第五章 用户设备配置

目录

在本章中，我们将深入研究那些你希望接入 Asterisk 系统的用户设备的配置，比如各种 VoIP 电话机。在 Asterisk 中为这些设备配置一个 Channel 是相当简单的，但是你要了解你还需要配置这些设备本身，从而让它们知道向哪里发送呼叫。换句话说，在 Asterisk 系统中配置终端设备包括两个方面：1）让 Asterisk 知道这些设备；2）让这些设备知道 Asterisk。

Asterisk 怎样用 SIP 协议工作

SIP 是一个点对点的协议，尽管在一般的应用中，我们实际上是把一些终端配置为用户端（Client），而把一些网关配置为服务器（Server），但 SIP 协议仍旧按照认为终端之间是点对点的关系来工作。这意思是说，SIP 电话机总是期望与另一部 SIP 电话机建立直接连接（Connection），而不是通过 PBX 中转。

现实中，许多 SIP 处理都发生在服务器上，具体到 Asterisk，通常的作法是认为 PBX 在所有的连接之间。在 Asterisk 系统中，当我们用一部电话机向另一部电话机发起一个呼叫时，实际上有两个独立的呼叫在发生：一个是从主叫到 Asterisk，而另一个则是从 Asterisk 到被叫。Asterisk 负责把这两个呼叫桥接起来。

因此，从 SIP 电话机的角度来看，你需要把它配置为把所有的呼叫都发给 Asterisk，即使这台设备有能力绕过 Asterisk 直接与被叫设备建立连接。SIP 协议是非常复杂和灵活的，这使得配置 SIP 终端看起来有些难，因为这些 SIP 终端提供的配置灵活性远远高于我们对其配合 Asterisk 所要求的功能。

尽管大部分设备都支持基于 WEB 的网管界面，当你需要为多于一个设备做配置时，我们仍然推荐基于服务器的配置流程。在基于服务器的配置流程中，我们只需要将服务器的 IP 地址告诉终端，这些终端就会从服务器自动下载包含必需参数的配置文件。举例来说，这可以通过安放在 FTP 服务器上的 XML 文件来实现。具体的下载过程，以及参数文件里面的参数定义，各个设备厂商的实现方法各有不同。在本章中，我们只是从 Asterisk 的角度来讨论这些配置方法。

5.1 给电话命名的概念

在开始为我们的电话在 Asterisk 上做配置之前，我们想先推荐一些最佳的电话机命名方法，以澄清用户（users），分机号码（extension numbers），以及电话机本身的概念。

在 Asterisk 中，系统关心的是 channel name。在这里根本没有用户（user）的概念[注1]，而分机（extensions）只是用来确定需要采用的呼叫流程。

举例来说，你可以通过 dialplan 定义当分机号码 100 被呼叫时，我办公桌上的电话会振铃。但是你也可以简单的把分机号 100 定义为语音邮箱（voicemail box），或是定义为一个会议室（conference room）。我还甚至还可以把分机号码 100 定义为在周一到周五的早九点到晚 5 点转接到我的桌上电话，而在其它时间转接到其他人的电话。类似地，当某个设备呼出电话时，也可以配置为在上班时间显示白天的号码，而在其余时间显示下班的号码。

Asterisk Extensions

Extension 的概念在 Asterisk 中是非常重要的。在大多数 PBX 中，分机（extension）只是一个号码，当你拨叫这个号码时，会导致一部电话振铃。而在 Asterisk 中，extension 是 dialplan 中一个指令组的名字。如果你把 Asterisk extension 看作一个脚本名（script name），那就对了。一个 Asterisk extension 名字可以是一个数字（例如 100），这个数字号码可以导致一个电话振铃；但是它也可以是一个代表运行一系列 dialplan 应用的名字（如 voicemail）。

在本书后续章节，我们会对 Asterisk extension 有更详细的讨论。但是首先，我们先来做一些电话的配置。

在 extension 的名字和 extension 实际作用之间的抽象，是 Asterisk 中的一个非常重要的概念。一个 extension 100 可以做很多件事，这取决于到底有多少个变量被编程到系统中。这对于实现诸如“公用办公桌”（hot-desking）这样的特性非常有意义。

公用办公桌（hot-desking）是指这样一种特性，它允许用户在某个电话机上登录，并通过这部电话机接打电话。举例来说，然我们假设我们有三个销售员，他们一般都在拜访客户而不在办公室，但是他们每个月也都会在办公室呆几天以处理文案工作。由于他们一般不会同时出现在办公室，除了给他们每个人分配一部独立电话机外，我们还可以让他们共享一部电话（对于更大规模来说，应该是共享一个电话机池，比如说三个电话机组成的电话机池）。这个场景可以证明将用户和分机的概念与物理电话机的概念分开带来的便利性（和必要性）。

所以，什么是给一部电话机命名的坏名字呢？用人名给电话机命名，例如[SimonLeBon]，对 Asterisk 来说就是个坏名字，因为这部电话还可能被例如 Joan Jett 或 Rick Astley 使用。同样的理由也解释了为什么我们不希望用分机号码命名一个电话机：类似[100]这样的名字对一部电话机来说也是个坏名字，因为你将来可能重新配置这部话机用于号码 160，或者这部电话机可能在“公用办公桌”的场景中被多个人使用。从安全的角度看，使用数字化的账号名称也是个非常糟糕的主意，这一点将在第 26 章进一步讨论。

命名电话机的一种流行办法是使用它的 MAC 地址。MAC 地址可以作为电话机的唯一标识，无论你把这部话机放在哪里使用。MAC 地址也不直接与这部话机的使用者，当前分机号码等相关。有些公司会在他们的产品标签上印刷有唯一的设备序列号，这个序列号也是命名电话机的不错选择。因为这个序列号同样不与特定的使用者有任何逻辑相关，而只与设备本身相关。

如何命名你的电话机取决于你，但是我们仍然希望将电话机的概念与使用者、在网络中的地址等概念分开，因为这些概念在 Asterisk 处理的范围之外，并且可能在任何时候改变。

在这本书中，你会发现我们使用类似 MAC 地址的命名（例如 **0000FFFF0001** 和 **0000FFFF0002**）来区分不同的电话。你也可以用与硬件一一匹配的其它名字（或者其它具有唯一性的字符串）。

最后，要说清楚的一点是，我们对于 VoIP 电话机设备命名的建议并不是技术要求。你完全可以用任何你喜欢的方式命名，当然这些命名需要符合 Asterisk 的命名规范（以 Alpha 字母或数字开始，并且没有空格，就可以了）。

5.2 硬件电话，软件电话，和模拟终端适配器

一般而言，你提供给客户作为电话的终端可能有三种，它们是硬件电话（hardphones），软件电话（softphones），以及模拟终端适配器（Analog Terminal Adaptors，ATAs）。

硬件电话是一种物理设备。它看起来就跟一个办公电话一样：它有听筒，有数字按键，等等。它直接联接到网络上，它就是当人们谈及 VoIP 电话机（或 SIP 电话机）时所想到的设备。

软件电话是一种运行在笔记本电脑或台式机上的应用程序。声音处理必需通过 PC 的声卡进行，所以你一般还需要一个能跟这个程序很好配合的头戴耳机。最近，软件电话已经可以在智能手机上使用，这给你的智能手机提供了除蜂窝网络之外的方法来联接其它网络。软件电话的界面一般看起来和硬件电话差不多，但这不是必须的。

ATA 是设计用来允许传统模拟电话（或者其它模拟设备，如传真机、数字无绳电话、寻呼放大器，等等）连接到 SIP 网络上的。它一般被设计为一个三明治大小的盒子，有一个 RJ-11 接口用来连接传统电话（一般被称为 FXS 接口），一个 RJ-45 接口用来连接以太网，还有一个电源接口。某些型号的 ATA 还支持不止一部电话机。

硬件电话的优势在于手柄可以为语音通讯提供很好的声学特性。任何质量良好的电话都是被设计为拾取正常的人声频率，滤除不希望的背景噪音，并且对波形整型的。自从电话网络存在以来，人们就使用传统电话，而且，人们一般希望使用已经习惯的东西。硬件电话可以提供与传统电话类似的使用方式与 Asterisk 通讯，这对大多数人来说是有吸引力的。而且，如果使用硬件电话，就不需要你的计算机整天都开着了。

硬件电话的主要缺点是不能便携，并且价格比较高。要知道如今市场上已经出现了不少质量优良而免费的软件电话产品。而且，如果你的办公桌面积有限，你也不会希望再增加任何东西在上面。如果你经常变更办公地点，不能便携的硬件电话或许也不适合你（当然，在每个办公地点各放一部硬件电话也是个解决方案）。

软件电话可以通过把它安装到已有的便携设备上来解决便携性问题，例如安装在你的笔记本电脑或智能手机上。软件电话的低费用（通常有免费版，或者大约 USD$30.0 的付费全功能版本）也是个有吸引力的优点。因为许多软件电话是免费的，所以看起来你的第一部连接 Asterisk 的电话会是软件电话了。因为软件电话也只是一种软件而以，所以它们可以很容易的被安装和升级。而且它们一般还可以利用一些其它外设提供更多特性，例如，通过一个摄像头进行视频通话，或者把你电脑里的一份文档作为传真发送出去。

软件电话的主要缺点包括它不能永远在线，每次打电话时需要先戴上头戴耳机，以及这样一个事实：许多计算机操作系统实际上每天都会随机选择一段时间来完成一些并不是用户要求它们做的任务，而当这些任务对 CPU 资源消耗很大时，往往会引起软件电话不能正常工作。

模拟终端适配器的优点是可以让你把模拟设备接入 SIP 网络，例如无绳电话（在很多情况下，无绳电话依然比 wifi 电话更有优势），寻呼放大器，振铃器等。当 SIP 网络工作不正常时，模拟终端适配器有时也用于自动切换回到旧的 PSTN 网络上。

模拟终端适配器的主要缺点是你无法通过老式的模拟电话设备获得和 SIP 电话机一样的功能特性。毕竟这些模拟电话技术已经存在一个多世纪了。

当我们使用 Asterisk 时，并不需要考虑选择使用软件电话，硬件电话，还是模拟终端适配器。我们完全可能用一个分机号码（extension number）同时呼叫多个设备，例如桌面电话，笔记本电脑上的软件电话，手机，甚至一个用在工厂的闪光灯（在工厂使用闪光灯指示振铃是因为工厂环境太嘈杂了，以至于我们无法听到铃音）。

Asterisk 如今可以令人满意的提供多种与外部世界交互的方法，有些方法在几年前还只能在梦中想想。我们看到，随着社交网络的流行，出现了更多的统一通讯应用。专注于通讯的，例如 Skype；专注于网络服务的，例如 Google；软件终端越来越灵活和流行。语音和应用之间的界限也不断模糊，软件电话非常有利于快速响应这种变化。

尽管如此，我们依然喜欢使用硬件电话。

5.3 配置Asterisk

在本节中，我们将描述如何在 /etc/asterisk/ 目录下创建 *sip.conf* 和 *iax.conf* 配置文件，这两个配置文件是用来定义你系统中的 SIP 设备和 IAX2 设备的参数的。

虽然 Asterisk 允许设备用许多不同的协议工作，但是，SIP 和 IAX2 是其中最流行和最成熟的模块。如果你第一次创建 Asterisk，你最好不要理会其它协议（例如 Skinny/SCCP，Unistim，H.323，以及 MGCP），而首先用好 SIP 和 IXA2。协议的配置是类似的，而且示例配置文件也包含了足够的信息和例子。所以一旦你具备了基础，其它协议也可以非常容易的使用。

对于 channel 配置文件，例如 *sip.conf* 和 *iax.conf* ，包含了 channel driver（例如 chan_iax2.so 或 chan_sip.so ）的配置信息，以及其它电话设备连接及与 Asterisk 通讯所需的必要信息。

关于 channel driver 的一般信息放在配置文件的开头，**[general]** section 中。包括设备名在内的所有的 section 名都被放置在一对方括弧中。在 section 名之后的内容作用在整个 section 范围上。**[general]** section 包含了一些设备配置的默认值，这些默认值可以被具体设备 section 或模板中的配置覆盖。Asterisk 同样有在代码中设计（hardcoded）的默认值，所以除了一些必须配置的配置项外，如果你满意 Asterisk 的默认值的话，可以忽略大部分配置项。

Asterisk 按照如下次序检查配置项：

1. 检查对应 channel 的 section；
2. 检查模板中的 section；
3. 检查**[general]** section；
4. 使用代码中设计（hardcoded）的默认值；

这就意味着，如果你没有配置某个配置项，并不等于你的设备并不能获得这个配置项的取值。如果你不确定默认值是否合适，你最好在 channel 配置文件或相关的模板文件中明确的配置每个配置项的取值。

这个概念十分重要！

5.3.1 Channel配置文件如何与Dialplan工作

尽管我们还没有讨论过 Asterisk 的 Dialplan，但是先看看 channel 配置文件（*sip.conf*，*iax.conf*）和 dialplan（*entensions.conf*）之间的关系也是十分有用的。Dialplan 是 Asterisk 系统的心脏：它控制着呼叫逻辑作用于任意 channel 之间的任意连接的方式，例如当一部设备

拨打分机 101 时或者收到一个来电应该发生什么？相关 channel 的配置文件与 extensions.conf 一起负责了呼叫路由处理。Figure 5-1 为 *sip.conf* 和 *extensions.conf* 之间的关系提供了一个图示的说明。

当一个呼叫进入 Asterisk 后，与之匹配的 channel 配置文件就会被启用（例如，*sip.conf*）。Channel 配置文件同时也负责处理认证以及定义与之相对应的 Dialplan 入口。

一旦 Asterisk 决定了如何处理一个 channel，它就会将呼叫控制传递给 Dialplan 中对应的 section。Channel 配置文件中的 context 配置项用于定义这个 channel 在 dialplan 的入口（Dialplan 中包含了处理及路由一个呼叫所需要的全部信息）。

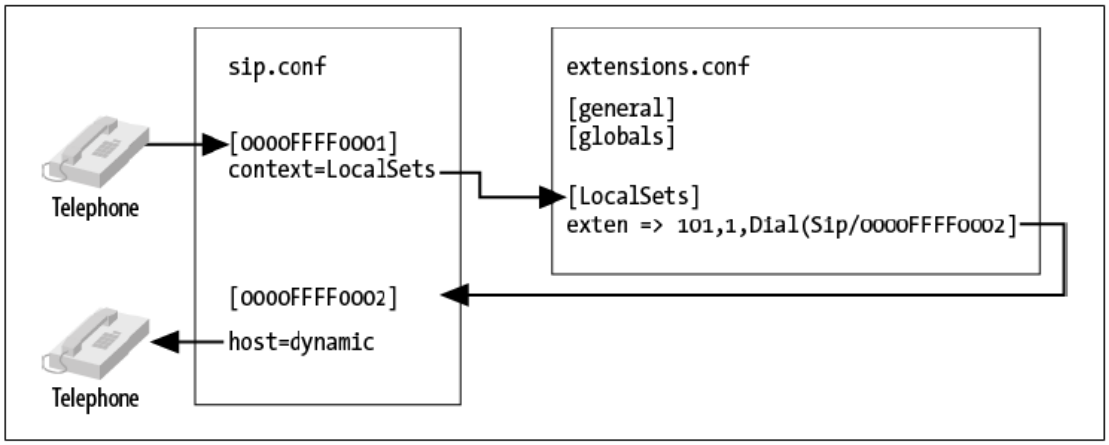

Figure 5-1. Relationship of sip.conf to extensions.conf

与呼叫进入 Dialplan 相反，如果在 dialplan 中已经定义好当分机号码 101 被呼叫时将拨打 0000FFFF0002 设备，那么对应的 channel 配置文件将用于决定如何将这个呼叫转出 dialplan 到网络上的一部电话（包括认证、编码等等）。

需要特别记住的一点是，channel 配置文件不仅控制一个呼叫怎样进入系统，还要负责控制它如何离开系统。举例来说，当一个分机拨叫另一个分机，channle 配置文件不仅用于将呼叫传递给 dialplan，还要负责将呼叫从 dialplan 传递给被叫设备。

5.3.2 Sip.conf

SIP [注2] channel modules 在 Asterisk 所有 modules 中是最成熟和特性最丰富的。这是因为 SIP 协议实在十分流行，它已经成为 VoIP 通讯的事实工业标准，并被应用在成千上万的各种设备和 PBX 上。如果你检查 Asterisk 源码 *./configs* 目录下的 *sip.conf.sample* 文件，有会发现非常非常多的配置项。幸运的是，这些配置项的默认值一般都能适合你的应用，因此你可以非常简单的创建一个能使大多数 SIP 电话机与你的 Asterisk 正常工作的配置文件。

首先，让我们在 */etc/asterisk* 目录下创建一个命名为 *sip.conf* 的配置文件。并将下面的内容粘贴到 *sip.conf* 中：

```
[general]
context=unauthenticated         ; default context for incoming calls
allowguest=no                   ; disable unauthenticated calls
srvlookup=yes                   ; enabled DNS SRV record lookup on outbound calls
udpbindaddr=0.0.0.0             ; listen for UDP requests on all interfaces
tcpenable=no                    ; disable TCP support

[office-phone](!)   ; create a template for our devices
type=friend         ; the channel driver will match on username first, IP second
context=LocalSets   ; this is where calls from the device will enter the dialplan
host=dynamic        ; the device will register with asterisk
nat=yes             ; assume device is behind NAT
                    ; *** NAT stands for Network Address Translation, which allows
                    ; multiple internal devices to share an external IP address.
secret=s3CuR#p@s5   ; a secure password for this device -- DON'T USE THIS PASSWORD!
dtmfmode=auto       ; accept touch-tones from the devices, negotiated automatically
disallow=all        ; reset which voice codecs this device will accept or offer
allow=ulaw          ; which audio codecs to accept from, and request to, the device
allow=alaw          ; in the order we prefer

; define a device name and use the office-phone template
[0000FFFF0001](office-phone)

; define another device name using the same template
[0000FFFF0002](office-phone)
```

打开你刚才创建的 *sip.conf* 文件，我们来逐条研究一下。

我们创建了四个 section，第一个是 **[general]** section。这是个标准 section，它必须出现在所有 channel modules 的配置文件的开头。**[general]** section 包含了定义协议如何工作的通用配置项，以及可以用来定义一些配置项的默认值。

例如，我们定义 context 的默认值为 **unauthenticated** ，以保证不可靠用户的来电可以进入 dialplan。名字 **unauthenticated** 就明显的表明，这个 context 定义的呼叫处理的是“不可靠”的呼叫，因此它们不能拨叫外线到 PSTN 网络（这可能造成金钱损失，或者被用于身份欺骗）。你还应该了解，其实这里的 **unauthenticated** 可以被任何你希望使用的其它名字取代，但是要求在 *extensions.conf* 中有个同名 section 来定义呼叫流程。

下一个配置项是 **allowguest** ，这个参数用于禁止我们收到不可靠的呼叫。需要注意的是，对有些 channels 来说，你可能希望允许不可靠的呼叫。对公司来说，一个常见的需要允许不可靠用户呼叫的例子是：当你希望允许客户通过例如电子邮件这样的 URL 来呼叫你时。如果我们希望允许客户通过它们不可靠的电话呼叫我们的话，我们就需要允许 **allowguest** ，并通过上一段描述的 **unauthenticated** 来定义它的呼叫处理流程。

你也许觉得疑惑你问什么会希望允许不可靠的呼叫。原因在于，假设你把你的 SIP URL 印在你的名片上（例如，sip:leif.madsen@shifteight.org），如果你对不可靠呼叫的处理只是简单挂断的话，人们就无法成功呼叫你的 SIP URL。对不可靠呼叫 context 的正确处理是将来电转接到受控的环境中。你可能希望能接听这些电话而不需要信任他们。

下一个配置项是 **srvlookup**， 它被用于使能或禁止 Asterisk 执行 DNS SRV 查询。这一功能主要被 Asterisk 用于连接外部的服务供应商。我们将在第 12 章对 Asterisk 的 DNS 功能做更深入的探讨。

再下一个配置项是 **udpbindaddr** [注3]，它可以通过设置为一个具体的 IP 地址或 **0.0.0.0** 来告诉 Asterisk 用哪个网络接口侦听 UDP 协议（UDP 协议一般用于承载声音信道）。如果定义为 **0.0.0.0**，就意味着指示 channel driver 侦听所有的网络接口。另一个选择是通过指定具体的 IP 地址从而指定侦听的网络接口。

目前 Asterisk 中的 **updbindaddr** 和 **tcpbindaddr** 配置项只能选择“全部或唯一”。换句话说，假设你的系统有三张网卡，你无法限定 VoIP 流只发生在其中两张网卡上：你只能选择指定 VoIP 工作在一个网卡上，或全部网卡上。

IPv6 在 sip.conf

对于 Asterisk 1.8 来说，Asterisk 对于 SIP 和 RTP 流均支持 IPv6。所有 /etc/asterisk/sip.conf 中相关 IP 地址的配置项都可以接受 IPv4 或 IPv6 地址。下面的以 **updbindaddr** 为例说明：

udpbindaddr value	Description
192.168.100.50	Bind to a specific IPv4 address.
2001:db8::1	Bind to a specific IPv6 address
0.0.0.0	Bind to all IPv4 addresses on the system.
::	Bind to all IPv4 *and* IPv6 addresses.

配置项 **tcpenable** 允许我们通过 TCP 接受 SIP 呼叫请求。现在我们先禁止这一配置项，因为 UDP 方法目前更成熟（也更通用），而且我们现在希望尽量减少你配置的障碍。我们建议，当你已经能够熟练的用 UDP 配置你的设备时，再测试对 TCP 的支持。

还有两个配置项 **tlsenbale** 和 **tlsbindaddr** 用于使能 SIP over TLS （加密 SIP）。我们将在第七章讨论 SIP over TLS 的配置。

跟着我们定义的一个 section 是个模板，我们命名为 **[office-phone](!)** 。由于我们已经把它创建为一个模板，所以我们可以将其中的取值应用到所有的设备上。

在 section 名后面跟随一个 **(!)** 告诉 Asterisk 把这个 section 作为一个模板处理。利用模板我们可以减少为每个设备重复的增加或修改配置项。模板是非常有用的，并且所有的 Asterisk 配置文件都支持模板。如果你希望针对某个设备修改此前在模板里已经定义的某个配置项取值，你只要在这个设备的配置 section 下直接重新设置就可以，新的取值将覆盖模板中的取值。使用模板并不是必须的，但是它的确非常方便并且被广泛的使用。

在 **[office-phone]** 模板中，我们定义了几个用于认证和控制呼叫的配置项。第一个配置项是 **type** ，我们配置它的值是 **friend** 。这是告诉 channel driver 首先尝试匹配名字，然后再尝试匹配 IP 地址。

SIP 配置匹配和 type 配置项

在本章的我们提供的 *sip.conf* 例子中，我们配置 **type=friend**。对于 **type** 来说，还有两个可选值可用：**user** 和 **peer** 。它们的区别在于 Asterisk 如何理解来电的 SIP 请求。基本规则如下表所示：

type =	Description
peer	Match incoming requests to a configuration entry using the source IP address and port number.
user	Match incoming requests to a configuration entry using the username in the From header of the SIP request. This name is matched to a section in *sip.conf* with the same name in square brackets.
friend	This enables matching rules for both peer and user. This is the setting most commonly used for SIP phones.

当一个呼叫请求被 Asterisk 收到并认证后，被请求的电话号码会被 dialplan 中对应于设备配置文件中 context 取值的 section 处理。在我们的例子中，这个 context 的取值是 **LocalSets** 。

当我们需要向某部电话发起呼叫请求时，我们需要用到 **host** 配置项。如果定义这个值为 **dynamic** ，就是告诉 Asterisk 被叫电话会告诉我们它的 IP 地址，而不需要我们指定一个静态地址。如果我们希望指定静态地址，只需要把 **dynamic** 替换为一个类似 **192.168.128.30** 的 IP 地址就可以了。

配置项 **nat** 用于通知 Asterisk 使能某些功能，从而可以使得安装在 NAT 之后的 SIP 电话机间也能正常通话。这一点非常重要，因为 SIP 协议报文中包含有 IP 地址信息。如果一个 SIP 电话机安装在私有网络上，它可能会把它的私有网络地址填写到 SIP 消息中，而这个私有网络地址信息一般是不能直接使用的。

设备的密码定义在 **secret** 参数中。尽管这个配置项不是必须的，但是你需要了解的是，一些讨厌的家伙们运行网络欺骗脚本去搜寻具有不可靠的密码和简单的设备名的 VoIP 账号（例如用 100 作为设备名，用 1234 作为密码）的现象是非常常见的。采用像 MAC 地址这样不那么常用的设备名称，以及不太容易猜到的密码，我们可以明显的降低我们的系统暴露到外部世界的风险。

5.4 加载你的新channel配置

5.4.1 Asterisk CLI

了解你的 Asterisk 系统发生了什么的最好办法是利用 Asterisk CLI。Asterisk CLI 接口提供了几个不同的输出级别以便于你了解你的系统上究竟发生了什么，并且提供了丰富而有用的实用程序以使得你能影响你运行中的 Asterisk 系统。让我们从唤醒你的 Asterisk CLI 并为你的 channel modules 加载配置文件开始：

```
$ sudo asterisk -r
*CLI> module reload chan_sip.so
*CLI> module reload chan_iax2.so
```

检查下你已经安装的新 channel：

```
*CLI> sip show peers
*CLI> sip show users
*CLI> iax2 show peers
*CLI> iax2 show users
```

现在，你的 Asterisk 系统应该已经被配置好可以处理你的电话设备的注册了。直到注册完成，这些电话设备才能打出或接听电话。由于每种电话设备各有不同，详细的讨论电话设备的配置不是本书要讨论的内容。

5.5 测试并确保你的电话设备成功注册

一旦你的电话设备成功注册到 Asterisk，你就可以通过 Asterisk CLI 查询这个电话设备的地址和状态。

关于注册（registration）的一个常见的误解是，认为注册是指一个设备通过认证，从而获得呼出电话的权限。这是错误的看法。认证的唯一作用是允许设备标明它在网络中的地址，因此 Asterisk 就可以知道如何向它发送与其相关的呼叫。

呼出电话的认证是一个完全独立的过程，并且在每次呼叫时都会发生，而不管它是否成功注册。这就意味着你的电话可能能呼出，但不能呼入。这一般发生在设备没有成功注册时（所以 Asterisk 不知道它的网络地址），但它依然可以正确的认证（因此 Asterisk 可以接受它发出的呼叫）。

为了检查设备的注册状态，首先唤醒 Asterisk CLI：

```
$ sudo asterisk -r
```

输入如下命令，则返回 Asterisk 所有已知设备的列表（不管其状态如何）：

```
*CLI> sip show peers
Name/username              Host            Dyn Nat ACL Port     Status
0000FFFF0001/0000FFFF0001  192.168.1.100    D   N      5060     Unmonitored
0000FFFF0002/0000FFFF0002  192.168.1.101    D   N      5060     Unmonitored
```

你可能注意到 **Name/username** 区域并不总能现实设备的全名。这是因为这个区域被限制为 25 个字符。

注意这个例子中的 **Status** 是 **Unmonitored** 。这是因为我们没有在 *sip.conf* 中使用 **qualify=yes** 配置。

5.6 模拟电话

有两种方法可以将模拟电话连接到 Asterisk 系统。一个方法是使用 ATA，这是最通常使用 SIP 协议连接 Asterisk 的方式。为 ATA 做的 Asterisk 配置与为任何基于 SIP 的电话机做的一样。另一个方法是利用诸如 Digium 这样公司生产的电话板卡，直接把模拟电话连接到 Asterisk 服务器上。Digium 销售的电话板卡可以增加到你的服务器上并提供 FXS 接口用于连接模拟电话（或传真机）。出于说明配置的目的，我们将以 Digium AEX440E 卡为例进行说明。AEX440E 是由 AEX410 半长 PCI-e 卡再加上 4 个 FXS 模块组成的，AEX440E 支持硬件回声抵消。

无论你使用那种硬件，请翻阅厂商文档查找详细硬件配置要求。

Asterisk 和 DAHDI 都被安装好了（参考第三章）。注意 DAHDI 必须在你安装 Asterisk 之前安装。当你安装 DAHDI 时，请确保也安装了 *init* 脚本。这是为了保证当系统启

动时你的硬件能被正确的初始化。Init 脚本可以从 *DAHDI-tools* 安装包中安装。

Init 脚本利用 */etc/dahdi/modules* 文件决定哪些 modules 会被加载以支持系统中安装的硬件。Init 脚本的安装会尝试自动配置这个文件，但你应该检查这个文件以确保正确：

```
# Autogenerated by tools/xpp/dahdi_genconf (Dahdi::Config::Gen::Modules) on
# Tue Jul 27 10:31:46 2010
# If you edit this file and execute tools/xpp/dahdi_genconf again,
# your manual changes will be LOST.
wctdm24xxp
```

DAHDI 需要的配置文件还包括：*/etc/dahdi/system.conf* 。这个文件看起来如下所示：

```
# Specify that we would like DAHDI to generate tones that are
# used in the United States.
loadzone = us
defaultzone = us

# We have 4 FXS ports; configure them to use FXO signaling.
fxoks = 1-4
```

以上配置假设在美国使用。关于国际化的说明请参见第九章。

如果你所选用的板卡不支持硬件回声抵消，则需要在 */etc/dahdi/system.conf* 中增加一行以使能软件回声抵消：

```
echocanceller = mg2,1-4
```

MG2 是推荐的回音抵消器，它包含在 DAHDI 发行包的正式版本中。我们还有另外一个开源的回声抵消器可供选择，称为 OSLEC（Open Source Line Echo Canceller）。很多用户都报告说 OSLEC 的效果非常好。更多关于安装 OSLEC 的信息，请参考网页 http://www.rowetel.com/blog/oslec.html

现在，我们利用 init 脚本来安装适当的 modules 并初始化硬件：

```
$ sudo /etc/init.d/dahdi start
Loading DAHDI hardware modules:
  wctdm24xxp:                                [  OK  ]

Running dahdi_cfg:                           [  OK  ]
```

现在，DAHDI 已经被配置好了。是时候进行 Asterisk 的相关配置了。一旦 Asterisk 安装完成，请确保 **chan_dahdi** 已经被安装。如果它没有安装，请检查它是否在 */usr/lib/asterisk/modules/* 下。如果它在那里，编辑 */etc/asterisk/modules.conf* 来加载 chan_dahdi.so。如果 chan_dahdi 在硬盘上不存在，说明在 Asterisk 安装前并没有安装 DAHDI；请立即返回安装 DAHDI（参考第三章）。你可以通过下述命令检查它的存在：

```
*CLI> module show like chan_dahdi.so
Module                         Description                              Use Count
chan_dahdi.so                  DAHDI Telephony Driver                   0
1 modules loaded
```

下一步，你必须配置 */etc/asterisk/chan_dahdi.conf*。这是 **chan_dahdi** 模块的配置文件，是 Asterisk 和 DAHDI 间的接口。它看起来如下：

```
[trunkgroups]

; No trunk groups are needed in this configuration.

[channels]

; The channels context is used when defining channels using the
; older deprecated method.  Don't use this as a section name.

[phone](!)
;
; A template to hold common options for all phones.
;
usecallerid = yes
hidecallerid = no
callwaiting = no
threewaycalling = yes
transfer = yes
echocancel = yes
echotraining = yes
immediate = no
context = LocalSets
signalling = fxo_ks ; Uses FXO signaling for an FXS channel

[phone1](phone)
callerid = "Mark Michelson" <(256)555-1212>
dahdichan = 1

[phone2](phone)
callerid = "David Vossel" <(256)555-2121>
dahdichan = 2

[phone3](phone)
callerid = "Jason Parker" <(256)555-3434>
dahdichan = 3

[phone4](phone)
callerid = "Matthew Nicholson" <(256)555-4343>
dahdichan = 4
```

你可以通过 *dahdi show channel* CLI 命令来检查 Asterisk 已经加载的配置：

```
*CLI> dahdi show channels
   Chan Extension  Context         Language   MOH Interpret    Blocked    State
 pseudo            default                    default                     In Service
      1            LocalSets                  default                     In Service
      2            LocalSets                  default                     In Service
      3            LocalSets                  default                     In Service
      4            LocalSets                  default                     In Service
```

如果要获得指定 channel 的详细信息，你可以运行 *dahdi show channel* **1**

5.7 测试你的设备的基本Dialplan

我们并不打算现在深入讨论 dialplan，但是一个基本的，可以帮助我们测试新注册设备的 dialplan 还是有帮助的。请将下述内容输入到 */etc/asterisk/extensions.conf* 中：

```
[LocalSets]

exten => 100,1,Dial(SIP/0000FFFF0001) ; Replace 0000FFFF0001 with your device name

exten => 101,1,Dial(SIP/0000FFFF0002) ; Replace 0000FFFF0002 with your device name

;
; These will allow you to dial each of the 4 analog phones configured
; in the previous section.
;
exten => 102,1,Dial(DAHDI/1)
exten => 103,1,Dial(DAHDI/2)
exten => 104,1,Dial(DAHDI/3)
exten => 105,1,Dial(DAHDI/4)

exten => 200,1,Answer()
    same => n,Playback(hello-world)
    same => n,Hangup()
```

这个基本的 dialplan 可以允许你通过分机 100 和 101 拨打你之前配置的 SIP 电话机。而之前安装的模拟板卡的四个线路则可以通过分机 102 到 105 分别拨打。如果你拨打分机 200，你会听到一个 **hello-world** 提示音。所有这些分机都可以随意编号，你可以用任何你想用的号码。这并不是一个完整的 dialplan，我们将在后续章节进一步完成它。

你需要重载你的 dialplan 以使修改生效。你可以在 Linux shell 下通过下述命令实现重载：

```
$ sudo asterisk -rx "dialplan reload"
```

或者在 Asterisk CLI 下输入：

```
*CLI> dialplan reload
```

现在，你可以在你新配置的两个分机间拨打电话了。我们可以通过 Asterisk CLI 观察通话过程。你会看到类似下面这样的信息（同时你呼叫的电话应该响铃）：

```
-- Executing [100@LocalSets:1] Dial("SIP/0000FFFF0001-0000000c",
   "SIP/0000FFFF0001") in new stack
-- Called 0000FFFF0001
-- SIP/0000FFFF0001-0000000d is ringing
```

如果这些没有发生， 你就需要重新检查下你的配置文件了，请确保不要敲错字。

5.8 深入观察：你的第一次呼叫

为了帮助你思考 Asterisk 到底是如何工作的，我们将简述下当同一个 Asterisk 系统下的两部分机利用 SIP 协议彼此呼叫时，到底发生了什么。

请记住这里实际发生了两个呼叫：一个是从主叫分机到 Asterisk，另一个是从 Asterisk 到被叫分机。SIP 是一个点到点的协议，并且从 SIP 协议的观点看，就是有两个呼叫在发生。SIP 协议并不知道 Asterisk 在为这两个呼叫进行桥接。每个分机只知道它跟 Asterisk 连接，并不知道另一个分机的情况。这就是 Asterisk 一般被归为 B2BUA（Back to Back User Agent）的原因。这也是为什么 Asterisk 可以非常容易的将不同的协议桥接在一起的原因。

对于你刚才进行的呼叫来说，会话流程如 Figure 5-2 中描述。
关于 SIP 协议如何工作的进一步详细信息，请参考附件 B 和 SIP RFC 文档 http://www.ietf/rfc/rfc3261.txt。

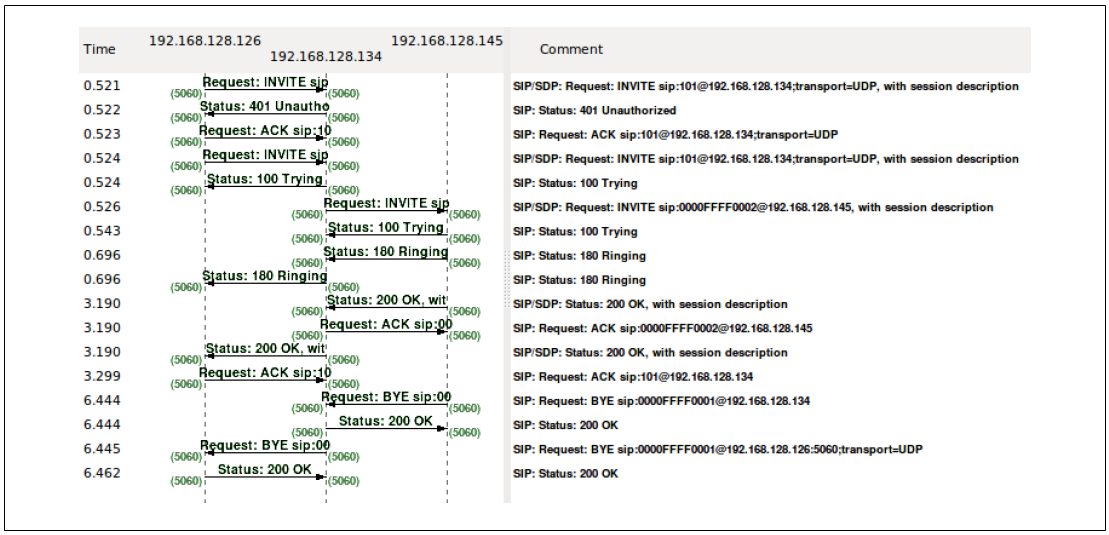

Figure 5-2. SIP dialogs

5.9 结论

在本章中，我们学习了设备命名的最佳实践是分离用户、分机号码，以及设备本身的概念。并且学习了如何在 channel 配置文件中定义配置和认证参数。下一步，我们将深入研究 Asterisk 的神奇之处 —— dialplan，并且观察这一简单的方法如何获得了巨大的成果。

注释：

注 1：实际上，Asterisk 也曾经尝试利用 user.conf 文件实现用户（users）和设备（devices）概念的抽象，然而，这一作法主要是用在 Asterisk GUI 中。用 Dialplan 来实现相关的概念，要更易于理解和更加灵活。

注 2：通读 SIP 的 RFC 文档需要非常长的时间，但是我们可以只读前 25 页。SIP RFC 的前 25 页对 SIP 协议做了很好的介绍。请参考 http://www.itef.org/rfc/rfc3261.txt

注 3：与 udpbindaddr 互补的另一个参数配置项是 tcpbindaddr，用于定义 TCP 侦听的 IP；