菊风业务框架

VoIP Client

发表日期: 2015-07-02 访问 http://www.juphoon.com 了解更多产品信息

菊风 VoIP SDK 用户开发指南

菊风系统软件有限公司
http://www.juphoon.com
电话: +86-574-87287820
传真: +86-574-87304379

目 录

表目录

图目录

1. 产品概述

MTC (Multimedia Talk Client)是菊风公司的 VoIP 客户端框架，实现了 NGN/软交换/3GPP IMS VoIP 标准规定的客户端业务功能，包括客户端登陆/注销、基本音视频通话、音视频补充业务等功能。MTC 的目的就是简化 GUI 客户端对 VoIP 业务开发的复杂度，使得 GUI 开发人员可以轻装上阵，甚至不需要对 VoIP 业务标准有深入的了解。

1.1 用途

本手册的用途是帮助程序开发人员正确使用 VoIP Client 开发框架 MTC 来开发 VoIP 客户端软件产品。

1.2 阅读对象

本手册的读者应具备基本 C 语言编程知识、对 VoIP 业务标准 和 MTF 用户接口有基本了解。

1.3 内容简介

本手册详细介绍了 VoIP SDK (MTC) 的使用方法。

1.4 缩略词和缩写词

下表罗列了在本手册中使用的缩略词和缩写词。

缩略/缩写词	完整描述
ZOS	Zero Operation System
DNS	Domain Name System
MSF	Multi-Service Framework
MTF	Multimedia Telephone Framework
MTC	VoIP Client Framework
SIP	Session Initiation Protocol
URI	Uniform Resource Identifier
IMS	IP Multimedia Subsystem

表 1-1 缩略词和缩写词

1.5 参考

[1]《MTC VoIP Client API 参考指南》

2. 软件设计

2.1 客户端框架

菊风针对 VoIP 规范功能要求的客户端开发框架，称为 Multimedia Talk Client (MTC)，用户可以通过 MTC 接口可以实现符合 VoIP 标准的客户端。 客户端框架如下图所示：

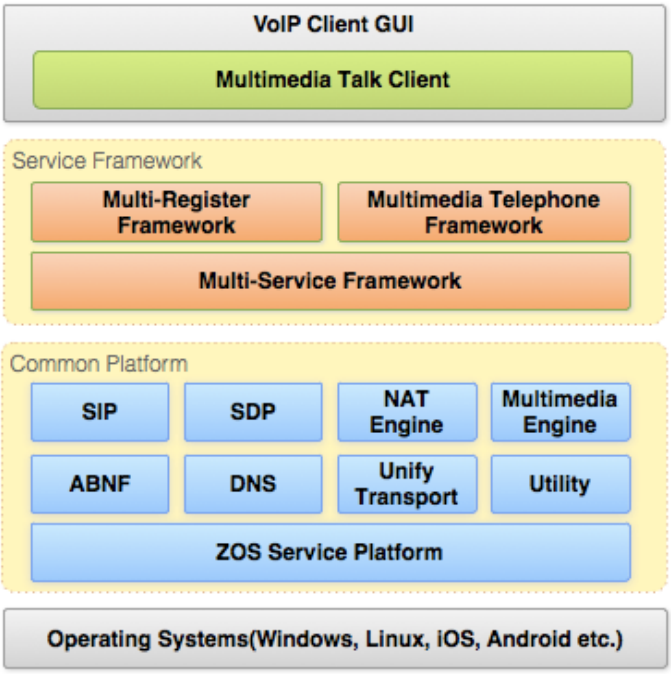

图 2-1 VoIP 客户端架构图

MTC 框架中包括以下功能模块：

- 登陆和注销
- 语音呼叫
- 补充业务（如呼叫等待等）
- 日志管理
- 业务参数管理

MTC 通过 MSF(Multi-Service Framework) 经由 SIP 协议栈实现与 VoIP 服务器的业务流程交互。

VoIP GUI 程序通过调用 MTC API 发起业务请求，例如登陆，发起语音呼叫等。 MTC 里面只是实现了基本的 GUI 界面操作之下的业务逻辑，使用 MTC 客户端的 GUI 集成开发会更加快速和简洁。

2.2 核心状态机

客户端在运行中分为以下状态：

- 未登陆状态
- 注册中状态
- 登陆成功状态

- 取消注册状态（注销后回退到未登陆状态）

从 MTC 的角度考虑，在不同状态下用户的操作范围是不一样的。比如登陆中状态不操作联系人列表、发起 VoIP 音视频会话等功能。登陆成功后可以操作所有功能。

MTC 的核心状态机是根据用户请求和 MTF 的事件状态来驱动整个 VoIP 业务逻辑。从客户端角度看，通过 MTC 业务请求接口和 GUI 回调机制以简便的方式实现业务功能。

GUI 客户端无须使用 MTC 的核心状态机接口，本节的目的是简单介绍 MTC 的业务逻辑实现过程。

2.3 用户数据管理

用户数据管理主要包括以下几个部分：

- 联系人管理
- 呼叫记录管理
- 短消息记录管理

联系人信息、呼叫记录和短消息记录等数据属于静态数据文件，保存在客户端本地

MTC 载入用户数据后，在内存中形成数据结构信息，GUI 可以根据数据的 ID 来引用数据内容

2.4 业务参数配置

MTC 中业务参数配置接口包括以下部分：

- 本地 IP 地址、日志基本等基本参数
- 用户信息
- SIP 本地和服务器参数
- 音视频媒体参数
- VoIP 的特殊业务参数

3. License 激活和回收

对于使用 License 控制的 VoIP SDK，需要调用相关接口实现 license 的激活和回收功能。

3.1 License激活和回收

3.1.1 接口和回调

接口名称	接口描述	GUI 回调
MtcLcs.Mtc_LcsActivate	根据 MtcCli.Mtc_CliInit 的返回值， 判断是否发起 license 激活请求。	存在
MtcLcs.Mtc_LcsDeactivateDevice	当设备不使用时，删除 license ，并回收 license	存在

表 3-1 License 激活和回收接口

License 激活和回收需要界面(GUI)处理的回调:

用户操作	操作详细描述	
MtcLcs.Mtc_LcsActivate	回调函数 mtcLcsActivated 的 bActivated 参数为 true	License 激活请求成功。 需要在该回调中重新调用 MtcCli.Mtc_CliInit 进行初始化
	回调函数 mtcLcsActivated 的 bActivated 参数为 false	License 激活请求失败
MtcLcs.Mtc_LcsDeactivateDevice	回调函数 mtcLcsDeviceDeactivatedM 的 bDeactivated 参数为 true	License 回收请求成功
	回调函数 mtcLcsDeviceDeactivatedM 的 bDeactivated 参数为 false	License 回收请求失败

表 3-2 License 回调处理

3.1.2 License 激活流程

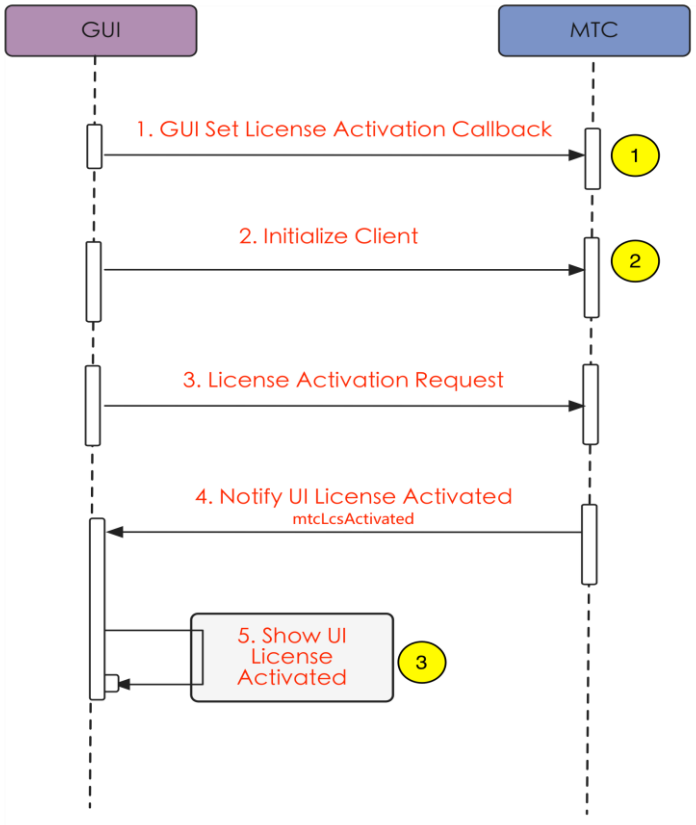

图 3-1 License 激活过程

1	设置 license 请求的成功的回调 MtcLicenseCb.registerCallback，必选。

2	使用 MtcCli.Mtc_CliInit 启动客户端系统（比如初始化 ZOS， XML 等）。
3	如 MtcCli.Mtc_CliInit 返回值为 MtcLcsConstants.MTC_LCS_ERR_NEED_ACT_LICSEN，则调用 MtcLcs.Mtc_LcsActivate 发起 license 激活请求。 由回调函数 mtcLcsActivated 上报 license 激活请求的结果。 如果回调函数的参数 bActivated 为 true，表示 license 请求成功，此时需要在 回调函数中重新发起 MtcCli.Mtc_CliInit 进行客户端的初始化。 如果回调函数的参数 bActivated 为 false，表示 license 请求失败，此时可由界面反馈错误结果。

表 3-3 客户端 License 激活过程说明

3.1.3 License 解绑流程

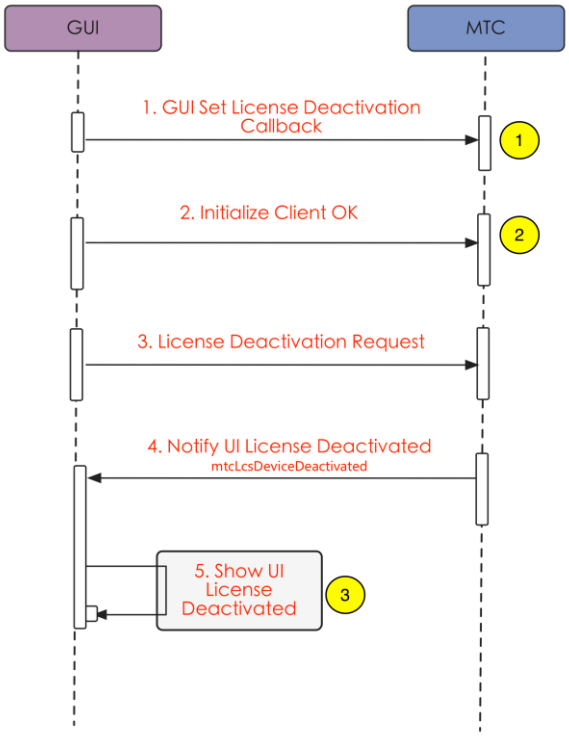

图 3-2 License 解绑回收过程

1	设置 license 请求的成功的回调 MtcLicenseCb.registerCallback，必选。
2	客户端已成功获得 license 并初始化。
3	使用 MtcLcs.Mtc_LcsDeactivateDevice 发起 license 解绑回收请求。 由回调函数 mtcLcsDeviceDeactivated 上报 license 解绑回收请求的结果。 如果回调函数的参数 bDeactivated 为 true，表示 license 回收解绑成功， 如果回调函数的参数 bDeactivated 为 false，表示 license 回收解绑失败。 此时可由界面反馈成功或者错误提示。

表 3-4 客户端解绑回收过程说明

3.1.4 License 激活和解绑示例代码

```
//设置 license 激活的回调
```

```
    Mtc_LcsCbSetActivatedOk((PFN_MTCLCSACTIVATECB) OnLicenseActivateOk);
//回调函数实现
    void OnLicenseActivateOk(bool bActivated)
    {
        if(bActivated)
        {
            //license 激活成功，重新初始化客户端
            int iRet = Mtc_CliInit(CUtf8String(wzProfilePath));
            if(iRet == ZOK)
            {
                PostMessage(hwnd, UM_LAUNCHCLR, 0, 0);
                return;
            }
            else   if(iRet == ZFAILED)
            {
                MessageBox(NULL,_T("Mtc_CliInit Error"), _T("ERROR"), MB_OK);
                exit(0);
            }
        }
        else
        {
            MessageBox(NULL,_T("License Activated Error"), _T("ERROR"), MB_OK);
            exit(0);
        }
     }
//发起 License 激活请求
    if(MTC_LCS_ERR_NEED_ACT_LICSEN == iRet)
    {
        Mtc_LcsCbSetActivatedOk((PFN_MTCLCSACTIVATECB)OnLicenseActivateOk);
        //向服务器 192.168.0.156 发起 license 请求并保存 license.sign 文件到目录 wzLicensePath
        Mtc_LcsActivate(CUtf8String("192.168.0.156"),
        CUtf8String("roytel"), ZNULL, CUtf8String(wzLicensePath));
        return FALSE;
    }
//解绑回收 license
    private void btnDeactivation_Click(object sender, RoutedEventArgs e)
    {
        string sLicensePath = System.AppDomain.CurrentDomain.BaseDirectory;
        sLicensePath = sLicensePath + "license.sign";
        //删除本地 license 并发起 license 解绑和回收请求
        int iRet = MtcLcs.Mtc_LcsDeactivateDevice("192.168.0.156", sLicensePath);
        if (MtcLcs.MTC_LCS_FILE_NOT_EXIST == iRet)
        {
            MessageBox.Show("License file not exist,\ns cannot deactivate the device!");
```

```
        }
    }
```

表 3-5 license 激活和回收示例代码

4. 客户端管理

客户端管理涉及一下过程处理：

- 销毁和初始化
- 启动和停止
- 参数设置

4.1 初始化和销毁

4.1.1 接口与回调

接口名称	接口描述
Mtc_Cli.Mtc_CliInit	初始化 ZOS、XML 等资源，初始媒体模块，获取用户列表，启动业务引擎
Mtc_Cli.Mtc_CliDestroy	关闭所有的业务、协议栈等任务，释放所有内存、媒体等资源

表 4-1 客户端初始化和销毁接口

4.1.2 初始化操作过程

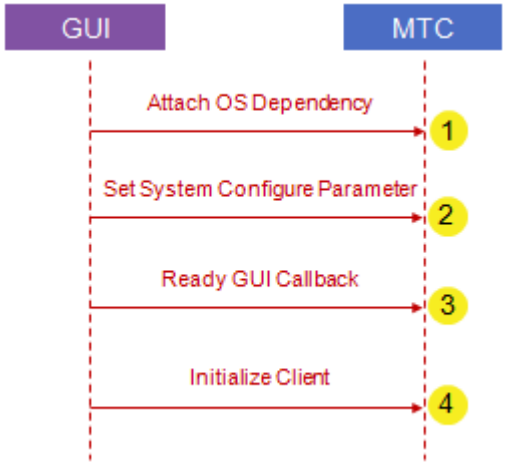

图 4-1 客户端初始化过程

1	设置一些预设参数
2	设置 MTC 业务操作回调 (GUI Callback
3	使用 Mtc_Cli.Mtc_CliInit 启动客户端系统（比如初始化 ZOS，XML 等）。 系统启动，用户信息等还没有加载。

表 4-2 客户端初始化过程说明

4.1.3 销毁操作过程

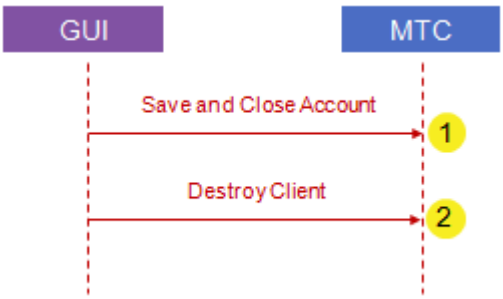

图 4-2 客户端销毁过程

1	用户需要确保会话被管理、配置和日志信息已经保存好。一旦客户端停止，所有相关内存资源会进入无效状态。
2	调用 Mtc_Cli.Mtc_CliDestroy

表 4-3 客户端销毁过程说明

4.2 启动和停止

4.2.1 接口与回调

接口名称	接口描述
Mtc_Cli.Mtc_CliStart	启动 SIP 等协议栈任务、MTF 等业务组件、初始化 MTC 状态
Mtc_Cli.Mtc_CliStop	复位 MTC 业务状体、停止 MTF 等业务组件、停止 SIP 等协议栈任务

表 4-4 客户端启动和停止接口

4.2.2 启动操作过程

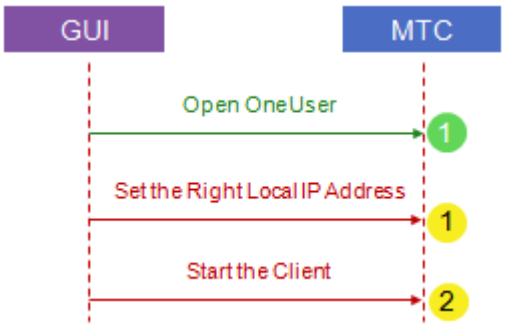

图 4-3 客户端启动过程

1	在启动客户端前，需要确认用户账号已经被正确的打开（Mtc_Cli.Mtc_CliOpen）。原因是协议栈、尤其是业务框架有很多配置信息是跟账号绑定，如果账号没有被打开，则就没有这些配置参数，将会导致启动服务无法正常使用。
1	MTC 目前提供的本地 IP 地址的接口有 Mtc_CliDb.Mtc_CliDbGetLocalIp，此接口只能返回一个本地 IP 地址。在实际部署中，PC 会存在多个网卡和 IP 地址，因此，建议本地 IP 地址采用操作系统提供的接口直接获取和设置，在 Mtc_Cli.Mtc_CliStart 启动前用 Mtc_CliDb.Mtc_CliDbSetLocalIp 设置选择的本地 IP 地址。
2	在启动前（Mtc_Cli.Mtc_CliStart），用户需要确保 MTF 等业务数据内存区中存放的本地 IP 地址是正确的地址。原因是 SIP 协议栈等任务启动时会根据本地 IP 地址绑定服务端口（比如 5060）,如果地址不正确，将导致绑定失败

表 4-5 客户端启动过程说明

4.2.3 停止操作过程

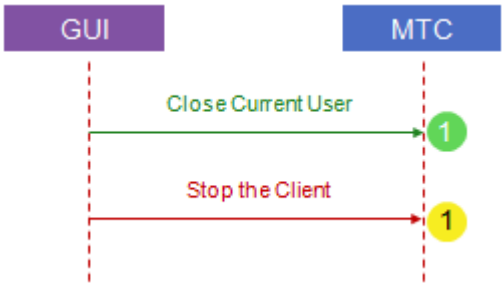

图 4-4 客户端停止过程

1	在停止客户端前，需要确认用户账号已经被正确的关闭（Mtc_Cli.Mtc_CliClose）。
1	用户使用 Mtc_Cli.Mtc_CliStop 停止客户端服务（协议及其业务）

表 4-6 客户端停止过程说明

4.3 参数设置

客户端参数分为以下两种参数类型：

- 全局基本参数 （MTC 没有此方面的参数）
- 用户相关的网络、业务和媒体参数

客户端参数存放在每个用户的账号文件信息中，用户如果读取或修改参数，需要首先打开用户账号，否则获取的参数都是默认数值，在用户账号打开后会被覆盖掉。

4.3.1 音频设置

4.3.1.1 输入设备设置

接口名称	接口描述
MtcMedia. Mtc_AudioGetInputDev	获取当前输入设备名称
MtcMedia. Mtc_AudioSetInputDev	设置当前输入设备名称
MtcMedia. Mtc_AudioGetInputDevCnt	获取所有输入设备数量
MtcMedia. Mtc_AudioEnumInputDev	获取指定索引的输入设备名称，索引值范围是 0 至【设备数量 - 1】
MtcMedia. Mtc_AudioSetInputVol	设置麦克风输入音量，范围 0 至 100
MtcMedia. Mtc_AudioGetInputVol	获取麦克风输入音量
MtcMedia. Mtc_AudioMuteInput	设置麦克风静音状态

表 4-7 音频输入设备接口

4.3.1.2 输出设备设置

接口名称	接口描述
MtcMedia. Mtc_AudioGetOutputDev	获取当前输出设备名称
MtcMedia. Mtc_AudioSetOutputDev	设置当前输出设备名称
MtcMedia. Mtc_AudioGetOutputDevCnt	获取所有输出设备数量
MtcMedia. Mtc_AudioEnumOutputDev	获取指定索引的输出设备名称，索引值范围是 0 至【设备数量 - 1】
MtcMedia. Mtc_AudioSetOutputVol	设置扬声器输出音量，范围 0 至 100
MtcMedia. Mtc_AudioGetOutputVol	获取扬声器输出音量
MtcMedia. Mtc_AudioMuteOutput	设置扬声器静音状态

表 4-8 音频输出设备接口

4.4 样例代码

4.4.1 单帐号初始化

```
/* 初始配置文件目录为 profiles */
Mtc_Cli.Mtc_CliInit(…);

/* 定义业务对象类，继承 GUI 回调接口 */
MtcCallCb.mtcCallCbIncoming(…);
MtcCallCb.mtcCallCbAlerted(…);
MtcCallCb.mtcCallCbTalking(…);
MtcCallCb.mtcCallCbTermed(…);

/* 使用 profiles 目录下配置文件初始化模块 */
Mtc_Cli.Mtc_CliOpen(…);

/* 可以在此调用 db 设置接口设置参数 */

/* 启动 MTC 各个模块 */
if (Mtc_Cli.Mtc_CliStart() != MtcCommon.ZOK)
{
    Mtc_Cli.Mtc_CliDestory();
    Return MtcCommon.ZFAILED;
}

/* 其他初始化工作 */

/* 启动登录 */
Mtc_Cli.Mtc_CliLogin();
```

5. 用户管理

用户管理包括以下三部分：

- 账号管理，包括读取已有账号列表、创建新账号等操作
- 用户登录和注销，负责接入和取消电信运营商的服务网络
- 账号参数设置，包括本地网络地址、SIP 账号和服务器网络信息、音视频能力设置等

如果 GUI 开发者采用自有的账号管理，则读取、创建、删除、选择账号等操作接口无需考虑，但是打开和关闭用户是必须要做的，目的是完成用户数据的载入（此时是默认值，需要重新设置）

5.1 账号管理

用户与账号在服务个体是相同的，但是名称上是有不同的，比如 Alice 在移动运营商获得的业务账号是 8000，本地账号管理（如 Alice_CMCC）则是用户 Profile 的管理，所以，在 MTC 账号管理中，账号名称只是 Profile 的名称，创建后就不能更改（除非删除）。所以在名称说法有以下约定：

- 账号名称：用户 Profile 的名称，MTC 中也是文件夹的名称，如 Alice_CMCC。
- 用户名称：服务提供商提供的业务账号，如 8000。
- 用户昵称：用户个性化的显示名称，如 Alice Huang。

账号管理只是对账号目录进行管理，当新建或选择账号时，并不意味着账号的配置参数被载入。用户必须选择正确的账号，通过打开账号操作才能完成账号的配置参数的载入，然后才能设置各种参数。

5.1.1 用户接口

接口名称	接口描述
MtcProf.Mtc_ProfGetUserSize	获取所有账号数目
MtcProf.Mtc_ProfGetUser	根据账号索引获取账号名称，索引范围是 0 到帐号数目-1
MtcProf.Mtc_ProfGetCurUser	获取当前账号
MtcProf.Mtc_ProfCreateUser	创建账号目录
MtcProf.Mtc_ProfDeleteUser	删除账号配置、日志文件和目录
MtcProf.Mtc_ProfExistUser	账号名称是否有效
MtcProf.Mtc_ProfSaveProvision	保存当前用户的配置文件
MtcProf.Mtc_ProfSaveContact	保存当前用户的联系人信息
MtcProf.Mtc_ProfSaveCallLogs	保存当前用户的呼叫历史记录
MtcProf.Mtc_ProfSaveImLogs	保存当前用户的 IM 历史记录
MtcProf.Mtc_ProfGetAbsoPath	获取绝对路径

表 5-1 用户管理接口

5.1.2 读取账号列表

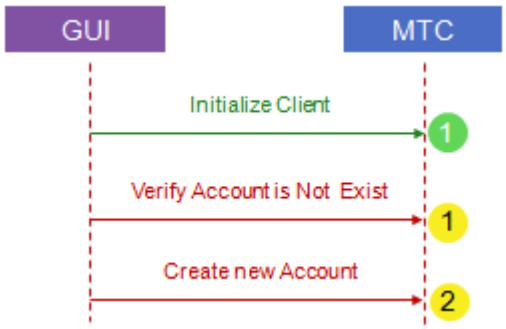

图 5-1 读取账号列表过程

1	在读取账号列表前，需要确保客户端资源已经成功的初始化（Mtc_Cli.Mtc_CliInit）。用户在初始化客户端时，还可以设置账号管理的目录（默认是 Profiles 目录）
1	使用 MtcProf.Mtc_ProfGetUserSize 获取账号目录中的所有账号（子目录）数目。

	MtcProf.Mtc_ProfGetUser
2	使用 MtcProf.Mtc_ProfGetUser 获取索引位置的账号名称

表 5-2 读取账号列表说明

5.1.3 获取当前账号

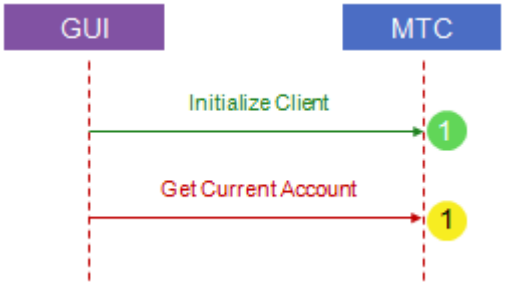

图 5-2 获取当前账号过程

1	在获取账号前，需要确保客户端资源已经成功的初始化 (Mtc_Cli.Mtc_CliInit)。
1	使用 MtcProf.Mtc_ProfGetCurUser 获取当前账号。如果没有创建或选择过账号，则返回空、

表 5-3 获取当前账号说明

5.1.4 创建新账号

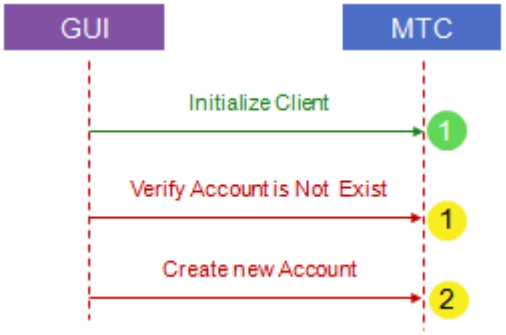

图 5-3 创建新账号过程

1	在创建账号前，需要确保客户端资源已经成功的初始化 (Mtc_Cli.Mtc_CliInit)。
1	使用 MtcProf.Mtc_ProfExistUser 判断账号是否存在
2	使用 MtcProf.Mtc_ProfCreateUser 创建新账号，MTC 会创建新文件夹

表 5-4 创建新账号说明

5.1.5 删除账号

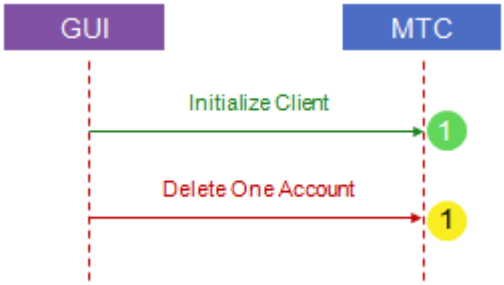

图 5-4 删除账号过程

1	在获取账号前，需要确保客户端资源已经成功的初始化 (Mtc_Cli.Mtc_CliInit)。
1	使用 MtcProf.Mtc_ProfDeleteUser 删除账号。如果账号存在，则账号的用户文件和目录将被删除。

表 5-5 删除账号过程说明

5.2 打开与关闭

用户打开是指用户在选择账号后，把用户数据文件从账号目录中载入的过程。用户打开过程包括将各个业务和应用组件数据与 MSF 的数据组件实现绑定，然后加载用户账号中的数据文件内容到内存中。如果账号名称为空，则用户打开将只是初始化默认的数据内容。

用户关闭是指用户数据从内存中写入目录文件中，同时解除 MSF 数据组件与业务和应用组件的绑定。如果用户打开的时候账号为空，则写入操作将不被执行。

5.2.1 用户接口

接口名称	接口描述
Mtc_Cli.Mtc_CliOpen	打开对应账号名称的用户，载入用户数据
Mtc_Cli.Mtc_CliClose	写入用户数据，并且关闭用户

表 5-6 打开和关闭接口

5.2.2 打开操作过程

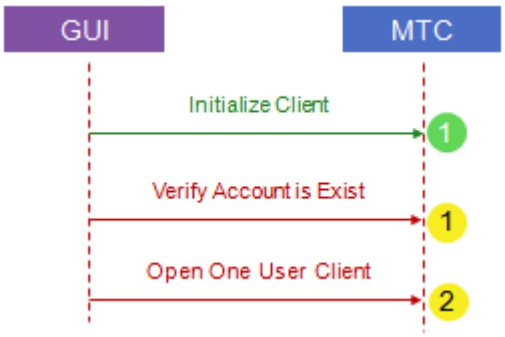

图 5-5 用户打开过程

1	在创建账号前，需要确保客户端资源已经成功的初始化 (Mtc_Cli.Mtc_CliInit)。
1	使用 MtcProf.Mtc_ProfExistUser 判断账号是否存在。如果在打开用户前已经有判断，则忽略此步骤。
2	使用 Mtc_Cli.Mtc_CliOpen 打开用户，把用户数据载入内存中

表 5-7 用户打开过程说明

5.2.3 关闭操作过程

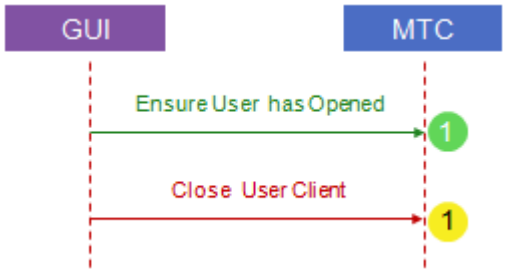

图 5-6 用户关闭过程

1	在关闭用户前，需要确认用户账号已经被正确的打开（Mtc_Cli.Mtc_CliOpen）。
1	用户使用 Mtc_Cli.Mtc_CliClose 关闭用户。在用户被关闭后，GUI 可以选择重新打开用户获取服务。

表 5-8 用户关闭过程说明

5.3 登陆与注销

客户端支持本地和服务器两种模式下登陆。本地登陆是指客户端无须向服务器发起注册，即可登陆客户端。但是，客户端在商用部署时往往采用服务器模式下登陆。用户可以使用 Mtc_DbGetUserReg 接口判断登陆模式是否是服务器模式。

5.3.1 接口与回调

用户实现登陆与注销的接口有：

接口名称	接口描述	GUI 回调存在
Mtc_Cli.Mtc_CliLogin	发起登陆请求	存在
Mtc_Cli.Mtc_CliLogout	发起注销请求	存在

表 5-9 登陆与注销用户接口

登陆与注销的接口相关的 GUI 回调处理：

用户操作	操作详细描述	
Mtc_Cli.Mtc_CliLogin	MtcCliCb.mtcCliCbRegOk	SIP REGISTER 请求已经被服务器接受
	MtcCliCb.mtcCliCbRegFailed	SIP REGISTER 请求已经被服务器拒绝或发送失败
	MtcCliCb.mtcCliCbLclLoginOk	本地登陆成功
	MtcCliCb.mtcCliCbServLoginOk	服务器登陆成功
	MtcCliCb.mtcCliCbLoginFailed	服务器登陆失败
Mtc_Cli.Mtc_CliLogout	MtcCliCb.mtcCliCbLclLogout	本地注销成功
	MtcCliCb.mtcCliCbServLogout	服务器注销成功

表 5-10 登陆与注销 GUI 回调

5.3.2 发起登陆过程

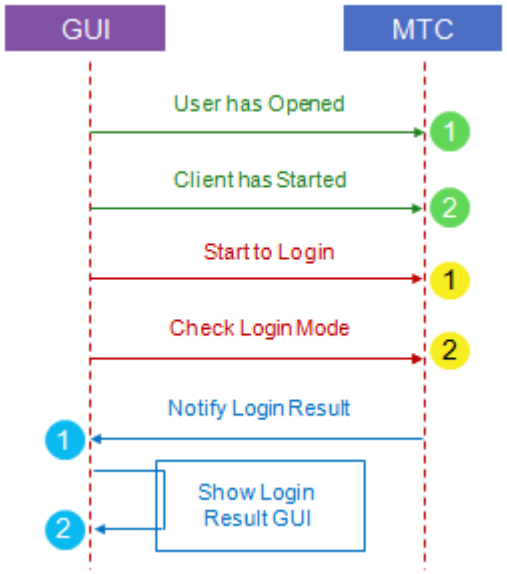

图 5-7 发起登陆过程

1	用户账号已经成功打开 (Mtc_Cli.Mtc_CliOpen)。
2	客户端已经成功启动 (Mtc_Cli.Mtc_CliStart)
1	使用 Mtc_Cli.Mtc_CliLogin 发起登陆请求。
2	获取登陆模式，如果是本地登陆，则设置好相关的 GUI 窗口布局，准备显示登陆成功窗口。
1	在客户端初始化的时候设置好 MtcCliCb.mtcCliCbServLoginOk， MtcCliCb.mtcCliCbLoginFailed GUI 回调，登陆结果将会由这两个回调发起通知。
2	在 MtcCliCb.mtcCliCbServLoginOk 回调中显示登陆成功窗口。在 MtcCliCb.mtcCliCbLoginFailed 显示登陆失败提示信息。
!	建议在登陆成功后，加载联系人列表、呼叫和短消息历史记录等本地用户数据。 如果是本地登陆模式，在 Mtc_Cli.Mtc_CliLogin 函数里面会立即调用登陆通知 GUI 回调，因此对于建议先调整好登陆成功的窗口布局，然后再发起登陆。在 Mtc_Cli.Mtc_CliLogin 函数返回后不建议做窗口调整操作（除非此函数立即返回失败） 如果是服务器模式，除非本地资源不足或发送失败，否则登陆结果会异步通知，Mtc_Cli.Mtc_CliLogin 会立即返回。

表 5-11 发起登陆过程说明

5.3.3 取消登陆过程

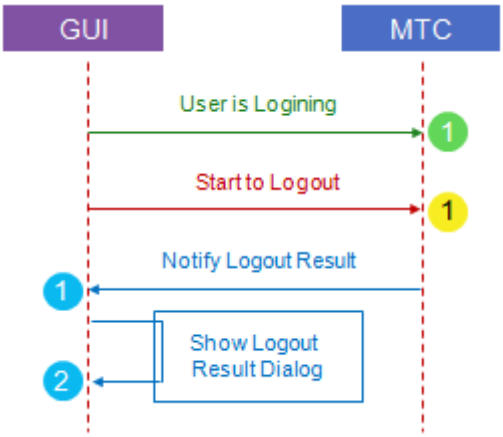

图 5-8 取消登陆过程

1	客户端已经向服务器发起注册，处于正在登陆中。
1	使用 Mtc_Cli.Mtc_CliLogout 发起取消请求。 如果是本地登陆模式，可以立即准备登陆准备窗口。
1	在客户端初始化的时候设置好 MtcCliCb.mtcCliCbLclLogout， MtcCliCb.mtcCliCbServLogout GUI 回调，取消登陆结果将会由这两个回调发起通知。
2	在 MtcCliCb.mtcCliCbLclLogout 和 MtcCliCb.mtcCliCbServLogout 回调中处理取消登陆结果，显示对应的 GUI 窗口。 如果是本地登陆模式，在 Mtc_Cli.Mtc_CliLogout 函数里面会立即调用 MtcCliCb.mtcCliCbLclLogout 通知 GUI 回调， 如果是服务器模式，除非本地资源不足或发送失败(MtcCliCb.mtcCliCbLclLogout)，否则取消登陆结果会异步通知(MtcCliCb.mtcCliCbServLogout)，Mtc_Cli.Mtc_CliLogout 会立即返回。

表 5-12 取消登陆过程说明

5.3.4 发起注销过程

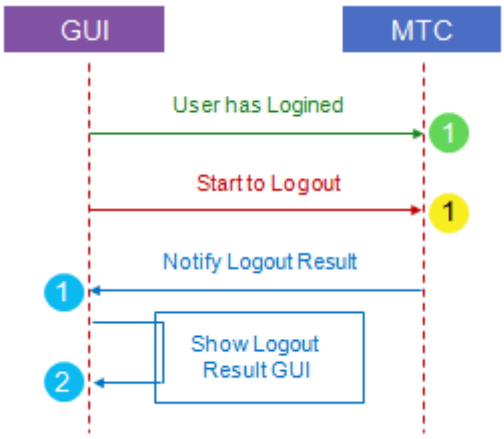

图 5-9 发起注销过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。
1	使用 Mtc_Cli.Mtc_CliLogout 发起注销请求。
!	如果是本地登陆模式，可以立即准备登陆准备窗口。 pfnShowLclLogou 会在 Mtc_Cli.Mtc_CliLogout 中被调用。 建议在 Mtc_Cli.Mtc_CliLogout 返回后，调用 Mtc_Cli.Mtc_CliStop 和 Mtc_Cli.Mtc_CliClose 关闭当前账号，然后重新选择账号并显示 GUI。
1	在客户端初始化的时候设置好 MtcCliCb.mtcCliCbLclLogout， MtcCliCb.mtcCliCbServLogout GUI 回调，取消登陆结果将会由这两个回调发起通知。
2	在 MtcCliCb.mtcCliCbLclLogout 和 MtcCliCb.mtcCliCbServLogout 回调中处理取消登陆结果，显示对应的 GUI 窗口。
!	如果是本地登陆模式，在 Mtc_Cli.Mtc_CliLogout 函数里面会立即调用 MtcCliCb.mtcCliCbLclLogout 通知 GUI 回调， 如果是服务器模式，除非本地资源不足或发送失败(MtcCliCb.mtcCliCbLclLogout)，否则取消登陆结果会异步通知(MtcCliCb.mtcCliCbServLogout)，Mtc_Cli.Mtc_CliLogout 会立即返回。

表 5-13 发起注销过程说明

5.3.5 被动注销过程

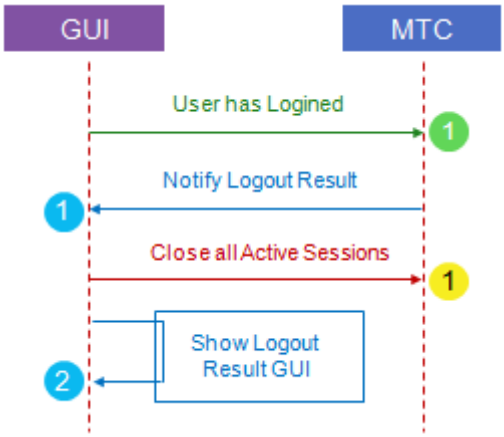

图 5-10 被动注销过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。客户端必须要打开订阅注册状态的选项，接口为 Mtc_DbSetSubsReg。此后，在客户端被服务器注销的时候才会得到通知。
1	在客户端初始化的时候设置好 MtcCliCb.mtcCliCbServLogout GUI 回调，被动注销将会由此回调发起通知。在此回调中，会携带被注销的事件和具体参数。 事件包括：

类型	说明	GUI 处理
MtcCliConstants .MTC_CLI_REG_ ERR_TIMEOUT	注册已经超时	MtcCliCb.mtcCliCbServLogout 回调中显示未登录状态
MtcCliConstants .MTC_CLI.MTC_CLI_ REG_ERR_DEACTED	注册信息无效	MtcCliCb.mtcCliCbServLogout 回调中显示未登录状态
MtcCliConstants .MTC_CLI.MTC_CLI_ REG_ERR_PROBATION	注册被拒绝，需要间隔一段时间之后再发起注册	MtcCliCb.mtcCliCbServLogout 回调中显示未登录状态，并根据超时时间启动注册倒计时窗口，计时结束后，重新发起登录过程

1	在 MtcCliCb.mtcCliCbServLogout 回调中，使用 MtcCall.Mtc_SessTerm 等接口关闭处于活动状态的音视频、短消息等会话。
2	在 MtcCliCb.mtcCliCbServLogout 回调中处理显示对应的 GUI 窗口。

表 5-14 被动注销过程说明

6. 会话管理

MTC 会话管理支持以下业务：

- 语音呼叫
- 多路呼叫
- 呼叫保持

- 呼叫转移
- 呼叫前转
- 呼入阻止
- 媒体能力更新
- 发送 DTMF
- 主叫号码显示 Caller-ID
- 设置静音
- 设置音量

6.1 接口与回调

用户实现会话操作的接口有：

接口名称	接口描述	GUI 回调存在
MtcCall.Mtc_SessCall	发起会话呼叫	存在
MtcCall.Mtc_SessAnswer	应答会话呼入	存在
MtcCall.Mtc_SessTerm	终止会话	存在
MtcCall.Mtc_SessHold	保持会话	存在
MtcCall.Mtc_SessUnhold	取消保持会话	存在
MtcCall.Mtc_SessUTrsf	发起非协商会话转移	存在
MtcCall.Mtc_SessATrsf	发起协商会话转移	存在
MtcCall.Mtc_SessDiv	对呼入的会话进行呼叫前转	存在
MtcCall.Mtc_SessDtmf	发送 DTMF 信息	
MtcCall.Mtc_SessHasTalk	判断当前是否是通话状态	
MtcCall.Mtc_SessHasHold	判断当前是否处于呼叫保持状态	
MtcCall.Mtc_SessHasHeld	判断当前是否处于呼叫被保持状态	
MtcCall.Mtc_SessAudioLostRatio	获取音频丢包率	
MtcCall.Mtc_SessGetMicMute	获取麦克风静音状态	
MtcCall.Mtc_SessSetMicMute	设置麦克风静音状态	
MtcCall.Mtc_SessGetSpkMute	获取扬声器静音状态	
MtcCall.Mtc_SessSetSpkMute	设置扬声器静音状态	
MtcCall.Mtc_SessSetMixVoice	将会话加入本地混音	
MtcCall.Mtc_SessGetState	获取会话状态	
MtcCall.Mtc_SessGetPeerUri	获取会话对方的 URI，从 From 获取	

MtcCall.Mtc_SessGetPeerId	获取会话对方的 Caller-ID，根据本地设置从 PAI 和 From 获取	
MtcCall.Mtc_SessPeerIsFocus	判断对方是否是会议主持方	
MtcCall.Mtc_SessHasOfferAnswer	判断会话是否完成了媒体协商	
MtcCall.Mtc_SessGetEarlyMediaStatus	获取早期媒体的状态	

表 6-1 会话用户接口

会话接口相关的 GUI 回调处理：

用户操作	**操作详细描述**	
MtcCall.Mtc_SessCall	MtcCallCb.mtcCallCbOutgoing	会话正在呼出中
	MtcCallCb.mtcCallCbAlerted	对方已经收到呼叫，处于振铃中
	MtcCallCb.mtcCallCbTalking	对方接受了呼叫请求，进入通话状态
	MtcCallCb.mtcCallCbTermed	对方拒绝或终止了呼叫请求
	MtcCallCb.mtcCallCbError	会话发生错误
*	MtcCallCb.mtcCallCbIncoming	收到远端的一个呼叫请求
MtcCall.Mtc_SessAnswer	MtcCallCb.mtcCallCbTalking	接受远端的呼叫请求，进入通话状态
*	MtcCallCb.mtcCallCbMdfyed	本方的媒体属性被修改
*	MtcCallCb.mtcCallCbRefered	当前会话被对方转移
*	MtcCallCb.mtcCallCbRedirect	当前会话被对方前转
*	MtcCallCb.mtcCallCbRpled	当前会话被另一个会话代替
*	MtcCallCb.mtcCallCbError	当前会话处理发生错误
MtcCall.Mtc_SessHold	MtcCallCb.mtcCallCbMdfyAcpt	对方接受修改媒体属性的请求

	MtcCallCb.mtcCallCbError	呼叫保持拒绝或处理错误
MtcCall.Mtc_SessUnhold	MtcCallCb.mtcCallCbMdfyAcpt	对方接受修改媒体属性的请求
	MtcCallCb.mtcCallCbError	呼叫保持解除拒绝或处理错误
MtcCall.Mtc_SessUpdate	MtcCallCb.mtcCallCbMdfyAcpt	对方接受修改媒体属性的请求
	MtcCallCb.mtcCallCbError	媒体能力更新拒绝或处理错误
MtcCall.Mtc_SessUTrsf	MtcCallCb.mtcCallCbTrsfAcpt	对方接受呼叫转移请求
	MtcCallCb.mtcCallCbTrsfTerm	呼叫转移终止
MtcCall.Mtc_SessATrsf	MtcCallCb.mtcCallCbTrsfAcpt	对方接受呼叫转移请求
	MtcCallCb.mtcCallCbTrsfTerm	呼叫转移终止
	MtcCallCb.mtcCallCbError	取消视频通话能力拒绝或处理错误

表 6-2 会话 GUI 回调

6.2 语音呼叫

6.2.1 发起呼叫过程

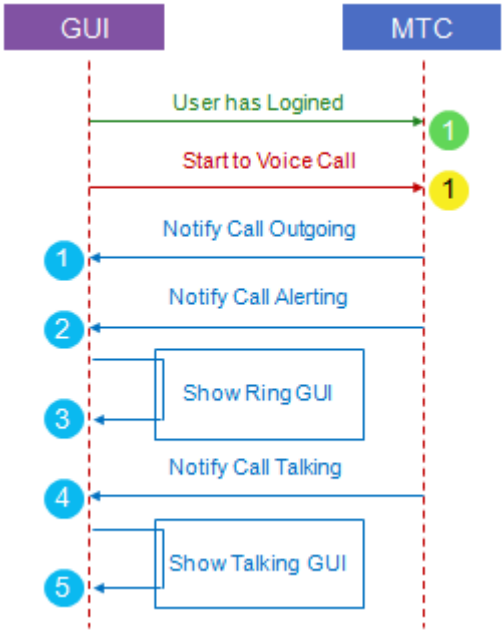

图 6-1 发起呼叫过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。
1	GUI 调用 MtcCall.Mtc_SessCall 发起呼叫。 pcUri 要确保是正确的 SIP URI 或 TEL URI。当用户在拨叫窗口中输入对方电话号码是，要先完成 SIP URI/TEL URI 的组装。组装方法有两种： a. 本地登陆模式组装 如果是有效的 IP 地址，建议组装为 sip:unknown@ip_address 如果是有效的域名，建议组装为 sip:unknown@domain b. 服务器登陆模式，直接封装为 sip:phone@domain 或者 tel:phone dwCookie 是用于实现会话与 GUI 会话窗口的绑定(比如句柄),此值可用可不用。如果想使用 dwCookie 映射会话， GUI 可在后续会话状态通知中使用 MtcCall.Mtc_SessGetCookie 获取 dwCookie。无论是否使用 dwCookie，GUI 应该提供根据 dwSessId 找到 GUI 会话窗口的方法，使得收到会话状态通知 (比如 MtcCallCb.mtcCallCbAlerted) 时能够找到已经存在的 GUI 会话窗口，更新界面显示。 输出参数 pdwSessId，一般是保存在 GUI 会话窗口中，用户后续会话操作。
1	MtcCallCb.mtcCallCbOutgoing 表明会话请求已经下发，等待会话响应 一般情况下，GUI 会话窗口可忽略此状态，但是如果会话窗口中的呼出状态由 MtcCallCb.mtcCallCbOutgoing 驱动的话，可以在回调中找到会话窗口，然后显示呼出状态 (如 Calling)

2	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃或者网络侧播放铃音（包括多媒体彩铃）
3	GUI 会话窗口更新会话状态（如 Ringing）。 GUI 可以通过 MtcCall.Mtc_SessHasEarlyMedia 接口判断是否存在早期媒体，如果存在表明网络侧会播放铃音（彩铃）。
4	MtcCallCb.mtcCallCbTalking 回调表明对方已经接听呼叫，语音流会双向接通。
5	GUI 会话窗口更新会话状态（如 Talking）。

表 6-3 发起呼叫过程说明

6.2.2 接听呼叫过程

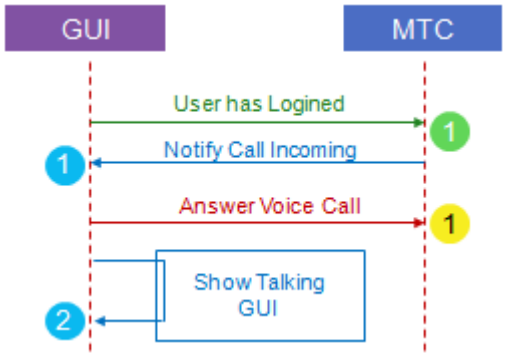

图 6-2 接听呼叫过程

1	客户端已经在服务器中成功注册，正处于登陆成功状态。
1	MtcCallCb.mtcCallCbIncoming 表明有会话邀请，等待用户接听 GUI 在此回调中可以创建新的 GUI 会话窗口，显示会话状态（如 Call Incoming）。如果是语音呼叫，则 MtcCall.Mtc_SessHasVideo 会返回 ZFALSE
1 !	GUI 调用 MtcCall.Mtc_SessAnswer 接受会话邀请。 GUI 可以使用 MtcCall.Mtc_SessGetPeerId 获取主叫号码显示（Caller-ID），同时此函数包括了对显示限制的处理。MtcCall.Mtc_SessGetPeerUri 用于获取远端 URI （From UI）。用户可以根据此 2 种信息查找对应的联系人是否存在，如果联系人存在号码，则还需要分析 URI 中的号码。 dwCookie 是用于实现会话与 GUI 会话窗口的绑定(比如句柄),此值可用可不用。如果想使用 dwCookie 映射会话， GUI 可在后续会话状态通知中使用 MtcCall.Mtc_SessGetCookie 获取 dwCookie。无论是否使用 dwCookie，GUI 应该提供根据 dwSessId 找到 GUI 会话窗口的方法，使得收到会话状态通知（比如 MtcCallCb.mtcCallCbTermed）时能够找到已经存在的 GUI 会话窗口，更新界面显示。
2	GUI 会话窗口更新会话状态（如 Talking）。

表 6-4 接听呼叫过程说明

6.2.3 取消呼叫过程

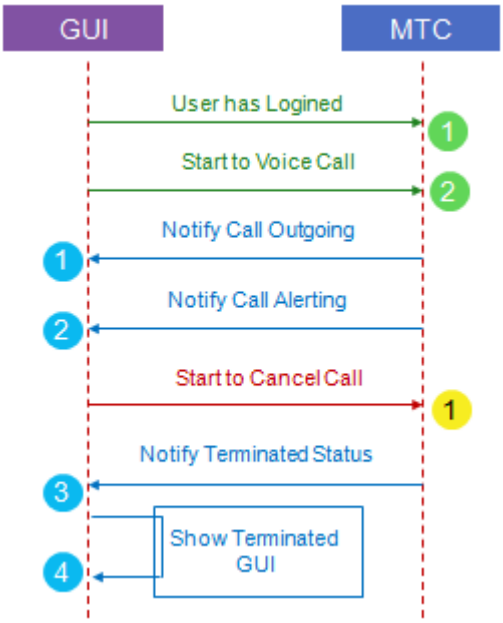

图 6-3 取消呼叫过程

1	客户端已经在服务器中成功注册，正处于登陆成功状态。
2	GUI 调用 MtcCall.Mtc_SessCall 发起呼叫。
1	MtcCallCb.mtcCallCbOutgoing 表明会话请求已经下发，等待会话响应
2	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃或者网络侧播放铃音（包括多媒体彩铃）
1	GUI 调用 MtcCall.Mtc_SessTerm 取消呼叫，dwReason 是拒绝的原因，设置 MtcCliConstants.EN_MTC_CALL_TERM_REASON_NORMAL 与被叫拒绝呼入请求不同，主叫在取消发起的呼叫时并没有取消原因这一说法，dwReason 还要填值只是出于接口一致的需要，事实上填成任何值都没问题
3	MtcCallCb.mtcCallCbTermed 表明有会话已经被终止
4	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。

表 6-5 取消呼叫过程说明

6.2.4 拒绝呼叫过程

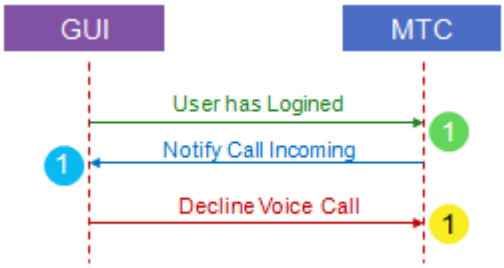

图 6-4 拒绝呼叫过程

1	客户端已经在服务器中成功注册，正处于登陆成功状态。
1	MtcCallCb.mtcCallCbIncoming 表明有会话邀请，等待用户拒绝
1	GUI 调用 MtcCall.Mtc_SessTerm 拒绝会话邀请。dwReason 是拒绝的原因，可以是 MtcCliConstants.EN_MTC_CALL_TERM_REASON_BUSY 或 MtcCliConstants.EN_MTC_CALL_TERM_REASON_NORMAL。 MtcCliConstants.EN_MTC_CALL_TERM_REASON_BUSY 的意思是被叫正忙 ,服务器在收到此响应时可能会触发相应业务，如 CFB。此函数会立即返回。

表 6-6 拒绝呼叫过程说明

6.2.5 终止呼叫过程

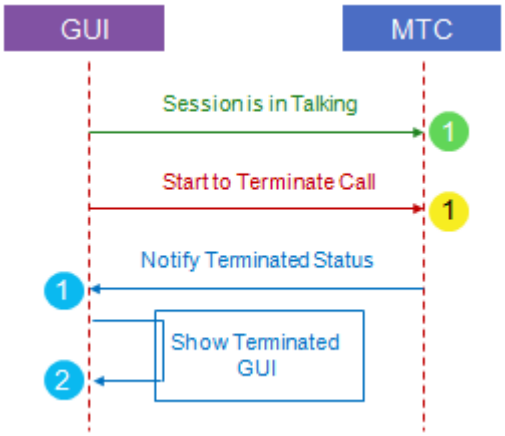

图 6-5 主动终止会话过程

1	客户端已经在服务器中成功注册，且处于通话中状态。
1	GUI 调用 MtcCall.Mtc_SessTerm 终止会话。dwReason 设置为 MtcCliConstants.EN_MTC_CALL_TERM_REASON_NORMAL 。此函数会立即返回。
1	MtcCallCb.mtcCallCbTermed 表明有会话已经被终止

2	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。

表 6-7 主动终止会话过程说明

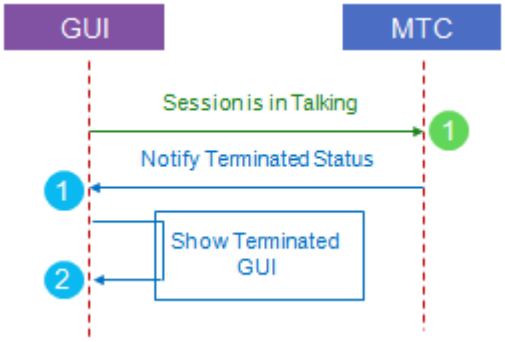

图 6-6 被动终止会话过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。
1	MtcCallCb.mtcCallCbTermed 表明有会话已经被对方终止，其中 dwReason 可以是以下值： MtcCallConstants .MTC_CALL_TERM_BYE 通话正常结束，收到 BYE 消息 MtcCallConstants .MTC_CALL_TERM_CANCEL 呼叫建立之前取消了呼叫，收到 CANCEL 消息 MtcCallConstants .MTC_CALL_TERM_TIMEOUT 呼叫超时 MtcCallConstants .MTC_CALL_TERM_BUSY 对方正忙，收到 486 消息 MtcCallConstants .MTC_CALL_TERM_DECLINE 呼叫被拒绝，收到 603 消息 GUI 可以不使用 MtcCall.Mtc_SessTerm 关闭会话，因此会话已经在回调后会被 MTC 关闭。但是继续调用对会话状态不会有影响。
2	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。

表 6-8 被动终止会话过程说明

6.2.6 主叫号码显示

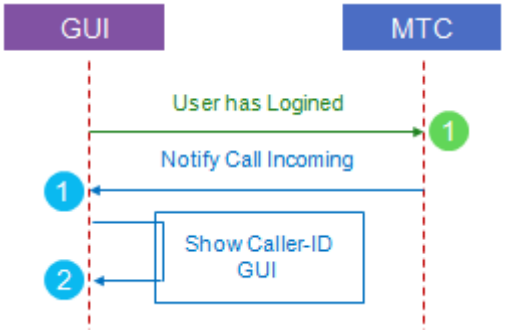

图 6-7 主叫号码显示过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。

1	MtcCallCb.mtcCallCbIncoming 表明有会话邀请，等待用户接听。
⚠	GUI 可以使用 MtcCall.Mtc_SessGetPeerId 获取主叫号码显示（Caller-ID），同时此函数包括了对显示限制的处理。MtcCall.Mtc_SessGetPeerUri 用于获取远端 URI （From UI）。用户可以根据此 2 种信息查找对应的联系人是否存在，如果联系人存在号码，则还需要分析 URI 中的号码。
2	GUI 会话窗口更新会话状态（如 Caller-ID）。

表 6-9 主叫号码显示说明

6.2.7 发送 DTMF 过程

DTMF (双音多频)是与呼叫服务器交互的语音频率的电话信号。可以选择 Auto 和 SIP Info 两种承载方式。

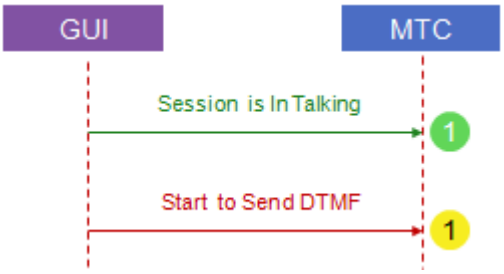

图 6-8 发送 DTMF 过程

1	客户端已经在服务器中成功注册，登陆成功，正处于通话状态。
1	GUI 调用 MtcCall.Mtc_SessDtmf 发送 DTMF 信号， 调用 MtcRing.Mtc_RingPlay 播放 DTMF 声音。

表 6-10 发送 DTMF 过程说明

6.3 视频呼叫

6.3.1 发起呼叫过程

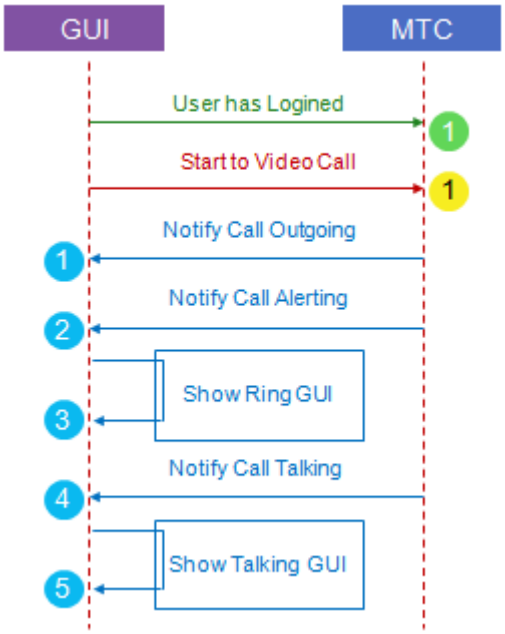

图 6-9 发起视频呼叫过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。
1	GUI 调用 MtcCall.Mtc_SessCall 发起呼叫。设置输入参数中 bVoice 为 true，bVideo 为 true 除非本地资源不足或发送失败，否则此函数会立即返回。
1	回调（通过 MtcCallCb.Mtc_CallCbSetOutgoing 设置） 表明会话请求已经下发，等待会话响应
2	回调（MtcCallCb.Mtc_CallCbSetAlerted）表明对方已经振铃或者网络侧播放铃音（包括多媒体彩铃）
3	GUI 会话窗口更新会话状态（如 Ringing）。
4	回调（MtcCallCb.Mtc_CallCbSetTalking）表明对方已经接听呼叫，视频流会双向接通。
5	GUI 会话窗口更新会话状态（如 Talking），视频显示窗口会显示视频图像。

表 6-11 发起视频呼叫过程说明

6.3.2 接听呼叫过程

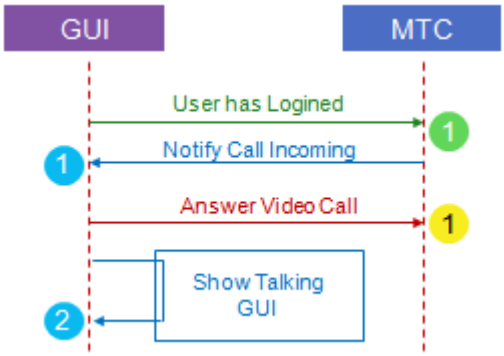

图 6-10 接听视频呼叫过程

1	客户端已经在服务器中成功注册，处于登陆成功状态。
1	回调（通过 MtcCallCb.Mtc_CallCbSetIncoming 设置）表明有会话邀请，等待用户接听
!	GUI 在此回调中可以创建新的 GUI 会话窗口，显示会话状态（如 Call Incoming）。如果是视频呼叫，则 MtcCall.Mtc_SessHasVideo 会返回 true
1	GUI 调用 MtcCall.Mtc_SessAnswer 接受会话邀请。设置输入参数中 bVoice 为 true，bVideo 为 true。除非本地资源不足或发送失败，否则此函数会立即返回。
2	GUI 会话窗口更新会话状态（如 Talking），使用 MtcCall.Mtc_SessStartVideo 启动本地显示图像，视频视频图像显示窗口会显示图像。

表 6-12 接听视频呼叫过程说明

6.3.3 取消呼叫过程

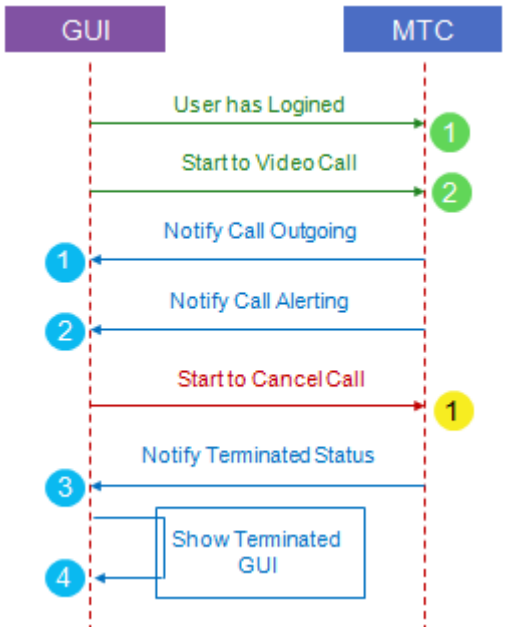

图 6-11 取消呼叫过程

1	客户端已经在服务器中成功注册，正处于登陆成功状态。
2	GUI 调用 MtcCall.Mtc_SessCall 发起呼叫。
1	回调（MtcCallCb.Mtc_CallCbSetOutgoing） 表明会话请求已经下发，等待会话响应
2	回调（MtcCallCb.Mtc_CallCbSetAlerted）表明对方已经振铃或者网络侧播放铃音（包括多媒体彩铃）
1	GUI 调用 MtcCall.Mtc_SessTerm 取消呼叫，dwReason 是拒绝的原因，填 MtcCallConstants.EN_MTC_CALL_TERM_REASON_BUSY ⚠ 与被叫拒绝呼入请求不同，主叫在取消发起的呼叫时并没有取消原因这一说法，dwReason 还要填值只是出于接口一致的需要，事实上填成任何值都没问题
3	回调（MtcCallCb.Mtc_CallCbSetTermed）表明有会话已经被终止
4	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。

表 6-13 取消呼叫过程说明

6.3.4 拒绝呼叫过程

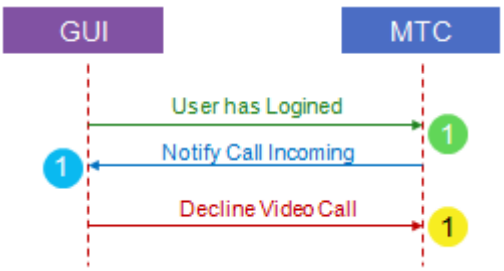

图 6-12 拒绝视频呼叫过程

1	客户端已经在服务器中成功注册，正处于登陆成功状态。
1	回调（通过 MtcCallCb.Mtc_CallCbSetIncoming 设置）表明有会话邀请，等待用户拒绝。通过 MtcCall.Mtc_SessHasVideo 的返回值（ZTRUE）来判断是否是视频呼叫。
1	GUI 调用 MtcCall.Mtc_SessTerm 拒绝会话邀请。dwReason 是拒绝的原因，可以是 EN_MTC_CALL_TERM_REASON_NORMAL 或 EN_MTC_CALL_TERM_REASON_BUSY 或 EN_MTC_CALL_TERM_REASON_DECLINE。EN_MTC_CALL_TERM_REASON_BUSY 的意思是被叫正忙，服务器在收到此响应时可能会触发相应业务，如 CFB。此函数会立即返回。

表 6-14 拒绝视频呼叫过程说明

6.3.5 终止呼叫过程

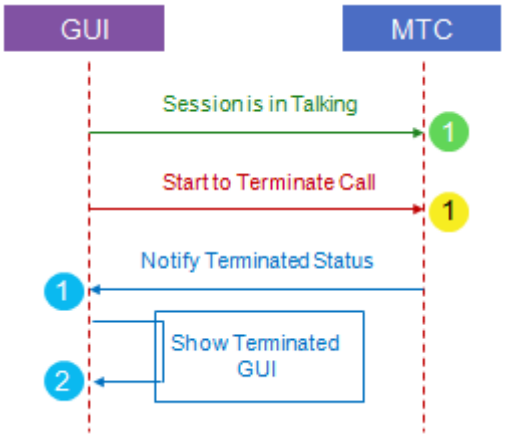

图 6-13 主动终止视频会话过程

1	客户端已经在服务器中成功注册，且处于通话中状态。
1	GUI 调用 MtcCall.Mtc_SessTerm 终止会话。dwReason 设置为 MtcCallConstants.EN_MTC_CALL_TERM_REASON_NORMAL 。此函数会立即返回。用户可以使用 MtcCall.Mtc_SessStopVideo 可以停止视频图像显示，甚至关闭视频图像显示窗口。
1	回调（通过 MtcCallCb.Mtc_CallCbSetTermed 设置） 表明有会话已经被终止
2	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。

表 6-15 主动终止视频会话过程说明

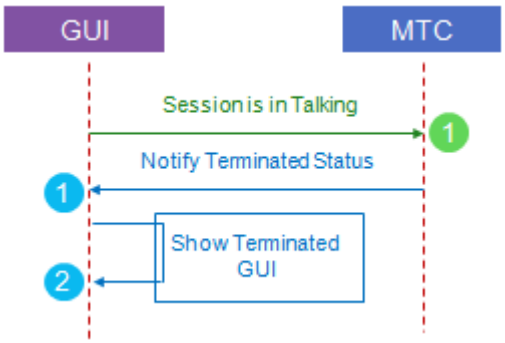

图 6-14 被动终止视频会话过程

1	客户端已经在服务器中成功注册，处于通话中状态。
1	回调（通过 MtcCallCb.Mtc_CallCbSetTermed 设置）表明有会话已经被对方终止。用户可以使用 MtcCall.Mtc_SessStopVideo 可以停止视频图像显示，甚至关闭视频图像显示窗口。其中 dwReason 可以是以下值: EN_MTC_CALL_TERM_BYE 通话正常结束，收到 BYE 消息

EN_MTC_CALL_TERM_CANCEL	呼叫建立之前取消了呼叫，收到 CANCEL 消息
EN_MTC_CALL_TERM_TIMEOUT	呼叫超时
EN_MTC_CALL_TERM_BUSY	对方正忙，收到 486 消息
EN_MTC_CALL_TERM_DECLINE	呼叫被拒绝，收到 603 消息
EN_MTC_CALL_TERM_TRSFED	呼叫被转移
EN_MTC_CALL_TERM_REDIRECT	呼叫被重定向
EN_MTC_CALL_TERM_REPLACE	会话被替换
2 GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。	

表 6-16 被动终止视频会话过程说明

6.4 呼叫前转

6.4.1 无条件前转

用户签约该业务后，前转由网络侧完成，终端不需要参与。

6.4.2 遇忙前转

用户签约该业务后，当终端已经达到最大呼叫数时，对于新呼入的呼叫终端响应 486，触发网络侧前转业务。GUI 不需要做处理。

6.4.3 无应答前转

用户签约该业务后，当用户在一定时间内对新呼入的呼叫无任何操作，则终端响应 480，触发网络侧前转业务。GUI 只需要参考接听呼叫流程显示界面，其他不需要做特殊处理。

6.4.4 不可及前转

用户签约该业务后，前转由网络侧完成，终端不需要参与。

6.4.5 呼叫改向

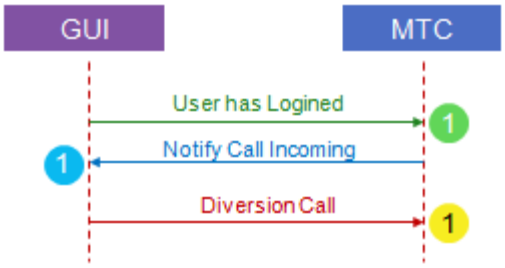

图 6-15 呼叫前转过程

1 客户端已经在服务器中成功注册，处于正处于登陆成功状态。

1	MtcCallCb.mtcCallCbIncoming 表明有会话邀请，等待用户阻止
1	GUI 调用 MtcCall.Mtc_SessDiv 把当前的呼入转移到另一个终端，此终端的 URI 由 pcUri 参数指定。此操作在服务器端会触发 CD 业务。

表 6-17 呼叫前转过程说明

6.4.6 未注册前转

用户签约该业务后，前转由网络侧完成，终端不需要参与。

6.5 呼叫等待

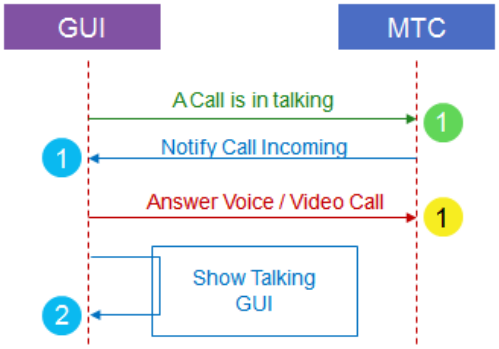

图 6-16 呼叫等待过程

1	客户端已经有一路正在通话中的呼叫。
1	MtcCallCb.mtcCallCbIncoming 表明有新的会话邀请，等待用户接听
⚠	GUI 在此回调中判断已经有一路正在通话的呼叫，则应该提示给用户一路新的呼叫正在等待应答。同时播放的提示声音应该与正常接听呼叫的声音有所区别。
1	GUI 调用 MtcCall.Mtc_SessAnswer 接受会话邀请。除非本地资源不足或发送失败，否则此函数会立即返回。
⚠	如果用户决定接听新的一路呼叫，则 GUI 应该首先将前一路呼叫结束或者保持。所以建议，在呼叫等待提示界面应有两个选项“结束当前通话，并接听”和“保持当前通话，并接听”。
2	GUI 会话窗口更新会话状态（如 Talking）。

表 6-18 呼叫等待过程说明

6.6 呼叫保持

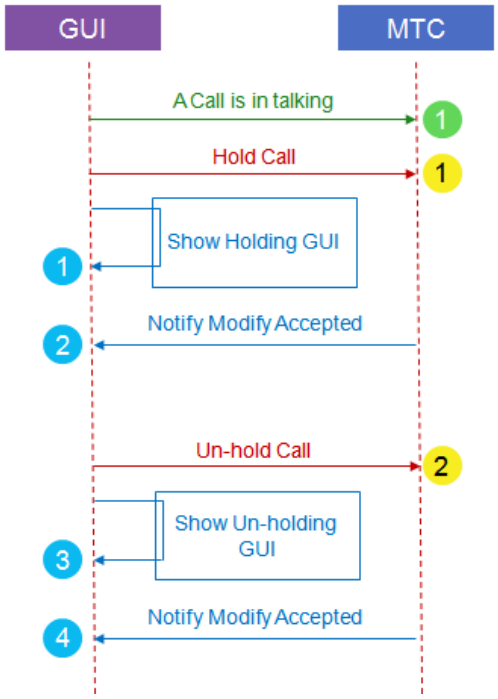

图 6-17 呼叫保持过程

1	客户端已经有一路正在通话中的呼叫。
1	GUI 调用 MtcCall.Mtc_SessHold 保持呼叫。
1	呼叫保持需要信令交互，不能立刻完成。GUI 应该显示正在保持呼叫，并限制用户呼叫保持成功之前，不能解除呼叫保持。
2	MtcCallCb.mtcCallCbHoldOk 指示呼叫保持操作成功，此时 GUI 应该将呼叫标记不同颜色或其他特征以区别正常呼叫。
2	GUI 调用 MtcCall.Mtc_SessUnhold 解除保持呼叫。
3	呼叫保持需要信令交互，不能立刻完成。GUI 应该显示正在解除保持呼叫，并限制用户在解除呼叫保持之前，不能呼叫保持。
4	MtcCallCb.mtcCallCbUnHoldOk 指示解除呼叫保持操作成功，此时 GUI 应该将呼叫标记成为正常呼叫。

表 6-19 呼叫保持过程说明

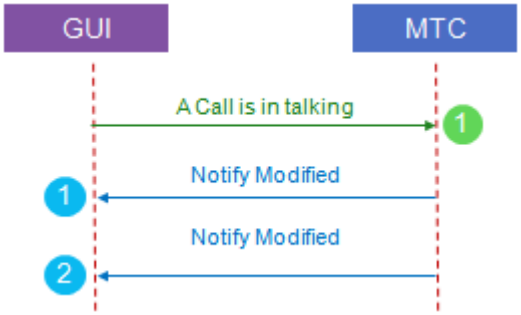

图 6-18 呼叫被保持过程

1	客户端已经有一路正在通话中的呼叫。
1	MtcCallCb.mtcCallCbHeld 指示呼叫被对方保持，GUI 应该将呼叫标记不同颜色或其他特征以区别正常呼叫。
2	MtcCallCb.mtcCallCbUnHeld 指示对方已经解除呼叫保持，此时 GUI 应该将呼叫标记成为正常呼叫。

表 6-20 呼叫被保持过程说明

6.7 呼叫限制

6.7.1 呼入阻止

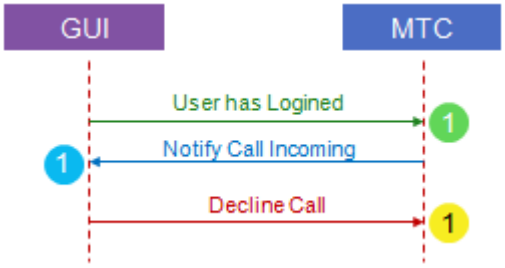

图 6-19 呼入阻止过程

1	客户端已经在服务器中成功注册，处于正处于登陆成功状态。
1	MtcCallCb.mtcCallCbIncoming 表明有会话邀请，等待用户阻止
1	GUI 调用 MtcCall.Mtc_SessTerm 拒绝会话邀请。dwReason 填成 MtcCliConstants.EN_MTC_CALL_TERM_REASON_DECLINE。MtcCliConstants.EN_MTC_CALL_TERM_REASON_DECLINE 的意思是被叫动态阻止了当前的呼入，服务器在收到此响应时会触发 CB 业务。

表 6-21 呼入阻止过程说明

6.8 呼叫转移

6.8.1 盲转

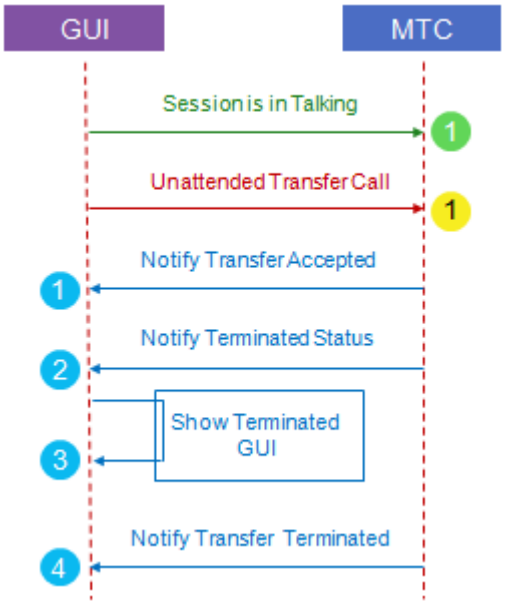

图 6-20 盲转过程

1	客户端和远端用户 1 处于通话中。
1	GUI 调用 MtcCall.Mtc_SessUTrsf 转移会话，pcUri 参数指明了转移的目的地，如用户 2 的 URI。转移业务如果成功执行，用户 1 将和用户 2 建立通话。
1	MtcCallCb.mtcCallCbTrsfAcpt 表明呼叫转移业务已被服务器接受
!	被服务器接受并不表示转移已经成功，接受的意思是服务器通过检查策略接受了呼叫转移的业务请求。
2	MtcCallCb.mtcCallCbTermed 表明会话已经被终止
3	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。
4	MtcCallCb.mtcCallCbTrsfTerm 表明呼叫转移业务已经结束
!	由于网络的原因，MtcCallCb.mtcCallCbTrsfTerm 可能发生在 MtcCallCb.mtcCallCbTermed 之前，GUI 要考虑到这种可能性。

表 6-22 盲转过程说明

6.8.2 咨询转

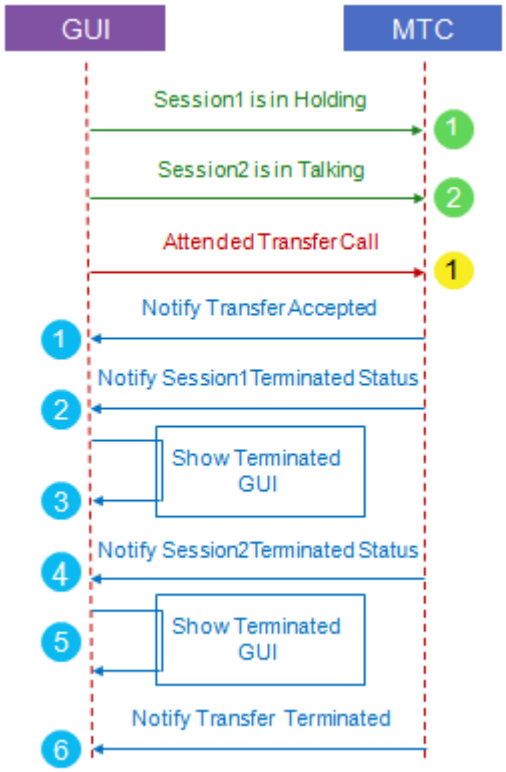

图 6-21 咨询转过程

1	客户端和用户 1 成功建立了会话，此会话处于保持中。
2	客户端和用户 2 成功建立了会话，此会话处于通话中。
1	GUI 调用 MtcCall.Mtc_SessATrsf 转移会话，dwTrsfSessId 参数指明了转移的目的地。当 dwTrsfSessId 是客户端和用户 1 建立的会话 ID 时，表示客户端想把和用户 2 的会话转移成用户 2 和用户 1 之间的会话。转移业务成功完成时，客户端和用户 1 和用户 2 的所有会话都会终止，用户 2 和用户 1 处于通话中。
1	MtcCallCb.mtcCallCbTrsfAcpt 表明呼叫转移业务已被服务器接受
!	被服务器接受并不表示转移已经成功，接受的意思是服务器通过检查策略接受了呼叫转移的业务请求。
2	MtcCallCb.mtcCallCbTermed 表明和用户 1 的会话已经被终止
3	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。
4	MtcCallCb.mtcCallCbTermed 表明和用户 2 的会话已经被终止
5	GUI 会话窗口更新会话状态（如 Ended），甚至关闭 GUI 会话窗口。
6	MtcCallCb.mtcCallCbTrsfTerm 表明呼叫转移业务已经结束
!	由于网络的原因，MtcCallCb.mtcCallCbTrsfTerm 和两个 MtcCallCb.mtcCallCbTermed 之间的先后顺序有可能会不一样，GUI 要考虑到这种可能性。

表 6-23 咨询转过程说明

6.9 铃音播放

6.9.1 本地铃音

本地铃音主要分为两类：按键铃音和事件铃音。

<table>
<tr><th>类型</th><th>描述</th><th>接口</th></tr>
<tr><td>按键铃音</td><td>用户按下拨号盘上的数字和星、井按键时播放的声音，其类型定义在 MtcCliConstants.EN_MTC_TONE_TYPE。
铃音播放应该在 GUI 界面对应拨号盘对应按键处理中调用。</td><td>MtcRing.Mtc_RingPlay</td></tr>
<tr><td>事件铃音</td><td>发生某一事件时播放的声音，现在提供的声音类型定义在 MtcCliConstants.EN_MTC_RING_TYPE。此部分铃音播放是在 MTC 内部处理直接调用。如果需要 GUI 处理，可以修改 MTC 内部实现。具体类型包括
<table>
<tr><th>类型</th><th>说明</th></tr>
<tr><td>MtcCliConstants. EN_MTC_RING_RING</td><td>收到语音呼叫</td></tr>
<tr><td>MtcCliConstants. EN_MTC_RING_RING_BACK</td><td>发起呼叫</td></tr>
<tr><td>MtcCliConstants. EN_MTC_RING_TERM</td><td>呼叫结束</td></tr>
<tr><td>MtcCliConstants. EN_MTC_RING_CALL_FAILED</td><td>呼叫失败</td></tr>
</table></td><td>MtcRing.Mtc_RingPlay</td></tr>
</table>

6.9.2 早期媒体

早期媒体是指在建立通话之前，呼叫发起方会收到来自网络侧的媒体数据，一般为彩铃或者多媒体彩铃。

- 彩铃：只有声音媒体
- 多媒体彩铃：包括声音和视频媒体

6.9.2.1 彩铃

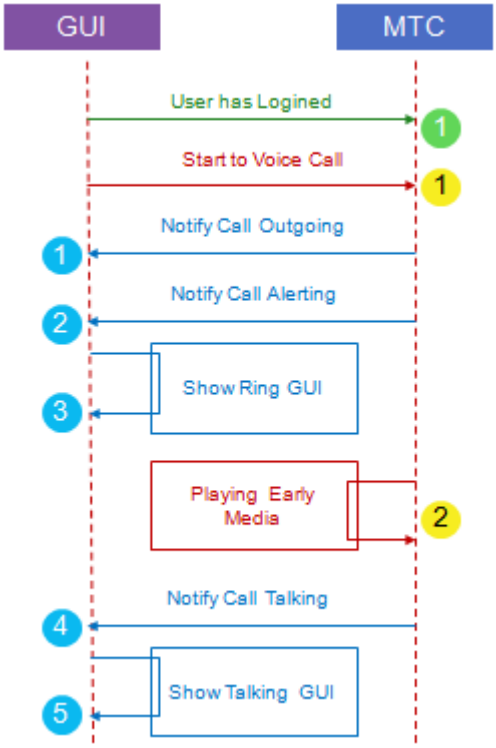

图 6-22 彩铃过程

彩铃业务的过程与一般发起语音呼叫的过程基本一致，差别在于收到 18x 响应中，是否带有早期媒体的 SDP 协商参数。如果没有，则 MTC 则会播放本地铃音。如果有，则会播放从网络侧收到的早期媒体数据。此业务实现，不需要 GUI 做特殊处理。具体过程如下：

1	客户端已经在服务器中成功注册，处于登陆成功状态。
1	GUI 调用 MtcCall.Mtc_SessCall 发起语音呼叫。除非本地资源不足或发送失败，否则此函数会立即返回。 pcUri 要确保是正确的 SIP URI 或 TEL URI。当用户在拨叫窗口中输入对方电话号码是，要先完成 SIP URI/TEL URI 的组装。组装方法有两种： a. 本地登陆模式组装 如果是有效的 IP 地址，建议组装为 sip:unknown@ip_address 如果是有效的域名，建议组装为 sip:unknown@domain b. 服务器登陆模式，直接封装为 sip:phone@domain 或者 tel:phone dwCookie 是用于实现会话与 GUI 会话窗口的绑定(比如句柄) ,此值可用可不用。如果想使用 dwCookie 映射会话， GUI 可在后续会话状态通知中使用 MtcCall.Mtc_SessGetCookie 获取 dwCookie。无论是否使用 dwCookie，GUI 应该提供根据 dwSessId 找到 GUI 会话窗口的方法，使得收到会话状态通知 (比如 MtcCallCb.mtcCallCbAlerted) 时能够找到已经存在的 GUI 会话窗口，更新界面显示。 输出参数 pdwSessId，一般是保存在 GUI 会话窗口中，用户后续会话操作。
1	MtcCallCb.mtcCallCbOutgoing 表明会话请求已经下发，等待会话响应

!	一般情况下，GUI 会话窗口可忽略此状态，但是如果会话窗口中的呼出状态由 MtcCallCb.mtcCallCbOutgoing 驱动的话，可以在回调中找到会话窗口，然后显示呼出状体（如 Calling）
2	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃，即收到 18x 响应。
2	如果响应中携带早期媒体的 SDP 参数，则 MTC 会播放网络侧收到的早期媒体数据，即彩铃。
3	GUI 会话窗口更新会话状态（如 Ringing）。
!	GUI 可以通过 MtcCall.Mtc_SessGetEarlyMediaStatus 接口判断是否存在早期媒体，如果存在表明网络侧会播放铃音（彩铃）。
4	MtcCallCb.mtcCallCbTalking 回调表明对方已经接听呼叫，语音流会双向接通。
5	GUI 会话窗口更新会话状态（如 Talking）。

表 6-24 彩铃过程说明

6.10 会话定时器

MTC 默认支持会话定时器功能，GUI 不需要有对应处理。

6.11 拨号 URI

GUI 调用 MtcCall.Mtc_SessCall 发起语音或视频呼叫。其中 URI 的参数需要 GUI 根据终端本地策略基于 MtcUri .Mtc_UriFormat 等接口组装成为 SIP URI 或者 TEL URI。

6.11.1 SIP URI

6.11.1.1 本地登陆模式组装

如果是有效的 IP 地址 ,建议组装为 sip:unknown@ip_address ,例如 sip:unknown@192.168.0.78 ,其中 192.168.0.78 为用户输入内容。

如果是有效的域名 ,建议组装为 sip:unknown@domain ,例如 sip:unknown@sip.juphoon.com ,其中 sip.juphoon.com 为用户输入内容。

6.11.1.2 服务器登陆模式

直接封装为 sip:phone@domain，例如 sip:13486651032@sip.juphoon.com，其中 13486651032 为用户输入内容，sip.juphoon.com 为用户设置的域名。

6.11.2 TEL URI

直接封装为 tel:phone，例如 tel:13486651032，其中 13486651032 为用户输入内容。

6.11.3 模拟 POTS 终端业务定制

GUI 不需要特殊处理，只需要根据本地策略，将用户输入组装成 SIP URI 或 TEL URI，并发起呼叫即可。

6.12 会话举例

6.12.1 基本呼叫过程

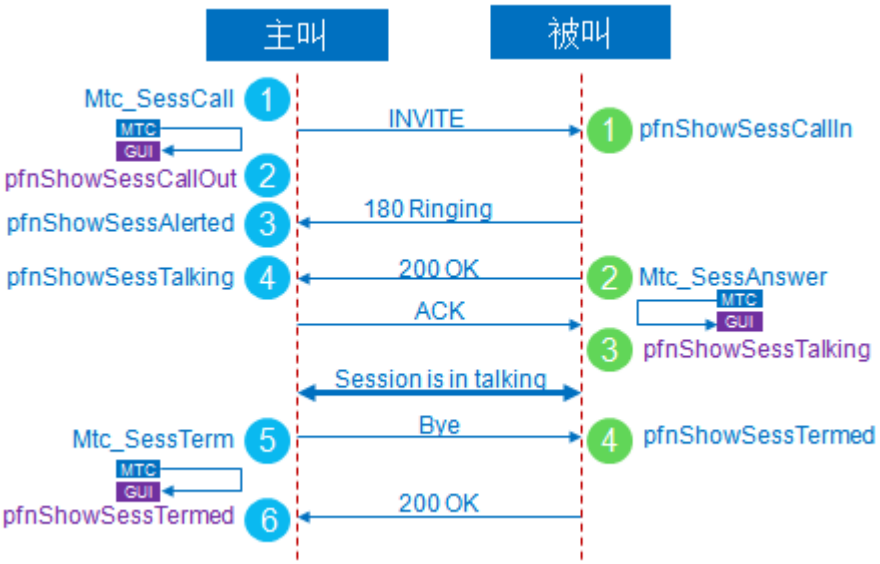

图 6-23 基本呼叫过程

1	主叫方调用 MtcCall.Mtc_SessCall 发起呼叫。输入参数中，bVoice 设置为 ZTRUE 表示加入音频，bVideo 为 ZTRUE 表示加入视频。pcUri 表示想要呼叫的对方 URI。pcUri，dwCookie，pdwSessId 参数说明见 5.2.1。
2	MtcCallCb.mtcCallCbOutgoing 回调表明发起会话请求已经下发，等待会话响应。
1	MtcCallCb.mtcCallCbIncoming 回调表明有会话邀请，等待用户接听。
3	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃。
2	被叫方调用 MtcCall.Mtc_SessAnswer 接受会话邀请。输入参数中，bVoice 设置为 ZTRUE 表示加入音频，bVideo 为 ZTRUE 表示需要加入视频。pcUri 表示想要呼叫的对方 URI。pcUri，dwCookie，pdwSessId 参数说明见 5.2.1。
3	MtcCallCb.mtcCallCbTalking 回调表明接听呼叫，语音流会双向接通。双方建立会话成功。
4	MtcCallCb.mtcCallCbTalking 回调表明对方已经接听呼叫，语音流会双向接通。双方建立会话成功。
5	主叫方调用 MtcCall.Mtc_SessTerm 终止会话。dwReason 表示终止会话的原因，在终止会话时应该设置为 MtcCliConstants.EN_MTC_CALL_TERM_REASON_NORMAL 。其他业务终止会话原因详见 5.2.5 终止原因的英文描述。
4	被叫方 MtcCallCb.mtcCallCbTermed 回调表明此会话已经被终止。
6	主叫方 MtcCallCb.mtcCallCbTermed 回调表明终结会话请求已经下发，此会话已经被终止。

表 6-25 基本呼叫过程说明

6.12.2 取消呼叫过程

图 6-24 取消呼叫过程

1	主叫方调用 MtcCall.Mtc_SessCall 发起呼叫。输入参数中，bVoice 设置为 ZTRUE 表示加入音频，bVideo 为 ZTRUE 表示需要加入视频。pcUri 表示你想要呼叫的对方 URI。pcUri，dwCookie，pdwSessId 参数说明见 5.2.1。
2	MtcCallCb.mtcCallCbOutgoing 回调表明会话请求已经下发，等待会话响应。
1	MtcCallCb.mtcCallCbIncoming 回调表明有会话邀请，等待用户接听。
3	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃或者网络侧播放铃音（包括多媒体彩铃）
4	主叫方调用 MtcCall.Mtc_SessTerm 终止会话。dwReason 设置成 MtcCliConstants.EN_MTC_CALL_TERM_REASON_NORMAL。与被叫拒绝呼入请求不同，主叫在取消发起的呼叫时并没有取消原因这一说法，dwReason 还要填值只是出于接口一致的需要，事实上填成任何值都没问题。其他业务终止会话原因详见 5.2.5 终止原因的英文描述。
5	主叫方 MtcCallCb.mtcCallCbTermed 回调表明取消会话请求已经下发，此会话被终止。
2	被叫方 MtcCallCb.mtcCallCbTermed 回调表明此会话已经被终止。

表 6-26 基本呼叫过程说明

6.12.3 拒绝呼叫过程

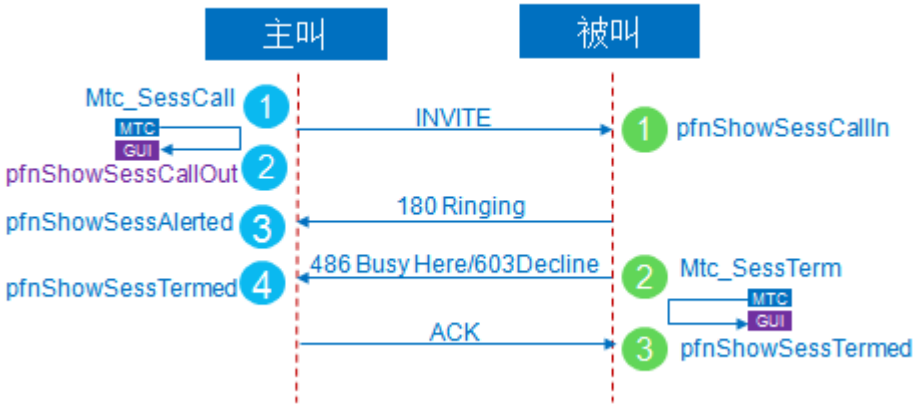

图 6-25 拒绝呼叫过程

1	主叫方调用 MtcCall.Mtc_SessCall 发起呼叫。输入参数中，bVoice 设置为 ZTRUE 表示加入音频，bVideo 为 ZTRUE 表示需要加入视频。pcUri 表示你想要呼叫的对方 URI。pcUri，dwCookie，pdwSessId 参数说明见 5.2.1。
2	MtcCallCb.mtcCallCbOutgoing 回调表明会话请求已经下发，等待会话响应。
1	MtcCallCb.mtcCallCbIncoming 回调表明有会话邀请，等待用户接听。
3	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃或者网络侧播放铃音（包括多媒体彩铃）
2	被叫方调用 MtcCall.Mtc_SessTerm 终止会话。dwReason 表示终止会话的原因，被叫主动拒绝会话时应设置成 MTC_CALL_TERM_BUSY_HERE 或者 MTC_CALL_TERM_DECLINE 。
3	被叫方 MtcCallCb.mtcCallCbTermed 回调表明此会话已经被终止
4	主叫方 MtcCallCb.mtcCallCbTermed 回调表明拒绝呼叫请求已经下发，此会话已经被终止。

表 6-27 拒绝呼叫过程说明

6.12.4 呼叫改向过程

图 6-26 呼叫改向过程

1	主叫方调用 MtcCall.Mtc_SessCall 发起呼叫。输入参数中，bVoice 设置为 ZTRUE 表示加入音频，bVideo 为 ZTRUE 表示需要加入视频。
1	被叫 MtcCallCb.mtcCallCbIncoming 回调表明有会话邀请，等待用户接听。
2	MtcCallCb.mtcCallCbAlerted 回调表明对方已经振铃。
2	被叫方 GUI 调用 MtcCall.Mtc_SessDiv 把当前的呼入转移到另一个终端 此终端的 URI 由 pcUri 参数指定。
3	MtcCallCb.mtcCallCbRedirect 回调表明会话被转向到另外一个终端。GUI 应相应调整界面显示。
3	被叫方 MtcCallCb.mtcCallCbTermed 回调表明此会话已经被终止

表 6-28 呼叫改向说明

6.12.5 呼叫保持过程

图 6-27 呼叫保持过程

1	发起保持方 GUI 调用 MtcCall.Mtc_SessHold 保持呼叫。
1	被保持方 MtcCallCb.mtcCallCbHeld 回调指示呼叫被对方保持
2	MtcCallCb.mtcCallCbHoldOk 指示呼叫保持操作成功。
3	发起保持方调用 MtcCall.Mtc_SessUnhold 解除保持呼叫。
2	MtcCallCb.mtcCallCbUnHeld 指示对方已经解除呼叫保持。
4	MtcCallCb.mtcCallCbUnHoldOk 指示解除呼叫保持操作成功。

表 6-29 呼叫保持过程说明

6.12.6 呼叫转移过程

6.12.6.1 盲转过程

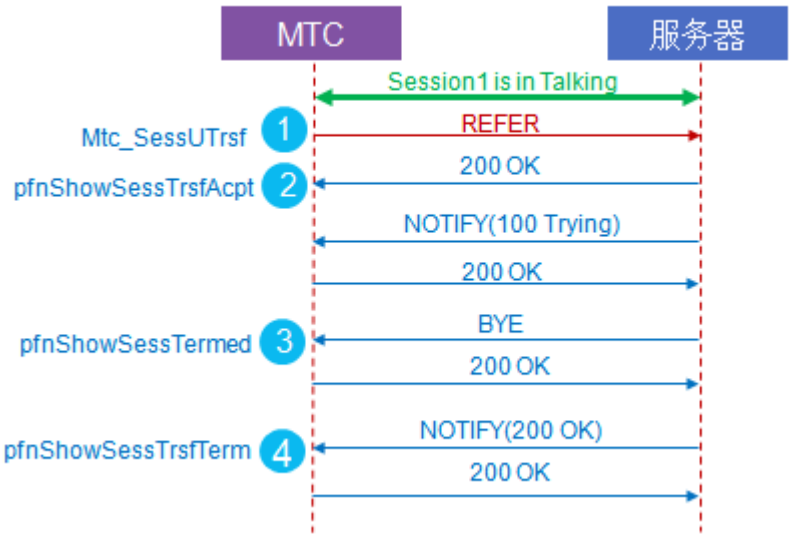

图 6-28 呼叫转移盲转过程

1	GUI 调用 MtcCall.Mtc_SessUTrsf 转移会话，pcUri 参数指明了转移的目的地，如用户 2 的 URI。
2	MtcCallCb.mtcCallCbTrsfAcpt 表明呼叫转移业务已被服务器接受。被服务器接受并不表示转移已经成功，接受的意思是服务器通过检查策略接受了呼叫转移的业务请求。
3	MtcCallCb.mtcCallCbTermed 回调表明会话已经被终止。
4	MtcCallCb.mtcCallCbTrsfTerm 表明呼叫转移业务已经结束。由于网络的原因，MtcCallCb.mtcCallCbTrsfTerm 可能发生在 MtcCallCb.mtcCallCbTermed 之前。

表 6-42 呼叫转移盲转过程说明

6.12.6.2 咨询转过程

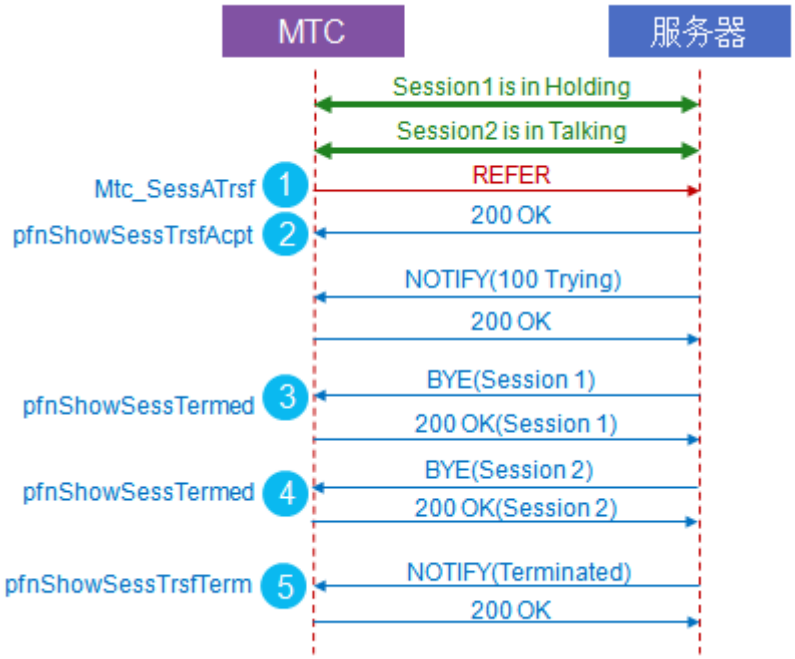

图 6-29 呼叫转移咨询转过程

1	GUI 调用 MtcCall.Mtc_SessATrsf 转移会话，dwTrsfSessId 参数指明了转移的目的地。当 dwTrsfSessId 是客户端和用户 1 建立的会话 ID 时，表示客户端想把和用户 2 的会话转移成用户 2 和用户 1 之间的会话。转移业务成功完成时，客户端和用户 1 和用户 2 的所有会话都会终止，用户 2 和用户 1 处于通话中。
2	MtcCallCb.mtcCallCbTrsfAcpt 表明呼叫转移业务已被服务器接受。被服务器接受并不表示转移已经成功，接受的意思是服务器通过检查策略接受了呼叫转移的业务请求。
3	MtcCallCb.mtcCallCbTermed 表明和用户 1 的会话已经被终止。
4	MtcCallCb.mtcCallCbTermed 表明和用户 2 的会话已经被终止。
5	MtcCallCb.mtcCallCbTrsfTerm 表明呼叫转移业务已经结束。由于网络的原因，MtcCallCb.mtcCallCbTrsfTerm 和两个 MtcCallCb.mtcCallCbTermed 之间的先后顺序有可能会不一样。

表 6-43 呼叫转移咨询转说明

7. 媒体录制存储

7.1 音视频录制存储接口

7.1.1 接口

接口名称	接口描述
MtcCall. Mtc_SessRecMicStart	录制麦克风声音。 参数说明： pcFileName：录制内容保存的文件名称。如：“c:\documetns\a.wav”。 ucFileType：录制的文件编码类型。类型定义参考：EN_MTC_MFILE_TYPE。 如：EN_MTC_MFILE_TYPE.EN_MTC_MFILE_WAV_PCMU
MtcCall. Mtc_SessRecMicStop	停止录制麦克风声音
MtcCall .Mtc_SessRecPlayStart	录制某路通话的扬声器声音。 参数说明： dwSessId： 某路通话的会话 ID 。如果 ID 为非法值，则录制混音。 pcFileName： 文件名。同上。 ucFileType：文件编码类型。同上。
MtcCall.Mtc_SessRecPlayStop	停止录制某路通话的扬声器声音。 参数说明： dwSessId: 某路通话的会话 ID 。如果 ID 为非法值，则停止录制混音。
MtcCall. Mtc_SessRecCallStart	录制整个会话的声音(包括麦克风和扬声器的声音)。 参数说明： pcFileName： 文件名。同上。 ucFileType：文件编码类型。同上。
MtcCall. Mtc_SessRecCallStop	停止录制整个会话声音。
MtcCall.Mtc_SessRecRecvVideoStart	录制收到的视频数据。 参数说明： dwSessId：录制的会话 ID。 pcFileName：录制文件文件名。同上。 ucFileType：文件类型。同上。 iWidth：视频的宽度(像素)

	iHeight：视频高度(像素) bWithAudio：是否录制音频数据
MtcCall. Mtc_SessRecRecvVideoStop	停止录制收到的视频数据。 参数说明： dwSessId：录制的会话 ID。
MtcCall.Mtc_SessRecSendVideoStart	录制发送的视频数据。 参数说明： dwSessId：录制的会话 ID。 pcFileName：录制文件文件名。同上。 ucFileType：文件类型。同上。 iWidth：视频的宽度(像素) iHeight：视频高度(像素) bWithAudio：是否录制音频数据
MtcCall.Mtc_SessRecSendVideoStop	停止录制发送的视频数据。 参数说明： dwSessId：录制的会话 ID。
MtcCall. Mtc_SessRenderSnapshot	远端视频截屏。 参数说明: dwSessId：录制的会话 ID。 pcFileName：录制文件文件名。同上。 ucFileType：文件类型。参考 EN_MTC_CALL_IMG_TYPE。如： EN_MTC_CALL_IMG_TYPE.EN_MTC_CALL_IMG_JPEG
MtcCall. Mtc_SessCaptureSnapshot	本地采集视频截屏。 参数说明: dwSessId：录制的会话 ID。 pcFileName：录制文件文件名。同上。 ucFileType：文件类型。参考 EN_MTC_CALL_IMG_TYPE。如： EN_MTC_CALL_IMG_TYPE.EN_MTC_CALL_IMG_JPEG

表 7-1 音视频录制和存储接口