UC和UC、UC和eSpace之间消息交互（文本、表情、截图）

# 1 模块初始化

## 流程说明

在使用IM组件之前，需要先进行初始化操作

## 代码示例

步骤1：

// 初始化IM部分底层数据

TUP\_RESULT tIMRet = tup\_im\_init();

if (TUP\_SUCCESS != tIMRet)

{

return FALSE;

}

步骤2：

// 初始化离线文件组件

char logPath[100] = G:\\UC\\test\\demo\\IMMSGDemo\\IMMSGDemo";

char \*pcLog = logPath;

OFFLINE\_INITCONFIG stInitConfig;

memset(&stInitConfig, 0, sizeof(OFFLINE\_INITCONFIG));

stInitConfig.stLogConfig.enLevel = OFFLINE\_LOG\_DEBUG;

stInitConfig.stLogConfig.uiLogFileSize = 15;

stInitConfig.stLogConfig.pcFilePath = pcLog;

stInitConfig.uiHandleNum = 100;

stInitConfig.bUseSvn = TUP\_FALSE;

tIMRet = tup\_offlinefile\_init(&stInitConfig);

if (TUP\_SUCCESS != tIMRet)

{

AfxMessageBox(\_T("tup\_offlinefile\_init failed!"));

return FALSE;

}

## 接口描述

TUP\_API TUP\_RESULT tup\_im\_init();

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_init |
| 接口类型 | API |
| 描述 | 功能描述：初始化im子系统，使用im之前调用该接口。  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_offlinefile\_initlogin(const OFFLINE\_INITCONFIG\* pstConfig, const OFFLINE\_LOGININFO\* pstLoginInfo, const OFFLINE\_USERINFO\* pstUserInfo, TUP\_ULONG\* pLogInResult);

|  |  |
| --- | --- |
| 接口名称 | tup\_offlinefile\_init |
| 接口类型 | API |
| 描述 | 功能描述：初始化离线文件组件，使用离线文件之前调用该接口。  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

# 2 登陆

## 2.2 流程说明

### 2.2.1 初始化登录信息和读取表情配置文件（Emotion.ini），存储到map容器（在OnInitDialog函数中执行）

### 2.2.2 设置登录参数并登录

#### 2.2.2.1 设置IM登录参数

#### 2.2.2.2 设置offlinefile登录参数（IM登录成功后执行此操作）

### 2.2.3 启用心跳服务

## 2.3 代码示例

### 2.3.1 对话框初始化示例

// 初始化登录对话框

m\_ipAddress.SetAddress(172,22,8,61);

m\_editPort.SetWindowTextW(TEXT("8011"));

m\_editAccount.SetWindowTextW(TEXT("wf01"));

m\_editPSD.SetWindowTextW(TEXT("1qaz@WSX"));

// 读取表情配置文件(Emotion.ini),存储到map容器

TCHAR emotionPath[\_MAX\_PATH];

memset(emotionPath, 0, \_MAX\_PATH);

GetCurrentDirectory(\_MAX\_PATH, emotionPath);

CString strEmotionPath(emotionPath);

strEmotionPath += \_T("\\Emotion\\Emotion.ini");

m\_strDefaultEmotionPath = emotionPath;

m\_strDefaultEmotionPath += \_T("\\Emotion\\");

for (int i = 0; i < EMOTION\_CUSTOM\_MAXCOUNT; ++i)

{

CString strTemp(\_T(""));

strTemp.Format(\_T("%d"), i + 1);

TCHAR emotionKey[MAX\_PATH] = {0};

GetPrivateProfileString((LPCTSTR)strTemp, \_T("character"), \_T(""), emotionKey, MAX\_PATH, strEmotionPath);

TCHAR emotionValue[MAX\_PATH] = {0};

GetPrivateProfileString((LPCTSTR)strTemp, \_T("file"), \_T(""), emotionValue, MAX\_PATH, strEmotionPath);

m\_EmotionReplace.insert(make\_pair<CString, CString>(CString(emotionKey), CString(emotionValue)));

}

### 2.3.2 登录操作示例代码

// 获取IP数据

BYTE field0, field1, field2, field3;

m\_ipAddress.GetAddress(field0, field1, field2, field3);

char IP[100];

memset(IP, 0, 100);

sprintf\_s(IP, "%d.%d.%d.%d", field0, field1, field2, field3);

// 获取服务器IP和端口

CString cstrPort;

m\_editPort.GetWindowText(cstrPort);

std::string strPort = CTools::UNICODE2UTF(cstrPort);

int iPort = atoi(strPort.c\_str());

// 设置服务器IP和端口

TUP\_RESULT tRet = tup\_im\_setserveraddress(IP, iPort);

if (TUP\_SUCCESS != tRet)

{

AfxMessageBox(TEXT("tup\_im\_setserveraddress failed."));

}

// 设置IM回调消息上报函数地址

tRet = tup\_im\_register\_callback(NotifyCallBack::IMNotify);

if (TUP\_SUCCESS != tRet)

{

AfxMessageBox(TEXT("tup\_im\_register\_callback failed."));

}

// 使能消息分发，否则消息不会上报

tRet = tup\_im\_setdispatchmessage(TUP\_TRUE);

if (TUP\_SUCCESS != tRet)

{

AfxMessageBox(TEXT("tup\_im\_setdispatchmessage failed."));

return;

}

// 获取控件上的登录账号和密码

IM\_S\_LOGIN\_ARG arg;

CString account;

m\_editAccount.GetWindowText(account);

CTools::CString2Char(account, arg.account, IM\_D\_MAX\_ACCOUNT\_LENGTH);

CString cstrPSD;

m\_editPSD.GetWindowText(cstrPSD);

CTools::CString2Char(cstrPSD, arg.password, IM\_D\_MAX\_PASSWORD\_LENGTH);

// 版本号固定

CString cstrVersion = TEXT("v1.1.11.103");

CTools::CString2Char(cstrVersion, arg.version, IM\_D\_MAX\_VERSION\_LENGTH);

// 登录UC服务器

IM\_S\_LOGIN\_ACK ack;

tRet = tup\_im\_login(&arg, &ack);

// 初始化loginInfo参数字段

loginInfo.enTlsMode = EN\_TLS\_VERIFY\_NONE;

loginInfo.stTlsPara.pcCaCertPath = HTTP\_NULL;

loginInfo.stTlsPara.pcClientCertPath = HTTP\_NULL;

loginInfo.stTlsPara.pcClientCertType = HTTP\_NULL;

loginInfo.stTlsPara.pcClientKeyPath = HTTP\_NULL;

loginInfo.stTlsPara.pcClientKeyType = HTTP\_NULL;

loginInfo.stTlsPara.pcClientPrivKeyPw = HTTP\_NULL;

loginInfo.stTlsPara.enVerifyServerMode = EN\_VERIFY\_SERVER\_PEER;

NEW\_NULL\_MEMORY\_TPL(loginInfo.pcLocalIP, 20);

std::string localIP(IP);

(void)memcpy\_s(loginInfo.pcLocalIP, 20, localIP.c\_str(), static\_cast<UINT>(localIP.size()));

loginInfo.timeout = 0;

loginInfo.tryConnectTimeout = 0;

loginInfo.enServerMode = OFFLINE\_TRANSFER\_CLOUDSTORE;

loginInfo.bEncryTrans = TUP\_TRUE;//云盘为TRUE

OFFLINE\_SERVERINFO stHttpsSer1;

memset(&stHttpsSer1, 0, sizeof(OFFLINE\_SERVERINFO));

OFFLINE\_SERVERINFO stHttpsSer2;

memset(&stHttpsSer2, 0, sizeof(OFFLINE\_SERVERINFO));

if (OFFLINE\_TRANSFER\_UM == loginInfo.enServerMode)

{

stHttpsSer1.pcUrl = "https://10.174.3.140:18820/umserver2/UM";

}

else if (OFFLINE\_TRANSFER\_CLOUDSTORE == loginInfo.enServerMode)

{

stHttpsSer1.pcUrl = "https://clouddrive-008.huawei.com";

}

OFFLINE\_SERVERINFO stHttpSer1;

memset(&stHttpSer1, 0, sizeof(OFFLINE\_SERVERINFO));

OFFLINE\_SERVERINFO stHttpSer2;

memset(&stHttpSer2, 0, sizeof(OFFLINE\_SERVERINFO));

if (OFFLINE\_TRANSFER\_UM == loginInfo.enServerMode)

{

stHttpSer1.pcUrl = "http://10.174.3.140:18820/umserver2/UM";

}

else

{

stHttpSer1.pcUrl = "http://clouddrive-008.huawei.com";

}

OFFLINE\_SERVERLIST stLstHttpsSer;

memset(&stLstHttpsSer, 0, sizeof(OFFLINE\_SERVERLIST));

stLstHttpsSer.ulCnt = 2;

stLstHttpsSer.stServerUrl[0] = stHttpsSer1;

stLstHttpsSer.stServerUrl[1] = stHttpsSer2;

loginInfo.stHttpsSerList = stLstHttpsSer;

OFFLINE\_SERVERLIST stLstHttpSer;

memset(&stLstHttpSer, 0, sizeof(OFFLINE\_SERVERLIST));

stLstHttpSer.ulCnt = 2;

stLstHttpSer.stServerUrl[0] = stHttpSer1;

stLstHttpSer.stServerUrl[1] = stHttpSer2;

loginInfo.stHttpSerList = stLstHttpSer;

loginInfo.ulMaxFileSize = 30 \* 1024u \* 1024u;// 30M

// 探测服务器刷新时间

loginInfo.lServerExpiredAt = 1000 \* 60;// 1分钟

loginInfo.bPartTrans = TUP\_FALSE;

// 初始化userInfo参数字段

userInfo.pcLoginName = "wwx302776";

userInfo.pcPassword = "WF#hf034439";

NEW\_NULL\_MEMORY\_TPL(userInfo.pcDeviceType, 512);

(void)memcpy\_s(userInfo.pcDeviceType, 512, "pc", 7);

NEW\_NULL\_MEMORY\_TPL(userInfo.pcDeviceName, 512);

TCHAR szBuff[256] = {0};

DWORD dwSize = 256;

if (::GetComputerName(szBuff ,&dwSize))

{

CString strTempp(szBuff);

std::string name = CTools::UNICODE2UTF(strTempp);

(void)memcpy\_s(userInfo.pcDeviceName, 512, name.c\_str(), static\_cast<UINT>(name.size()));

}

std::string sysinfo, sysversion, syssn;

GetDeviceInfo(sysinfo, sysversion, syssn);

unsigned int oslength = sysinfo.length() + 1;

NEW\_NULL\_MEMORY\_TPL(userInfo.pcDeviceOS, oslength);

unsigned int snlength = syssn.length() + 1;

NEW\_NULL\_MEMORY\_TPL(userInfo.pcDeviceSN, snlength);

unsigned int verlength = sysversion.length() + 1;

NEW\_NULL\_MEMORY\_TPL(userInfo.pcDeviceVersion, verlength);

(void)memcpy\_s(userInfo.pcDeviceOS, oslength, sysinfo.c\_str(), sysinfo.length());

(void)memcpy\_s(userInfo.pcDeviceSN, snlength, syssn.c\_str(), syssn.length());

(void)memcpy\_s(userInfo.pcDeviceVersion, verlength, sysversion.c\_str() ,sysversion.length());

tRet = tup\_offlinefile\_login(&loginInfo, &userInfo, &loginRes);

// 启用心跳服务

TUP\_RESULT tRet = tup\_im\_sendheartbeat();

## 2.4 接口描述

TUP\_RESULT tup\_im\_setserveraddress(const TUP\_CHAR\* ip, TUP\_UINT16 port);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_setserveraddress |
| 接口类型 | API |
| 描述 | 功能描述：设置服务器地址  参数说明：param1:服务器地址  Param2:服务器端口  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_register\_callback(TUP\_IM\_FN\_CALLBACK\_PTR callback);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_register\_callback |
| 接口类型 | API |
| 描述 | 功能描述：注册回调  参数说明：回调函数  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_setdispatchmessage(TUP\_BOOL isDispatch);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_setdispatchmessage |
| 接口类型 | API |
| 描述 | 功能描述：使能分发消息  参数说明：TUP\_BOOL  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_login(const IM\_S\_LOGIN\_ARG\* arg, IM\_S\_LOGIN\_ACK\* ack);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_login |
| 接口类型 | API |
| 描述 | 功能描述：登录  参数说明：param[in] 登录请求  Param[out]:登录结果  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_offlinefile\_login(const OFFLINE\_LOGININFO\* pstLoginInfo, const OFFLINE\_USERINFO\* pstUserInfo, TUP\_ULONG\* pLogInResult);

|  |  |
| --- | --- |
| 接口名称 | tup\_offlinefile\_login |
| 接口类型 | API |
| 描述 | 功能描述：设置用户账号服务器地址  参数说明：param[in]:登录信息  param[in]:登录用户信息  Param[out]:登录结果  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_getuserinfo(const IM\_S\_GETUSERINFOARG\* arg,IM\_S\_USERINFO\* ack);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_getuserinfo |
| 接口类型 | API |
| 描述 | 功能描述：获取用户信息请求  参数说明：账号  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_API TUP\_RESULT tup\_im\_sendheartbeat();

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_sendheartbeat |
| 接口类型 | API |
| 描述 | 功能描述：发送心跳消息  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

# 发送和接收消息

## 发送

### 流程说明

1. 启用IM输入状态服务
2. 精确查找接收方用户的信息
3. 对发送消息的格式化处理
4. 设置发送消息请求参数
5. 发送消息
   * 1. 代码示例

// 启用IM输入状态服务

TUP\_RESULT tRet = tup\_im\_notifyiminputting(&m\_statusArg);

// 精确查找接收方用户的信息

tRet = tup\_im\_getuserinfo(&argRecv, &userInfoRecv);

**// 消息长度<500的发送格式**

sendMSG.Replace(\_T(" "), \_T("&nbsp;"));

sendMSG.Replace(\_T("\r\n"), \_T("&lt;BR&gt;"));// 使用的控件是richedit所以回车是\r\n

sendMSG.Replace(\_T("<BR>"), \_T("&amp;lt;BR&amp;gt;"));

// 用TinyXml去实现汉字会出现乱码(编码问题),所以直接用eSpace特定格式去生成发送内容

// 纯文本发送格式

strSend = \_T("<r><c>&lt;imbody&gt;&lt;imagelist/&gt;&lt;html&gt;&lt;![CDATA[&lt;FONT style=&apos;font-size:14px&apos; face=&apos;Segoe UI&apos;&gt;");

strSend += sendMSG;

strSend += CString(\_T("&lt;/FONT&gt;]]&gt;&lt;/html&gt;&lt;content&gt;"));

strSend += sendMSG;

strSend += \_T("&lt;/content&gt;&lt;/imbody&gt;</c></r>");

strSend.Replace(\_T("&amp;"), \_T("&"));

// 纯表情符号发送格式

<r><c>&lt;imbody&gt;&lt;imagelist/&gt;&lt;html&gt;&lt;![CDATA[&lt;FONT style=&apos;font-size:14px&apos; face=&apos;Segoe UI&apos;&gt;&lt;IMG style=&quot;WIDTH: 26px; HEIGHT: 26px&quot; border=0 hspace=0 src=&quot;file:///UC.InternalImg//emotion/15.gif&quot;&gt;&lt;/FONT&gt;]]&gt;&lt;/html&gt;&lt;content&gt;/ZC&lt;/content&gt;&lt;/imbody&gt;</c></r>

// 图文混合发送格式（文本和表情符号，eSpace不支持截图和文本混合发送）

<r><c>&lt;imbody&gt;&lt;imagelist/&gt;

&lt;html&gt;&lt;![CDATA[&lt;FONT style=&apos;font-size:14px&apos; face=&apos;Segoe UI&apos;&gt;Hello&lt;IMG style=&quot;WIDTH: 26px; HEIGHT: 26px&quot; border=0 hspace=0 src=&quot;file:///UC.InternalImg//emotion/12.gif&quot;&gt;Bingo&lt;/FONT&gt;]]&gt;&lt;/html&gt; &lt;content&gt;&lt;![CDATA[Hello/;&gt;Bingo]]&gt;&lt;/content&gt;

&lt;/imbody&gt;</c></r>

**//消息长度超过大于500小于1000发送格式**

先进行压缩再进行编码操作，最终发送格式

<r><g>1</g><c>压缩编码后的内容</c></r>

**//消息长度大于1000禁止发送**

**//截图发送格式**

<imbody><imagelist><img><imgname>20151105160726.png</imgname><imgdata \_\_type\_\_="url">/download/1419571487\_0\_18\_1018\_201511\_20151105\_3652973482465386263-32303135313130353136303732362e706e67.png\_true\_121fdd69e8dd58258ff679014fd747deee0705395634b13dbcdfca92b6bc9a15/allowredirect</imgdata></img></imagelist><html><![CDATA[<FONT style='font-size:14px' face='Segoe UI'><IMG style="BORDER-BOTTOM: #cccccc 1px solid; BORDER-LEFT: #cccccc 1px solid; WIDTH: 154px; HEIGHT: 180px; BORDER-TOP: #cccccc 1px solid; BORDER-RIGHT: #cccccc 1px solid" id='20151105160726.png"' onmouseup=ShowImageMenu(this) ondblclick=ShowImage(this) border=0 hspace=3 vspace=3 src="file:///UC.ExternalImg//temp/20151105160726.png"></FONT>]]></html><content>/:um\_begin{/download/1419571487\_0\_18\_1018\_201511\_20151105\_3652973482465386263-32303135313130353136303732362e706e67.png\_true\_121fdd69e8dd58258ff679014fd747deee0705395634b13dbcdfca92b6bc9a15/allowredirect|Img|1|20151105160726.png|0|212;82;[OfflineTransmissionCode][NULL]}/:um\_end</content></imbody>

对上面这段格式进行压缩再编码生成最终发送格式为：

<r><g>1</g><c>压缩编码后的截图内容</c><r>

// 发送即时消息

tRet = tup\_im\_sendim(&m\_SendImArg);

// 发送截图消息

先调用接口tup\_offlinefile\_uploadfile将图片上传到UM服务器，再调用接口tup\_im\_sendim将图片相关信息发送给接收方

* + 1. 接口描述

1. TUP\_RESULT tup\_im\_notifyiminputting(const IM\_S\_CHAT\_STATUS\_ARG\* arg);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_notifyimputting |
| 接口类型 | API |
| 描述 | 功能描述：通知IM输入状态  参数说明：arg请求  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_sendim(const IM\_S\_SENDIM\_ARG\* arg);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_sendim |
| 接口类型 | API |
| 描述 | 功能描述：发送im消息  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_API TUP\_RESULT tup\_offlinefile\_uploadfile(const OFFLINE\_UP\_FILEINFO\* pUpFileInfo, OFFLINEHANDLE\* pFileHandle,

OffLineTransProgressNotify offLineProgNotifyFunc, OffLineFileRspNotify offLineRspNotifyFunc);

|  |  |
| --- | --- |
| 接口名称 | tup\_offlinefile\_uploadfile |
| 接口类型 | API |
| 描述 | 功能描述：文件上传  参数说明：Input:pUpFileInfo 文件上传信息  Input:offLineProgNotifyFunc过程回调函数  Input:offLineRspNotifyFunc 上传结束响应回调  Output:pFileHandle文件句柄  返回值： TUP\_RESULT  TUP\_SUCCESS 成功  错误码 失败 |
|  |  |

## 接收

### 3.1.1 接收消息后先解析<g>标签，判断消息是否被压缩，1表示压缩，0表示未压缩，如果消息被压缩，则先进行解码再解压缩操作，然后替换转义标签符号

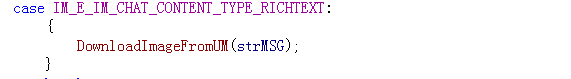
### 3.1.2 纯文本消息

因为UC和eSpace传回的类型不一致，因此采用以下方式判断解析



### 3.1.3 富文本消息

只能接收来自TUP端的富文本消息，eSpace的富媒体消息是在纯文本消息里进行处理的



### 3.1.4 如果是富媒体消息，先调用接口tup\_offlinefile\_downloadfile从UM服务器上下载图片再进行界面显示

### 3.1.5 接口描述

|  |  |
| --- | --- |
| 接口名称 | tup\_offlinefile\_downloadfile |
| 接口类型 | API |
| 描述 | 功能描述：文件下载  参数说明：Input:pDownFileInfo 文件下载信息  Input:offLineProgNotifyFunc过程回调函数  Input:offLineRspNotifyFunc响应回调  Output:pFileHandle文件句柄  返回值： TUP\_RESULT  TUP\_SUCCESS 成功  错误码 失败 |
|  |  |

# 退出

## 流程说明

### 登录状态：已登录

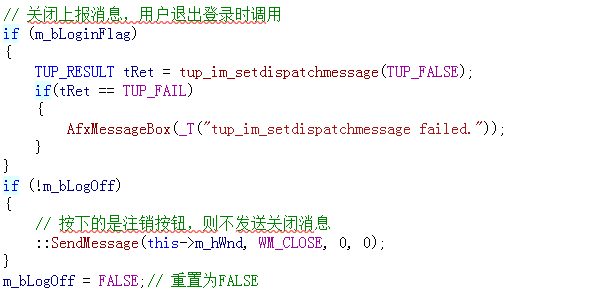
1. 取消底层模块中未处理的消息处理
2. 注销UC服务器的登录
3. 向窗口发送WM\_CLOSE消息

### 登录状态：未登录

直接向窗口发送WM\_CLOSE消息

## 4.2 代码示例





## 4.3 接口描述

tup\_im\_setcancelsendingmessage

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_setcancelsendingmessage |
| 接口类型 | API |
| 描述 | 功能描述：取消底层模块中未处理的消息处理  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_logout();

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_logout |
| 接口类型 | API |
| 描述 | 功能描述：退出  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_im\_setdispatchmessage(TUP\_BOOL isDispatch);

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_setdispatchmessage |
| 接口类型 | API |
| 描述 | 功能描述：关闭上报消息  参数说明：TUP\_BOOL  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

## 4.4注销

和退出流程相同，但是注销不会向窗口发送WM\_CLOSE消息，可以继续执行登录或退出操作

# 关闭对话框后组件反初始化

1. 释放IM部分底层数据

tIMRet = tup\_im\_uninit();

1. 去初始化离线文件

tIMRet = tup\_offlinefile\_uninit();

1. 接口描述

TUP\_RESULT tup\_im\_uninit();

|  |  |
| --- | --- |
| 接口名称 | tup\_im\_uninit |
| 接口类型 | API |
| 描述 | 功能描述：卸载im库  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |

TUP\_RESULT tup\_offlinefile\_uninit();

|  |  |
| --- | --- |
| 接口名称 | tup\_offlinefile\_uninit |
| 接口类型 | API |
| 描述 | 功能描述：文件传输去初始化  参数说明：无  返回值：  TUP\_SUCCESS 成功  错误码 失败 |