

---

菊风协议构架

# Juphoon Phone Framework (JPF)

发表日期: 2006-5-22

访问 <http://www.juphoon.com> 了解更多产品信息

---

## 使用菊风 JPF 构建 SIP 电话机

宁波菊风系统软件有限公司

<http://www.juphoon.com>

电话: +86-574-87287820

传真: +86-574-87304379

文件编号: 100-000-02-01

Copyright © 2007, Juphoon System Software Corporation.

版权所有

## 目录

使用菊风JPF 构建SIP电话机 .....	1
<b>1. 框架概要 .....</b>	<b>5</b>
1.1 目的 .....	5
1.2 功能 .....	5
1.3 应用场景 .....	5
<b>2. 多业务框架 .....</b>	<b>6</b>
2.1 框架介绍 .....	6
2.2 协作组件 .....	6
2.2.1 模块分解 .....	7
2.2.2 协作路径 .....	7
2.3 用户协作 .....	7
2.4 主要概念 .....	8
2.4.1 Endpoint .....	8
2.4.2 Connection .....	9
2.4.3 Stream .....	9
2.4.4 Call .....	9
2.4.5 Call Flow .....	9
2.4.6 Session .....	10
2.4.7 Dialog .....	10
2.4.8 Transaction .....	11
2.5 线程模型 .....	11
2.5.1 User Thread .....	11
2.5.2 JPF Thread .....	11
2.5.3 SUA Thread .....	13
2.5.4 Audio Thread .....	13
2.5.5 SIP Stack Thread .....	13
<b>3. JPF 操作对象 .....</b>	<b>14</b>
3.1 操作接口 .....	14
3.2 状态报告 .....	14
3.3 应用处理 .....	19
<b>4. JPF 业务接口 .....</b>	<b>20</b>
4.1 端点操作 .....	20
4.2 连接操作 .....	20
4.3 状态通知 .....	21
4.4 数据管理 .....	22
<b>5. JPF 业务特性 .....</b>	<b>23</b>
5.1 REGISTER .....	23

---

5.2	BASIC CALL .....	25
5.3	CALLER ID .....	26
5.4	CALL HOLD.....	27
5.5	CONSULTATION HOLD .....	28
5.6	UNATTENDED TRANSFER.....	29
5.7	ATTENDED TRANSFER.....	31
5.8	CALL WAITING.....	33
5.9	MULTIPLE LINE APPEARANCE .....	34
5.10	MUTE .....	35
5.11	REDIAL.....	35
5.12	DO NOT DISTURB.....	36
<b>6.</b>	<b>JPF 协议特性.....</b>	<b>37</b>
6.1	REGISTER .....	37
6.2	UN-REGISTER.....	37
6.3	OUTBOUND PROXY .....	37
6.4	PROXY/WWW AUTHENTICATION.....	38
6.5	STUN .....	38
6.6	DTMF .....	38

## 表格列表

表 2-1 SIP Phone 业务组件协作路径说明.....	7
表 4-1 端点操作接口 .....	20
表 4-2 连接操作接口 .....	21
表 4-3 数据管理接口 .....	22
表 5-1 Register 用户环境.....	25
表 5-2 Basic Call 用户环境.....	26
表 5-3 Caller ID 用户环境 .....	27
表 5-4 Call Hold 用户环境.....	28
表 5-5 Consultation Hold用户环境 .....	29
表 5-6 Unattended Transfer用户环境 .....	31
表 5-7 Attended Transfer用户环境 .....	33
表 5-8 Call Waiting 用户环境.....	34
表 5-9 Multiple Line Appearance 用户环境 .....	35
表 5-10 Mute 用户环境.....	35
表 5-11 Redial用户环境.....	36
表 5-12 Do Not Disturb 用户环境.....	36
表 6-1 Register 数据操作接口 .....	37
表 6-2 Proxy 数据操作接口 .....	38

## 图片列表

图 2-1 SIP Phone 多业务框架模块图.....	6
图 2-2 SIP Phone 多业务框架协作图.....	6
图 2-3 SIP Phone 多业务框用户交互.....	8
图 2-4 SIP Phone 多业务框架主要概念.....	8
图 2-5 Basic Call 呼叫流程图 .....	10
图 2-6 多业务框架线程模型 .....	11
图 3-1 JPF 架构图 .....	14
图 5-1 Register 序列图.....	24
图 5-2 Basic Call 序列图.....	25
图 5-3 Caller ID 序列图 .....	26
图 5-4 Call Hold 序列图.....	27
图 5-5 Consultation Hold 序列图 .....	29
图 5-6 Unattended Transfer 序列图 .....	30
图 5-7 Attended Transfer 序列图 .....	32
图 5-8 Call Waiting 序列图.....	33
图 5-9 Multiple Line Appearance 序列图 .....	34
图 5-10 Mute 序列图.....	35
图 5-11 Do Not Disturb 序列图.....	36

## 1. 框架概要

### 1.1 目的

实现 SIP Phone 多业务框架的目的有：

- 为多种SIP Phone 终端提供统一的应用框架，一套代码可以同时为PC/PDA 上的 Softphone, 或者 SIP 电话机等多种终端服务
- 分离业务控制和信令控制功能
- 分离业务实现与实际媒体控制功能

### 1.2 功能

SIP Phone 多业务框架实现的功能有：

- 实现语音通话功能
- 实现视频通话功能
- 实现用户交互操作
- 实现 SIP 标准特性
- 实现 SIP 基本呼叫特性（Register, Reliable Response, Heart Beat, ...）
- 实现多种补充业务（Call Hold, Call Transfer, 3-way Conference, ...）
- 实现其他 SIP 业务（IM, Presence, PTT, ...）

### 1.3 应用场景

SIP Phone 多业务框架可以应用在以下设备中：

- SIP Audio/Video Telephone
- SIP Software Phone
- Residential Gateway

## 2. 多业务框架

### 2.1 框架介绍

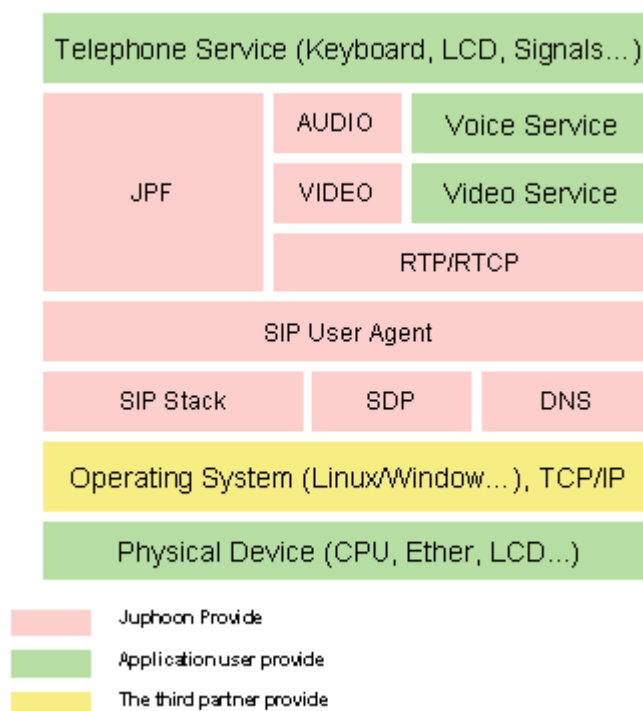


图 2-1 SIP Phone 多业务框架模块图

### 2.2 协作组件

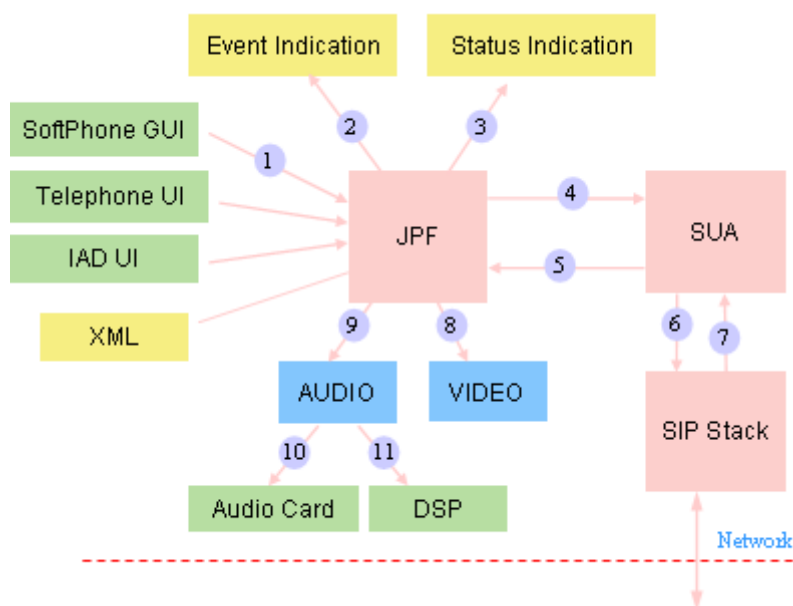


图 2-2 SIP Phone 多业务框架协作图

### 2.2.1 模块分解

在多业务框架中主要有以下模块构成

- SIP 协议栈，实现 SIP 信令的事务控制和会话管理
- SUA (SIP User Agent)，实现了 SIP 多个呼叫（一端对一端为一个呼叫）的业务信令呼叫控制
- JPF (Juphoon Phone Framework)，实现每个端点的业务管理和媒体控制(打开关闭媒体通道，负责信令中的媒体协商)，同时负责接收用户的呼叫和业务控制命令，并且向用户通报接入的呼叫和相关操作状态
- Softphone GUI, Telephone UI, IAD UI 是接近用户级的实现，举例来说，对于Softphone GUI，就是实现了一个电话机的界面，有呼叫按钮，接听按钮 等等；对于 Telephone UI 来说就是实现了摘机，挂机，拨号，拍叉等按键的驱动
- AUDIO, VIDEO 是一层抽象的媒体控制层，JPF 只需要使用 AUDIO, VIDEO提供的操作接口（比如 Audio\_StrmCreate 创建一个媒体流）即可实现对媒体通道的开关、流向控制（比如只发送，只接收等）。AUDIO 层负责跟具体声卡驱动或DSP驱动接口的实现，同时负责维护 RTP 媒体流的控制，而且，所有与 Audio QoS 相关的实现都在此模块来实现

### 2.2.2 协作路径

编号	发起方	接收方	功能描述
1	Softphone GUI	JPF	JPF 模块提供呼叫业务相关的用户接口，使得 Softphone GUI 能根据用户操作来实现业务
2, 3	JPF	Softphone GUI	JPF 提供来对方的呼叫和业务相关的回调，并且提供业务状态指示的回调，Softphoen GUI 只需在回调实现相关的用户操作显示即可，比如收到呼入的指示，则可以打开接听对话框，等待用户接听。
4	JPF	SUA	SUA 提供呼叫对方的业务控制的消息接口
5	SUA	JPF	SUA 提供来自对方的业务控制的消息接口
6	SUA	SIP Stack	SUA 使用 SIP Stack 的原语接口创建和管理会话
7	SIP Stack	SUA	SIP Stack 提供来自对方的会话控制的消息接口
8, 9	JPF	AUDIO, VIDEO	JPF 使用媒体层的接口来控制媒体通道
10	AUDIO	声卡	AUDIO 使用声卡的驱动来放音和接收采样数据
11	AUDIO	DSP	AUDIO 使用 DSP 的驱动来放音和接收采样数据

表 2-1 SIP Phone 业务组件协作路径说明

### 2.3 用户协作

用户需要提供相关的协作组件：

- DSP Service  
提供语音Codec(比如G.711 A/U, G.723.1, G.729等) 和语音 QoS 组件 (比如回声消除, 静音抑制等)
- Video Service  
提供语音采样和显示组件
- Telephone Service  
提供摘机、挂机、拨号的驱动接口, LCD 显示组件等

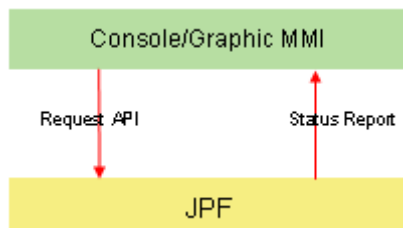


图 2-3 SIP Phone 多业务框用户交互

## 2.4 主要概念

在框架中主要有以下基本概念, 如图中所示:

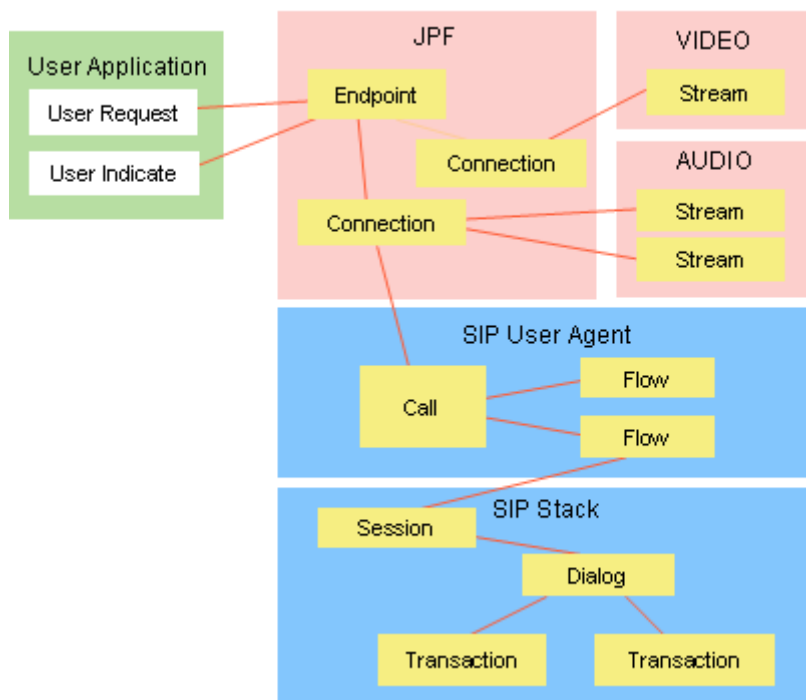


图 2-4 SIP Phone 多业务框架主要概念

### 2.4.1 Endpoint

Endpoint (端点) 是一个表示用户的逻辑概念, 可以对应于Softphone的登录用户, 或者对应于 SIP 电话机, 或者是 SIP 网关的一个端口。每个端点具有一些公有的设备属性, 比如

本地IP、本地的 SIP 信令端口、设备的NAT, STUN等配制信息等。此外，每个端点具有其特有的属性，比如用户 Uri, 注册帐号和密码等。每个端点可以实现不同的业务，比如基本通话、会议电话、即时消息等等。

用户可以通过端点接口要对端点执行注册操作或者获取端点信息。具体的业务是跟具体的 Connection 相关联的。

每个端点具有设备内唯一的数字编号（ID），用户在创建端点的时候获取端点ID，端点的操作是通过端点 ID 来指向对应的端点。因此，用户必须要设立关联控制块来存储端点ID。

### 2.4.2 Connection

Connection（连接）表明了两个 Endpoint 之间的业务关系，比如基本通话、呼叫转移、即时消息等。一个连接只能提供不会冲突的业务，比如在基本通话业务中可以实现呼叫保持，或者发起呼叫转移业务等。像即时消息这些可以在同一连接中发起，也可以在不同的连接中发起。

对于需要多方配合的业务比如三方会议，端点中会维持多个连接状态，每个连接表明与某个远端端点的一个基本通话，此时的会议业务会启动对语音Stream的混音功能。

在每个连接中保留了2个端点之间的一些状态信息，用户可以通过连接接口获取连接双方的地址信息，获取连接的通话状态信息。同时，连接也提供了业务操作接口，比如呼叫保持，呼叫转移等等。

每个连接具有设备内唯一的数字编号（ID），用户在创建连接的时候获取连接ID，连接的操作是通过连接 ID 来指向对应的连接。因此，用户必须要设立关联控制块来存储连接ID。

用户可以提供控制句柄来跟新创建的连接实现关联。当用户收到事件通知的时候，用户可以根据连接的接口获取与事件相关联的句柄，这样就可以快速定位用户管理结构。

### 2.4.3 Stream

Stream 表示媒体数据流，包括语音数据和视频数据。每个流具有方向性，主要有只接收、只发送、同时接收与发送、既不接收也不发送等四个方向。用户可以使用连接接口获取当前的媒体流状态。

### 2.4.4 Call

Call（呼叫）是一个非正式的术语，它是指在端点之间的一些通讯行为，通常用于建立多媒体对话。在语义上是属于协议层面，用户不需要去理解呼叫，只需去关心业务，具体的业务是通过协议呼叫完成的。从模块层次上，呼叫控制存在于SIP User Agent (SUA)模块中。

在存在于两个端点之间的一个连接中，可以实现多个业务，而在 SUA 侧只会对应于一个 Call，即业务控制相关的信息是通过 SUA 的 CALL 负责维护和传递的。

在 SUA 中，每个呼叫具有唯一的数字编号（ID），JPF 跟 SUA 之间的通信通过连接 ID 和 呼叫 ID 来标示对应的呼叫控制块。

### 2.4.5 Call Flow

Call Flow（呼叫流程）表明了呼叫是一个过程，比如基本通话的呼叫流程如下图：

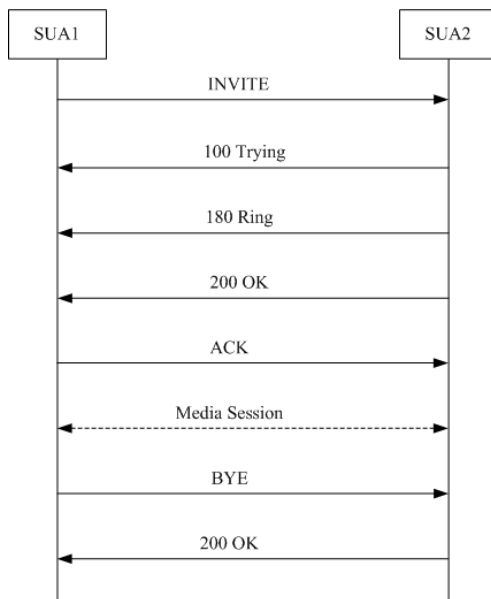


图 2-5 Basic Call 呼叫流程图

在 SUA 的实现过程中，根据呼叫功能分成不同的流程，比如基本呼叫流程、呼叫转移流程等。

在一个呼叫中可以存在多个不冲突的呼叫流程。比如注册流程和基本呼叫流程是有冲突的，所以是属于不同的呼叫。比如基本呼叫流程可以跟呼叫转移流程并存，所以，在实现 Transfer 业务的时候，在一个呼叫中可能存在两个流程。

流程的设计跟实际的业务会有所不同，比如基本通话和呼叫保持业务是在一个基本呼叫流程中。

在 SUA 中，每个流程具有唯一的数字编号（ID），而从 JPF 层面它不需要关心流程ID，只需要关心呼叫ID即可。

### 2.4.6 Session

Session（会话）是SIP 协议中的一个基本概念，表示一个由多媒体发送方和接受方组成的集合，并且包括在发送方和接受方之间得数据流。一个多媒体会议是一个典型的多媒体会话。

在业务框架中，会话可以包括一个或多个Dialog，从 SIP 协议角度解释就是具有相同的 Call-Id、相同的From-Tag和不同的To-Tag的Dialog的集合。从模块层次上，呼叫控制存在于 SIP Stack 模块中。

每个会话具有唯一的数字编号（ID），同时也会存放会话用户的ID，即 SUA 中的呼叫ID。

当在作为主叫时，SUA 在发起新的呼叫时，需要传递呼叫 ID 给 SIP Stack，而 SIP Stack 在回复呼叫时传送会话 ID 给 SUA，通过这样 SUA 和 SIP Stack 完成呼叫和会话的关联。作为被叫，关联过程与之相反。

### 2.4.7 Dialog

Dialog（对话）是 SIP 协议中的一个基本概念，表示一个在两端之间持续一段时间的会话关系。SIP 协议是通过From-Tag 和 To-Tag 来唯一标识一个对话的，tag 是一个随机的加密串，要求具有时空唯一性。

创建 Dialog 的方法主要有 INVITE, SUBSCRIBE, REFER。具体创建过程参见 RFC3261、RFC3265、RFC3515。

每个对话具有唯一的数字编号 (ID)，同时也会存放对话用户的ID，即 SUA 中的流程ID。

当在作为主叫时, SUA 在发起新的呼叫时, 需要传递流程 ID 给 SIP Stack, 而 SIP Stack 在回复呼叫时传送对话 ID 给 SUA, 通过这样 SUA 和 SIP Stack 完成流程和对话的关联。作为被叫, 关联过程与之相反。

### 2.4.8 Transaction

Transaction (事务) 是 SIP 协议中的一个基本概念, 表示单个请求和这个请求的所有响应组成。在 SIP 协议中, 一个如果 INVITE 请求收到了非成功的最终响应, 则 INVITE 事务包括对应 ACK 请求。

每个事务具有唯一的数字编号 (ID), 事务无须存放用户 ID, 但用户在收到事务通知时需要保存事务 ID, 而且用户在发送对应请求的响应时需要把显示的把之前的事务 ID 传递给 SIP Stack。如果用户只是发送请求, 只需要把事务 ID 设置为 ZMAXULONG 传给 SIP Stack 即可。

## 2.5 线程模型

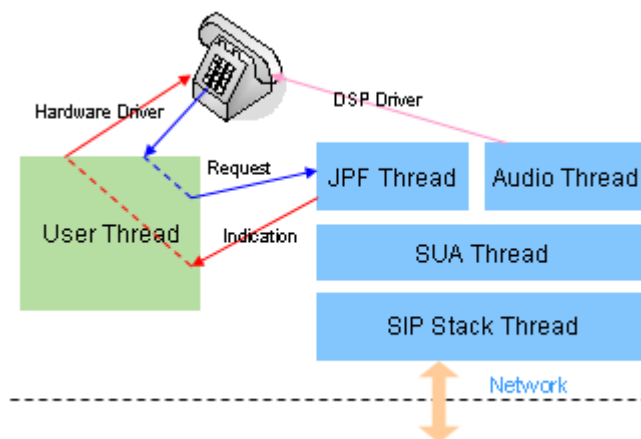


图 2-6 多业务框架线程模型

### 2.5.1 User Thread

用户线程负责驱动界面与 JPF 线程之间的事件传递和处理。当用户执行某一个动作时, 用户线程会收到各种驱动信号或界面事件, 然后把这些事件通知给 JPF 线程。当如用户线程收到 JPF 线程的状态通知时, 产生驱动信号, 然后通过界面程序通知给用户。比如对于 SIP 电话机, 用户线程处理或产生键盘驱动、LCD 驱动等信号, 同时负责传递处理 JPF 线程的呼叫状态事件。

### 2.5.2 JPF Thread

用户应用程序通过直接调用 JPF 提供的接口启动 JPF 任务。直接调用 JPF 启动函数。请看下面的例子。

```
ZINT main()
{
    /* init zos config */
    Zos_SysCfgInit();

    /* zos system init */
    if (Zos_SysInit() != ZOK)
        return -1;

    /* jpf database init */
    Jpf_DbInit(JCPE_CFG_FILE_NAME);

    /* sdp abnf init */
    if (Sdp_AbnfInit() != ZOK)
        return -1;

    /* start utal */
    if (Utal_Start() != ZOK)
        return -1;

    /* start sip stack */
    if (Sip_Start() != ZOK)
        return -1;
    /* start rtp stack */
    if (Rtp_Start() != ZOK)
        return -1;

    /* start sua */
    if (Sua_Start() != ZOK)
        return -1;

    /* start audio */
    if (Audio_Start() != ZOK)
        return -1;

    /* start dns */
    if (Dns_Start() != ZOK)
        return -1;
}
```

```
/* start stun */
if (Stun_Start() != ZOK)
    return -1;

/* start jpf */
if (Jpf_Start() != ZOK)
    return -1;

/* start jcphone */
if (Jcpe_Start() != ZOK)
    return -1;

/* init zos shell */
if (Zsh_Init(ZNULL) != ZOK)
    return -1;

return 0;
}
```

### 2.5.3 SUA Thread

SUA 线程是 SIP User Agent，就是处理 SIP 的呼叫信令流程，基本分为基本呼叫流程，注册呼叫流程REFER 呼叫流程等。SUA 线程对于 JPF 线程来说是一个屏蔽了与呼叫协议相关的流程处理细节的一个层实例。

当 SUA 处理来自 JPF 线程的事件时，SUA 根据呼叫流程状态发送不同的 SIP Stack 的原语事件给 SIP Stack 事件。反之则进行相反的处理。

### 2.5.4 Audio Thread

语音线程是一个抽象的语音处理层，对于 JPF 用户来说是屏蔽了具体的语音编解码和质量处理、语音报文传输的细节。

语音线程要根据具体的 SIP 终端实体提供语音设备和编解码的适配处理。比如一个 SIP 电话使用了某个 DSP 芯片，则用户需要在 Audio 层中修改响应的播放音的驱动接口，修改语音数据的采样数据的处理接口等适配工作。

JPF 用户只需要使用语音模块的接口来实现具体的语音功能。比如播放振铃音，JPF 用户不需要发一个事件给语音线程，实际上只需要调用语音的Audio\_TonePlay 接口即可。

### 2.5.5 SIP Stack Thread

SIP Stack 线程负责处理 SIP 协议的会话处理、对话处理、事务处理、传输处理。SIP Stack 会把各种 SIP 消息和内部事件（比如等待超时事件），根据消息内容和消息顺序，找到对应的会话和对话。在SIP Stack 中的会话、对话中保留有对应的用户 ID。用来标识业务对象。

### 3. JPF 操作对象

Juphoon Phone Framework (JPF) 是基于多业务框架实现的一个 SIP Phone 的应用框架，它主要包括以下组件：

- Endpoint / Connection API  
执行用户对内部对象的操作动作，提供获取内部对象的状态接口
- Status Report  
把各个内部对象状态变化报告给用户
- Application Process  
保证 JPF 与用户和 SUA 三者之间正确的多业务处理

JPF 的架构图如下：

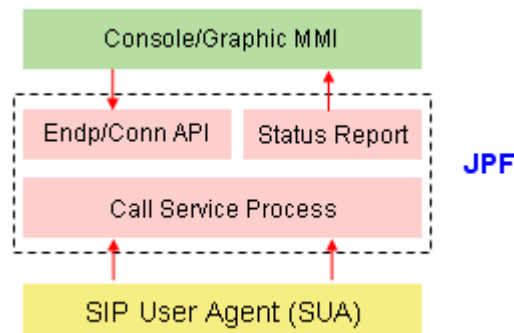


图 3-1 JPF 架构图

#### 3.1 操作接口

操作接口分为命令接口和状态接口，命令接口是直接产生业务功能的接口，比如发起呼叫，接听呼叫等，状态接口是获取操作对象的状态信息，比如呼叫状态，对端的SIP地址信息等。

把操作接口分成两类的目的是提供两个独立功能的接口集合。从使用方式上来说，命令接口是在收到驱动信号或界面事件触发的时候用户直接调用的，而状态接口一般是在用户在收到状态报告的时候，通过其来获取相关的信息（比如对端 SIP Uri 等），使得用户能够显示信息和判断执行后续命令。

操作接口按照操作对象可以分为端点操作接口和连接操作接口。这两个对象接口功能各异，但操作方式类似。从操作顺序来看，首先必须通过端点操作接口创建端点，才能执行连接的各种操作。

#### 3.2 状态报告

状态报告是一个通知用户关于命令操作结果、内部操作对象状态变化的方式，实际上是采用回调的方式告知用户，然后用户通过状态类型来判断是何种状态信息，用户就可以通过状态接口获取相关信息，并使用命令接口执行后续命令。

状态报告可以根据操作对象的不同分为注册状态报告和连接状态报告。用户基本上都要注册这两种回调接口来获取状态报告信息。注册接口是 **Jpf\_UsrRegCallback**，用户把自己实现的回调接口通过此函数向 JPF 注册。请看下面的例子：

```
/* jcphone framework event process register response */
ZINT Jcpe_ActFeProcRegStat(ZULONG dwEndpId, ZUCHAR ucStatType)
{
    if (ucStatType == EN_JPFE_REG_STAT_NOT_REGED)
    {
        JPF_LOG_INFO((JPF_LOGID, "endpoint not registered.));
    }
    else if (ucStatType == EN_JPFE_REG_STAT_REG_ACPTED)
    {
        JPF_LOG_INFO((JPF_LOGID, "endpoint register accepted.));
    }
    else if (ucStatType == EN_JPFE_REG_STAT_REG_REJED)
    {
        JPF_LOG_INFO((JPF_LOGID, "endpoint register rejected.));
    }
    else if (ucStatType == EN_JPFE_REG_STAT_REG_ERR)
    {
        JPF_LOG_INFO((JPF_LOGID, "endpoint register error.));
    }
    else if (ucStatType == EN_JPFE_REG_STAT_UNREG_ERR)
    {
        JPF_LOG_INFO((JPF_LOGID, "endpoint unregister error.));
    }

    return ZOK;
}
```

```
/* jcphone framework event process connection status */
ZINT Jcpe_ActFeProcConnStat(ZULONG dwConnId, ZUCHAR ucStatType)
{
    switch (ucStatType)
    {
        case EN_JPFE_CONN_STAT_CALLOUT:
            JPF_LOG_INFO((JPF_LOGID, "connection <%p> is callout.",
dwConnId));
            break;
        case EN_JPFE_CONN_STAT_ALERTED:
            /* play ringback tone */
            Jpf_ConnPlayTone(EN_AUDIO_TONE_RING_BACK,
JPF_TONE_FOREVER);

            JPF_LOG_INFO((JPF_LOGID, "connection <%p> is alerted.",
dwConnId));
            break;
    }
}
```

```
case EN_JPFE_CONN_STAT_CALLIN:
    /* play ring tone */
    Jpf_ConnPlayTone(EN_AUDIO_TONE_RING, JPF_TONE_FOREVER);
    Zos_Printf("%s\r\n",
               "Calling Even! Press 'a' to accept, or 'r' to
reject");

    /* save connection id */
    g_dwJcpeConnId = dwConnId;

    JPF_LOG_INFO((JPF_LOGID, "connection <%p> remote is call
in.", dwConnId));
    break;
case EN_JPFE_CONN_STAT_CONNED:
    /* stop tone */
    Jpf_ConnStopTone();

    /* save connection id */
    g_dwJcpeConnId = dwConnId;

    JPF_LOG_INFO((JPF_LOGID, "connection <%p> is connected.",
dwConnId));
    break;
case EN_JPFE_CONN_STAT_TERMINATE:
    /* stop tone */
    Jpf_ConnStopTone();

    /* delete the connection id */
    g_dwJcpeConnId = ZMAXULONG;

    JPF_LOG_INFO((JPF_LOGID, "connection <%p> is terminated.",
dwConnId));
    break;
case EN_JPFE_CONN_STAT_MDFYING_ACPTED:
case EN_JPFE_CONN_STAT_MDFYING_ERR:
case EN_JPFE_CONN_STAT_MDFYED:
case EN_JPFE_CONN_STAT_HOLD_ERR:
case EN_JPFE_CONN_STAT_CHOLD_TERMED:
case EN_JPFE_CONN_STAT_CHOLD_ERR:
case EN_JPFE_CONN_STAT_REFERED:
case EN_JPFE_CONN_STAT_REFERER_TERMED:
case EN_JPFE_CONN_STAT_REFERER_ERR:
case EN_JPFE_CONN_STAT_TRSF_ACPTED:
case EN_JPFE_CONN_STAT_TRSF_TERMED:
```

```
case EN_JPFE_CONN_STAT_TRSF_ERR:
case EN_JPFE_CONN_STAT_HEARTBEAT_ERR:
case EN_JPFE_CONN_STAT_REDIRECTED:
    break;
default:
    JPF_LOG_ERROR((JPF_LOGID, "unknown status type.));
    break;
}

return ZOK;
}
```

在上面的例子中Jcpe\_ActFeProcRegStat就是由用户实现的获取注册状态信息的回调函数，而Jcpe\_ActFeProcConnStat是由用户实现的获取连接状态信息的回调函数。用户必须向JPF注册这两个函数，实现方法如下：

```
/* jphone framework process register response */
typedef ZINT (*PFN_JPFPROCREGSTAT) (ZULONG dwEndpId, ZUCHAR
ucStatType);

/* jphone framework process connection status */
typedef ZINT (*PFN_JPFPROCCONNSTAT) (ZULONG dwConnId, ZUCHAR
ucStatType);

/* jphone framework event process by user */
typedef struct tagJPFE_PROC
{
    PFN_JPFPROCREGSTAT pfnRegStat; /* register response process */
    PFN_JPFPROCCONNSTAT pfnConnStat; /* connection status process */
} ST_JPFE_PROC;

ZINT Jcpe_RegCallback(PFN_JPFPROCREGSTAT pfnRegStat,
                      PFN_JPFPROCCONNSTAT pfnConnStat)
{
    ST_JPFE_PROC stEvtProc;

    /* register call back */
    stEvtProc.pfnRegStat = pfnRegStat;
    stEvtProc.pfnConnStat = pfnConnStat;

    return Jpf_UsrRegCallback(&stEvtProc);
}

ZINT main()
{
    char cmd[128];
    ST_ZOS_SSTR stCalleeUri;

    if (Jcpe_Initialize() != ZOK)
        return -1;

    if (Jcpe_RegCallback(Jcpe_ActFeProcRegStat,
                        Jcpe_ActFeProcConnStat) != ZOK)
        return -1;

    if (Jcpe_Start() != ZOK)
        return -1;

    if (Jcpe_EndpCreate(&g_dwJcpeEndpId) != ZOK)
        return -1;
}
```

```
.....  
}
```

在上面的例子中，Jcpe\_RegCallback负责向JPF注册函数Jcpe\_ActFeProcRegStat和函数Jcpe\_ActFeProcConnStat。而函数Jcpe\_RegCallback就是通过调用注册接口Jpf\_UsrRegCallback进行注册的。

### 3.3 应用处理

应用处理实际上是 JPF 的内部状态机，主要分为以下几类：

- Register  
用于处理用户注册服务
- Basic Call  
用于处理基本呼叫业务
- Consultation Hold  
用于处理 Consultation Hold 业务
- Attended Transfer  
用于处理 Attended Transfer 业务
- Unattended Transfer  
用于处理 Unattended Transfer 业务
- Refer  
用于处理 Refer 业务，即收到对方的 REFER 请求。

这些业务操作在不互相冲突下互相共存，即在基本呼叫的基础上同时存在其他增值业务，比如呼叫保持。

JPF 业务功能的扩展和维护也是存在于应用处理模块。当 JPF 收到用户事件、SUA 事件的时候都去驱动应用处理状态机，然后应用处理状态根据业务状态通知事件给用户或SUA。实际上用户状态事件的回调也是在应用处理过程中被隐式调用的。

## 4. JPF 业务接口

### 4.1 端点操作

端点操作接口主要有创建、删除、注册等接口，如下表

接口名称	接口描述
Jpf_EndpCreate	创建一个端点
Jpf_EndpDelete	删除一个端点
Jpf_EndpReg	向注册服务器注册一个端点
Jpf_EndpUnreg	向注册服务器注销一个端点
Jpf_EndpGetCookie	获取与端点相关联的用户句柄
Jpf_EndpGetLocalAddr	获取端点的 SIP 地址
Jpf_EndpGetLocalUri	获取端点的 SIP Uri
Jpf_EndpGetRegState	获取端点的注册状态
Jpf_EndpGetConnNum	获取存在于端点中的有效连接的数目
Jpf_EndpGetConnItem	根据连接的索引编号获取对应的连接 ID

表 4-1 端点操作接口

### 4.2 连接操作

连接操作接口主要有发起呼叫、接听呼叫，终结呼叫等接口，如下表

接口名称	接口描述
Jpf_ConnCall	发起一个呼叫
Jpf_ConnAnswer	接听一个呼叫
Jpf_ConnTerminate	终结一个呼叫
Jpf_ConnHold	呼叫保持
Jpf_ConnCHold	Consultation 呼叫保持
Jpf_ConnUnhold	解除呼叫保持
Jpf_ConnUTrsf	Unattended 呼叫转接
Jpf_ConnATrsf	Attended 呼叫转接
Jpf_ConnReferAcpt	接受呼叫转接
Jpf_ConnReferRej	拒绝呼叫转接
Jpf_ConnPlayTone	开始播放音频
Jpf_ConnStopTone	停止播放音频
Jpf_ConnStartDtmf	开始发送 DTMF 信号
Jpf_ConnStopDtmf	停止发送 DTMF 信号
Jpf_ConnGetMuted	获取语音 Mute 状态
Jpf_ConnSetMuted	设置语音 Mute 状态
Jpf_ConnGetHolded	获取呼叫保持状态

Jpf_ConnGetEndpId	获取连接所属的端点 ID
Jpf_ConnGetCookie	获取连接相关联的用户句柄
Jpf_ConnGetType	获取连接类型
Jpf_ConnGetState	获取连接状态
Jpf_ConnGetCHoldState	获取 Consultation 呼叫保持状态
Jpf_ConnGetUTrsfState	获取 Unattended 呼叫转接状态
Jpf_ConnGetATrsfState	获取 Attended 呼叫转接状态
Jpf_ConnGetReferState	获取呼叫转接状态
Jpf_ConnGetAudioState	获取语音媒体流的状态
Jpf_ConnGetVideoState	获取视频媒体流的状态
Jpf_ConnGetLocalAddr	获取本地 SIP 地址
Jpf_ConnGetPeerAddr	获取对端 SIP 地址
Jpf_ConnGetLocalUri	获取本地 SIP Uri
Jpf_ConnGetPeerUri	获取对端 SIP Uri
Jpf_ConnGetCHoldUri	获取 Consultation Hold Uri
Jpf_ConnGetReferToUri	获取 Refer To Uri
Jpf_ConnGetReferByUri	获取 Refer By Uri

表 4-2 连接操作接口

### 4.3 状态通知

状态通知需要注册以下两个回调接口：

- **PFN\_JPFPROCREGSTAT**
- **PFN\_JPFPROCCONNSTAT**

这两个回调接口分别接收注册状态通知和连接状态通知，详细地函数定义如下：

```

/* jphone framework process register response */
typedef ZINT (*PFN_JPFPROCREGSTAT) (ZULONG dwEndpId, ZUCHAR
ucStatType);

/* jphone framework process connection status */
typedef ZINT (*PFN_JPFPROCCONNSTAT) (ZULONG dwConnId, ZUCHAR
ucStatType);

/* jphone framework event process by user */
typedef struct tagJPFE_PROC
{
    PFN_JPFPROCREGSTAT pfnRegStat; /* register response process */
    PFN_JPFPROCCONNSTAT pfnConnStat; /* connection status process */
} ST_JPFE_PROC;

```

回调注册接口是 **Jpf\_UsrRegCallback**，函数定义如下：

```
/* jphone framework register callback */  
ZINT Jpf_UsrRegCallback(ST_JPFE_PROC *pstEvtProc);
```

#### 4.4 数据管理

JPF 提供部分的数据管理功能，实际的产品数据管理则要根据需求进行修改。

下表列举部分的数据接口，详细的接口信息参见《JPF Function Definition》。

接口名称	接口描述
Jpf_DbGetLocallp	获取本地的 IP 地址
Jpf_DbGetLogLevel	获取当前日志级别
Jpf_DbGetUsrName	获取本地用户名
Jpf_DbGetUsrUri	获取本地用户 Uri
Jpf_DbSetLocallp	设置本地的 IP 地址
Jpf_DbSetLogLevel	设置当前日志级别
Jpf_DbSetUsrName	设置本地用户名
Jpf_DbSetUsrUri	设置本地用户 Uri

表 4-3 数据管理接口

## 5. JPF 业务特性

JPF 实现以下 SIP 业务:

- Register
- Basic Call
- Caller ID
- Call Hold
- Consultation Hold
- Unattended Transfer
- Attended Transfer
- Call Waiting
- Multiple Line Appear
- Mute
- Redial
- Do Not Disturb

### 5.1 Register

Register 是提供用户注册和用户注销功能, 用户应用程序与 JPF 等模块的呼叫过程如下图:

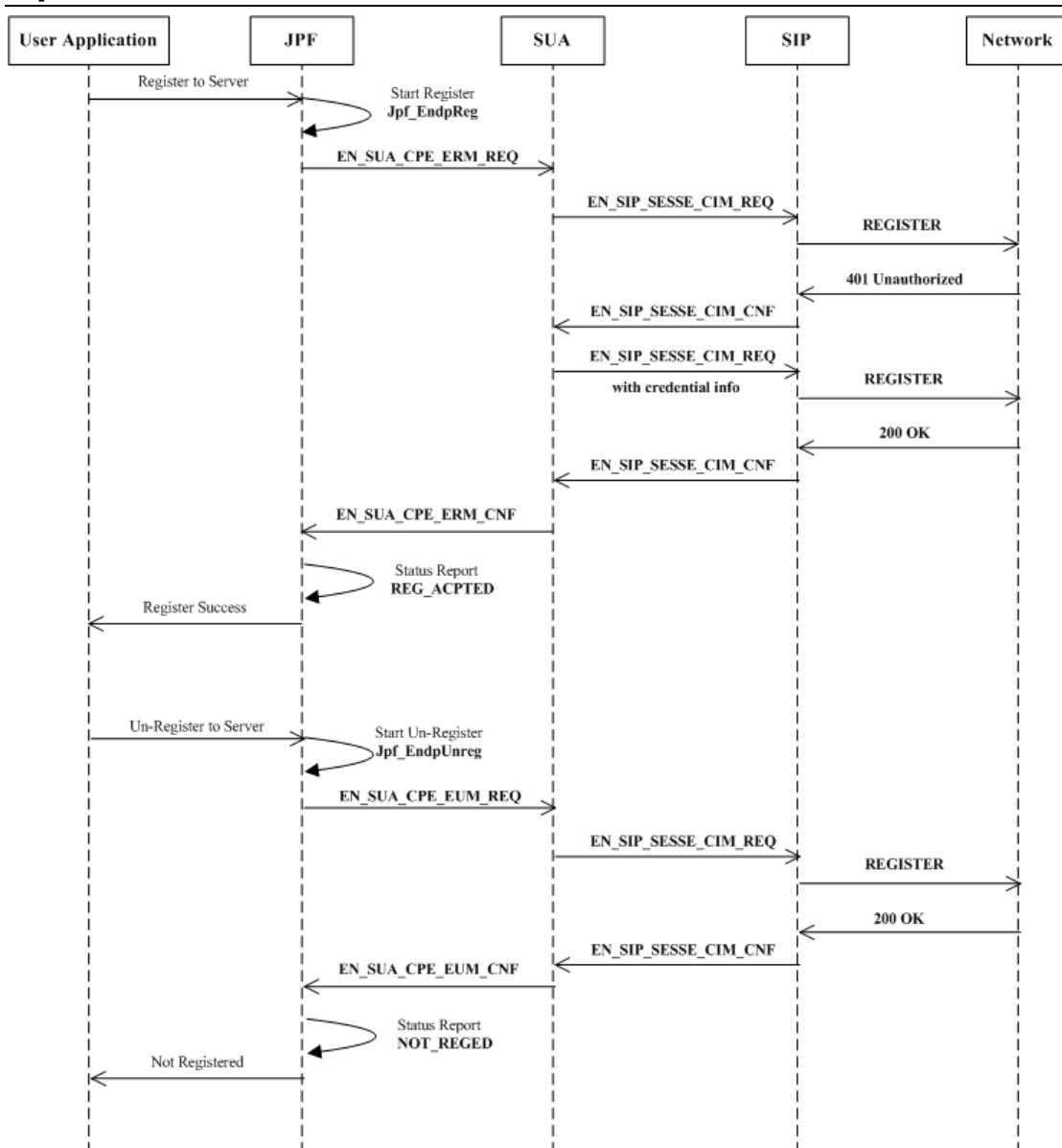


图 5-1 Register 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b>
呼叫回调	用户需要实现注册状态通知回调函数，函数类型是 <b>PFN_JPFPROCREGSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册。
信息获取	首先根据回调函数传入的状态类型来判断注册状态。 用户可以使用 <b>Jpf_EndpGetCookie</b> 来获取用户程序的句柄， <b>Jpf_EndpGetRegState</b> 来获取注册的状态。
补充说明	用户可以通过 <b>Jpf_DbSetRegDomain</b> , <b>Jpf_DbSetRegAccount</b> , <b>Jpf_DbSetRegPasswd</b> , <b>Jpf_DbSetProxyIp</b> , <b>Jpf_DbSetProxyUdpPort</b> , <b>Jpf_DbSetProxyTcpPort</b> 等接口设置注册配置。

用户可以通过 **Jpf\_DbSetRegEnable** 接口来设置是否进行注册，

表 5-1 Register 用户环境

### 5.2 Basic Call

Basic Call 是提供基本通话功能，用户应用程序与 JPF 等模块的呼叫过程如下图：

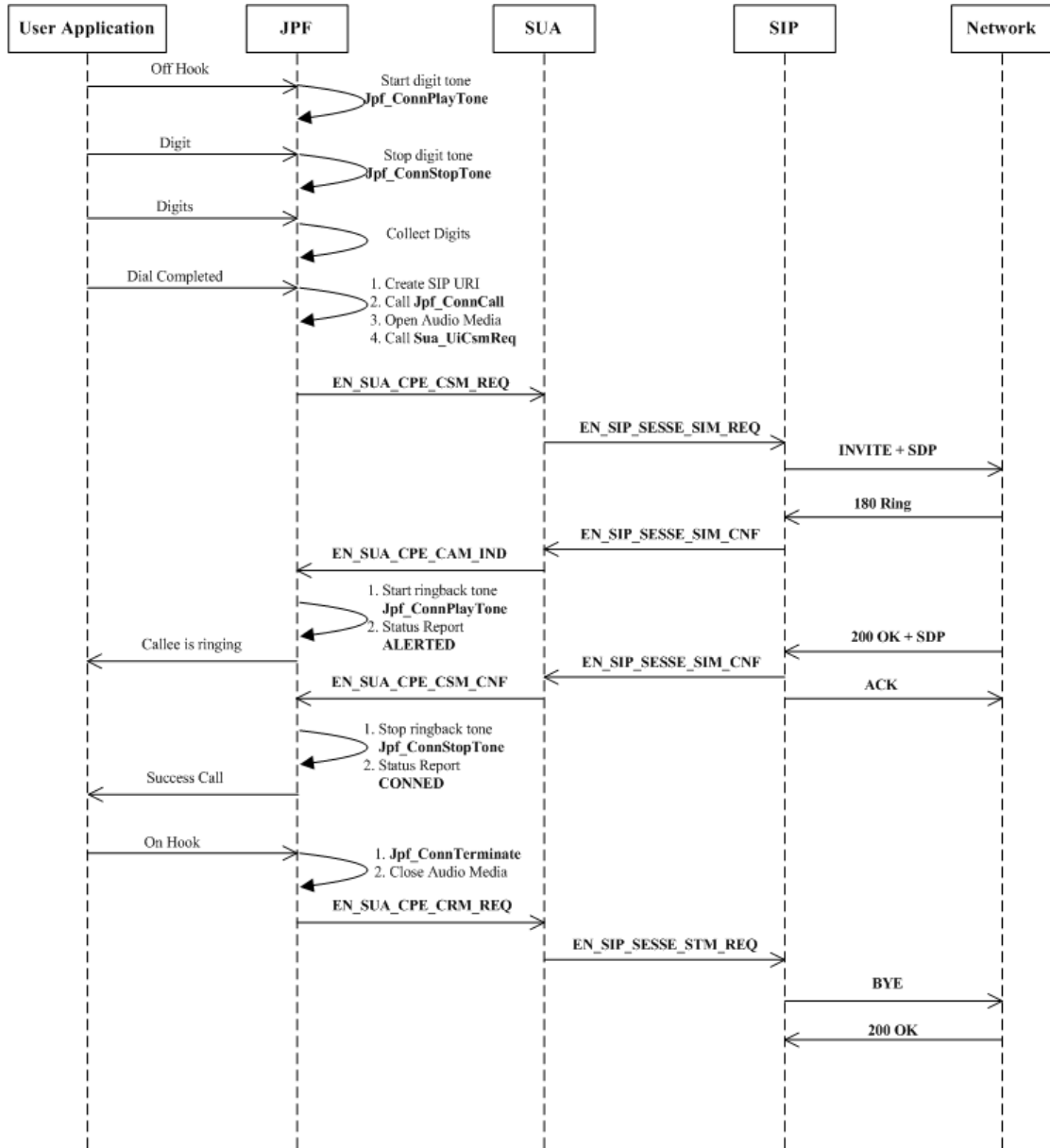


图 5-2 Basic Call 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b>
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进

	行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID, 使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄; 使用 <b>Jpf_ConnGetState</b> 来获取连接状态; 使用 <b>Jpf_ConnGetLocalAddr</b> , <b>Jpf_ConnGetPeerAddr</b> 等函数获取连接双方的信息。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> , <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息, 可以通过 <b>Jpf_DbSetUsedCodecs</b> , <b>Jpf_DbSetAecEnable</b> , <b>Jpf_DbSetAnrEnable</b> , <b>Jpf_DbSetAgcEnable</b> , <b>Jpf_DbSetAsdEnable</b> , <b>Jpf_DbSetLiveToneEnable</b> 等接口设置媒体能力和选项

表 5-2 Basic Call 用户环境

### 5.3 Caller ID

Caller ID 是指在收到新的呼叫时, 用户可以获得并显示主叫的用户信息。用户应用程序与 JPF 等模块的呼叫过程如下图:

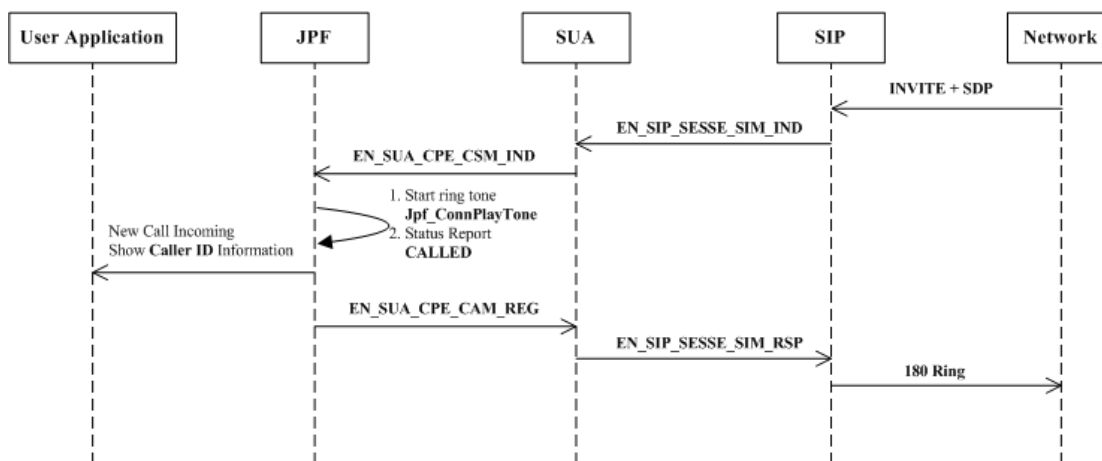


图 5-3 Caller ID 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建, 创建方法是 <b>Jpf_EndpCreate</b>
呼叫回调	用户需要实现连接状态通知回调函数, 函数类型是 <b>PFN_JPFPROCCONNSTAT</b> , 在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID, 使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄; 使用 <b>Jpf_ConnGetState</b> 来获取连接状态; 使用 <b>Jpf_ConnGetPeerAddr</b> , <b>Jpf_ConnGetPeerUri</b> 等函数获取连接主叫的用户信息。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> , <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息。

表 5-3 Caller ID 用户环境

Caller ID 的格式是 [UserName] <sip:UserInfo@UserAddress>, 其中 UserName 是可选的, 详细地 Caller ID 信息是存在于 INVITE 消息中的 From Header 中。举例如下:

"Bob Liu" <sip:bob@juphoon.com>  
 <sip:+8657587304379@juphoon.com>

### 5.4 Call Hold

Call Hold 是提供呼叫保持业务, 用户应用程序与 JPF 等模块的呼叫过程如下图:

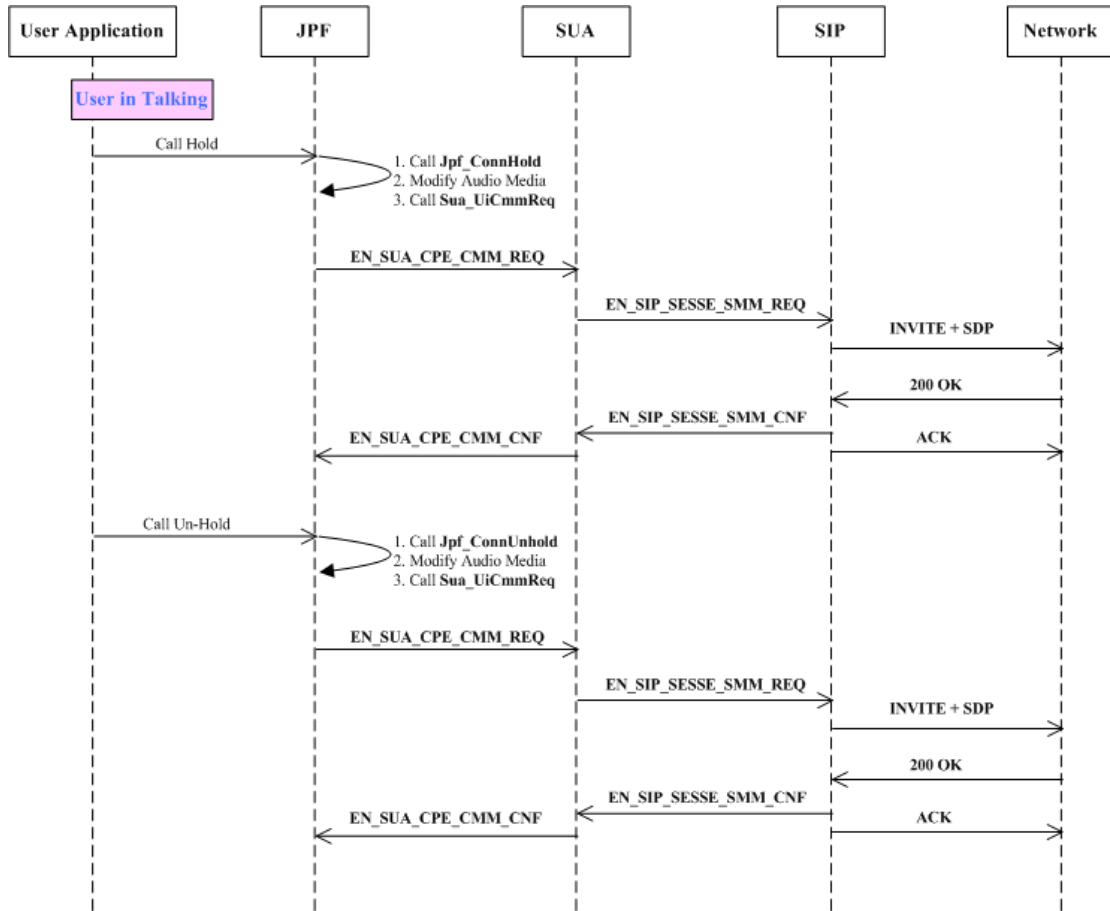


图 5-4 Call Hold 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建, 创建方法是 <b>Jpf_EndpCreate</b> 用户连接已经进入正常通话状态
呼叫回调	用户需要实现连接状态通知回调函数, 函数类型是 <b>PFN_JPFPROCCONNSTAT</b> , 在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态, 主要是判断 Hold 是否出错, 连接是否终结。

补充说明	无
------	---

表 5-4 Call Hold 用户环境

### 5.5 Consultation Hold

Consultation Hold提供顾问模式的呼叫保持功能, 用户应用程序与 JPF 等模块的呼叫过程如下图:

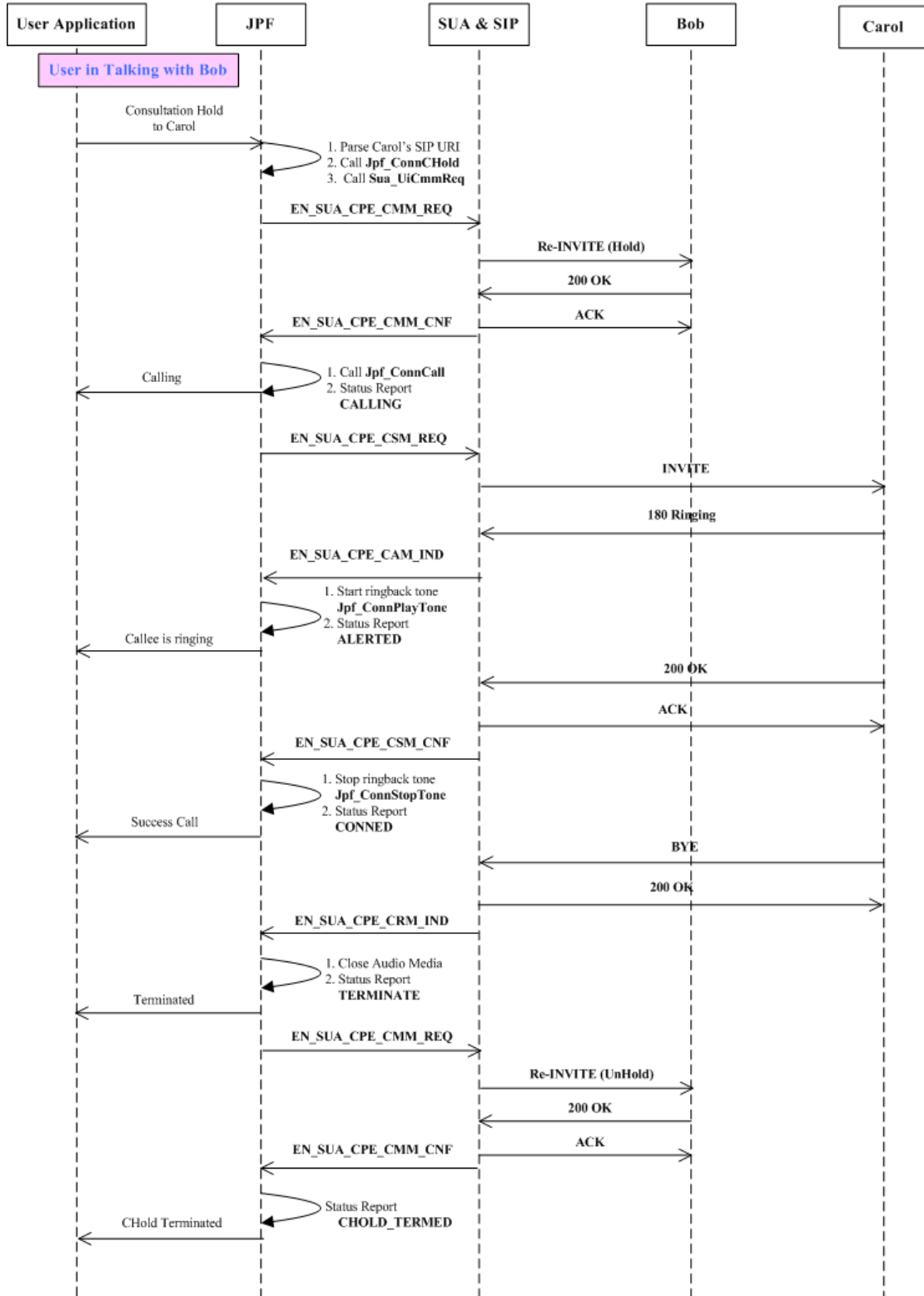


图 5-5 Consultation Hold 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b> 。 用户必须已经与对端建立呼叫。
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID，使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄；使用 <b>Jpf_ConnGetState</b> 来获取连接状态；使用 <b>Jpf_ConnGetLocalAddr</b> ， <b>Jpf_ConnGetPeerAddr</b> 等函数获取连接双方的信息。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> ， <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息，可以通过 <b>Jpf_DbSetUsedCodecs</b> ， <b>Jpf_DbSetAecEnable</b> ， <b>Jpf_DbSetAnrEnable</b> ， <b>Jpf_DbSetAgcEnable</b> ， <b>Jpf_DbSetAsdEnable</b> ， <b>Jpf_DbSetLiveToneEnable</b> 等接口设置媒体能力和选项

表 5-5 Consultation Hold用户环境

## 5.6 Unattended Transfer

Unattended Transfer提供非参与模式的呼叫转移功能，用户应用程序与 JPF 等模块的呼叫过程如下图：

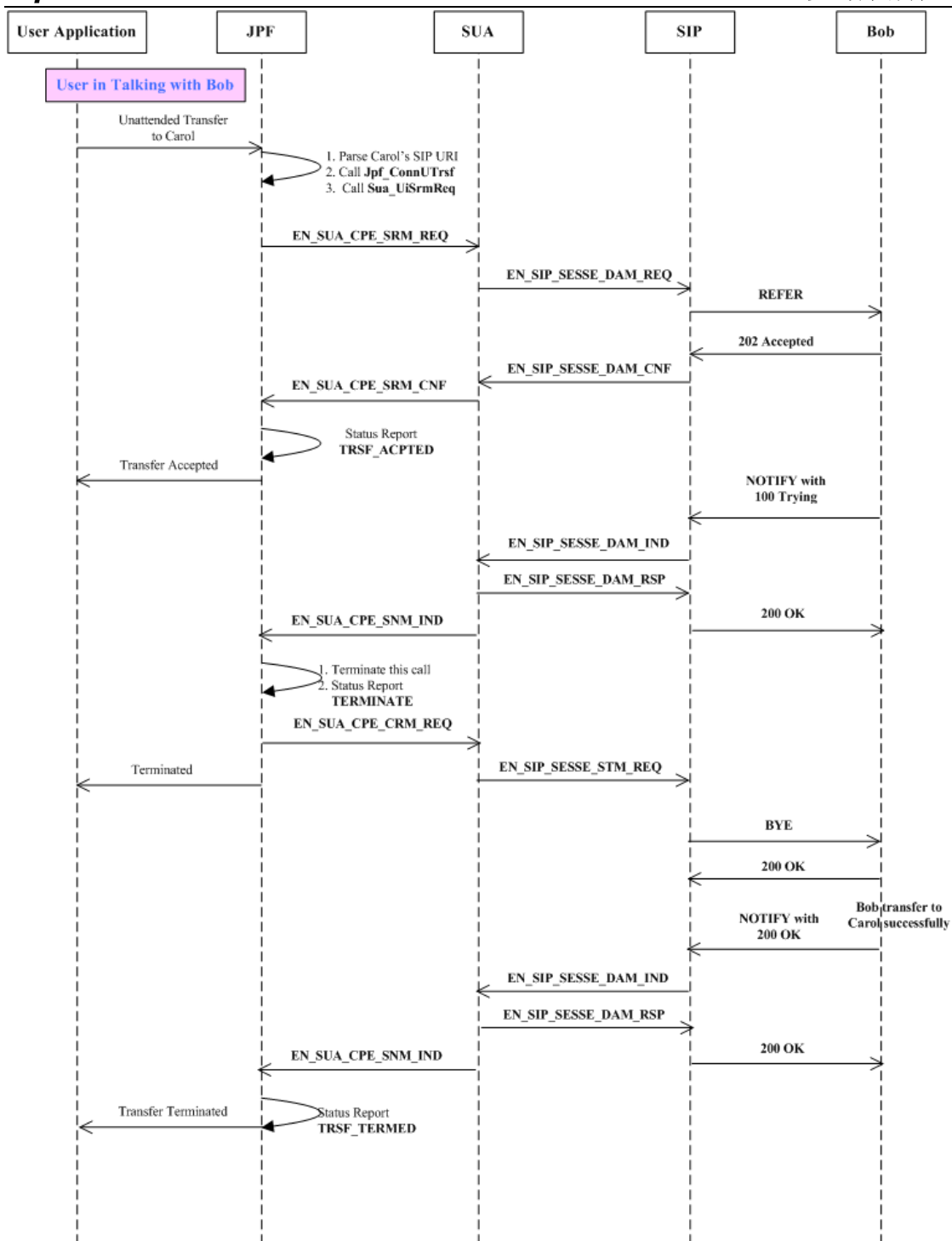


图 5-6 Unattended Transfer 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b> 。 用户必须已经与对端建立呼叫。
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进

	行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID, 使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄; 使用 <b>Jpf_ConnGetState</b> 来获取连接状态; 使用 <b>Jpf_ConnGetLocalAddr</b> , <b>Jpf_ConnGetPeerAddr</b> 等函数获取连接双方的信息。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> , <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息, 可以通过 <b>Jpf_DbSetUsedCodecs</b> , <b>Jpf_DbSetAecEnable</b> , <b>Jpf_DbSetAnrEnable</b> , <b>Jpf_DbSetAgcEnable</b> , <b>Jpf_DbSetAsdEnable</b> , <b>Jpf_DbSetLiveToneEnable</b> 等接口设置媒体能力和选项

表 5-6 Unattended Transfer用户环境

## 5.7 Attended Transfer

Attended Transfer提供参与模式的呼叫转移功能, 用户应用程序与 JPF 等模块的呼叫过程如下图:

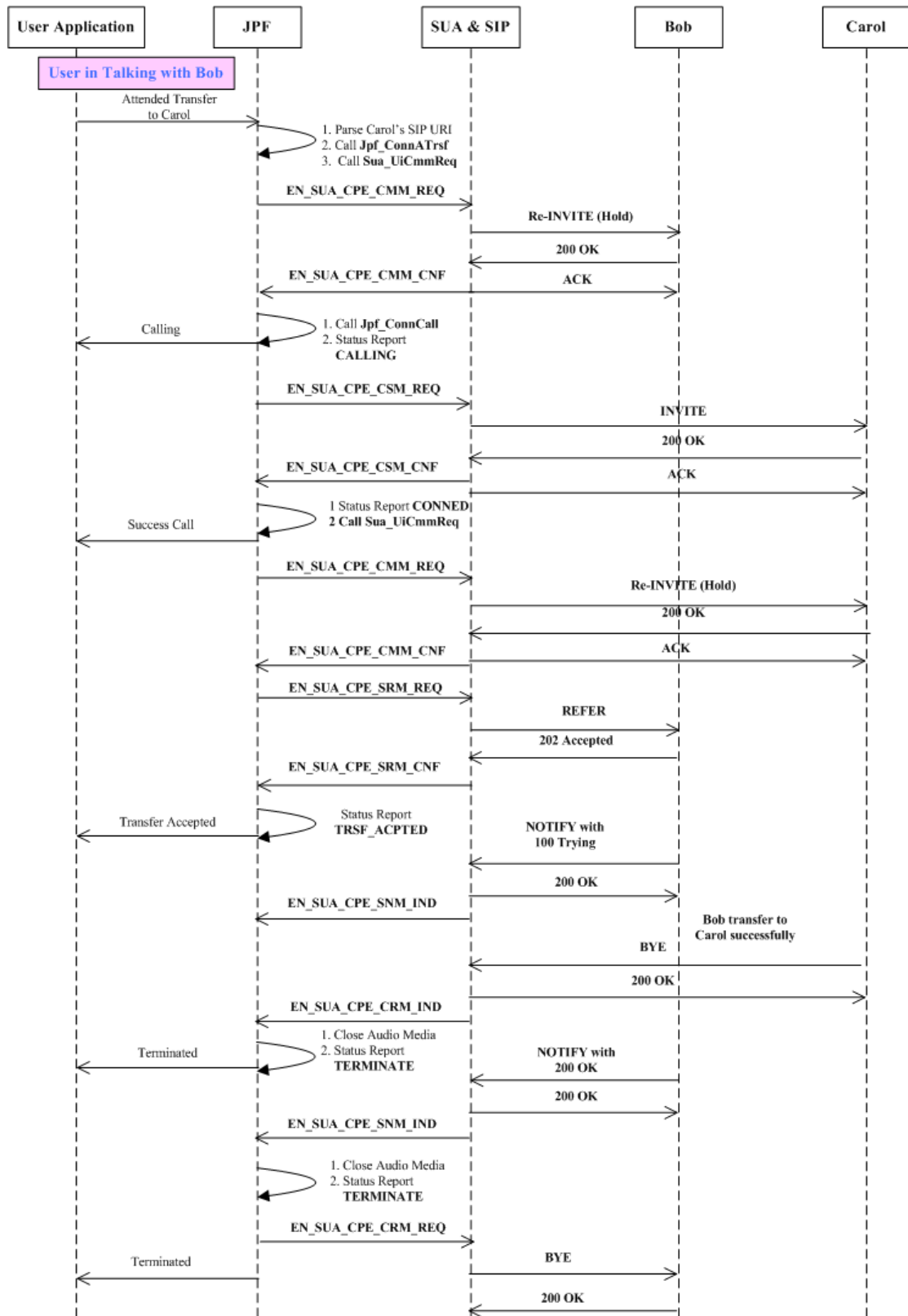


图 5-7 Attended Transfer 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b> 。 用户必须已经与对端建立呼叫。
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID，使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄；使用 <b>Jpf_ConnGetState</b> 来获取连接状态；使用 <b>Jpf_ConnGetLocalAddr</b> ， <b>Jpf_ConnGetPeerAddr</b> 等函数获取连接双方的信息。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> ， <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息，可以通过 <b>Jpf_DbSetUsedCodecs</b> ， <b>Jpf_DbSetAecEnable</b> ， <b>Jpf_DbSetAnrEnable</b> ， <b>Jpf_DbSetAgcEnable</b> ， <b>Jpf_DbSetAsdEnable</b> ， <b>Jpf_DbSetLiveToneEnable</b> 等接口设置媒体能力和选项

表 5-7 Attended Transfer用户环境

### 5.8 Call Waiting

Call Waiting 是指用户正在与一方通话时收到了另一方的呼叫。用户应用程序与 JPF 等模块的呼叫过程如下图：

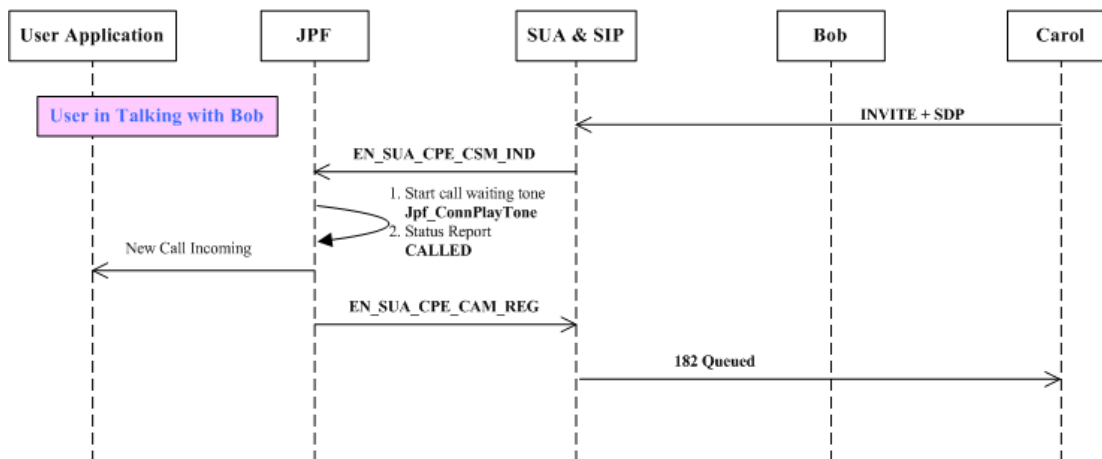


图 5-8 Call Waiting 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b> 。 用户必须已经与某一方建立了呼叫。
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册

信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID, 使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄; 使用 <b>Jpf_ConnGetState</b> 来获取连接状态; 使用 <b>Jpf_ConnGetPeerAddr</b> , <b>Jpf_ConnGetPeerUri</b> 等函数获取连接主叫的用户信息; 使用 <b>Jpf_ConnAnswer</b> 来接受呼叫; 使用 <b>Jpf_ConnTerminate</b> 来拒绝呼叫。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> , <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息。

表 5-8 Call Waiting 用户环境

### 5.9 Multiple Line Appearance

Multiple Line Appearance 是指用户可以同时与多方进行通话。用户应用程序与 JPF 等模块的呼叫过程如下图:

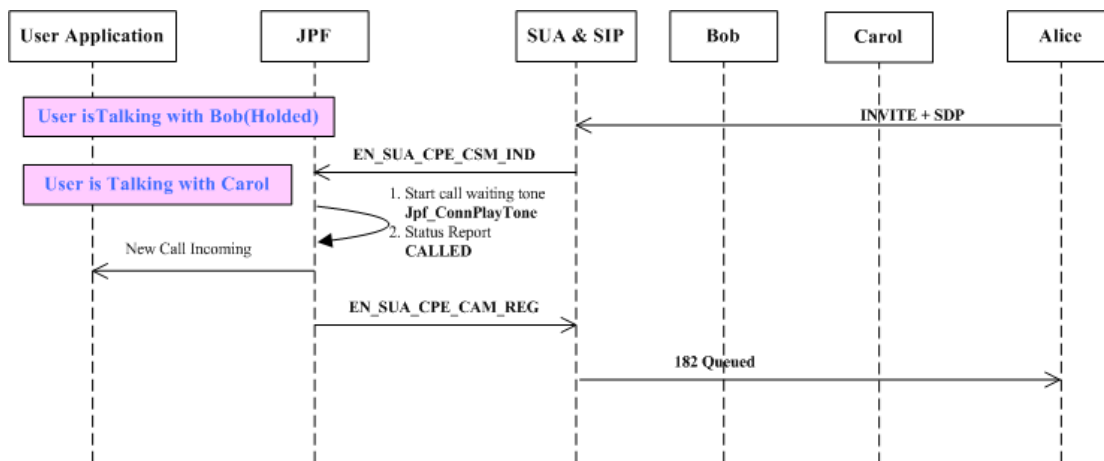


图 5-9 Multiple Line Appearance 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建, 创建方法是 <b>Jpf_EndpCreate</b> 用户必须已经至少与某一方建立了呼叫。
呼叫回调	用户需要实现连接状态通知回调函数, 函数类型是 <b>PFN_JPFPROCCONNSTAT</b> , 在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点 ID, 使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄; 使用 <b>Jpf_ConnGetState</b> 来获取连接状态; 使用 <b>Jpf_ConnGetPeerAddr</b> , <b>Jpf_ConnGetPeerUri</b> 等函数获取连接主叫的用户信息; 使用 <b>Jpf_ConnAnswer</b> 来接受呼叫; 使用 <b>Jpf_ConnTerminate</b> 来拒绝呼叫。
补充说明	用户可以通过 <b>Jpf_DbSetUsrName</b> , <b>Jpf_DbSetUsrUri</b> 等接口设置用户信息。

表 5-9 Multiple Line Appearance 用户环境

### 5.10 Mute

Mute 是指通话过程中设置语音状态，使得用户开关语音流。用户应用程序与 JPF 等模块的呼叫过程如下图：

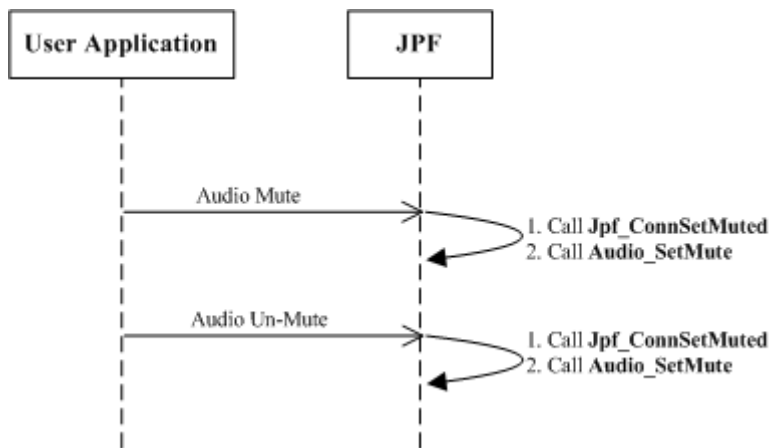


图 5-10 Mute 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b> ，连接必须已经创建，也就是说连接至少是在呼入或呼出状态。
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态，主要是判断 Mute 是否出错，连接是否终结。
补充说明	用户可以通过 <b>Jpf_ConnGetMuted</b> 接口获取 Mute 的开关状态。

表 5-10 Mute 用户环境

### 5.11 Redial

Redial 是指用户可以向最近一次呼出的对端发起新的呼叫。用户使用过程主要是要保存最近一次呼出的呼叫信息（对端信息），然后按照 Basic Call 的方式发起新的呼叫。

环境名称	环境描述
前置条件	用户端点必须首先已经创建，创建方法是 <b>Jpf_EndpCreate</b> 。必须存在上次呼出呼叫的信息，无论呼叫是否成功。
呼叫回调	用户需要实现连接状态通知回调函数，函数类型是 <b>PFN_JPFPROCCONNSTAT</b> ，在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态。 用户可以在连接回调接口中使用 <b>Jpf_ConnGetEndpId</b> 来获取对应的端点

	ID, 使用 <b>Jpf_ConnGetCookie</b> 来获取用户句柄; 使用 <b>Jpf_ConnGetState</b> 来获取连接状态; 使用 <b>Jpf_ConnGetLocalAddr</b> , <b>Jpf_ConnGetPeerAddr</b> 等函数获取连接双方的信息。
补充说明	用户可以通过 <b>Jpf_DbSetUserName</b> , <b>Jpf_DbSetUserUri</b> 等接口设置用户信息, 可以通过 <b>Jpf_DbSetUsedCodecs</b> , <b>Jpf_DbSetAecEnable</b> , <b>Jpf_DbSetAnrEnable</b> , <b>Jpf_DbSetAgeEnable</b> , <b>Jpf_DbSetAsdEnable</b> , <b>Jpf_DbSetLiveToneEnable</b> 等接口设置媒体能力和选项

表 5-11 Redial用户环境

### 5.12 Do Not Disturb

DND 是指用户不希望接听任何的呼叫。用户应用程序与 JPF 等模块的呼叫过程如下图:

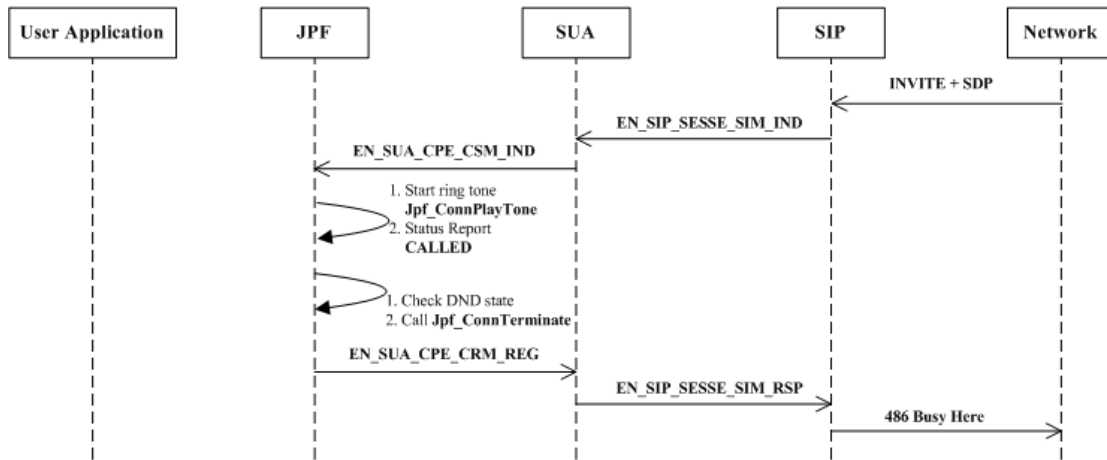


图 5-11 Do Not Disturb 序列图

环境名称	环境描述
前置条件	用户端点必须首先已经创建, 创建方法是 <b>Jpf_EndpCreate</b>
呼叫回调	用户需要实现连接状态通知回调函数, 函数类型是 <b>PFN_JPFPROCCONNSTAT</b> , 在初始化的时候用 <b>Jpf_UsrRegCallback</b> 进行注册
信息获取	首先根据回调函数传入的状态类型来判断连接状态, 如果是新的呼叫则立即发送拒绝消息。
补充说明	用户可以通过 <b>Jpf_DbGetDndEnable</b> , <b>Jpf_DbSetDndEnable</b> 等接口设置 DND 标志。

表 5-12 Do Not Disturb 用户环境

## 6. JPF 协议特性

JPF 实现以下 SIP 协议特性:

- Register
- Register Refresh
- Un-Register
- Outbound Proxy
- Proxy/WWW Authentication
- Stun
- DTMF

### 6.1 Register

如果用户设置注册标志或者修改注册信息，那么 JPF 都可以启动注册过程，注册数据操作接口有：

接口名称	接口描述
Jpf_DbGetRegDomain	获取注册域名
Jpf_DbGetRegAccount	获取注册帐号
Jpf_DbGetRegPasswd	获取注册密码
Jpf_DbGetRegEnable	获取注册标志
Jpf_DbSetRegDomain	设置注册域名
Jpf_DbSetRegAccount	设置注册帐号
Jpf_DbSetRegPasswd	设置注册密码
Jpf_DbSetRegEnable	设置注册标志

表 6-1 Register 数据操作接口

注册服务器中的地址信息是保存在 Proxy IP 地址信息中，具体接口参见对应接口文档。当用户注册成功后，SUA 会根据 EXPIRES Header 中 Expire 值启动计时器。当计时器超时会重新启动注册过程。

### 6.2 Un-Register

如果用户取消注册标志，那么 JPF 就会启动用户向服务器注销账号信息的过程

### 6.3 Outbound Proxy

如果用户设置了Proxy 服务器的信息，那么用户的所有呼叫业务都会通过 Proxy 服务器来完成，Proxy 数据操作接口有：

接口名称	接口描述
Jpf_DbGetProxyIp	获取 Proxy 服务器 IP 地址
Jpf_DbGetProxyUdpPort	获取 Proxy 服务器 SIP UDP 端口号
Jpf_DbGetProxyTcpPort	获取 Proxy 服务器 SIP TCP 端口号
Jpf_DbSetProxyIp	设置 Proxy 服务器 IP 地址

Jpf_DbSetProxyUdpPort	设置 Proxy 服务器 SIP UDP 端口号
Jpf_DbSetProxyTcpPort	设置 Proxy 服务器 SIP TCP 端口号

表 6-2 Proxy 数据操作接口

#### 6.4 Proxy/WWW Authentication

如果当用户进行注册（REGISTER）或发起呼叫（INVITE）时会收到 401 或者 407 的响应，表示 Proxy 服务器需要对用户进行身份验证。此时，用户需要重新构造消息请求，把用户帐号和密码凭证发给 Proxy 服务器。

#### 6.5 Stun

如果用户在防火墙或NAT之后，那么就可以使用 STUN 开关来实现 NAT 穿透。

Stun 数据操作接口就是 **Jpf\_DbGetStunEnable** 和 **Jpf\_DbSetStunEnable**。

#### 6.6 DTMF

用户可以使用 **Jpf\_ConnDtmf** 接口来发送带外（RFC 2833）和带内 DTMF 信号。至于接收 DTMF 信号，用户需要在 AUDIO 层中设置接收 DTMF 的回调函数。

详细信息参见 RFC 2833。