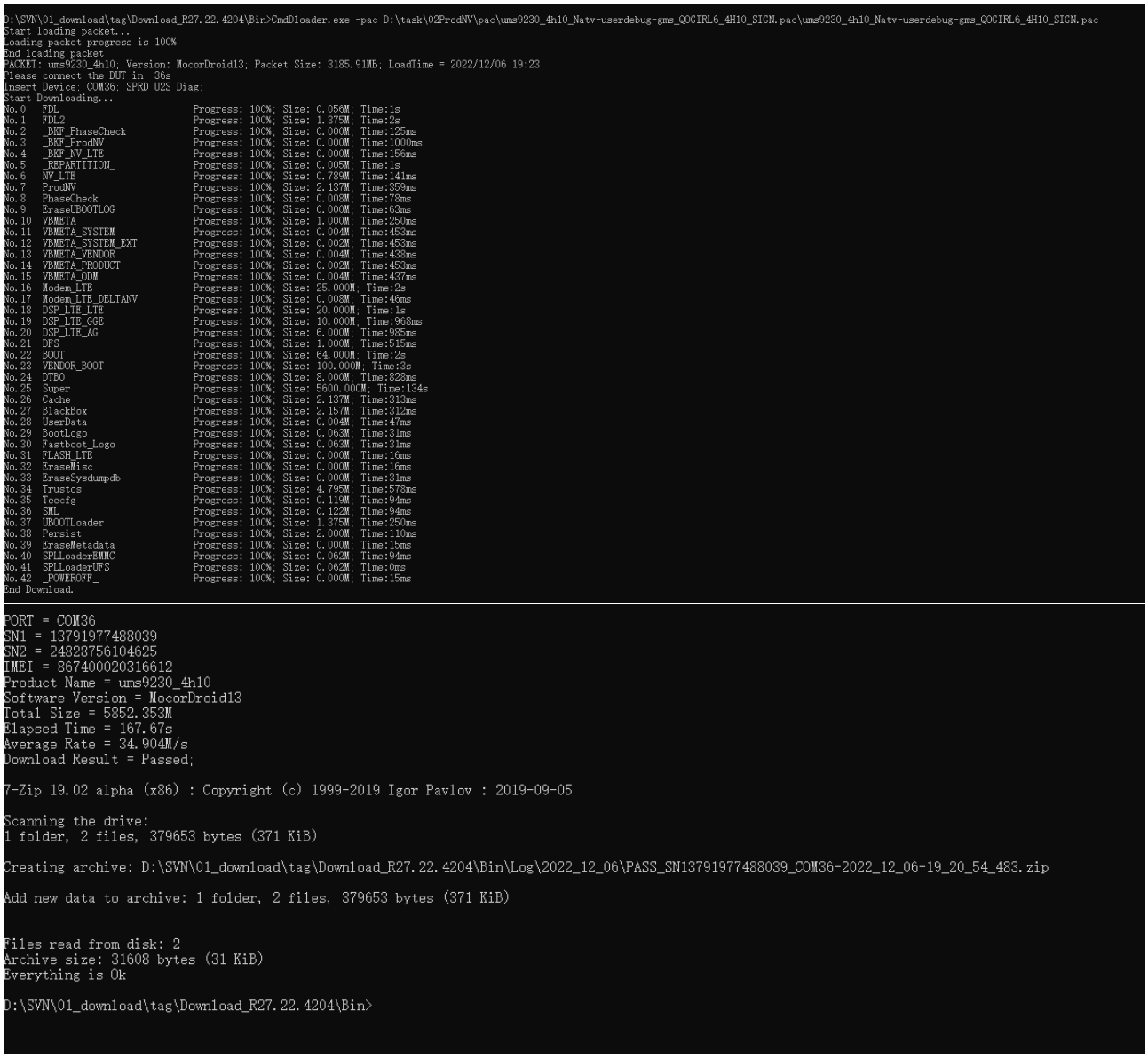


图3-2 运行效果



3.2 参数说明

3.2.1 Pac

【概述】

必选参数。指定下载资源所在路径。

所有命令均不区分大小写。

【参数值】

字符串。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac
```

3.2.2 Version

【概述】

可选参数。下载工具发布形态类型。

【参数值】

数值。值范围：

0：ResearchDownload （默认）

1：FactoryDownload

2：UpgradeDownload

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -version 1
```

3.2.3 Port

【概述】

可选参数。指定下载端口。未指定时默认为 0，自动识别端口。

命令行一拖多情况下，必须指定端口下载。

【参数值】

数值。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -port 3
```

3.2.4 Policy

【概述】

可选参数。指定 pac 的加载策略。

【参数值】

数值。值范围：

0：进行 CRC 校验，生成所有 img 文件（默认）

1：进行 CRC 校验，根据最大 size 生成文件

2：不进行 CRC 校验，根据最大 size 生成文件，最快加载 pac 速度

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -policy 1
```

3.2.5 Baudrate

【概述】

可选参数。指定下载端口波特率。

未指定该参数时，使用默认值 115200。

【参数值】

数值。值范围：57600，115200（默认），230400，460800，921600，1000000，2000000，3250000，4000000。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -baudrate 460800
```

3.2.6 SkipFileID

【概述】

可选参数。指定跳过下载分区。可指定一个或多个分区，分区之间用逗号分隔。

未指定该参数时，不跳过任何下载分区，依据实际的配置文件执行。

【参数值】

字符串。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -SkipFileID
UserData,EraseMetadata
```

3.2.7 Timeout

【概述】

可选参数。指定等待连接 DUT 时长，单位 ms。默认等待时长为 60000 ms。

【参数值】

数值。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -timeout 120000
```


3.2.8 WriteSN1/ WriteSN2

【概述】

可选参数。写设备序列号。

未指定该参数时，工具随机生成 SN。

SN 默认长度为 14。

【参数值】

字符串。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -writesn1 12345678912345 -
writesn2 11223344556677
```

3.2.9 MutiFileID/ MutiProjectID/ MutiProjectBase

【概述】

可选参数。指定共软体参数 FileID，ProjectID，ProjectBase。

命令行中共软体三个参数必须同时存在。

【参数值】

字符串。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -MutiFileID OCDT -
MutiProjectID 20701 -MutiProjectBase UNISOC
```

3.2.10 ReadNV

【概述】

可选参数。回读 NV 文件到指定路径。

【参数值】

字符串。

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -ReadNV
E:\Download_file\ReadNV
```


3.2.11 SendFlag

【概述】

可选参数。在 miscdata 的 928 字节处写标志位。

该功能需要软件支持，仅支持 UpgradeDownload 和 FactoryDownload 工具。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -sendflag
```

3.2.12 WriteEfuse

【概述】

可选参数。写熔丝。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -WriteEfuse
```

3.2.13 WriteTimeStamp

【概述】

可选参数。写时间戳。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -WriteTimeStamp
```

3.2.14 Poweroff

【概述】

可选参数。下载完成后退出下载模式。

不能与 reset 同时使用。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -Poweroff
```

3.2.15 Reset

【概述】

可选参数。下载完成后重启设备。

不能与 poweroff 同时使用。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -reset
```

3.2.16 EZMode

【概述】

可选参数。极简模式下载。

该模式不会提示连接 DUT，也不会打印任何下载中的信息。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -EZMode
```

3.2.17 DDDP

【概述】

可选参数。分区下载进度仅打印 1%和 100%。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -pac D:\test.pac -DDDP
```

3.2.18 Help

【概述】

可选参数。打印帮助信息。

如果不输入任何参数，或者输入的参数信息有误，同样会输出使用帮助说明。

【参数值】

NA

【示例】

```
D:\Download_R27.22.4204\Bin>CmdDloader.exe -help
```


4 开发参考

4.1 接口说明

4.1.1 DLFW_Startup

初始化库内部资源，当外部模块加载 DLFramework 库后应首先调用本接口，与 DLFW_Cleanup 对应。

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_Startup( DL_TYPE_ENUM eDLType, LPCSPCALLBACK_PARAM lpCallBack, LPVOID pLogUtil );
```

【参数说明】

参数名称	输入/输出	说明
eDLType	IN	Download type；参考 DL_TYPE_ENUM 定义
lpCallBack	IN	用于注册 Callback；参考 SPCALLBACK_PARAM
pLogUtil	IN	ISpLog 实例对象指针，用于日志记录

【接口说明】

NA

【接口返回】

加载成功返回 SP_OK，否则返回错误代码。

4.1.2 DLFW_Cleanup

销毁内部资源，当外部模块释放 DLFramework 库前应调用本接口。

【接口定义】

```
SP_EXPORT void SP_API DLFW_Cleanup( void );
```

【参数说明】

NA

【接口说明】

NA

【接口返回】

NA

4.1.3 DLFW_LoadPacket

加载和解析 Packet 文件。

【接口定义】

```
SP_EXPORT SPRESULT SP_API DLFW_LoadPacket( LPCWSTR lpszPacketPath, LPCWSTR
lpszImageFilesDir, uint32 u32Timeout = INFINITE );
```

【参数说明】

参数名称	输入/输出	说明
lpszPacketPath	IN	Packet 的路径
lpszImageFilesDir	IN	解压 Packet 包的目录
u32Timeout	IN	解压 Packet 包超时时间

【接口说明】

- 解压 Packet 时，会起线程进行

【接口返回】

加载成功返回 SP_OK，否则返回错误代码。

4.1.4 DLFW_LoadFiles

加载已经解压的 Packet 散件。

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_LoadFiles( LPCWSTR lpszImagesDir, uint32 u32Timeout =
INFINITE );
```

【参数说明】

参数名称	输入/输出	说明
lpszImagesDir	IN	已解压 Packet 包散件的目录

u32Timeout	IN	加载 Packet 包散件超时时间

【接口说明】

- 加载 Packet 包散件时，会起线程进行；
- Packet 包散件的目录必须包含 FileList.ini，散件加载通过解析 FileList.ini 实现；

【接口返回】

加载成功返回 SP_OK，否则返回错误代码。

4.1.5 DLFW_ReloadSettings

Framework 公共资源重新读取 ini 配置文件，并对部分资源进行初始化。

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_ReloadSettings();
```

【参数说明】

NA

【接口说明】

- 读取的配置文件包括：Download.ini、BMFileType.ini、BMTimeOut.ini、BMError.ini、BinPack.ini、MCPType.ini、DLFramework.ini

【接口返回】

加载成功返回 SP_OK，否则返回错误代码。

4.1.6 DLFW_SyncParameters

同步 Framework 公共资源的参数到 Task 任务资源，同时完成 Task 内部资源的初始化；

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_SyncParameters();
```

【参数说明】

NA

【接口说明】

- 完成 Packet 的加载之后，才会执行 Task 内部资源的初始化；
- 同步信息包括：INI 配置，Packet 的配置（XML、Packet 头信息），通过 API 接口修改的配置等；

【接口返回】

加载成功返回 SP_OK，否则返回错误代码。

4.1.7 DLFW_DeviceMoniter

下载串口监测；

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_DeviceMoniter( BOOL bStart = TRUE );
```

【参数说明】

参数名称	输入/输出	说明
bStart	IN	TRUE：启动下载串口监测；FLASE：停止监测；

【接口说明】

- 监测串口的拔插动作；
- 对串口进行确认，是否是下载端口、AT 端口、过滤端口、固定端口等；
- 通过 Callback(CBK_STRING_USB_INSERT 和 CBK_STRING_USB_REMOVE)向 UI 层发送消息；
- 启动监测串口功能时候，会对已存在的串口进行扫描，满足条件，会向 UI 层发送消息；

【接口返回】

加载成功返回 SP_OK，否则返回错误代码。

4.1.8 DLFW_CreateTask

创建 DUT 测试序列任务。

【接口定义】

```
SP_EXPORT SP_HANDLE SP_API DLFW_CreateTask( LPCSPCALLBACK_PARAM lpCallBack, LPVOID
pLogUtil, BOOL bUart = FALSE );
```

【参数说明】

参数名称	输入/输出	说明
lpCallback	IN	注册 AAL 的回调
pLogUtil	IN	ISpLog 实例对象指针，用于日志记录
bUart	IN	UART 下载

【接口说明】

- 若需支持多线程 DUT 测试，需要为每个线程创建一个任务句柄。

【接口返回】

任务创建成功返回任务句柄，否则返回 NULL。

4.1.9 DLFW_FreeTask

释放创建的测试任务。

【接口定义】

```
SP_EXPORT void      SP_API DLFW_FreeTask( SP_HANDLE hTask );
```

【参数说明】

参数名称	输入/输出	说明
hTask	IN	任务句柄

【接口说明】

NA

【接口返回】

NA

4.1.10 DLFW_RunTask

执行测试任务。

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_RunTask( SP_HANDLE hTask, DWORD dwPort );
```

【参数说明】

参数名称	输入/输出	说明
hTask	IN	任务句柄
dwPort	IN	下载的串口号

【接口说明】

- 内部创建一个工作线程下载 Image 从 PC 到 UE。
- 本接口通过线程实现异步执行，非阻塞式，即创建了工作线程即可返回，而非等到整个测试序列执行完毕后才返回。

【接口返回】

NA

4.1.11 DLFW_StopTask

中断测试任务执行。

【接口定义】

```
SP_EXPORT SPRESULT  SP_API DLFW_StopTask( SP_HANDLE hTask );
```

【参数说明】

参数名称	输入/输出	说明
hTask	IN	任务句柄

【接口说明】

NA

【接口返回】

NA

4.1.12 DLFW_SetProperty

设置属性

【接口定义】

```
SP_EXPORT SPRESULT SP_API DLFW_SetProperty( INT nProperty, INT nFlags, LPCVOID lpValue );
```

【参数说明】

参数名称	输入/输出	说明
nFlags	IN	属性标记
nProperty	IN	属性类别，参见附录 B
lpValue	IN	属性值，根据 nProperty 不同数据结构不同

【接口说明】

NA

【接口返回】

执行成功返回 SP_OK，否则返回错误代码。

4.1.13 DLFW_GetProperty

获取属性。

【接口定义】

```
SP_EXPORT SPRESULT SP_API DLFW_GetProperty( INT nProperty, INT nFlags, LPVOID lpValue );
```

【参数说明】

参数名称	输入/输出	说明
nFlags	IN	属性标记
nProperty	IN	属性类别，参见附录 B
lpValue	OUT	属性值，根据 nProperty 不同数据结构不同

【接口说明】

NA

【接口返回】

执行成功返回 SP_OK，否则返回错误代码。

4.1.14 DLFW_TaskSetProperty

设置属性

【接口定义】

```
SP_EXPORT SPRESULT SP_API DLFW_TaskSetProperty( SP_HANDLE hTask, INT nProperty, INT
nFlags, LPCVOID lpValue );
```

【参数说明】

参数名称	输入/输出	说明
hTask	IN	任务句柄
nFlags	IN	属性标记
nProperty	IN	属性类别，参见附录 B
lpValue	IN	属性值，根据 nProperty 不同数据结构不同

【接口说明】

NA

【接口返回】

执行成功返回 SP_OK，否则返回错误代码。

4.1.15 DLFW_TaskGetProperty

获取属性。

【接口定义】

```
SP_EXPORT SPRESULT SP_API DLFW_TaskGetProperty( SP_HANDLE hTask, INT nProperty, INT nFlags, LPVOID lpValue );
```

【参数说明】

参数名称	输入/输出	说明
hTask	IN	任务句柄
nFlags	IN	属性标记
nProperty	IN	属性类别，参见附录 B
lpValue	OUT	属性值，根据 nProperty 不同数据结构不同

【接口说明】

NA

【接口返回】

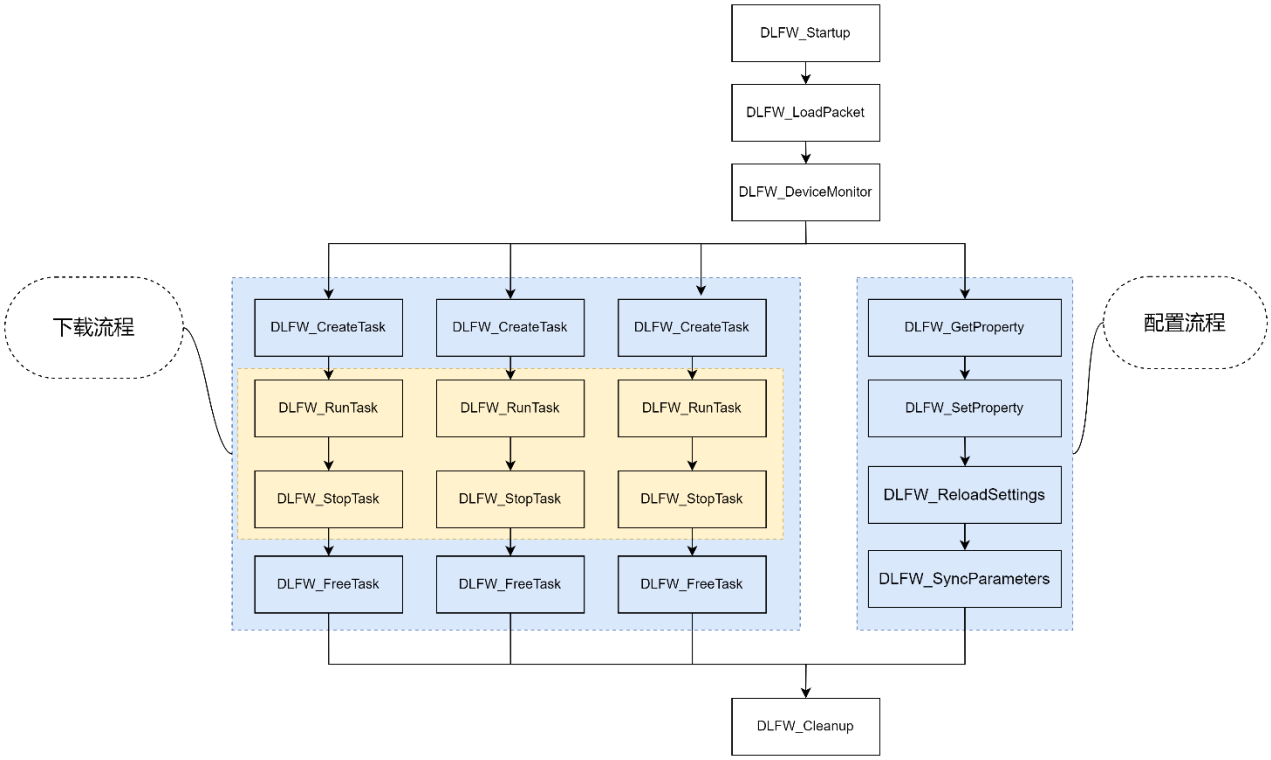
执行成功返回 SP_OK，否则返回错误代码。

4.2 调用流程

4.2.1 多线程

对于多线程（多 DUT），需要为每个线程（DUT）创建一个测试任务，彼此之间独立。

图4-1 多线程执行流程图



说明

若 Packet 不变更，仅仅是更换 DUT 或者循环测试，那么直接调用 DLFW_RunTask 即可开启一次新的测试。

4.3 数据结构

4.3.1 回调数据

回调函数原型定义：

```
typedef BOOL ( CALLBACK* LPSPDATACALLBACK )( LPCSPCALLBACK_DATA lpcbData, LPCVOID
pParam );

typedef struct _tagSPCALLBACK_PARAM
{
    LPSPDATACALLBACK pFunc;
    LPCVOID pParam;

    STRUCT_INITIALIZE( _tagSPCALLBACK_PARAM );
} SPCALLBACK_PARAM, *LPSPCALLBACK_PARAM;
typedef const SPCALLBACK_PARAM* LPCSPCALLBACK_PARAM;
```


参数名称	说明
pFunc	回调函数指针
pParam	回调参数，CAL 层调用回调时将此数据返回

回调数据结构：

```
typedef enum : uint16
{
    CALLBACK_LOAD_PACKET_PROGRESS = 0,
    CALLBACK_PACKET_INFO,
    CALLBACK_DL_STEP_DESCRIPTION,
CALLBACK_DL_IMAGE_BEGIN,
CALLBACK_DL_END,
    CALLBACK_STRING_INFO,
    CALLBACK_STEP_INFO,
    CALLBACK_PROGRESS_INFO,

    MAX_CALLBACK_DATA
} CALLBACKDATA_TYPE;


typedef struct
{
    CALLBACKDATA_TYPE eType;
    LPCVOID lpData;

} SPCALLBACK_DATA, *LPSPCALLBACK_DATA;
typedef const SPCALLBACK_DATA* LPCSPCALLBACK_DATA;
```

参数名称	说明
eType	指定回调数据类型，参见 CALLBACKDATA_TYPE 枚举定义
lpData	根据 eType 不同，结构定义不同，参见附录 A

4.4 日志输出

Download 内部通过 iSpLog 模块打印和输入各个模块的日志提供研发和产线工程师定位和分析问题。

表4-1 日志类型

类型	说明
Trace.log	下载流程中生成的 Trace Log
DLFramework.log	非下载流程中生成的 Trace Log 读写 ini、加载 Packet、生成 Packet、USB Moniter、UI 和 DLL 参数同步等

表4-2 日志 Module 分类

DLL	Module	说明
DLFramework.dll	DLFWTask	任务资源模块
	DLFramework	公共资源模块
	BMOprObserv	下载流程操作
	DevMoniter	设备端口监测
	NV	NV 相关
BMAFrame9.dll	BMAFrame	下载流程执行模块
BMPlatform9.dll	BMPlatform1	下载功能实现模块
	BMPlatform2	下载功能实现模块
	BMPlatform3	下载功能实现模块
Callback.dll	CALLBACK	回调功能模块
Config.dll	Config	参数配置模块

表4-3 日志关键字分类

查看	方法
模块信息	AuthenticationLib.dll, **FileVersion**:27.22.36.1, **Vendor**: UNISOC TECHNOLOGIES INC.
查看下载所有 OperationType	打开 Trace.log 可以查找：[ENTER] -> OnOperationStart(

查看下载指令	需要配置 Log Level 为 5，可以通过 UI 状态栏配置，也可以修改: Log Level: DLFramework.ini -> Log -> Level 数据长度：DLFramework.ini -> Log -> DataSize（显示 PC 工具和 UE 交互的数据长度） [2022-05-20 13:52:38:016][BMPlatform3][4 - Data] --> 16(0x00000010) Bytes 00000000h: 7E 00 02 02 10 01 20 90 E1 C8 00 00 0A 08 30 94 : ~..... .....0.	
	打开 Trace.log 可以查找：CMD:	
查找下载的 Base 和 Size	打开 Trace.log 可以查找：StartData	
查看读取和写入 INI 参数	Get Section	通过 ini 文件中的 Section 整体读取
	Get String	通过 ini 文件中的 Section + key 读取指定参数
	Default String	通过 ini 文件中的 Section + key 无法读取指定参数，使用默认参数
	Set String	通过 ini 文件中的 Section + key 写入指定参数
	Set Section	通过 ini 文件中的 Section 整体写入
查看不同模块参数传递	GetProperty	从其它模块获取参数
	SetProperty	设置参数到其它模块
NV	CheckNV	检查 NV
	FindNV	查找 NV
	PreserveNV	保存 NV
Backup	BackupFiles	备份文件在 UE 侧
	SaveBackupFile	备份文件在 PC 侧
跳过下载文件	SkipFile	SkipFile(Cache)
查看创建资源	Creat	Object：其它 DLL 内部资源对象；无 Object：本 DLL 内部资源对象
查看释放资源	Release	Object：其它 DLL 内部资源对象；无 Object：本 DLL 内部资源对象
查看 Function（函数）开始	[ENTER] ->	[ENTER] -> DisconnectChannel(Channel: 0x429a6c8)
查看 Function（函数）介绍	[LEAVE] <-	[LEAVE] <- DisconnectChannel( )
查看下载的最终结果	PASS , 0 表示成功	[3 - Information] OnEnd(Result: 0)
	FAIL，非 0 表示失败	[3 - Information] OnEnd(Result: 262) [1 - Error] [UB1254]Software has not supported this feature

5 附录

5.1 附录 A – 回调数据结构

5.1.1 CALLBACK_LOAD_PACKET_PROGRESS

【类型】CALLBACK_LOAD_PACKET_PROGRESS

【说明】加载 Packet 的进度。

```
/*
SPCALLBACK_DATA:: eType = CALLBACK_LOAD_PACKET_PROGRESS;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_LOAD_PACKET_PROGRESS;
*/
typedef enum
{
    CBK_PROG_BEGIN = 0,
    CBK_PROG_PROCEED = 1,
    CBK_PROG_END = 2,
} LOAD_PACKET_PROGRESS_E;

typedef struct
{
   LOAD_PACKET_PROGRESS_E    eType;
   UINT32                    u32Progress;

} CALLBACKDATA_LOAD_PACKET_PROGRESS, * LPCALLBACKDATA_LOAD_PACKET_PROGRESS;
typedef const CALLBACKDATA_LOAD_PACKET_PROGRESS* LPCCALLBACKDATA_LOAD_PACKET_PROGRESS;
```

参数名称	说明
eType	加载 Packet 的过程；参考 LOAD_PACKET_PROGRESS_E 定义
u32Progress	当 eType 等于 CBK_PROG_PROCEED 时，加载 Packet 的进度；

5.1.2 CALLBACK_PACKET_INFO

【类型】CALLBACK_PACKET_INFO

【说明】Packet 的信息

```
/*
SPCALLBACK_DATA:: eType = CALLBACK_PACKET_INFO;
SPCALLBACK_DATA::lpData = CALLBACKDATA_PACKET_INFO;
*/
typedef struct
{
    LPCWSTR lpName;
    LPCWSTR lpVersion;
    LPCWSTR lpLoadTime;
    uint64* pu64Size;

} CALLBACKDATA_PACKET_INFO, * LPCALLBACKDATA_PACKET_INFO;
typedef const CALLBACKDATA_PACKET_INFO* LPCCALLBACKDATA_PACKET_INFO;
```

参数名称	说明
lpName	Packet 名称
lpVersion	Packet 版本
lpLoadTime	加载 Packet 的时间
*pu64Size	Packet 的大小

5.1.3 CALLBACK_DL_STEP_DESCRIPTION

【类型】CALLBACK_DL_STEP_DESCRIPTION

【说明】下载步骤的描述信息

```
/*
SPCALLBACK_DATA:: eType = CALLBACK_DL_STEP_DESCRIPTION;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_DL_STEP_DESCRIPTION;
*/
typedef struct
{
    LPCWSTR lpFileID;
    LPCWSTR lpFileType;
```

```
} CALLBACKDATA_DL_STEP_DESCRIPTION, *LPCALLBACKDATA_DL_STEP_DESCRIPTION;
typedef const CALLBACKDATA_DL_STEP_DESCRIPTION* LPCCALLBACKDATA_DL_STEP_DESCRIPTION;
```

参数名称	说明
lpFileID	参考 Packet 里面 XML 的定义：SchemeList –> File -> ID
lpFileType	参考 Packet 里面 XML 的定义：SchemeList –> File -> Type

5.1.4 CALLBACK_DL_IMAGE_BEGIN

【类型】CALLBACK_DL_IMAGE_BEGIN

【说明】下载的 Image 的大小。

```
/*
SPCALLBACK_DATA:: eType = CALLBACK_DL_IMAGE_BEGIN;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_DL_IMAGE_SIZE;
*/
typedef struct
{
    uint32  u32Port;
    uint64* pu64Size;
} CALLBACKDATA_DL_IMAGE_SIZE, * LPCALLBACKDATA_DL_IMAGE_SIZE;
typedef const CALLBACKDATA_DL_IMAGE_SIZE* LPCCALLBACKDATA_DL_IMAGE_SIZE;
```

参数名称	说明
u32Port	端口号
* pu64Size;	Image 的大小

5.1.5 CALLBACK_DL_END

【类型】CALLBACK_DL_END

【说明】下载最终结果。

```
/*
SPCALLBACK_DATA:: eType = CALLBACK_DL_END;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_DL_END;
```

```
*/
typedef struct
{
    uint32  u32Port;
    uint32  u32Stage;
    uint32  u32ErrCode;
    LPCWSTR lpErrMsg;
} CALLBACKDATA_DL_END, * LPCALLBACKDATA_DL_END;
typedef const CALLBACKDATA_DL_END* LPCCALLBACKDATA_DL_END;

enum DL_STAGE
{
    DL_NONE_STAGE,
    DL_CHK_BAUD,
    DL_CONNECT,
    DL_ERASE_FLASH,
    DL_DL_STAGE,
    DL_READ_STAGE,
    DL_RESET_STAGE,
    DL_READCHIPTYPE_STAGE,
    DL_READNVITEM_STAGE,
    DL_CHANGEBUAD_STAGE,
    DL_FINISH_STAGE,
    DL_UNPLUGGED_STAGE,
    DL_PAUSED,
    DL_SECOND_ENUMPORT,
    DL_SET_FIRST_MODE
} ;
```

参数名称	说明
u32Port	端口号

<table>
<tr><td>u32Stage</td><td>参考 DL_STAGE 定义<table><tr><th>Type</th><th>说明</th></tr><tr><td>DL_NONE_STAGE</td><td>None 状态</td></tr><tr><td>DL_CHK_BAUD</td><td>检查波特率状态</td></tr><tr><td>DL_CONNECT</td><td>链接状态</td></tr><tr><td>DL_ERASE_FLASH</td><td>擦除 Flash 状态</td></tr><tr><td>DL_DL_STAGE</td><td>下载状态</td></tr><tr><td>DL_READ_STAGE</td><td>ReadNV 状态</td></tr><tr><td>DL_RESET_STAGE</td><td>重启状态</td></tr><tr><td>DL_READCHIPTYPE_STAGE</td><td>读取 ChipType 状态</td></tr><tr><td>DL_READNVITEM_STAGE</td><td>读取 NVItem 状态</td></tr><tr><td>DL_CHANGEBUAD_STAGE</td><td>改变波特率状态</td></tr><tr><td>DL_FINISH_STAGE</td><td>完成状态</td></tr><tr><td>DL_UNPLUGGED_STAGE</td><td>没有连接 DUT 状态</td></tr><tr><td>DL_PAUSED</td><td>暂停状态</td></tr><tr><td>DL_SECOND_ENUMPORT</td><td>二次枚举</td></tr><tr><td>DL_SET_FIRST_MODE</td><td>设置启动模式</td></tr></table></td></tr>
<tr><td>u32ErrCode</td><td>错误代码；参考工具 Doc/ Download Error Message Description.xlsx</td></tr>
<tr><td>lpErrMsg</td><td>错误信息；</td></tr>
</table>

5.1.6 CALLBACK_STRING_INFO

【类型】CALLBACK_STRING_INFO

【说明】用于传递字符串信息


```
typedef enum : uint32
{
CBK_STRING_POWER_MANAGE = 0,
CBK_STRING_WARN_MSG,
CBK_STRING_ERROR_MSG,
CBK_STRING_IMEI,
    CBK_STRING_MCP,
    CBK_STRING_SN1,
    CBK_STRING_SN2,
    CBK_STRING_USB_INSERT,
    CBK_STRING_USB_REMOVE,
    CBK_STRING_WARN_ERASE = 0x1000,

} STRING_INFO_TYPE_E;

/*
SPCALLBACK_DATA:: eType = CALLBACK_STRING_INFO;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_STRING_INFO;
*/
typedef struct
{
    uint32  u32Port;
    STRING_INFO_TYPE_E  eType;
    LPCWSTR lpInfo;
} CALLBACKDATA_STRING_INFO, *LPCALLBACKDATA_STRING_INFO;
typedef const CALLBACKDATA_STRING_INFO* LPCCALLBACKDATA_STRING_INFO;
```

参数名称	说明
u32Port	端口号

<table>
<tr><td>eType</td><td>参考 STRING_INFO_TYPE_E 定义<table><tr><th>Type</th><th>说明</th></tr><tr><td>CBK_STRING_POWER_MANAGE</td><td>Power Manage 信息</td></tr><tr><td>CBK_STRING_WARN_MSG</td><td>警告信息</td></tr><tr><td>CBK_STRING_ERROR_MSG</td><td>错误信息</td></tr><tr><td>CBK_STRING_IMEI</td><td>IMEI</td></tr><tr><td>CBK_STRING_MCP</td><td>McpType 信息</td></tr><tr><td>CBK_STRING_SN1</td><td>SN1</td></tr><tr><td>CBK_STRING_SN2</td><td>SN2</td></tr><tr><td>CBK_STRING_USB_INSERT</td><td>串口插入信息（串口名称）</td></tr><tr><td>CBK_STRING_USB_REMOVE</td><td>串口移除信息（串口名称）</td></tr><tr><td>CBK_STRING_WARN_ERASE</td><td>擦除 IMEI 和全擦的警告信息</td></tr></table></td></tr>
<tr><td>lpInfo</td><td>信息</td></tr>
</table>

5.1.7 CALLBACK_STEP_INFO

【类型】CALLBACK_STEP_INFO

【说明】当前 Step

```
typedef enum : uint16
{
    CBK_STEP_CHECK_BAUDRATE = 0, //检查下载波特率
    CBK_STEP_CONNECT,
    CBK_STEP_ERASE_FLASH,
    CBK_STEP_READ_FLASH,
    CBK_STEP_RESET,
    CBK_STEP_SET_FIRST_MODE,
    CBK_STEP_READ_CHIPTYPE,
    CBK_STEP_READ_NVITEM,
    CBK_STEP_UN_TRANSCODE,
```



```
    CBK_STEP_REQUIRE_SN,
    CBK_STEP_DL_BEGIN,
    CBK_STEP_DL_IMAGE,
CBK_STEP_MUTI_SOFTWARE,
CBK_STEP_STOP_AUTODLOADER
} STEP_INFO_TYPE_E;

/*
SPCALLBACK_DATA:: eType = CALLBACK_STEP_INFO;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_STEP_INFO;
*/
typedef struct
{
    uint32  u32Port;
    STEP_INFO_TYPE_E  eType;
} CALLBACKDATA_STEP_INFO, * LPCALLBACKDATA_STEP_INFO;
typedef const CALLBACKDATA_STEP_INFO* LPCCALLBACKDATA_STEP_INFO;
```

<table>
<tr><th>参数名称</th><th>说明</th></tr>
<tr><td>u32Port</td><td>端口号</td></tr>
<tr><td>eType</td><td>参考 STEP_INFO_TYPE_E 定义
<table><tr><th>Type</th><th>说明</th></tr><tr><td>CBK_STEP_CHECK_BAUDRATE</td><td>检查波特率</td></tr><tr><td>CBK_STEP_CONNECT</td><td>连接 UE</td></tr><tr><td>CBK_STEP_ERASE_FLASH</td><td>擦除 Flash</td></tr><tr><td>CBK_STEP_READ_FLASH</td><td>读取 Flash</td></tr><tr><td>CBK_STEP_RESET</td><td>重启</td></tr><tr><td>CBK_STEP_SET_FIRST_MODE</td><td>设置下次启动模式</td></tr><tr><td>CBK_STEP_READ_CHIPTYPE</td><td>读取 ChipType</td></tr><tr><td>CBK_STEP_READ_NVITEM</td><td>读取 NV 信息</td></tr></table></td></tr>
</table>

	CBK_STEP_UN_TRANSCODE	读取 Transcode
	CBK_STEP_REQUIRE_SN	请求 SN 输入
	CBK_STEP_DL_BEGIN	下载开始
	CBK_STEP_DL_IMAGE	下载 Image 开始
	CBK_STEP_MUTI_SOFTWARE	共软体请求选择项目
	CBK_STEP_STOP_AUTODLOADER	停止/退出下载工具

5.1.8 CALLBACK_PROGRESS_INFO

【类型】CALLBACK_PROGRESS_INFO

【说明】进度信息。

```
typedef enum : uint16
{
    CBK_PROGRESS_DL_IMAGE = 0,
    CBK_PROGRESS_READ_FLASH,
} PROGRESS_INFO_TYPE_E;

/*
SPCALLBACK_DATA:: eType = CALLBACK_PROGRESS_INFO;
SPCALLBACK_DATA:: lpData = CALLBACKDATA_PROGRESS_INFO;
*/
typedef struct
{
    uint32  u32Port;
    PROGRESS_INFO_TYPE_E  eType;
    uint32  u32Progress;
} CALLBACKDATA_PROGRESS_INFO, * LPCALLBACKDATA_PROGRESS_INFO;
typedef const CALLBACKDATA_PROGRESS_INFO* LPCCALLBACKDATA_PROGRESS_INFO;
```

参数名称	说明
u32Port	端口号
eType	参考 PROGRESS_INFO_TYPE_E 定义

	Type	说明
	CBK_PROGRESS_DL_IMAGE	下载 Image 的进度
	CBK_PROGRESS_READ_FLASH	读取 Flash 的进度
u32Progress	进度百分比	

5.2 附录 B – 属性

DLFramework 库提供 DLFW_SetPropery、DLFW_GetProperty、DLFW_TaskSetPropery（Task）和 DLFW_TaskGetProperty（Task、预留）接口提供灵活、扩展的属性设置。

5.2.1 DLFW_ATTR_CONFIG_OBJECT

获取 Config.dll 的实体指针。

【读取示例】

```
IConfig* pCfgObject = NULL;
DLFW_GetProperty( DLFW_ATTR_CONFIG_OBJECT, 0, &pCfgObject );
```

【设置示例】

NA

5.2.2 DLFW_ATTR_DEV_HOUND_OBJECT

获取 PortHound9.dll 的实体指针。

【读取示例】

```
IDevHound* pDevHound = NULL;
DLFW_GetProperty ( DLFW_ATTR_DEV_HOUND_OBJECT, 0, &pDevHound );
```

【设置示例】

NA

5.2.3 DLFW_ATTR_START_PATH

设置 exe 的启动路径。

【读取示例】

NA

【设置示例】

```
string_t strPath = _T("D:\\Download\\Bin");
DLFW_SetProperty( DLFW_ATTR_START_PATH, 0, ( LPVOID )strPath.c_str() );
```

5.2.4 DLFW_ATTR_PRD_VERSION

读取 Packet 的。

【读取示例】

```
LPTSTR lpString = NULL;
DLFW_GetProperty( DLFW_ATTR_PRD_VERSION, 0, &lpString );
```

【设置示例】

NA

5.2.5 DLFW_ATTR_PRODUCT_USE

设置和读取当前项目

【读取示例】

```
int nCurProduct = 0;
DLFW_GetProperty( DLFW_ATTR_PRODUCT_USE, 0, &nCurProduct );
```

【设置示例】

```
int nCurProduct = 0;
DLFW_SetProperty( DLFW_ATTR_PRODUCT_USE, 0, &nCurProduct );
```

5.2.6 DLFW_ATTR_PRODUCT_NAME_ALL

获取 XML(BMAConfig.xml 和 Packet 中的 XML)中所有的项目名称

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Transition Type Description >
*
* @param DLFW_ATTR_PRODUCT_NAME_ALL : - lppData    < LPWSTR* >
                                      - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_COMMON
{
    PVOID* lppData;
    PVOID* lppReserved;
    uint32 u32Size;
```



```
    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_COMMON );
} DLFW_ATTRDATA_COMMON, * LPDLFW_ATTRDATA_COMMON;
typedef const DLFW_ATTRDATA_COMMON* LPCDLFW_ATTRDATA_COMMON;
```

【读取示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_GetProperty( DLFW_ATTR_PRODUCT_NAME_ALL, 0, &data );
```

说明

参考代码：BOOL CMainPage::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

NA

5.2.7 DLFW_ATTR_PRODUCT_INFO_ALL

获取 XML(BMAConfig.xml 和 Packet 中的 XML)中所有的项目对应的信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Transition Type Description >
*
* @param DLFW_ATTR_PRODUCT_INFO_ALL : - lppData    < PPRODUCT_INFO_T* >
                                      - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_COMMON
{
    PVOID* lppData;
    PVOID* lppReserved;
    uint32 u32Size;

    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_COMMON );
} DLFW_ATTRDATA_COMMON, * LPDLFW_ATTRDATA_COMMON;
typedef const DLFW_ATTRDATA_COMMON* LPCDLFW_ATTRDATA_COMMON;
```

说明

PPRODUCT_INFO_T 定义参考：\Source\Common\IncludeEx\BMAGlobal.h

【读取示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_GetProperty( DLFW_ATTR_PRODUCT_INFO_ALL, 0, &data );
```


说明

参考代码：BOOL CMainPage::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

NA

5.2.8 DLFW_ATTR_CHECK_DL_FILES

读取检查下载的 Image 文件信息是否正确

【读取示例】

```
BOOL bPacketInit = FALSE;
DLFW_GetProperty(DLFW_ATTR_CHECK_DL_FILES, 0, &bPacketInit );
```

说明

bPacketInit 值等于 TRUE 时，表示下载的 Image 文件没有异常。

bPacketInit 值等于 FALSE 时，表示下载的 Image 文件存在异常，具体问题可以通过 TraceLog 进行分析；

【设置示例】

NA

5.2.9 DLFW_ATTR_PROJECT_CONFIG

读取共软体的所有 Project 信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Transition Type Description >
*
* @param DLFW_ATTR_PROJECT_CONFIG  : - lppData    < LPPROJECT_CONFIG >
                                    - u32Size    < uint32 >


*/
typedef struct _tagDLFW_ATTRDATA_COMMON
{
    PVOID* lppData;
    PVOID* lppReserved;
    uint32 u32Size;

    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_COMMON );
} DLFW_ATTRDATA_COMMON, * LPDLFW_ATTRDATA_COMMON;
typedef const DLFW_ATTRDATA_COMMON* LPCDLFW_ATTRDATA_COMMON;
```

说明

LPPROJECT_CONFIG 定义参考：\Source\Common\IncludeEx\BMAGlobal.h

【读取示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_GetProperty(DLFW_ATTR_PROJECT_CONFIG, 0, &data );
```

说明

参考代码：BOOL CMutiSoftSelectDlg::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

NA

5.2.10 DLFW_ATTR_PROJECT_ID

设置共软体选择的 Project 信息。

【读取示例】

NA

【设置示例】

```
string_t strID = _T("12345");
DLFW_SetProperty(DLFW_ATTR_PROJECT_ID, 0, ( LPVOID )strID.c_str() );
```

5.2.11 DLFW_ATTR_DL_FILE_NAME

读取所有下载的 Image 文件名称

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Type Description >
*
* @param DLFW_ATTR_DL_FILE_NAME     : - lppID      < LPWSTR* >
                                     - lppName    < LPWSTR* >
                                     - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_DLIMAGEINFO
{
    LPWSTR* lppID;
    LPWSTR* lppName;
    int32*  pn32Flag;
    uint64* pu64Size;
    uint64* pu64Offset;
    uint32  u32Size;
```

```
    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_DLIMAGEINFO );
} DLFW_ATTRDATA_DLIMAGEINFO, * LPDLFW_ATTRDATA_DLIMAGEINFO;
typedef const DLFW_ATTRDATA_DLIMAGEINFO* LPCDLFW_ATTRDATA_DLIMAGEINFO;
```

【读取示例】

```
DLFW_ATTRDATA_DLIMAGEINFO data;
DLFW_GetProperty( DLFW_ATTR_DL_FILE_NAME, 0, &data );
```

说明

参考代码：BOOL CMainPage::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

```
DLFW_ATTRDATA_DLIMAGEINFO data;
DLFW_SetProperty( DLFW_ATTR_DL_FILE_NAME, 0, &data );
```

说明

参考代码：BOOL CMainPage::SetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

5.2.12 DLFW_ATTR_DL_FILE_STATE

读取所有下载的 Image 文件的 Flag（是否要下载）信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Type Description >
*
* @param DLFW_ATTR_DL_FILE_STATE    : - lppID     < LPWSTR* >
                                     - pn32Flag   < int32* >
                                     - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_DLIMAGEINFO
{
    LPWSTR* lppID;
    LPWSTR* lppName;
    int32*  pn32Flag;
    uint64* pu64Size;
    uint64* pu64Offset;
    uint32  u32Size;

    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_DLIMAGEINFO );
} DLFW_ATTRDATA_DLIMAGEINFO, * LPDLFW_ATTRDATA_DLIMAGEINFO;
typedef const DLFW_ATTRDATA_DLIMAGEINFO* LPCDLFW_ATTRDATA_DLIMAGEINFO;
```

【读取示例】

```
DLFW_ATTRDATA_DLIMAGEINFO data;
DLFW_GetProperty(DLFW_ATTR_DL_FILE_STATE, 0, &data );
```

说明

参考代码：BOOL CMainPage::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

```
DLFW_ATTRDATA_DLIMAGEINFO data;
DLFW_SetProperty(DLFW_ATTR_DL_FILE_STATE, 0, &data );
```

说明

参考代码：BOOL CMainPage::SetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

5.2.13 DLFW_ATTR_DL_FILE_SIZE

读取所有下载的 Image 文件的 size 信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Type Description >
*
* @param DLFW_ATTR_DL_FILE_SIZE     : - lppID      < LPWSTR* >
                                     - pu64Size    < uint64* >
                                     - pu64Offset   < uint32 >
                                     - u32Size     < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_DLIMAGEINFO
{
    LPWSTR* lppID;
    LPWSTR* lppName;
    int32*  pn32Flag;
    uint64* pu64Size;
    uint64* pu64Offset;
    uint32  u32Size;

    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_DLIMAGEINFO );
} DLFW_ATTRDATA_DLIMAGEINFO, * LPDLFW_ATTRDATA_DLIMAGEINFO;
typedef const DLFW_ATTRDATA_DLIMAGEINFO* LPCDLFW_ATTRDATA_DLIMAGEINFO;
```

【读取示例】

```
DLFW_ATTRDATA_DLIMAGEINFO data;
```



```
DLFW_GetProperty(DLFW_ATTR_DL_FILE_SIZE, 0, &data );
```

说明

参考代码：BOOL CMainPage::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

NA

5.2.14 DLFW_ATTR_DL_DATA_INFO

读取所有下载的 Image 文件的 size 信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Type Description >
*
* @param DLFW_ATTR_DL_DATA_INFO     : - lppName    < LPWSTR* >
                                     - pu64Size    < uint64* >
                                     - pu64Offset   < uint32 >
                                     - u32Size     < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_DLIMAGEINFO
{
    LPWSTR* lppID;
    LPWSTR* lppName;
    int32*  pn32Flag;
    uint64* pu64Size;
    uint64* pu64Offset;
    uint32  u32Size;

    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_DLIMAGEINFO );
} DLFW_ATTRDATA_DLIMAGEINFO, * LPDLFW_ATTRDATA_DLIMAGEINFO;
typedef const DLFW_ATTRDATA_DLIMAGEINFO* LPCDLFW_ATTRDATA_DLIMAGEINFO;
```

【读取示例】

```
DLFW_ATTRDATA_DLIMAGEINFO data;
DLFW_GetProperty(DLFW_ATTR_DL_DATA_INFO, 0, &data );
```

说明

参考代码：BOOL CMainPage::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

NA

5.2.15 DLFW_ATTR_BACKUP_FILES

设置和读取所有备份的文件信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Transition Type Description >
*
* @param DLFW_ATTR_BACKUP_FILES     : - lppData    < PFILE_INFO_T* >
                                     - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_COMMON
{
    PVOID* lppData;
    PVOID* lppReserved;
    uint32 u32Size;
    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_COMMON );
} DLFW_ATTRDATA_COMMON, * LPDLFW_ATTRDATA_COMMON;
typedef const DLFW_ATTRDATA_COMMON* LPCDLFW_ATTRDATA_COMMON;
```

说明

PFILE_INFO_T 定义参考：\Source\Common\IncludeEx\BMAGlobal.h

【读取示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_GetProperty(DLFW_ATTR_BACKUP_FILES, 0, &data );
```

说明

参考代码：BOOL CPageCalibration::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_SetProperty(DLFW_ATTR_BACKUP_FILES, 0, &data );
```

说明

参考代码：BOOL CPageCalibration::SetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

5.2.16 DLFW_ATTR_BACKUP_NV_ENABLE

设置和读取备份 NV 的总开关

【读取示例】

```
BOOL bBackup = FALSE;
DLFW_GetProperty(DLFW_ATTR_BACKUP_NV_ENABLE, 0, &bBackup );
```

【设置示例】

```
BOOL bBackup = FALSE;
DLFW_SetProperty(DLFW_ATTR_BACKUP_NV_ENABLE, 0, &bBackup );
```

5.2.17 DLFW_ATTR_BACKUP_NV_FILES

设置和读取所有备份的文件信息

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Transition Type Description >
*
* @param DLFW_ATTR_BACKUP_NV_ITEMS  : - lppData    < PNV_BACKUP_INFO_T* >
                                     - lpReserved < LPWSTR* >
                                     - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_COMMON
{
    PVOID* lppData;
    PVOID* lppReserved;
    uint32 u32Size;
    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_COMMON );
} DLFW_ATTRDATA_COMMON, * LPDLFW_ATTRDATA_COMMON;
typedef const DLFW_ATTRDATA_COMMON* LPCDLFW_ATTRDATA_COMMON;
```

说明

PNV_BACKUP_INFO_T 定义参考：\Source\Common\DLFWDef\DLFWStructExDef.h

【读取示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_GetProperty(DLFW_ATTR_BACKUP_NV_FILES, 0, &data );
```

说明

参考代码：BOOL CNvBackupOptList::GetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

【设置示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_SetProperty(DLFW_ATTR_BACKUP_NV_FILES, 0, &data );
```

说明

参考代码：BOOL CNvBackupOptList::SetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

5.2.18 DLFW_ATTR_FLASH_OPERATION,

设置 Flash 的操作：读取、写入、擦除

【数据结构】

```
/**
* @brief DLFW_PROPERTY_TYPE_ENUM    : - Parameter  < Transition Type Description >
*
* @param DLFW_ATTR_FLASH_OPERATION  : - lppData    < PFILE_INFO_T >
                                     - u32Size    < uint32 >
*/
typedef struct _tagDLFW_ATTRDATA_COMMON
{
    PVOID* lppData;
    PVOID* lppReserved;
    uint32 u32Size;
    STRUCT_INITIALIZE( _tagDLFW_ATTRDATA_COMMON );
} DLFW_ATTRDATA_COMMON, * LPDLFW_ATTRDATA_COMMON;
typedef const DLFW_ATTRDATA_COMMON* LPCDLFW_ATTRDATA_COMMON;
```

说明

PFILE_INFO_T 定义参考：\Source\Common\IncludeEx\BMAGlobal.h

【读取示例】

NA

【设置示例】

```
DLFW_ATTRDATA_COMMON data;
DLFW_SetProperty(DLFW_ATTR_FLASH_OPERATION, 0, &data );
```

说明

参考代码：BOOL CPageFlashOptions::SetProperty()

data 使用时，需要自己在堆开辟和释放空间，用于数据的传递；

5.2.19 DLFW_ATTR_FLASH_ERASE_ALL

设置全擦

【读取示例】

NA

【设置示例】

```
BOOL bEraseAll = TRUE;
DLFW_SetProperty(DLFW_ATTR_FLASH_ERASE_ALL, 0, &bEraseAll );
```

5.2.20 DLFW_ATTR_POWER_OFF

设置和读取下载结束后对 UE 关机的操作

【读取示例】

```
BOOL bPowerOff = TRUE;
DLFW_GetProperty(DLFW_ATTR_POWER_OFF, 0, &bPowerOff );
```

【设置示例】

```
BOOL bPowerOff = TRUE;
DLFW_SetProperty(DLFW_ATTR_POWER_OFF, 0, &bPowerOff );
```

5.2.21 DLFW_ATTR_LOG_PATH

设置 Log 路径到 Task

【读取示例】

NA

【设置示例】

```
string_t strLogPath = _T("D:\\DownloadLog\\Task1 ");
DLFW_TaskSetProperty( DLFW_ATTR_LOG_PATH, 0, &strLogPath.c_str() );
```

5.2.22 DLFW_ATTR_BARCODE_SN

设置 SN 到 Task

【读取示例】

NA

【设置示例】

```
DLFW_ATTRDATA_BARCODE dlSN;
dlSN.pszSN1 = m_dlgCurBarcode.m_strBarcode.operator LPCWSTR();
dlSN.pszSN2 = m_dlgCurBarcode.m_strBarcode2.operator LPCWSTR();
dlSN.u8SN1Len = ( uint8 )m_dlgCurBarcode.m_strBarcode.GetLength();
dlSN.u8SN2Len = ( uint8 )m_dlgCurBarcode.m_strBarcode2.GetLength();
DLFW_TaskSetProperty( DLFW_ATTR_BARCODE_SN, dwPort, &dlSN ) ) ;
```

6 参考文档

1. 《UNISOC Simba Framework 集成指导说明.docx》

