第七章 外线连接

目录

在之前的章节里，我们已经介绍了很多重要知识，这些知识是对于使 Asterisk 系统能正产工作是必不可少的。然而，我们还没有完成一件对于任何 PBX 都必不可少的事情：即，将 Asterisk 与外部世界连接起来。在本章中，我们即来讨论这件工作。

之所以说 Asterisk 的软件架构具有重要意义，很大程度上是由于这样一个事实，Asterisk 平等的对待每一个 channel。这一点完全不同于传统 PBX，在传统 PBX 中，外线（用于连接外部世界）和分机（用于连接内部用户和资源）是完全不同的。事实上，Asterisk dialplan 用相似的方法处理所有 channel，这意味着你可以在 Asterisk 系统中非常容易的实现一些在传统 PBX 非常难以实现（或不可能实现）的工作。

然而，这种灵活性也是有代价的。由于系统并不会固有的知道内部资源（例如一部内线分机）和外部资源（例如，一个电信线路）的不同，Asterisk 系统需要你来保证你的 dialplan 恰当的处理了每种类型的资源。

7.1 干线（Trunking）基础

干线（Trunking）的目的是在两个实体间提供一个共享连接。举例来说，公路干线是把两个镇子连接起来的高速公路。铁路上广泛的使用术语“干线（trunk）”，来指代那些将支线铁路连接到一起的主要线路。

同样地，在电信领域，干线用于将两个系统连接起来。电信运营商利用通讯干线把他们的网络连接到一起，而对 PBX 来说，将 PBX 与外部世界连接起来的线路一般就被称作干线（尽管电信运营商一般不会认为这些是干线）。从技术的观点看，PBX 干线的定义并不像其它例子那么清晰（PBX 干线使用了与终端线路完全不同的技术），但作为一个概念，干线仍然是非常重要的。举例来说，在 VoIP 系统中，一切通讯实际上是点对点的（所以从技术的观点看，并不真实的存在干线），但是这个概念仍然是有用的，它可以帮助我们区分连接到外部世界的 VoIP 资源和连接到用户分机（例如一部 SIP 电话机）的 VoIP 资源。

大概最容易的理解方式是把干线理解为一组提供外线路由的线路集合。所以，在 Asterisk PBX 中，你可能有 VoIP 服务商提供长途呼叫的干线，有连接 PSTN 线路的干线，以及将你的不同地点的办公室连接在一起的干线。这些干线可能实际上是通过同一个网络连接，但在你的 dialplan 中，你可以用完全不同的方法处理它们。

尽管我们相信 VoIP 最终将完全取代 PSTN，许多 VoIP 线路使用的概念（例如电话号码）继续存在更多是由于历史原因而不是技术需求，但是我们认为，在我们完全进入 VoIP 时代前，讨论一下在 Asterisk 中如何使用传统 PSTN 线路还是非常有用的。

如果你的 Asterisk 系统计划只使用 VoIP 线路，那也没问题，请直接阅读本章的 VoIP 部分，我们会告诉你应当怎么做。不过我们强烈建议阅读 PSTN 部分，因为其中包含了很多有用的基础知识，虽然这些并不是理解和使用 Asterisk 所必须的。

7.2 用于外线连接的基本 Dialplan

在传统 PBX 中，访问外线一般都需要通过拨打一个号码前缀来实现[注1]。我们一般用数字 9 作为这个前缀。

在 Asterisk 中，同样可以指定 9 作为外线路由前缀，但是由于 Asterisk dialplan 要智能的多，完全没有必要强制用户在呼叫外线前加拨 9。典型地，你会为你的系统配置一个分机范围（如 100－199），以及一个 feature code 范围（*00 到*99）。任何这个范围之外的，匹配了你设定的国家码和区域码的呼叫，都会被视为外线呼叫。

如果你通过一个电信运营商提供全部的外线资源，你可以通过几条简单的样式匹配处理外线呼叫。本节提供的例子是针对 NANP（North American Numbering Plan）的。如果你的国家不在 NANP 中（NANP 服务于加拿大，美国，以及几个加勒比海国家），你可能会需要不同的样式匹配。

在 **[globals]** 部分中包含两个变量，叫做 LOCAL 和 TOLL [注2]。这些变量的目的是当你需要更换电信运营商时简化你的 dialplan 管理。它们允许你只修改 dialplan 的一个地方，但会对所有引用这个 channel 的地方生效：

```
[globals]
LOCAL=DAHDI/Go              ; assuming you have a PSTN card in your system
TOLL=SIP/YourVoipCarrier    ; as defined in sip.conf
```

在 **[external]** 部分中包含实际的 dialplan 代码，这些代码将识别呼叫的号码并将它们传递给 **Dial()** application [注3]:

```
[external]
exten => _NXXNXXXXXX,1,Dial(${LOCAL}/${EXTEN})     ; NANP 的 10 位号码样式匹配
exten => _NXXXXXX,1,Dial(${LOCAL}/${EXTEN})        ; NANP 的 7 位号码样式匹配
exten => _1NXXNXXXXXX,1, Dial(${LOCAL}/${EXTEN})  ; NANP 的长途号码样式匹配
exten => _011.,1, Dial(${TOLL}/${EXTEN})           ; NANP 的国际电话号码样式匹配

; 这部分的功能与上面相同。
; 这是为了喜欢拨外线前先拨“9”的用户设计的
exten => _9NXXNXXXXXX,1, Dial(${LOCAL}/${EXTEN:1})
exten => _9NXXXXXX,1, Dial(${LOCAL}/${EXTEN:1})
exten => _91NXXNXXXXXX,1, Dial(${LOCAL}/${EXTEN:1})
exten => _9011.,1, Dial(${LOCAL}/${EXTEN:1})
```

在任意分机或设备使用的 context 中，你可以用一个 **include=>** 指令来允许访问 **external** context：

```
[LocalSets]
```

include => external

非常重要的一点是，不要在可能处理来电的任何 context 中包含访问外线的代码。风险在于仿冒机器人程序将最终获得访问你外线的方法（你将被这种仿冒机器人程序的常见所吓到）。
我们怎么强调都不过分的是，你一定要保证没有外部资源可以访问你的付费电话线路。

7.3 PSTN 线路

公共交换电话网络（PSTN – Public Switched Telephone Network）已经存在超过一个世纪了。它是许多对我们今天有重大影响的技术的先驱，从 Internet 到 MP3 播放器。

7.3.1 传统 PSTN 干线

固定电话运营商采用两种技术交付电话线路：模拟的和数字的。

7.3.1.1模拟电话

最初的电话系统完全是模拟的。你通过声音发出的音频信号，被用于产生一种运载到通话的另一端的电讯号。这种电讯号具有与原始声音一样的特性。

模拟线路具有一些与其它你可能想连接到 Asterisk 的线路所不同的特性：

- 没有信令通道 —— 大部分信令是机电信号
- 断线监测通常会延迟数秒，而其并不是完全可靠的
- 极少的远端监测（例如，应答监测就不充分）
- 线路差异性，这意味着不同线路的音频特性是不同的，并且需要调试

你希望连接到你的 Asterisk 系统的模拟线路需要连接到 Foreign eXchange Office (FXO)端口。由于标准计算机并没有这种 FXO 端口，所以你必须购买 FXO 板卡并安装到你的系统上以实现连接传统模拟电话线[注4]。

FXO 和 FXS

对任何模拟线路来说，存在两个端点：局端（office 端，典型指 PSTN 的中心局），和终端（station 端，典型指电话，也包括板卡，如 Modem 板卡或 PBX 系统的线路板卡）。

中心局（the central office）负责提供：

- 提供馈电（通常是 48V 直流）
- 提供振铃电压（通常是 90V 交流）
- 产生拨号音
- 检测摘机状态（摘机或挂机）

- 发送补充信令，如主叫号码（Caller ID）

终端（the station）负责：

- 提供振铃器（或者至少能够以某种方法处理铃流电压）
- 提供拨号盘（或者其它发送 DTMF 的方法）
- 提供一个叉簧开关（hook switch）以指示线路的状态

一个 Foreign eXchange (FX) 端口通过它与什么连接而命名，而不是它做什么。所以，举例来说，一个 Foreign eXchange Office (FXO) 端口实际上是终端：它连接到中心局。一个 Foreign eXchange Station (FXS) 端口实际上是提供中心局服务的端口（换句话说，你可以把一个模拟终端插入到 FXS 端口上）。

正是由于这个原因，在 Asterisk 配置文件里的信令设置好像是反的：FXO 端口使用 FXS 信令；FXS 端口使用 FXO 信令。当你理解了物理端口类型的命名是基于它与什么连接时，在 Asterisk 中的信令名就更加容易理解了：由于 FXO 端口连接到中心局，所以它需要表现的像个终端，因此需要 FXS 信令。

注意，将 FXO 端口换成 FXS 端口，并不是你简单的做一些配置就可以实现的。FXS 端口和 FXO 端口需要完全不同的电路。大部分支持 Asterisk 的模拟板卡都支持通过安装在主板卡上的子卡来提供正确的 channel 类型，这就意味着你在定义你板卡的端口类型时具备了一些灵活性。

模拟端口通常不用于中型和大型系统。它们通常用于小型办公室（小于 10 根外线；少于 30 部电话）。你决定使用模拟线路可能是基于如下因素：

- 你所在区域的数字线路有限
- 费用（模拟线路在低密度时较便宜，但在高密度时较贵）
- 后勤因素（如果你已经安装的模拟线路，你可能希望保留它们）

从技术的观点看，你一般应当更倾向于选择数字线路而不是模拟线路。但是，现实并不总是顺应技术的发展，所以模拟线路还会继续存在一些年。

7.3.1.2数字电话

数字电话技术被开发出来是为了克服模拟电话的一些限制。数字线路的一些优点包括：

- 经过长距离后没有幅度衰减
- 减少了线路噪音（尤其是长途线路）
- 在一根线路上承载多个通话的能力
- 更快的呼叫建立和拆除速度
- 更丰富的信令信息（尤其是使用 ISDN 时）
- 对电信运营商而言费用更低
- 对用户而言费用更低（在高密度情况下）

在 Asterisk 系统中（或任何 PBX 系统），有如下几种数字线路你可能会用到：

T1（24 channels）

在加拿大和美国使用（大多数用于 ISDN-PRI）

E1（32 channels）

在世界上其它地区使用（ISDN-PRI 或 MFC/R2）

BRI（2 channels）

用于 ISDN-BRI 线路（Euro-ISDN）

注意，物理线路可以进一步通过运行在其上的协议来定义。举例来说，T1 线路可以被用于 ISDN-PRI，或者 CAS；而 E1 线路可以被用于 ISDN-PRI，CAS，或者 MFC/R2。我们将在下一节讨论这些不同的协议。

7.3.2 安装 PSTN 干线（PSTN Trunks）

基于你已经安装的硬件的不同，安装 PSTN 板卡的过程也会不同。我们将按照一般情况讨论安装，这将适合于所有的 Digium PSTN 板卡。其它制造商也会提供基于他们硬件的安装脚本，这将自动处理你的大部分工作。

7.3.2.1下载和安装 DAHDI

DAHDI 接口是 Digium Asterisk Hardware Device Interface 的缩写，DAHDI 定义了 PSTN 板卡和 Asterisk 之间通信的软件框架。即使你没有任何 PSTN 硬件，我也推荐你安装 DAHDI，因为这是获得一个有效时钟源的简单、可靠的方法[注 5]。完整的 DAHDI 安装指导可以在第 3 章找到。

禁止加载额外的 DAHDI Modules

DAHDI 默认会加载所有编译好的 modules 到内存。由于这不是必须的，让我们从现在开始禁止加载任何硬件 modules。如果在配置文件中没有任何 module 被加载，DAHDI 就会加载 **dahdi_dummy** 驱动，这个驱动为 Asterisk 提供了一个从内核获得时钟的接口，从而使诸如 MeetMe 和 IAX2 这样时钟依赖的 modules 可以正常工作。

从 DAHDI2.3.0 起，通过加载 dahdi_dummy 获得时钟的要求不再存在。相关的功能现在集成到 dahdi kernel module 中去了。

定义 DAHDI 会加载哪些 modules 的配置文件位于*/etc/dahdi/modules*。为了禁止加载额外的 modules，我们需要做的事就是编辑 modules 文件并通过在每一行前面加＃号注释掉所有的 modules。当你完成了这一点，你的 modules 配置文件应当看起来如下：

```
# Contains the list of modules to be loaded / unloaded by /etc/init.d/dahdi.
#
# NOTE: Please add/edit /etc/modprobe.d/dahdi or /etc/modprobe.conf if you
# would like to add any module parameters.
#
# Format of this file: list of modules, each in its own line.
# Anything after a '#' is ignored, likewise trailing and leading
# whitespace and empty lines.
# Digium TE205P/TE207P/TE210P/TE212P: PCI dual-port T1/E1/J1
# Digium TE405P/TE407P/TE410P/TE412P: PCI quad-port T1/E1/J1
# Digium TE220: PCI-Express dual-port T1/E1/J1
# Digium TE420: PCI-Express quad-port T1/E1/J1
#wct4xxp
# Digium TE120P: PCI single-port T1/E1/J1
# Digium TE121: PCI-Express single-port T1/E1/J1
# Digium TE122: PCI single-port T1/E1/J1
#wcte12xp
# Digium T100P: PCI single-port T1
# Digium E100P: PCI single-port E1
#wct1xxp
# Digium TE110P: PCI single-port T1/E1/J1
#wcte11xp
# Digium TDM2400P/AEX2400: up to 24 analog ports
# Digium TDM800P/AEX800: up to 8 analog ports
# Digium TDM410P/AEX410: up to 4 analog ports
#wctdm24xxp
# X100P - Single port FXO interface
# X101P - Single port FXO interface
#wcfxo
# Digium TDM400P: up to 4 analog ports
#wctdm
# Digium B410P: 4 NT/TE BRI ports
#wcb4xxp
# Digium TC400B: G729 / G723 Transcoding Engine
#wctc4xxp
# Xorcom Astribank Devices
#xpp_usb
```

你也可以利用 *dahdi_genconf modules* 来生成适当的空配置文件。*dahdi_genconf application* 会搜索你的系统硬件，如果什么都没有找到，会创建一个不会加载任何硬件 modules 的 *modules* 文件。

然后你可以通过重启你的 DAHDI 来卸载任何已经加载的现有驱动，并且只在初始化脚本中加载 **dahdi_dummy** modules：

```
$ sudo /etc/init.d/dahdi restart
Unloading DAHDI hardware modules: done
Loading DAHDI hardware modules:
No hardware timing source found in /proc/dahdi, loading dahdi_dummy
Running dahdi_cfg: [ OK ]
```

在你可以使用你的硬件之前，你还需要配置 */etc/dahdi/system.conf* 文件；这一过程在本章下两节描述。

7.3.2.2配置数字线路

电信运营商开发数字电话技术是为了减少长途线路费用，同时提升传输质量。整个 PSTN 骨干网已经完全实现数字化多年了。数字线路的关键是实现声音信号的数字化。不过数字干线也会允许更复杂和更可靠的信令。几个标准被不断开发出来，对每个标准而言，只是有部分的区别。

你可以使用 dahai_hardware 和 lsdahdi 帮助你确定你的系统包含那些电话技术硬件。你也可以使用 dahdi-genconf modules 来产生一个 /etc/asterisk/modules 文件，该文件是基于找到的硬件产生的。

7.3.2.2.1 PRI ISDN

Primary Rate Interface ISDN（一般称为 PRI）是一个设计为主要运行于 DS1（T1 或 E1）线路上的，用于运营商与客户之间的协议。PRI 使用一个 DS0 信道作为信令通道（称为 D 信道）。一个典型的 PRI 信道是划分为一组 B 信道（实际承载呼叫的承载信道），和一个用于信令的 D 信道。尽管最常见的作法是 PRI 线路承载在单一的物理线路上（例如 T1 和 E1），但是把 PRI 线路分散在多个 DS1 线路上也是可行的，甚至于有多个 D 信道[注6]。

尽管有许多不同的配置 PRI 的方法，我们希望避免这些选择把你弄糊涂（许多方法已经过时了或至少不再通用），并且提供了一些通用配置方法的例子。

当安装电话板卡硬件时，请确保升级了 */etc/dahdi/modules* 文件以使能与你的硬件适应的 *modules*，然后通过初始化脚本（*/etc/init.d/dahdi*）重载 DAHDI。你也可以使用 *dahdi_genconf modules* 命令生成 *modules* 文件。

大部分用于北美的 PRI 线路都使用具有下列特性的 T1 线路：

- Line code: B8ZS (bipolar with 9-zero substitution)
- Framing: ESF (extended superframe)

你还需要配置两个文件。其中 /etc/dahdi/system.conf 应按如下配置：

```
loadzone = us
defaultzone = us

span = 1,1,0,esf,n8zs
bchan = 1-23
enchocanceller = mg2,1-23
hardhdlc =24
```

另一个 */etc/asterisk/chan_dahdi.conf* 应按如下配置：

```
[trunkgroups]
```

```
[channels]

usecallerid = yes
hidecallerid = no
callwaiting = yes
callwaitingcallerid = yes
threeewaycalling = yes
transfer = yes
canpark = yes
cancallforward = yes
callreturn = yes
echocancel = yes
echocancelwhenbridge = yes
relaxdemf = yes
rxgain = 0.0
txgain = 0.0
group = 1
callgroup = 1
pickgroup = 1
pickupgroup = 1
immediate = no

switchtype = national ; commonly referenced to as NI2
context = from-pstn
group = 0
echocancel = yes
signaling = pri_cpe
channel => 1-23
```

某些运营商采用 Nortel 的 DMS 交换机，这种交换机一般采用 DMS100 协议而不是标准的 ISDN 协议。在这种情况下，你需要把 **switchtype** 设置为 **DMS100**：

```
switchtype = dms100
```

在美国和加拿大之外，PRI 线路主要承载在 E1 线路上。
在欧洲，用于 PRI 的 E1 线路一般具有如下特性：

- Line code: CCS
- Framing: HDB3 (high-density bipolar)

并且 */etc/asterisk/chan_dahdi.conf* 应当看起来如下：

```
[trunkgroups]

[channels]
usecallerid = yes
```

```
hidecallerid = no
callwaiting = yes
usecallingpres = yes
callwaitingcallerid = yes
threewaycalling = yes
transfer = yes
canpark = yes
cancallforward = yes
callreturn = yes
echocancel = yes
echocancelwhenbridged = yes
relaxdtmf = yes
rxgain = 0.0
txgain = 0.0
group = 1
callgroup = 1
pickupgroup = 1
immediate = no
switchtype = qsig
context = pri_incoming
group = 0
signalling = pri_cpe
channel => 1-15,17-31
```

7.3.2.2.2 BRI ISDN

Basic Rate Interface ISDN（通常称作 BRI，或者直接称作 ISDN）可以看作 PRI 的一个小兄弟。BRI 只提供 2 个 64Kbps 的 B 通道和一个 16Kbps 的 D 通道。BRI 在北美地区很少使用（我们不推荐使用 BRI，不管出于什么原因），但是在一些欧洲国家 BRI 广泛应用，甚至已经基本取代了模拟线路。

在 Asterisk 下的 BRI 支持主要取决于你安装了何种 BRI 板卡。BRI 板卡的厂商会提供专门的安装指导。

当安装电话板卡硬件时，请确保升级了 */etc/dahdi/modules* 文件以使能与你的硬件适应的 *modules*，然后通过初始化脚本（*/etc/init.d/dahdi*）重载 DAHDI。你也可以使用 *dahdi_genconf modules* 命令生成 *modules* 文件。

7.3.2.2.3 MFC/R2

MFC/R2 协议可以被认为是 ISDN 协议的先驱。它首先用在模拟线路上，但是现在也大量应用在 E1 线路上，E1 线路也用于承载 ISDN-BRI。MFC/R2 协议在加拿大、美国和西欧

并不常见，但是它在某些地区非常流行（特别是拉丁美洲和亚洲），主要是因为它适合于运营商提供较为廉价的服务。

这个协议有许多不同的风格，在每个国家都由不同的变体。

OpenR2 项目提供了 **libopenr2** 库，为了你的 Asterisk 系统能支持 R2 线路，你需要安装这个库。在安装 **libopenr2** 之前，你需要完成 DAHDI 的安装。

编译和安装的顺序是：

1. DAHDI
2. Libopenr2
3. Asterisk

一旦安装了 OpenR2 ，你就可以使用 r2test application 来查看一下支持的变体列表：

```
$ r2test -l
Variant                 Code Country
AR                      Argentina
BR                      Brazil
CN                      China
CZ                      Czech Republic
CO                      Colombia
EC                      Ecuador
ITU                     International Telecommunication Union
MX                      Mexico
PH                      Philippines
VE                      Venezuela
```

关于更多在 Asterisk 中配置 R2 的信息，请参阅 Asterisk 源码目录下（搜索 “mfcr2”）的 *configs/chan_dahdi.conf.sample* 。

此外，OpenR2 包含一些在不同国家实现 Asterisk 与网络连接的配置例子文件。如果需要阅读关于不同国家变体的信息，搜索 /doc/asterisk 目录并查看相应目录下的文档：

```
$ ls doc/asterisk/
ar   br   ec   mx   ve
```

作为一个例子，OpenR2 提供了一个用于在墨西哥连接 Telmex 或 Axtel 的例子配置文件。我们逐步展示这个例子。首先，你必须通过修改 */etc/dahdi/system.conf* 来配置 DAHDI，如下所示：

```
loadzone = us
defaultzone = us
span = 1,1,0,cas,hdb3
cas = 1-15:1101
cas = 17-31:1101
span = 2,1,0,cas,hdb3
cas = 32-46:1101
cas = 48-62:1101
```

下一步，你必须通过修改 */etc/astereisk/chan_dahdi.conf* 来配置 Asterisk，如下所示：

```
signalling = mfcr2
mfcr2_variant = mx
mfcr2_get_ani_first = no
mfcr2_max_ani = 10
mfcr2_max_dnis = 4
mfcr2_category = national_subscriber
mfcr2_mfback_timeout = -1
mfcr2_metering_pulse_timeout = -1
; this is for debugging purposes
mfcr2_logdir = log
mfcr2_logging = all
; end debugging configuration
channel => 1-15
channel => 17-31
```

7.3.2.3配置模拟线路

有许多公司生产用于 Asterisk 的 PSTN 板卡。这些板卡需要安装驱动程序，这样 Linux 才可以识别它们（DAHDI 包含了 Digium 的板卡驱动）。从这一点看，配置是被 Asterisk module **chan_dahdi** 处理的。

你可以使用 dahdi_hardware 和 lsdahdi 来确定你的系统包含哪些电话板卡硬件。

当安装电话板卡硬件时，请确保升级了 */etc/dahdi/modules* 文件以使能与你的硬件适应的 *modules*，然后通过初始化脚本（*/etc/init.d/dahdi*）重载 DAHDI。你也可以使用 *dahdi_genconf modules* 命令生成 *modules* 文件。

为了配置一张 FXO 板卡能够在 Asterisk 上工作，需要两个文件。

第一个文件并不是 Asterisk 配置文件，而是位于 */etc/dahdi* 目录下的文件[注7]。这个文件，*system.conf*，允许你定义一些基本参数，同时指定在系统中有效的 channels。我们的例子假设是一张有 4 个 FXO 端口的板卡，但是其它不同的组合也是可以的，这取决于你的硬件。

```
loadzone = us          ; tonezone defines sounds the interface must produce
                       ; (dialtone, busy signal, ringback, etc.)
defaultzone = us       ; define a default tonezone
fxsks = 1-4            ; which channels on the card will have these parameters
```

一旦你的板卡和 Channels 被系统识别，你还必须通过文件 */etc/asterisk/chan_dahdi.conf* 在 Asterisk 中配置它们：

```
[channels]
;
```

```
; To apply other options to these channels, put them before "channel".
;
signalling = fxs_ks       ; in Asterisk, FXO channels use FXS signaling
                          ; (and yes, FXS channels use FXO signaling)
channel => 1-4            ; apply all the previously defined settings to this channel
```

在这个例子中，我们告诉 Asterisk，系统中的头 4 个 DAHDI channels 是 FXO 端口。

7.3.2.3.1 The s extension

如果你使用模拟 channels 连接 PSTN，那么我们需要解释下 extension **s** 。当一个来电进入一个 context，但是没有指定目的 extension 时（例如，通过 PSTN 网络振铃一个 FXO 端口），它会被传递给 **s** extension。（这里 s 的意思是“start”，这个 extension 是当没有携带 extension 信息的来电的起始入口）。这个 extension 在 dialplan 中实现从一个部分到另一个部分的跳转时也十分有用。举例来说，如果我们对一系列 DID 号码的处理都是跳转到同一个地方，我们就可以将它们都跳转到 **s** extension 处理，而不是为每一个 DID 拷贝同样的 dialplan 代码。

因为这实在是我们的 dialplan 中需要的，让我们把它添加进去。我们将对来电执行三个动作（应答，播放一个声音文件，然后挂机），所以我们的 **s** extension 将需要 3 个 priorities。我们把这 3 个 priorities 放在**[incoming]**下面，因为我们决定所有的来电都从这个 context 开始[注8]：

```
[incoming]
exten => s,1,Answer()
     same => n,Playback(tt-weasels)
     same => n,Hangup()
```

显然的，你一般不会想应答一个电话然后挂掉。典型的，一个来电要么被一个自动应答机应答，要么直接振铃一部电话（或一组电话）。

7.4 VoIP

在电信世界里，VoIP 还是相对比较新的概念。在上个世纪或 VoIP 出现之前，把你和 PSTN 网络联系起来的唯一办法是利用本地电话公司提供的专用线路。现在，VoIP 允许你把两个用户联系起来而根本不引入 PSTN（尽管在大部分 VoIP 场景，仍旧存在 PSTN 元素，特别是在使用传统 E.164 号码的场合）。

7.4.1 PSTN 终结（termination）

在 VoIP 完全取代 PSTN 之前，我们都会需要将呼叫从 VoIP 网络连接到 PSTN 网络。这个过程称为终结（termination）。这意思是说，在 VoIP 网络上的某些点，连接 PSTN 的网关需要从 VoIP 网络接收呼叫并把它们连接到 PSTN 网络上。从 PSTN 网络的角度看，这些呼叫看起来就是从“终结点”发起的。

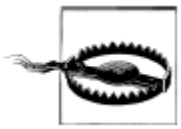
Asterisk 可以被用作终结设备。事实上，由于 Asterisk 非常方便处理协议转换，用作终结设备是 Asterisk 系统一种非常适合的应用。

为了提供 PSTN 终结，Asterisk 设备需要能够处理所有连接的 PSTN 网络的协议。一般来说，这意味着你的 Asterisk 设备需要 PRI 线路来处理 PSTN 连接，以及 SIP channels 来处理来自 VoIP 网络的呼叫。基本原理是相同的，无论你是运行一个为完全部署 VoIP 电话的办公室提供 PSTN trunks 的小型系统，还是运行一个在关键地点部署网关设备并为数千个用户提供 PSTN 终结服务的复杂网络。

来自 VoIP 网络的呼叫会进入 Dialplan 中为 SIP 来电指定的 context，并且 dialplan 会将呼叫中继到 PSTN 接口。这非常简单，支持 PSTN 终结的部分 dialplan 代码如下：

```
[from-voip-network]
exten => _X.,1,Verbose(2, Call from VoIP network to ${EXTEN})
    same => n,Dial(DAHDI/g0/${EXTEN})
```

在现实中，你一般需要处理更复杂的路由计划，需要考虑诸如地理位置、合作政策、费用、可用的资源等等。



由于大部分 PSTN 线路都允许你拨叫世界上的任意位置的任意号码，而且你都需要为已发生的费用买单，所以加强 PSTN 终结网关的安全性是怎么强调其重要性都不过分的事情。犯罪分子花费了很多努力去破解电话系统（特别是没有很好设置安全性的 Asterisk 系统），如果你没有很小心的关注安全性的各个方面，那么你很容易成为话费欺诈的牺牲品。这只是时间早晚问题而已。

千万不要允许任何不安全的 VoIP 连接访问包含 PSTN 终结的 context。

7.4.2 PSTN 再生

显然的，如果你希望将呼叫从 VoIP 网络转到 PSTN 网络，那么你可以也希望可以从 PSTN 接受呼叫到你的 VoIP 网络。这个过程一般称作再生（*origination*）。这简单的指出呼叫是从 PSTN 网络产生的。

为了实现再生，电话号码是必须的。因此你需要从本地电话公司获得用于连接到你的 Asterisk 系统的线路。根据你所在国家和地区的不同，有几种不同的线路可以提供用于 PSTN 再生，从最基本的 POTS 线路到运营商级的 SS7 线路。

用于 PSTN 再生目的的电话号码一般称为直接拔入号码（DIDs）。这种叫法在某些场合并不严谨（例如，传统模拟线路的号码就不能被认为是 DID），但是这个术语这么用已经非常流行了。从历史上说，DID 专指连接到客户端设备（CPE）的干线关联的号码。

由于电话号码是被传统电信业控制的，你要么从运营商直接获取电话号码，要么从那些大批购入号码再拆成较小的块转卖的公司处获得。如果你获得了诸如 PRI 这样的线路，你一般可以购买与这个线路一起交付的 DID 号码。

为了接受来自你用于 PSTN 再生的线路的呼叫，你一般需要能处理被叫号码。这是因为 PSTN 干线一般不止一个号码，并且运营商需要指出那个号码是被叫，这样你的 Asterisk 系

统才知道如何路由这个呼叫。这个被叫号码一般称作被叫号码识别服务（DINS，Dialed Number Identification Service）号码。DINS 号码和 DID 号码不一定必须一致[注9]，但一般它们是一致的。当你向运营商购买线路时，你需要要求它们发送 DNIS（如果他们不理解这个要求，你可能需要考虑其它运营商了）。

在 Dialplan 中，处理呼入线路的 context 需要知道如何处理呼入号码。如下例所示：

```
[from-pstn]
; This is the context that would be listed in the config file
; for the circuit (i.e. chan_dahdi.conf)

exten => _X.,1,Verbose(2,Incoming call to ${EXTEN})
        same => n,Goto(number-mapping,${EXTEN},1)

[number-mapping]
; This context is not strictly required, but will make it easier
; to keep track of your DIDs in a single location in your dialplan.
; From here you can pass the call to another part of the dialplan
; where the actual dialplan work will take place.

exten => 4165551234,1,Dial(SIP/0000FFFF0001)
exten => 4165554321,1,Goto(autoattendant-context,start,1)
exten => 4165559876,1,VoiceMailMain()          ; a handy back door for listening
                                               ; to voice messages
exten => i,1,Verbose(2,Incoming call to invalid number)
```

在 **number-mapping** context 中，明白的列出了所有需要处理的 DID 号码，再加上对未列出的 DID 号码的无效号码处理句柄（你可以把无效号码转给前台，或者自动应答机，或者任意播放一段无效提示的 context）。

7.4.3 VoIP to VoIP

最终，对 PSTN 的需要可能消失，大部分语音通讯将通过网络进行。

隐藏在 SIP 协议后的原始思想是它曾经是一个点到点（Peer-to-Peer）协议。从技术上讲，它今日仍旧是。然而，事情现在已经变得非常复杂了。诸如安全、隐私、公司政策、一体化、集中管理等等把事情变得非常复杂，远远超过了简单的在一部 SIP 话机里输入一个 URL 而其它地方的另一部 SIP 话机就会振铃响应的应用。

SIP 协议变得膨胀而复杂。实现一套基于 SIP 的系统和网络变得比实现传统电话 PBX 网络复杂的多。[注10]

我们并不打算在本书中讨论复杂的 VOIP 系统的设计和实现问题，但是我们会讨论一些将 Asterisk 系统配置为支持 VOIP 系统之间互联的方法。

7.4.4 配置 VoIP 干线（VoIP Trunks）

在 Asterisk 中，并不需要单独安装 VoIP 模块（除非由于某种原因，你没有编译 Asterisk 要求的模块）。有几种不同的 VoIP 协议都可以用于 Asterisk，但我们将集中讨论两种最流行的：SIP 和 IAX。

7.4.4.1在 Asterisk 系统之间配置 SIP 干线（SIP Trunks）

SIP 是最流行的 VoIP 协议——它是如此流行以至于许多人以为其它的 VoIP 协议已经作废了（它们并没有作废，但是这并不能否认 SIP 已经统治 VoIP 领域许多年了）。

SIP 协议是点对点（peer-to-peer）协议，而且并没有正式的干线特性。这意思是说，无论你将一部 SIP 话机连接到你的 SIP 服务器，还是把两个 SIP 服务器连接起来，对 SIP 连接来说都是一样的。

7.4.4.1.1 通过 SIP 连接两个 Asterisk 系统

将两个 Asterisk 系统连接起来，并允许在它们彼此间呼叫的需要是很普遍的需求。比如你有一家公司有两个办公地点并在每个地点部署了一台 PBX，或者比如你是一家公司的 PBX 管理员，而你非常喜欢 Asterisk 并在家里也架设了一个。本节对配置两个 Asterisk 服务器能够通过 SIP 互通提供了一个快速指导。在我们的例子中，我们将这两个服务器称为 **serverA** 和 **serverB** 。

首先必须修改的文件是 */etc/asterisk/sip.conf* 。这是配置 SIP 账号的主要配置文件。首先，如下内容必须被添加到 serverA 的 sip.conf 中。它定义了对端服务器需要使用的 SIP 账号：

```
[serverB]
;
; Specify the SIP account type as 'peer'. This means that incoming
; calls will be matched on IP address and port number. So, when Asterisk
; receives a call from 192.168.1.102 and the standard SIP port of 5060,
; it will match this entry in sip.conf. It will then request authentication
; and expect the password to match the 'secret' specified here.
;
type = peer
;
; This is the IP address for the remote box (serverB). This option can also
; be provided a hostname.
;
host = 192.168.1.102
;
; When we send calls to this SIP peer and must provide authentication,
```

```
; we use 'serverA' as our username.
;
username = serverA
;
; This is the shared secret with serverB. It will be used as the password
; when either receiving a call from serverB, or sending a call to serverB.
;
secret = apples
;
; When receiving a call from serverB, match it against extensions
; in the 'incoming' context of extensions.conf.
;
context = incoming
;
; Start by clearing out the list of allowed codecs.
;
disallow = all
;
; Only allow the ulaw codec.
;
allow= ulaw
```

请注意修改 host 选项以匹配你自己系统配置的 IP 地址

然后将下述代码输入到 **serverB** 的 */etc/asterisk/sip.conf* 中。它差不多和我们在 serverA 中的代码一样，除了账户名称和 IP 地址不同：

```
[serverA]
type = peer
host = 192.168.1.101
username = serverB
secret = apples
context = incoming
disallow = all
allow = ulaw
```

现在你应该通过 Asterisk CLI 命令验证一下这些配置是否已经成功载入 Asterisk 了。首先可以试一下 *sip show peers* 命令。如同这个命令名字所暗示的，它可以显示所有已经配置的 SIP 账号：

```
*CLI> sip show peers
Name/username     Host            Dyn  Forcerport  ACL     Port     Status
serverB/serverA   192.168.1.101                    5060     Unmonitored
1 sip peers [Monitored: 0 online, 0 offline Unmonitored: 1 online, 0 offline]
```

你也可以在 serverB 上试一下 sip show peers 命令。这个命令可以显示很多细节。

在两个 Asterisk 服务器之间配置 SIP 呼叫的最后一步是修改 */etc/asterisk/extensions.conf* 的 dialplan 文件。举例来说，如果你希望在 serverA 上向 6000 到 6999 发起的呼叫都被传递给 serverB，你需要在 serverA 的 dialplan 中增加如下一行：

```
exten => _6XXX,1,Dial(SIP/${EXTEN}@serverB)
```

7.4.4.1.2 将 Asterisk 系统连接到一个 SIP 服务提供商

如果你注册了一个 SIP 服务提供商，你或许已经获得了拨出或接受电话呼叫的服务。这种配置会根据你选择服务提供商的不同而稍有不同。理想情况下，你注册的 SIP 服务提供商会提供 Asterisk 的配置示例以帮助你尽快联通系统。如果他们没有提供，那么，我们将尝试给你提供一些通用配置以帮助你开始。

如果你能从你的 SIP 服务提供商收到呼叫，那么服务提供商一般会要求你的服务器能够注册到他们的服务器上。为了做到这一点，你必须在 */etc/asterisk/sip.conf* 的 **[general]** 部分中增加注册行：

```
[general]
...
register => username:password@your.provider.tld
...
```

下一步，你需要在 sip.conf 中为你的服务提供商创建一个入口点。如下例所示：

```
[myprovider]
type = peer
host = your.provider.tld
username = username
secret = password
; Most providers won't authenticate when they send calls to you,
; so you need this line to just accept their calls.
insecure = invite
dtmfmode = rfc2833
disallow = all
allow = ulaw
```

现在已经完成了账号定义，你还必须在 dialplan 中增加几行以允许你向服务提供商发起呼叫：

```
exten => _1NXXNXXXXXX,1,Dial(SIP/${EXTEN}@myprovider)
```

7.4.4.1.3 加密 SIP 呼叫

Asterisk 支持 TLS 加密 SIP 信令，以及 SRTP 加密媒体流。在本节中，我们将在两台 Asterisk 之间利用 SIP TLS 和 SRTP 建立呼叫。第一步是确保所有依赖软件都已经安装好了。确保你已经安装了 OpenSSL 和 LibSRTP。如果任何一个没有安装，请先安装这些依赖软件，然后再重装 Asterisk 以确保 Asterisk 已经包含了对 TLS 和 SRTP 的支持。一旦完成，请确保 res_srtp modules 已经被编译并安装。要安装 OpenSSL，CentOS 下的安装包是 **openssl-devel** ，Ubuntu 下的安装包是 **libssl-dev**。要安装 LibSRTP，CentOS 下的安装包是 **libsrtp-devel**，Ubuntu 下的安装包是 **libsrtp0-dev**。

下一步我们将配置 SIP TLS。你必须通过双方服务器上 */etc/asterisk/sip.conf* 中**[general]** 部分中的 **tlsenable** 来使能 TLS。你可以指定一个 IP 地址来绑定 TLS 的侦听 IP。这个例子中，我们采用 IPv6 的通配符地址，意思是允许 TLS 连接到系统的所有 IPv4 和 IPv6 地址上：

```
[general]
tlsenable = yes
tlsbindaddr = ::
```

再下一步就是获取证书了。为了验证配置和功能的目的，我们打算利用随 Asterisk 发布的 helper 脚本生成自签发（self-signed）证书。如果你是在真实环境中使用，你可能不希望使用自签发证书。可是，如果你想做，有不少应用程序可以帮助你很容易的管理你自己的认证授权（CA），比如 TinyCA。

我们需要用到的脚本是 *ast_tls_cert*，它在 Asterisk 源文件的 contrib./scripts/directory 目录下。我们需要生成一个 CA 证书和两个服务器证书。首先利用 *ast_tls_cert* 生成 CA 证书和 **serverA** 的服务器证书。然后再利用 *ast_tls_cert* 生成 **serverB** 的服务器证书：

```
$ cd contrib/scripts
$ mkdir certs
$ ./ast_tls_cert -d certs -C serverA -o serverA
$ ./ast_tls_cert -d certs -C serverB -o serverB -c certs/ca.crt -k certs/ca.key
$ ls certs
ca.cfg ca.crt ca.key serverA.crt serverA.csr serverA.key serverA.pem
serverB.crt serverB.csr serverB.key serverB.pem tmp.cfg
```

现在证书已经创建好了，它们需要被移动到 **serverA** 和 **serverB** 上的适当位置。我们将使用 /var/lib/asterisk/keys/ 目录来存放证书。将下列文件移动到 **serverA**：

- ca.crt
- serverA.pem

将下列文件移动到 **serverB**：

- ca.crt
- server.pem

将认证文件准备好后，我们就可以完成 Asterisk 的配置了。我们需要告诉 Asterisk 我们刚刚创建的服务器证书。虽然我们采用的是自签发证书，我们仍然需要指出证书的位置。在 **serverA** 中 /etc/asterisk/sip.conf 的**[general]**部分中，增加如下内容：

```
[general]
tlscertfile = /var/lib/asterisk/keys/serverA.pem
tlscafile = /var/lib/asterisk/keys/ca.crt
```

在 **serverB** 做同样的修改：

```
[general]
tlscertfile = /var/lib/asterisk/keys/serverB.pem
tlscafile = /var/lib/asterisk/keys/ca.crt
```

当你创建服务器证书时，Common Name 字段必须与服务器主机的 hostname 一致。如果你采用 *ast_tls_cert* 脚本，这可以通过-C 选项实现。如果呼叫时发生了服务器证书验证问题，你可能需要修复 Common Name 字段。另一个选择是，出于测试目的，你可以将*/etc/asterisk/sip.conf* 里 **[general]**部分中的 **tlsdont verify server** 选项设置为 **yes**，这样 Asterisk 即使在服务器证书验证失败的情况下也会允许呼叫进行。

在 7.4.4.1.1 一节，我们创建了在 serverA 和 serverB 之间呼叫电话的必要配置。现在我们修改一下这些配置文件，以让 Asterisk 知道在这两个服务器之间的呼叫是需要加密的。唯一的变化是在每个服务器的入口配置中增加 **transport = tls** 选项。

在 **serverA** 上：

```
[serverB]
type = peer
host = 192.168.1.102
username = serverA
secret = apples
context = incoming
disallow = all
allow = ulaw
transport = tls
```

在 **serverB** 上：

```
[serverA]
type = peer
host = 192.168.1.101
username = serverB
secret = apples
context = incoming
disallow = all
```

```
allow = ulaw
transport = tls
```

现在，当你利用 Dial(SIP/server) 或 Dial(SIP/server) 发起呼叫时，SIP 信令就会被加密。你可以通过修改 dialplan 设置 **CHANNEL(secure_bridge_signaling)=1** 来强制所有外呼呼叫被加密。

```
[default]
exten => 1234,1,Set(CHANNEL(secure_bridge_signaling)=1)
      same => n,Dial(SIP/1234@serverB)
```

在接收呼叫侧，你可以利用 **CHANNEL(secure_signaling)** dialplan 函数来检查呼入呼叫的信令是否被加密，参见下面的例子：

```
[incoming]
exten => _X.,1,Answer()
      same => n,GotoIf($["${CHANNEL(secure_signaling)}" = "1"]?secure:insecure)
      same => n(secure),NoOp(Signaling is encrypted.)
      same => n,Hangup()
      same => n(insecure),NoOp(Signaling is not encrypted.)
      same => n,Hangup()
```

当呼叫从采用这种配置的 serverA 发送到 serverB 时，你可以通过 Asterisk 控制台输出看到 dialplan 判断呼入呼叫的信令是加密的：

```
-- Executing [1234@incoming:1] Answer("SIP/serverA-00000000", "") in new stack
-- Executing [1234@incoming:2] GotoIf("SIP/serverA-00000000",
   "1?secure:insecure") in new stack
-- Goto (incoming,1234,3)
-- Executing [1234@incoming:3] NoOp("SIP/serverA-00000000",
   "Signaling is encrypted.") in new stack
-- Executing [1234@incoming:4] Hangup("SIP/serverA-00000000", "") in new stack
```

现在 SIP TLS 已经配置好了，我们还需要配置 SRTP 以实现媒体流的加密。幸运的是，与 SIP TLS 的配置相比，它非常容易配置。首先，确保 res_srtp module 已经被 Asterisk 加载了：

```
*CLI> module show like res_srtp.so
Module          Description              Use Count
res_srtp.so     Secure RTP (SRTP)        0
1 modules loaded
```

为了使能 SRTP，配置 **CHANEL(secure_bridge_media)**函数为 **1**：

```
[default]
exten => 1234,1,Set(CHANNEL(secure_bridge_signaling)=1)
      same => n,Set(CHANNEL(secure_bridge_media)=1)
```

```
same => n,Dial(SIP/1234@serverB)
```

这个配置指出呼出呼叫需要加密媒体。当呼叫通过 SIP 发出时，Asterisk 将要求 SRTP 被使用，否则呼叫会失败。

通过所有这些工具，你可以确保两个 Asterisk 服务器之间的呼叫是完全加密的。同样的技术也可以被用于加密 Asterisk 和 SIP 话机之间的呼叫。

Dialplan 函数提供了验证呼入呼叫是否加密以及强制呼出呼叫加密的机制。然而，请记住这些工具只能提供控制在呼叫路径上“一跳”（one hop）的加密。如果这个呼叫经过多个服务器，这些工具并不能保证整个呼叫路径都是加密的。仔细考虑加密呼叫的需求并执行所有必须的步骤以确保这些需求在整个呼叫路径上都被考虑到是非常重要的。安全性是复杂、艰苦的工作。

7.5 结论

最终，我们相信 PSTN 将整个消失。然而，在这一切发生之前，我们需要一个被广泛使用和信任的发布机制，这个机制允许组织和个人发布他们的地址信息，从而使得他们能够被找到。我们将在第 12 章研究一些可能的方法。

注释：

注1. 在按键电话系统中，每条外线在每部电话上都有一个对应的按键，对每条外线的访问通过按下对应的按键（line key）实现。

注2. 你可以命名为任何你希望的名字。名字“local”和“toll”对于 Asterisk dialplan 没有任何内建的意思。

注3. 更多关于样式匹配的信息，请参阅第 6 章。

注4. 如果你希望把你的传统家庭电话线连接到你的 Asterisk 系统上，你需要完全相同的板卡。

注5. 获得时钟源也有一些其它方法，如果你真的希望一个很紧凑的系统，运行一个不包含 DAHDI 的 Asterisk 系统是可能的，但是它不是我们想在这里讨论的。

注6. 有时线路会根据它包含了多少 B 信道和 D 信道来命名，所以一个单独的运行在北美 PRI 协议的 T1 可以被称作 23B+D，一个双 T1 线路加上一个备份 D 信道可以被称为 46B+2D。我们也见过 PRI 线路被称为 nB+nD，尽管这有点书呆子气了。

注7. 理论上，这些板卡可以用于任何支持 DAHDI 的软件；因此，基本的板卡配置文件就不应该是 Asterisk 的一部分。

注8. Context 的名字没有任何指定的名字。我们也可以命名这个 context 为[stuff_that_comes_in]，只要这个 context 在 channel 定义文件 sip.conf, iax.conf, chan-dahdi.conf, 等等中有指向，某个 channel 将通过这个 context 进入 dialplan。虽然这么说，我们仍然强烈建议你命名的 context 名字能够帮助你理解它们的目的。一些好的 context 名字包括 [incoming], [local_calls], [long_distance], [sip_telephones], [user_services], [experimental], [remote_locations]，等等。请一直记住 context 决定一个 channel 如何进入 dialplan，所以起个相关的名字。

注9. 在传统 PBX 系统中，DID 号码的作用是允许直接联系办公室中的一个分机。许多 PBX 系统并不支持号码翻译或变长号码等概念，因此电信运营商不得不通过 DID 号码来传递分机号，而不是实际拨打的号码（DNIS 号码）。举例来说，电话号码 416-555-1234 已经被映射到分机 100，因此运营商需要发送号码 100 给 PBX 而不是 DINS 号码 4165551234。如果你已经把老式的 PBX 更换为 Asterisk 系统，你会在适当的位置找到这段翻译，而且你需要获得一个主叫拨打的号码和运营商发个 PBX 的号码之间的映射表。运营商只发送 DINS 的后四位号码也很常见，此时 PBX 就把它翻译为内部号码。

注10. 市场上有许多专用的 PBX 系统，它们通过基本的配置就可以很好的工作。Asterisk 的部署要灵活得多，但是一点也不简单。