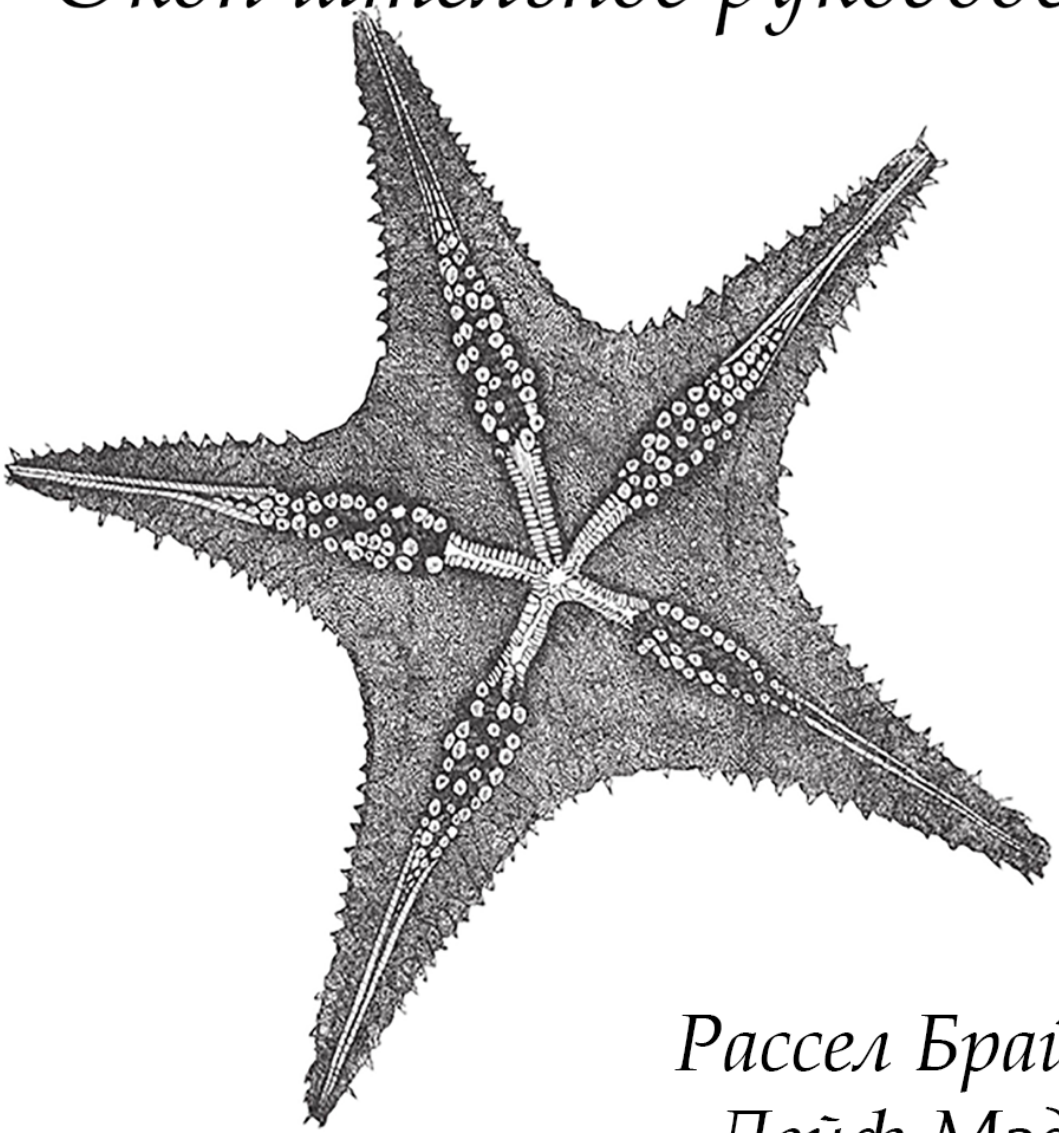
Будущее телефонии сейчас

4-е издание
Рассматривает Asterisk 11



Asterisk

Окончательное руководство



Рассел Брайант,
Лейф Мэдсен и
Джим Ван Меггелен

O'REILLY®

ЧЕТВЕРТОЕ ИЗДАНИЕ

Asterisk™: Окончательное руководство

Russell Bryant, Leif Madsen, and Jim Van Meggelen

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Asterisk™: Окончательное руководство, четвертое издание

Рассел Брайант, Лейф Мэдсен и Джим Ван Меггелен

Copyright © 2013 Рассел Брайант, Лейф Мэдсен и Джим Ван Меггелен. Все права защищены.

Отпечатано в ~~Соединенных Штатах Америки~~.

Опубликовано O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Книги O'Reilly можно приобрести для образовательных, деловых или рекламных целей. Интернет-издания также доступны для большинств названий (*http://my.safaribooksonline.com*). За дополнительной информацией обращайтесь в нашу компанию/учрежденческий коммерческий отдел: 800-998-9938 или *corporate@oreilly.com*.

Редакторы: Майк Лукидес и Натан Джепсон

Производственный редактор: Кристен Борг

Автор: Бекка Фрид

Корректор: Киль Ван Хорн

Индексатор: Фред Браун

Дизайнер Обложки: Карен Монтгомери

Дизайнер Интерьера: Дэвид Футато

Иллюстратор: Ребекка Демарест

Перевод, верстка и индексация перевода: Дмитрий Кузнецов

Май 2013: Четвертое Издание

История изменений для четвертого издания:

2013-05-07: первый релиз

См. *http://oreilly.com/catalog/errata.csp?isbn=9781449332426* для деталей о выпуске.

Nutshell Handbook, логотип Nutshell Handbook и логотип O'Reilly являются зарегистрированными товарными знаками O'Reilly Media, Inc. *Asterisk: Окончательное руководство*, изображение морской звезды и соответствующий торговый костюм являются товарными знаками O'Reilly Media, Inc.

Многие из обозначений, используемые производителями и продавцами для обозначения своих продуктов, заявляются в качестве торговых марок. Где эти обозначения появляются в этой книге и O'Reilly Media, Inc. было известно о торговой марке, обозначение было напечатано в шапках или буквицы.

Несмотря на то, что при подготовке данной книги были приняты все меры предосторожности, издатель и авторы не несут ответственности за ошибки или упущения, а также за ущерб, возникший в результате использования содержащейся в ней информации.

ISBN: 978-1-449-33242-6

[LSI]

Оглавление

Предисловие....16
Введение....20
Революция в телефонии....1
Asterisk and VoIP: Преодоление разрыва между традиционной и сетевой телефонией 2
Проект телефонии Zapata 2
Масштабные изменения требуют гибкости технологии 3
Asterisk: хакерская АТС 4
Asterisk: АТС профессионалов 4
Сообщество Asterisk 4
Списки рассылки Asterisk 5
Wikipedia Asterisk 6
IRC-каналы 6
Группы пользователей Asterisk 6
Проект документации Asterisk 6
Деловой пример 6
Вывод 6
Архитектура Asterisk....7
Модули 8
Приложения 10
Модули соединений 14
Модули записи деталей вызовов (CDR) 14
Модули журналирования событий канала (CEL) 15
Драйверы канала 15
Конверторы кодеков 17
Интерпретаторы формата 18
Функции диалплана 19
Модули УАТС 21
Модули ресурсов 21
Дополнительные модули 25
Тестовые модули 26
Файловая структура 26
Файлы конфигурации 26
Модули 26
Библиотека ресурсов 27
Spool 27
Журналирование (логирование) 27
Диалплан 27
Аппаратные средства 27
Управление версиями Asterisk 28
Предыдущие методологии выпуска 28
Текущая методология выпуска 29
Упрощение номеров версий 30
Вывод 30
Установка Asterisk....31
Шпаргалка по инсталляции 33
Установка дистрибутива 36
RHEL сервер 36
Ubuntu сервер 39
Зависимости программного обеспечения 42
Загружаем то, что нам нужно 43
Получение исходного кода из Subversion 43
Получение исходного кода используя wget 44
Как его устанавливать 44
DAHDI 44

LibPRI 46
Asterisk 46
Установка разрешений для файлов 47
Базовая конфигурация 47
Начальная настройка 47
safe_asterisk 51
make menuselect 54
Обновление Asterisk 58
Общие проблемы 59
-bash: wget: command not found (команда не найдена) 59
configure: error: no acceptable C compiler found in $PATH 60
make: gcc: command not found (команда не найдена) 60
configure: error: C++ preprocessor “/lib/cpp” fails sanity check 60
configure: error: *** Please install GNU make. It is required to build Asterisk! 60
configure: *** XML documentation will not be available because the ‘libxml2’ development package is missing. 60
configure: error: *** termcap support not found 60
You do not appear to have the sources for the 2.6.18-164.6.1.el5 kernel installed. 61
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root? 61
Модернизация (Upgrading) Asterisk 61
Вывод 62
Задачи начальной настройки 63
asterisk.conf 63
Раздел [directories] 63
Раздел [options] 64
Раздел [files] 67
Раздел [compat] 67
modules.conf 67
Секция [modules] 68
indications.conf 69
musiconhold.conf 70
Файл по-умолчанию musiconhold.conf 72
Дополнительные файлы конфигурации 73
Вывод 74
Конфигурация пользовательских устройств 75
Концепции именования телефонов 76
Телефоны, софтфоны и телефонные адаптеры 77
Настройка Asterisk 79
Как файлы конфигурации работают с диалпланом 80
sip.conf 81
Согласование конфигурации SIP и опции type 83
Изменение файлов конфигурации канала для вашей среды 89
Телефоны Digium с Астериск 90
Загрузка конфигураций ваших новых каналов 90
Asterisk CLI 90
Тестирование для обеспечения регистрации устройств 91
Аналоговые телефоны 91
Базовый диалплан для проверки ваших устройств 94
Под капотом: Ваш первый звонок 95
Вывод 95
Основы диалплана 97
Синтаксис диалплана 97
Примерный конфигурационный файл 97
Контекст 98
Расширения (Extensions) 99
Приоритеты 100
Приложения 102

Приложения Answer(), Playback() и Hangup() 102
Приложение Progress() 102
Простой диал-план 103
Hello World 103
Построение интерактивного диал-плана 104
Приложения Goto(), Background() и WaitExten() 104
Обработка неверных значений и тайм-аутов 106
Использование приложения Dial() 107
Использование переменных 110
Совпадение по шаблонам 112
NANP и мошенничество с тарифами 114
Шаблоны, используемые в других странах 114
Продвинутые возможности манипуляций с цифрами 115
Включения (инклюды) 116
Вывод 116
Внешние подключения 117
Основы транкинга 117
Фундаментальный диалплан для исходящих соединений 118
Линии PSTN (ТфОП) 119
Традиционные транки ТфОП 119
FXO и FXS 119
Установка ТфОП транков 121
Отключение загрузки дополнительных модулей DAHDI 121
VoIP 128
Как справиться с преобразованием сетевых адресов (NAT) 128
Сохранение открытости удаленного брандмауэра 130
Терминация ТфОП 134
Инициирование ТфОП 135
VoIP в VoIP 136
Неавторизованные вызовы через SIP 137
Настройка транков VoIP 137
Экстренный набор 143
Вывод 144
Голосовая почта 145
Комедийная почта 145
Раздел [general] 146
Внешняя проверка паролей голосовой почты 150
Раздел [zonemessages] 154
Раздел контекстов 154
Исходный файл voicemail.conf 158
Стандартные клавиши голосовой почты 158
Интеграция диалплана 159
Приложение диалплана VoiceMail() 159
Приложение диалплана VoiceMailMain() 162
Создание каталога Набор-по-имени 162
Использование Jitterbuffer'a 163
Внутреннее хранение 164
Файловая система Linux 164
ODBC 164
IMAP 164
Использование Asterisk в качестве автономного сервера голосовой почты 165
Интеграция Asterisk в среду SIP как автономный сервер голосовой почты 165
SMDI (Simplified Message Desk Interface) 168
Интеграция базы данных 169
Вывод 169
Интернационализация 170
Внешние устройства для сервера Asterisk 171

Подключение к ТфОП, DAHDI, карты Digium и аналоговые телефоны 173
Драйверы DAHDI 175
Asterisk 177
Caller ID 177
Язык и/или акцент подсказок 178
Штампы время/дата и произношение 179
Вывод — Простая шпаргалка 181
Погружение в диалплан 182
Выражения и манипуляции с переменными 182
Базовые выражения 182
Операторы 183
Функции диалплана 185
Синтаксис 185
Примеры функций диалплана 185
Условное ветвление 186
Приложение GotoIf() 186
Временное условное ветвление с GotoIfTime() 189
Макрос 191
Определение макроса 192
Вызов макроса из диалплана 192
Использование аргументов в макросе 193
GoSub 194
Определение подпрограмм 194
Вызов подпрограмм из диалплана 195
Использование аргументов в подпрограмме 195
Возврат из подпрограммы 196
Локальные (Local) каналы 198
Использование базы данных Asterisk (AstDB) 200
Хранение данных в AstDB 200
Извлечение данных из AstDB 200
Удаление данных из AstDB 201
Использование AstDB в диалплане 201
Создание приложения горячего стола с AstDB 202
Полезные функции Asterisk 206
Zapateller() 206
Парковка вызова 207
Конференц-связь с MeetMe() 208
Конференц-связь с ConfBridge() 209
Вывод 210
Парковка, пейджинг и конференц-связь 211
features.conf 211
Базовые особенности DTMF 211
Раздел [general] 212
Обработка таймаутов припаркованных вызовов с опцией comebacktoorigin 213
Раздел [featuremap] 214
Раздел [applicationmap] 214
Наследование переменных канала 215
Динамическое создание Feature-Map из диалплана. 216
Группировка сопоставлений приложений 217
Парковочные места 217
Пейджинг потолочный и «под носом» (aka система оповещения). 218
Места для отправки пейджинга 219
Многоадресный пейджинг на Cisco SPA-телефонах 222
Зонирование пейджинга 224
Расширенные возможности конференц-связи 224
Раздел [general] 224
Параметры профилей пользователей 225

Параметры для профилей соединения 226
Параметры меню ConfBridge 228
Включение PIN-кода 229
Ожидание присоединения маркированного пользователя 231
Использование меню ConfBridge() 232
Включение видеоконференций 234
Вывод 236
Маршрутизация интернет-вызовов 237
DNS и SIP URI 238
SIP URI 238
SRV записи 238
Приём вызовов в вашу систему 239
Набор SIP URI из Asterisk 245
ENUM и E.164 246
E.164 и ITU (МСЭ) 246
Североамериканский центр нумерации 247
ENUM 247
Asterisk и ENUM 248
Статус ENUM во всем мире 248
ISN, ITAD и freenum.org 248
Цель проекта freenum.org 249
Получили ISN? 249
Абонентские номера ITAD(ISNs) 250
Управление интернет-нумерацией 250
Административные домены IP-телефонии (ITAD) 250
Создание записи DNS для своего ITAD 251
Использование делегированных зон для географически распределенных офисов 252
Тестирование ITAD 252
Использование ISN в вашей системе Asterisk 253
Безопасность и идентификация. 254
Мошенничество с оплатой 255
Спам по интернет-телефонии (SPIT) 255
Распределенные атаки на отказ в обслуживании 256
Фишинг 256
Безопасность - это непрерывный процесс. 256
Вывод 256
Очереди автоматического распределения вызовов (ACD) 258
Создание простой очереди ACD 259
Участники очереди 262
Управление участниками очереди с помощью CLI 262
Определение участников очереди в файле queues.conf 264
Управления участниками очереди с помощью логики диалплана 264
Использование Pause и Unpause 265
Автоматическая регистрация в нескольких очередях 266
Введение в состояния устройств 269
Файл queues.conf 271
Файл agents.conf 278
Расширенные очереди 280
Приоритетная очередь (взвешивание очереди) 280
Приоритет участника очереди 282
Динамическое изменение штрафов (queuerules.conf) 283
Управление объявлениями 284
Воспроизведение объявлений между файлами музыки на удержание 284
Переполнение 288
Использование локальных (Local) каналов 290
Статистика очереди: файл queue_log 294
Вывод 296

Состояния устройств.....297
Состояния устройств 297
Проверка состояния устройства 298
Статусы внутренних номеров 299
Хинты 299
Проверка статуса внутреннего номера 300
SIP-присутствие 301
Конфигурация Asterisk 301
Использование пользовательских состояний устройств 302
Как пример 302
Распространение состояний устройств 304
Использование Corosync 304
Использование XMPP 308
Общие внешние линии 311
Установка приложений SLA 311
Обзор конфигурации 312
Пример системы клавиш с аналоговыми транками 312
Пример системы клавиш с SIP-транками 315
Пример системы альтернативных клавиш с SIP-транками 317
Пример общего внутреннего номера 320
Расширенная конфигурация 321
Ограничения 322
Создание службы обратного звонка 323
Вывод 324
Автосекретарь.....325
Автосекретарь не является IVR 325
Проектирование вашего автосекретаря 325
Приветствие 327
Главное меню 327
Тайм-аут 328
Неверный (Invalid) 329
Набор добавочного номера 329
Создание вашего автосекретаря 329
Запись подсказок 329
Рекомендуемый формат файла подсказки 329
Диалплан 331
Доставка входящих вызовов 332
IVR 332
Заключение 332
Интеграция реляционной базы данных.....333
Установка и настройка PostgreSQL и MySQL 333
Установка PostgreSQL для RHEL 333
Установка PostgreSQL для Ubuntu 334
Установка MySQL для RHEL 334
Установка MySQL для Ubuntu 334
Настройка PostgreSQL 335
Настройка доступа к базе данных PostgreSQL через IPv6 localhost 335
Настройка MySQL 336
Установка и настройка ODBC 337
Настройка ODBC для PostgreSQL 338
Настройка ODBC для MySQL 339
Настройка ODBC для Microsoft SQL 340
Проверка ODBC-коннектора 341
Компиляция модулей ODBC для Asterisk 342
Настройка res_odbc для подключения Asterisk к ODBC 342
Управление базами данных 343
Устранение неполадок с базой данных 343

Инъекция SQL 344
Мощь вашего диалплана с func_odbc 344
Отношения файлов конфигурации ODBC 344
Нежное введение в func_odbc 345
Веселимся с func_odbc: горячий стол 346
Использование функции ARRAY() 350
Использование SQL непосредственно в диалплане. 353
Мультистроковая функциональность с func_odbc 355
Использование Realtime 358
Статический Realtime 359
Слово о метриках 360
Когда использовать preload в modules.conf для Realtime модулей 361
Динамический Realtime 362
Предопределенные динамические имена для Realtime в extconfig.conf 362
Хранение записей деталей вызовов (CDR) 365
Установка имени системы для глобальных уникальных идентификаторов 366
Дополнительные параметры конфигурации для cdr_adaptive_odbc.conf 368
Хранение сообщений голосовой почты ODBC 369
Альтернативный метод централизации 369
Компиляция модуля app_voicemail для поддержки хранилища ODBC 370
Создание типа Large Object для PostgreSQL 370
Макет таблицы хранения голосовой почты ODBC 372
Настройка voicemail.conf для ODBC-хранилища 373
Тестирование хранилища голосовых сообщений ODBC 374
Интеграция базы данных с очередями 377
Хранение queues.conf в базе данных 378
Хранение параметров диалплана для очереди в базе данных 380
Запись queue_log в базу данных. 380
Заключение 381
Интерактивное голосовое меню 382
Что такое IVR? 382
Компоненты IVR 382
Совершенно вкусная IVR 384
Соображения по проектированию IVR 384
Модули Asterisk для построения IVR 384
CURL 385
func_odbc 385
AGI 385
AMI 385
Простой IVR используя CURL 385
Установка модуля cURL 385
Диалплан 386
Приложение для записи приглашений 386
Распознавание речи и преобразование текста в речь 387
Text-to-Speech 387
Распознавание речи 387
Заключение 388
Внешние службы 389
Интеграция календаря 389
Компиляция поддержки календаря в Asterisk 390
Настройка поддержки календаря для Asterisk 391
Запуск напоминаний календаря на вашем телефоне 393
Управление вызовами на основе информации календаря 397
Запись информации о вызове в календарь 398
Дополнительные функции 400
Интеграция голосовой почты с IMAP 401
Компиляция поддержки голосовой почты IMAP в Asterisk 401

Использование XMPP (Jabber) с Asterisk 406
Компиляция поддержки XMPP в Asterisk 407
Команды диалплана Jabber 407
chan_motif 412
Внешние сообщения (инфраструктура обмена сообщениями) 415
Конфигурация xmpp.conf 415
Конфигурация sip.conf 416
Конфигурирование диалплана 417
Интеграция LDAP 418
Конфигурирование OpenLDAP 418
Компиляция поддержки LDAP в Asterisk 420
Конфигурирование Asterisk для поддержки LDAP 420
Утилиты Text-to-Speech 422
Festival 423
Cepstral 424
Вывод 425
Fax 426
Что такое Fax? 426
Способы обработки факсов в Asterisk 426
spandsp 427
Получение spandsp 427
Компиляция и установка spandsp 427
Добавление библиотеки spandsp в ваш libpath 427
Перекомпиляция Asterisk с поддержкой spandsp 428
Отключение spandsp (Если вы хотите проверить Digium Fax) 428
Digium факс для Asterisk 428
Получение Digium FFA 428
Отключение Digium FFA (Если вы хотите проверить spandsp) 429
Обработка входящих факсов 429
Факс в TIFF 430
Факс в Email 431
Fax в PDF 431
Обнаружение факса 431
Использование T.38 432
Какой VoIP-провайдер? 433
Обработка исходящих факсов 433
Передача факса из Asterisk 433
Формат файла для отправки факсов 433
Эксперимент с отправкой в электронной почты по факсу 434
Сквозной факс 437
Использование буфера факса в chan_dahdi.conf 437
Шлюз T.38 438
Заключение 438
Интерфейс управления Asterisk (AMI) 439
Быстрый старт 439
AMI через TCP 440
AMI через HTTP 440
Конфигурация 441
manager.conf 442
http.conf 445
Обзор протокола 445
Кодировка сообщений 447
AMI через HTTP 448
Файлы вызовов 452
Пример использования 454
Инициирование вызова 454
Перенаправление вызова 455

Инициирование вызова с использованием Python и StarPy 456
Разработка фреймворков 461
CSTA 461
Интересные приложения 461
Flash-панель оператора 461
Заключение 462
Интерфейс шлюза Asterisk (AGI) 463
Быстрый старт 463
Варианты AGI 464
Process-Based AGI 464
DeadAGI Is Dead 465
FastAGI-AGI через TCP 466
Async AGI-AMI-Контролируемый AGI 466
Обзор связи AGI 467
Настройка сеанса AGI 467
Команды и ответы 469
Завершение сеанса AGI 472
Пример: База данных учетных записей доступа 473
Разработка фреймворков 476
Вывод 476
Кластеризация 477
Традиционные УАТС 478
Гибридные системы 479
Чистый Asterisk, нераспределенный 480
Asterisk и интеграция базы данных 481
Единая база данных 481
Реплицирование базы данных 482
Asterisk и предоставление статуса устройства 484
Распространение состояний устройств по локальной сети 484
Распространение состояний устройств по глобальной сети 485
Несколько очередей, несколько местоположений 486
Заключение 488
Распределённое универсальное распознавание номеров (DUNDi) 489
Как работает DUNDi? 489
Файл dundi.conf 491
Настройка Asterisk для использования с DUNDi 493
Общая конфигурация 493
Первоначальное определение пиров DUNDi 494
Создание контекстов сопоставления 496
Использование контекстов сопоставления с пирами 497
Разрешение удаленных подключений 498
Контроль ответов 500
Выполнение поиска из диалплана 503
Вывод 505
Системный мониторинг и журналирование 507
logger.conf 507
Просмотр журналов Asterisk 509
Логирование демоном Linux syslog 509
Проверка логирования 510
Ротация логов 510
Call Detail Records (CDR) — запись деталей вызовов 510
Содержание CDR 510
Приложения диалплана 512
cdr.conf 512
Конечные решения (Backends) 513
Пример Call Detail Records 518
Предостережения 519

CEL (Channel Event Logging) 519
Типы событий канала 519
Содержание событий канала 520
Приложения диалплана 521
Конечные решения (Backends) 522
Примеры событий канала 527
SNMP 531
Установка модуля SNMP для Asterisk 531
Настройка SNMP для использования OpenNMS в Asterisk 532
Мониторинг Asterisk с помощью OpenNMS 534
Заключение 535
Web-интерфейсы 537
Flash Operator Panel 537
Состояние очереди и отчеты 538
Отображение состояния очереди 538
Отчеты очереди 539
Детальная запись вызовов (CDR) 539
A2Billing 539
Вывод 539
Безопасность 540
Сканирование действительных учетных записей 540
Уязвимости аутентификации 542
Fail2ban 542
Установка 542
Конфигурация 543
Зашифрованные медиаданные 547
Уязвимости диалплана 547
Обеспечение безопасности сети Asterisk API 548
IAX2 отказ в обслуживании 548
Другие меры по снижению риска 550
Разрешения CLI 551
Ресурсы 552
Заключение—лучший идиот 552
Asterisk: Будущее телефонии 554
Проблемы традиционной телефонии 554
Закрытое мышление 555
Ограниченное соблюдение стандартов 555
Медленный цикл выпусков 555
Отказ отпустить прошлое и принять будущее 556
Сдвиг парадигмы 556
Обещание телефонии с открытым исходным кодом 556
Зуд, который вызывает Asterisk 557
Открытая архитектура 557
Соблюдение стандартов 558
Молниеносная реакция на новые технологии 558
Страстное сообщество 558
Некоторые вещи, которые уже возможны 558
Будущее Asterisk 562
Обработка речи 562
Высококачественный голос 563
Видео 564
WebRTC 565
Беспроводность 565
Универсальная система обмена сообщениями 566
Пиринг 566
Задачи 567
Возможности 570

Понимание телефонии.....571
Аналоговая телефония 571
Части аналогового телефона 571
Tip and Ring 573
Цифровая телефония 573
Имульсно-кодовая модуляция 574
Цифровая коммутируемая телефонная сеть 583
Типы линий связи 583
Цифровые сигнальные протоколы 584
Сети с коммутацией пакетов 586
Вывод 586
Протоколы VoIP.....587
Необходимость протоколов VoIP 588
Протоколы VoIP 588
IAX (“Inter-Asterisk eXchange” протокол) 588
SIP 589
H.323 591
MGCP 593
Проприетарные протоколы 593
Кодеки 593
G.711 594
G.726 594
G.729A 595
GSM 595
iLBC 595
Speex 595
G.722 596
MP3 596
Quality of Service 596
TCP, UDP и SCTP 596
Дифференцированное обслуживание 597
Гарантированный сервис 597
Негарантированная доставка 598
Эхо 598
Почему возникает эхо 599
Управление эхом на каналах DAHDI 599
Аппаратное эхоподавление 599
Asterisk и VoIP 600
Пользователи и пиры и друзья — о, боже! 600
Выражение register 601
Безопасность VoIP 601
Спам через Интернет Телефонию (SPIT) 602
Шифрование аудио с помощью Secure RTP 602
Спуфинг 602
Что можно сделать? 602
Вывод 603
Подготовка системы для Asterisk.....604
Выбор оборудования сервера 605
Проблемы производительности 606
Выбор процессора 608
Выбор материнской платы 609
Требования к электропитанию 611
Окружающая среда 612
Питание отвечающее стандартам и источники бесперебойного питания 612
Заземление 613
Электрическая цепь 614
Комната оборудования 614

Телефонное оборудование 615
Подключение к ТфОП 615
Подключение исключительно к сети с коммутацией пакетов 617
Эхоподавление 617
Типы телефонов 618
Физические телефоны 618
Софтфоны 620
Телефонные адаптеры 621
Терминалы связи 621
Особенности Linux 621
Вывод 622
Алфавитный указатель 623

Когда мы думали о том, кого могли бы попросить написать предисловие к четвертому изданию книги, было перебрано множество имен. У нас уже был Марк Спенсер (автор Asterisk), который написал его для первых двух выпусков книги. Затем Джон Тодд сделал фантастическую работу для третьего издания. После, окинув взором несколько имен (некоторых вы увидите ниже) мы подумали: «Эта книга, написана сообществом, как насчет предисловия сообщества?» Имея в виду эту идею, мы выбрали нескольких людей, которых мы уважаем, и которые используют Asterisk так же долго, как (если не дольше) мы сами. Когда мы думали о людях, мы хотели получить несколько точек зрения и свободно ответить на несколько вопросов об Asterisk. Вопросы, которые мы рассматривали, включали:

- Как Asterisk помогает сообществам?
- Какие мирские начинания внесла Asterisk?
- Где находится Asterisk и куда идет?
- Для чего развертывается Asterisk и какие потребности решает?

Это те вопросы, которые большинство людей, использующих Asterisk в течение длительного периода времени, либо спрашивают, либо задают сами себе. Задав эти вопросы следующим авторам, все они вернулись с различными точками зрения об Asterisk и о том, как она изменила телекоммуникационную отрасль и жизнь людей. Мы надеемся, что вам понравится читать об их мнениях так же, как и нам.

Мэтт Джордан (инженер-программист, Digium)

Когда Лейф попросил меня написать предисловие к обновленному изданию *Asterisk: The Definitive Guide*, он задал следующий вопрос: «Где находится Asterisk и куда идет?» Это означает, что далее будут некоторые прогнозы - Вы были предупреждены!

Чтобы ответить на первую часть, я посмотрел, как Asterisk развивалась в течении нескольких последних версий. Каждая версия построена на предыдущей, в то же время оставаясь верной тому, что сделало Asterisk отличной: свободная и открытая платформа для создания приложений телефонии. С течением времени и ландшафт телефонии изменился, Asterisk изменилась вместе с ним. Asterisk развивалась с новой функциональностью для удовлетворения меняющихся потребностей людей, которые её использовали и разрабатывали. Время от времени Asterisk подталкивает телекоммуникационную отрасль; временами она отталкивается от неё. Результатом этого толчка является состояние Asterisk сегодня - во многих вещах и во многих местах. Она управляет

телефонными системами в моем местном продуктовом магазине, аптеке и сети пиццерий - она является двигателем выбора, который обеспечивает УАТС каждого человека: от любителей до крупных предприятий.

Вторая часть сложнее. Поскольку Asterisk приближается к 15-летнему юбилею, вопрос, на мой взгляд, заключается не столько в том, «куда идет Asterisk», а «куда идет индустрия телефонии?». Сближение мобильных платформ, размещенных решений и WebRTC в корне изменит не только наше определение телефона, но и то, как компании развертывают свою инфраструктуру связи и что это значит для общения. Итак, как Asterisk реагирует на фундаментальные изменения в развертывании, работоспособности и использовании?

В моем сознании, как всегда - новаторски. Прокладывая путь к принятию стандартов для общения. Предоставляя новые API-интерфейсы, которые позволяют любому пользователю использовать Asterisk для создания коммуникационных приложений для самых разных бизнес-потребностей. И, наконец, желая меняться. Asterisk традиционно предоставляет приложения для вас - если вы хотите очереди вызовов, вы используете приложение Queue. Если вам нужна голосовая почта, вы используете VoiceMail. Поскольку мы продвигаемся вперед в следующих основных версиях Asterisk, я вижу, что акцент делается не на предоставлении вам функциональности, а на предоставлении строительных блоков для создания любых необходимых функций связи.

Это захватывающее время чтобы быть пользователем и разработчиком Asterisk, и я с нетерпением жду построения Asterisk вместе с вами.

Алекс Балашов (Директор Evariste Systems)

Asterisk чаще всего отмечает свою свободную лицензию и щедрый набор функций. Однако, как известно большинству пользователей технологии с открытым исходным кодом, она требует затрат на внедрение. Существенная ценность Asterisk, на мой взгляд, заключается не в экономической или технической эффективности свободной УАТС, а в разрушительном структурном воздействии, которое она оказывает на более широкую область инноваций, в которой она участвует. Это безвозвратно сместило разговор о возможностях в телефонии.

Asterisk не только предлагает открытый код для существующих вендоров АТС. Тот факт, что она может работать на аппаратных средствах ПК и небольших ПК-совместимых встраиваемых устройствах, вызвал тектонический сдвиг в сторону коммодитизации[1] бизнес-проблем, которые ранее разрешались только путем комплексного обмена данными, выполненного на дорогостоящем, запатентованном оборудовании или дорогостоящем лицензированном программном обеспечении как, например, системы интерактивного голосового меню (IVR), которые предоставляют интерфейсы самообслуживания для банков. У Asterisk есть многочисленные пути интеграции и API, которые позволяют подключаться к другим сервисам с использованием открытых стандартов и общепринятых протоколов, что резко снижает стоимость разговоров с другими системами. Невозможно переоценить трансформирующее воздействие, которое это имело, позволив совершенно новым бизнес-моделям взлететь, не привязываясь к до сих пор запретительному капитальному сопротивлению.

Asterisk может принять во внимание тот факт, что на данный момент есть совершенно новые, поколенческие ответы на вопрос: «Можем ли мы заставить телефонную систему сделать это?». Существующие поставщики телекоммуникационных услуг, независимо от степени, в которой они рассматривают Asterisk в качестве конкурента, вынуждены пересмотреть свои маркетинговые предложения с точки зрения возможностей, которые она заставила открыть. Asterisk изменила язык наблюдения - лексику, мыслительный процесс, основные экономические предположения систем бизнес-телефонии.

Функциональность Asterisk и привязанность к оборудованию также ослабили стены огромных крепостей телекоммуникационной монополии, которые ранее считались неприступными. Я стал свидетелем его использования с `libss7` в качестве элемента взаимосвязи конкурентных операторов в нескольких странах с формирующимся рынком, а также в качестве обхода платных приложений и недорогих услуг телефонных карт. Влияние этой волны давления, основанной Asterisk, инновационных недорогих альтернатив, является титаническим, создавая совершенно новые

1 Превращение продукта в сравнительно дешёвый товар массового потребления (викисловарь).

социальные связи, рабочие места и средства к существованию во всем мире, обогащая жизнь многих людей. Моя собственная семья разбросана по всему миру, и моя УАТС Asterisk сделала этот мир намного меньше. Мы просто не могли позволить себе общаться так тесно, регулярно и богато раньше.

В целом, я думаю, что в общем континууме технологического развития, Asterisk, возможно, хорошо запомнится тем, что внутри неё, и ещё больше тем, что она оттолкнула от себя своими мощными локтями.

Кевин Макаллистер (вице-президент по технологиям, CoreDial, LLC).

В конце лета 2005 года мне предложили работу в стартапе, который планировал предоставлять услуги IP-УАТС для предприятий. В то время я знал системное администрирование Linux, IP-сети и чуть ли не до смерти боялся голоса.

Первой причиной страха перед голосом было то, что я знал, как работает Интернет. Голос - это канарейка на угольной шахте, которой является Интернет - когда сеть ломается, голос умирает первым. Вторая причина боязни заключается в том, что люди очень хорошо знакомы с телефонами, и поэтому у них есть крепкие представления о том, как телефоны должны работать чтобы вести свой бизнес, и ожидают, что голосовое обслуживание будет более надежным, чем электричество. Я был ответственен за попытку преодолеть эту особенность для работы в этих сложных обстоятельствах.

План состоял в том, чтобы построить поверх существующего прототипа на основе Asterisk. Таким образом, у меня была не только сложная задача протолкнуть голос через мою сеть, но и узнать, как использовать сложную часть программного обеспечения с открытым исходным кодом, которая традиционно предлагает документацию, являющейся неполной и часто неточной. Я хотел это сделать, но не знал как заставить программу сделать это, и никак не мог выяснить.

К счастью, в то время была доступна только одна книга: *Asterisk ™: будущее телефонии*. Первая версия этой книги помогла мне быстро понять не только Asterisk, но основы VoIP. Презентация позволила мне увидеть, как я могу быстро объединить многие и очень гибкие функции программного обеспечения Asterisk для создания сложных и надежных функций, которые требуются моим клиентам.

Сейчас - почти восемь лет и бесчисленные миллионы успешных звонков спустя - я сам научился тому, что делать и что не делать при создании интернет-VoIP-систем. Но я продолжаю сильно полагаться на Asterisk и на авторов этой книги, чтобы помочь мне быстро понять это постоянно меняющееся и улучшающееся программное обеспечение. Вы выбрали отличную отправную точку для работы с Asterisk. Желаю вам еще большего успеха, чем у меня.

Брайан Капуч (отдел компьютерных наук, Колледж Святого Иосифа [Индиана])

«Чтож, Вы можете сделать это довольно легко». Это наблюдение снова и снова зажигалось в моем мозгу, когда я впервые начал взламывать Asterisk. Я уже играл ся с некоторыми ранними продуктами IP-телефонии, наслаждаясь некоторым успехом, но также испытывал много страданий, и вдруг однажды я впервые увидел упоминание об Asterisk в списке рассылки. Я получил код и собрал его (до того, как он стал версией!), и я навсегда запомню первый звук, который я услышал: голосовое приглашение Эллисон Смит «Comedian Mail». Я смеялся и смеялся. Что здесь происходит??!!??

Я копнул глубже, и это изменило мою жизнь. В то время я управлял беспроводным провайдером. Это было неописуемое волнение, что я смог создать сеть серверов Asterisk, работающих на дешевом eBay-оборудовании, подключенном к локальным линиям телефонной связи на пятнадцати POP[2], разбросанных на 500 квадратных миль в основном пустынных сельскохозяйственных угодий

2 PoP (англ. *Point of presence*) — точка присутствия, термин, употребляемый в телекоммуникациях для операторов первого уровня (wikipedia).

Индианы. Я не только получил свою собственную зону бесплатных звонков, они были бесплатны в те дни, когда минуты действительно что-то стоили.

Для другого проекта я подключил к Asterisk USB-камеру стоимостью 20 долларов, которую я купил на eBay, и получил видеонаблюдение за старой железной дорогой, которой я владею в Медаривилле, штат Индиана. Она была построена в 1853 году и еще не была оснащена современным электротехническим сервисом. Система работала от морской батареи, которую я менял раз в несколько дней.

Позже, в один летний вечер, мой друг Боб устроил фейк-взлом в отеле, пока я давал вечернюю презентацию на технической конференции в далекой Калифорнии. Моя аудитория и я смотрели на входную дверь через живой канал с камеры, и до того, как изображение входа Боба обновилось на мониторе, в моем мобильном телефоне зазвенел портативный SIP-телефон, в котором сообщалось о «ситуации безопасности» в моем отеле. Серверы Asterisk обрабатывали вызов «точка-точка» между отелем и конференц-центром.

Asterisk – это всевозможные функциональные возможности телефонии, для всех ситуаций, совершенно свободные для любого пользователя. Это еще один пример того, как продукты с открытым исходным кодом не только экономят много денег, но и делают работу лучше, чем коммерческие продукты от крупных игроков. Для меня Asterisk – это расширение возможностей, свобода от умирающей тирании жадных монополистов и новые миры, которые еще впереди. Независимо от того, какой новый коммуникационный трюк вы могли бы захотеть попробовать, вероятный ответ «Ну, вы можете сделать это довольно легко». Эта книга была создана великими друзьями, которые, как и я, почти присутствовали при создании. Они энергично и творчески поддерживают бесценное, дружественное и всестороннее руководство к одному из самых великих продуктов с открытым исходным кодом. Наслаждайтесь!

Это книга для тех, кто использует Asterisk.

Asterisk - это конвергентная платформа телефонии с открытым исходным кодом, которая предназначена в первую очередь для работы в Linux. Asterisk объединяет более 100 лет знаний о телефонии в надежный набор тесно интегрированных телекоммуникационных приложений. Сила Asterisk заключается в ее настраиваемой природе, дополненной непревзойденным соблюдением стандартов. Никакая другая УАТС не может быть развернута таким множеством творческих способов.

Приложения, такие как голосовая почта, организованная конференц-связь, очередь вызовов и агенты, музыка в ожидании и парковка вызова являются стандартными функциями, встроенными прямо в программное обеспечение. Кроме того, Asterisk может интегрироваться с другими бизнес-технологиями таким образом, о каком закрытые проприетарные УАТС вряд ли могут мечтать.

Asterisk может казаться довольно пугающим и сложным для нового пользователя, поэтому документация настолько важна для его роста. Документация снижает барьер для входа и помогает людям созерцать возможности.

Произведено с щедрой поддержкой O'Reilly Media, *Asterisk: The Definitive Guide* - четвертое издание того, что раньше называлось *Asterisk: The Future of Telephony*. Мы решили сменить имя, потому что Asterisk был настолько дико успешным, что он больше не является многообещающей технологией. Asterisk имеет достижения.

Эта книга была написана для и при участии сообщества Asterisk.

Аудитория

Эта книга предназначена для того, чтобы быть мягкой по отношению к новичкам в Asterisk, но мы предполагаем, что вы знакомы с базовым администрированием Linux, сетями и другими IT-дисциплинами. Если нет, мы рекомендуем вам изучить обширную и замечательную библиотеку книг, которую О'Рейли публикует по этим темам. Мы также предполагаем, что вы новичок в телекоммуникациях (как традиционная коммутируемая телефония, так и новый мир Voice over IP).

Однако, эта книга также будет полезна для более опытного администратора Asterisk.

Мы сами используем книгу как ссылку на функции, которые мы не использовали до этого.

Организация

Книга состоит из следующих глав:

Глава 1, Революция в телефонии.

Здесь мы рубим щепки и зажигаем огонь. Добро пожаловать в Asterisk!

Глава 2, Архитектура Asterisk

Обсуждает файловую структуру системы Asterisk.

Глава 3, Установка Asterisk

Охватывает получение, компиляцию и установку Asterisk.

Глава 4, Задачи начальной настройки

Описывает некоторые задачи первоначальной настройки вашей новой системы Asterisk. В этой главе рассматриваются некоторые файлы конфигурации, необходимые для всех установок Asterisk.

Глава 5, Конфигурация пользовательских устройств

Предоставляет рекомендации по настройке Asterisk, позволяющие таким устройствам как телефоны, подключаться и совершать вызовы.

Глава 6, Основы диалплана

Представляет сердце Asterisk - диалплан.

Глава 7, Внешнее подключение

Обсуждает, как настроить Asterisk для подключения к другим системам, таким как другие серверы Asterisk, провайдеры интернет-телефонии или обычная старая телефонная сеть.

Глава 8, Голосовая почта

Охватывает использование одного из самых популярных приложений, включенных в Asterisk - систему голосовой почты.

Глава 9, Интернационализация

Фокусируется на проблемах, которые должен знать администратор Asterisk при развертывании системы за пределами Северной Америки.

Глава 10, Погружение в диалплан

Пройдется по нескольким более продвинутым концепциям диалплана.

Глава 11, Парковка, пейджинг и конференц-связь

Описывает использование популярных функций телефонии, включенных в Asterisk: вызов парковки, пейджинг и конференц-связь.

Глава 12, Маршрутизация интернет-вызовов

Обнаруживает методы маршрутизации вызовов между различными административными доменами в Интернете.

Глава безнадежно устарела и доменное имя более недоступно (прим. Переводчика).

Глава 13, Очереди автоматического распределения вызовов (ACD)

Обсуждает, как создавать очереди вызовов в Asterisk.

Глава 14, Состояние устройств

Представляет концепцию состояний устройства и то, как они могут использоваться в качестве индикаторов присутствия.

Глава 15, Автосекретарь

Описывает, как построить систему меню, используя диалплан Asterisk.

Глава 16, Интеграция реляционной базы данных

Обсуждает различные способы интеграции Asterisk с базой данных.

Глава 17, Интерактивное голосовое меню.

Обсуждается, как использовать Asterisk для создания приложений, которые реагируют на входящий вызов.

Глава 18, Внешние службы

Предоставляет инструкции по подключению к внешним службам, включая LDAP, календари, IMAP для голосовой почты, XMPP, внеполосный обмен сообщениями и преобразование текста в речь.

Глава 19, Факс

Обсуждает различные варианты интеграции отправки и приема факсов с системой Asterisk.

Глава 20, Интерфейс управления Asterisk (AMI)

Представляет сетевой API для мониторинга и управления системой Asterisk.

Глава 21, Интерфейс шлюза Asterisk (AGI)

Представляет API Asterisk, который позволяет осуществлять управление вызовами на любом языке программирования.

Глава 22, Кластеризация

Обсуждает ряд подходов к кластеризации нескольких серверов Asterisk, когда требования развертывания превышают возможности одного сервера.

Глава 23, Распределенное универсальное распознавание номеров (DUNDi)

Рассматривает одноранговый протокол (peer-to-peer), родной для Asterisk, который может использоваться для маршрутизации вызовов.

Глава 24, Системный мониторинг и журналирование

Представлены некоторые интерфейсы, доступные для ведения лога и мониторинга системы Asterisk.

Глава 25, Веб-интерфейсы

Обзор некоторых веб-интерфейсов, которые дополняют установку Asterisk.

Глава 26, Безопасность

Обсуждает некоторые общие проблемы безопасности, о которых должны знать администраторы Asterisk.

Глава 27, Asterisk: будущее телефонии»

В заключение мы обсудим некоторые вещи, которые ожидаем увидеть от телефонии с открытым исходным кодом в ближайшем будущем.

Приложение A, «Понимание телефонии»

Изучаем технологии, используемые в традиционных телекоммуникационных сетях.

Приложение B, Протоколы VoIP.

Вникаем во все особенности Voice over IP.

Приложение C, Подготовка системы для Asterisk

Содержит информацию, которую вы должны знать и принимать во внимание при планировании развертывания Asterisk.

Программное обеспечение

Эта книга ориентирована на документирование Asterisk версии 11; однако многие из решений и большая часть информации в этой книге не зависит от версии. Linux - это операционная система, в которой мы запускали и тестировали Asterisk, и мы документировали инструкции по установке как для Red Hat Enterprise Linux (RHEL), так и для Ubuntu (Debian), где они отличаются друг от друга.

Условные обозначения используемые в книге

В этой книге используются следующие типографские условные обозначения:

Курсив

Указывает новые термины, URL-адреса, адреса электронной почты, имена файлов, расширения файлов, имена путей, каталоги и имена пакетов, а также утилиты, команды, модули, параметры Unix, и аргументы.

`Моноширинный`

Используется для отображения образцов кода, содержимого файла, взаимодействия с командной строкой, команд базы данных, имен библиотек и параметров.

`Моноширинный полужирный`

Указывает команды или другой текст, который должен быть введен буквально пользователем. Также используется для выделения кода.

`Моноширинный курсив`

Показывает текст, который следует заменить на значения, заданные пользователем.

`[ Ключевые слова и другие материалы ]`

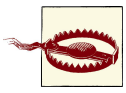
Указывает необязательные ключевые слова и аргументы.

`{ выбор-1 | выбор-2 }`

Определяет либо *`выбор-1`*, либо *`выбор-2`*.



Этот значок обозначает подсказку, предложение или общее примечание.



Этот значок указывает на предупреждение или предостережение.

Использование примеров кода

Эта книга предназначена для того, чтобы помочь вам выполнить свою работу. В целом, если эта книга содержит примеры кода, вы можете использовать его в своих программах и документации. Вам не нужно обращаться к нам за разрешением, если вы не воспроизводите значительную часть кода. Например, для написания программы, которая использует несколько фрагментов кода из этой книги, не требуется разрешение. Продажа или распространение примеров с CD-ROM из книг O'Reilly требует разрешения. Ответ на вопрос, с цитированием этой книги и примером кода, не требует разрешения. Включение значительного количества кода из примера этой книги в документацию вашего продукта требует разрешения.

Мы ценим, но не требуем, атрибуции. Обычно атрибуция включает название, автора, издателя и ISBN. Например: «*Asterisk: The Definitive Guide*, четвертое издание, Рассел Брайант, Лейф Мэдсен и

Джим Ван Меггелен (O'Reilly). Copyright 2013 Russell Bryant, Leif Madsen и Jim Van Meggelen, 978-1-449-332342-6.»

Если вы считаете, что использование вами примеров кода не соответствует справедливому использованию или предоставленному выше разрешению, не стесняйтесь обращаться к нам по *permissions@oreilly.com*.

Safari® Books Online



Safari Books Online (*www.safaribooksonline.com*) — это цифровая библиотека по требованию, которая предоставляет экспертный контент как в книжной, так и в видео форме от ведущих мировых авторов по технологиям и бизнесу.

Специалисты в области технологий, разработчики программного обеспечения, веб-дизайнеры и бизнес-профессионалы используют Safari Books Online в качестве основного ресурса для исследований, решения проблем, обучения и сертификации.

Safari Books Online предлагает широкий ассортимент продуктов и ценовые программы для организаций, правительственных учреждений, а также отдельных лиц. Подписчики имеют доступ к тысячам книг, учебные видеороликов и мануалов для публикации в одной полностью доступной для поиска базе данных от издателей, таких как O'Reilly Media, Prentice Hall Professional, Adison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, и десятки других. Для получения дополнительной информации о Safari Books Online, пожалуйста, посетите нас в Интернете.

Как связаться с нами

Пожалуйста, отправляйте комментарии и вопросы, касающиеся этой книги, издателю:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (в США или Канаде)

707-829-0515 (международный или местный)

707-829-0104 (факс)

У нас есть веб-страница для этой книги, где мы перечисляем ошибки, примеры и любую дополнительную информацию. Вы можете получить доступ к этой странице по адресу *http://bit.ly/asterisk_tdg_4E*.

Чтобы прокомментировать или задать технические вопросы об этой книге, отправьте письмо на *bookquestions@oreilly.com*.

Дополнительную информацию о наших книгах, курсах, конференциях и новостях см. на нашем веб-сайте по адресу *http://www.oreilly.com*.

Найдите нас на Facebook: *http://facebook.com/oreilly*

Следуйте за нами на Twitter: *http://twitter.com/oreillymedia*

Смотрите нас на YouTube: *http://www.youtube.com/oreillymedia*

Благодарности

Дэвиду Даффетту, спасибо за отличную главу о интернационализации, которая не была бы хорошо подана, написав её североамериканцы.

Затем мы хотим поблагодарить нашего фантастического редактора Майкла Лукидеса за его терпение в первом, втором и третьем изданиях этой книги, которые слишком долго выходили из-под земли и ожидались много долгих месяцев, чтобы наконец быть написанными. Майк предложил бесценную обратную связь и нашел невероятно тактичные способы указать нам переписать раздел (или главу), когда это было необходимо, и при этом заставить нас думать, что это была наша идея. Майк построил нас, когда мы упали, и вернул нас на землю, когда мы стали нахальными. Вы мастер, Майк, и видя, сколько книг получили ваш редакторский надзор, способствует пониманию того, почему O'Reilly Media - это успех.

Также благодаря остальным невоспетым героям в отделе производства О'Рейли. Это люди, которые берут нашу книгу и делают ее *книгой О'Рейли*.

Во время написания этой книги мы с удовольствием консультировались со многими людьми с особым опытом в различных областях. Их щедрые вклады времени и опыта сыграли важную роль в наших исследованиях. Благодарим Рэнди Ресник, организатор VoIP User Group; Кевин Флеминг; Ли Говард, автор iaxmodem и hylafax; Джошуа Колпа из Digium; Филлип Муллис из группы пользователей Asterisk в Торонто; Эллисон Смит - голос Asterisk; Флавио Е. Гонсалвес - автор книг по Asterisk, OpenSER и OpenSIPS; Дж. Окендо, гуру безопасности; Цафрир Коэн, источник знаний о безопасности и многом другом; Джефф Гельбах, для SNMP; Овидиу Сас, для ваших энциклопедических знаний SIP; Tomo Takebe, для некоторой помощи по SMDI; Майкл С. Уайт и e4 Strategies для оборудования Polycom; Стив Андервуд, за помощь с факсом и `spandsp`; и Ричард Гентнер и Джон Скрыт, за помощь в LDAP; Кинси Мур, обзор примера AMI Python; Лиза Улевич, которая помогла Алексу Балашову с его предисловием; и Кевину Макаллистеру, за то, что он позволил авторам играть на своем сервере Minecraft.

Кроме того, мы хотели бы поблагодарить Тилгмана Лешера за то, что помог обновить главу о том, что факс не умрет, и за предоставление надежный обзора других разделов книги.

Особую благодарность также следует отвести Джону Тодду за то, что он был одним из первых, кто написал всеобъемлющие инструкции по Asterisk, за все эти годы и за все другие вещи, которые вы делаете (и сделали) для сообщества Asterisk.

Открытая система публикации отзывов (OFPS)

Пока мы писали эту книгу, мы использовали O'Reilly Open Feedback Publishing System (OFPS), которая позволила нашей книге появляться в Интернете по мере ее написания. Члены сообщества смогли представить отзывов и комментариев, которые оказали нам огромную помощь. Ниже приводится список их имен или никнеймов:[1]

> Мэтью МакАуган, Мэтт Пусатери, Дэвид Ван Гиннекен, Астерикс Мания, Джаред Смит, Джейсон Паркер, Джованни Валлеси, Марк Петерсен, thp4, Дэвид Роу, tvc123, Фредерик Жан, Джон Тодд, Стивен Сокол, Лоран Штеффан, Роберт Дайли, Говард Харпер, Джозеф Ренсин, Говард Уайт, Джей Имес, Винсент Томассет, Дэйв Барнов, Себастьян Дио, Игорь Николаев, Аренд ван дер Колк, Анвар Хоссейн, крейгесмит, Нкабир, Анест, Николас Барнс, Алекс Нойман, Джастин Коркиннер, Стефан Шмидт, Пол Белэнджер, Йфинстрем, Род Монтгомери, Шей Эриссон, Гастон Драке, Ричард Гентнер, Майкл С Коллинз, Джо БобКутер, переигрыватель, препро, стгнет, Матье Д., Джефф Пилер, Билли Чиа, Антти Калласкари, Алана Грэма, Марка Петерсена и Уолтера Докеса.

Особая благодарность Мэтту Иордану за углубленный обзор AMI и других разделов четвертого издания книги.

Спасибо всем вам за ценный вклад в эту книгу.

Спасибо Шону Брайту, Эду Гаю, Саймону Дитнеру и Полу Беланджеру за помощь в разъяснении лучших практик для пользовательских и групповых политик для установки Asterisk. Раньше был простой способ установить Asterisk с правами *root* , но мы решили описать процесс установки, который больше соответствует лучшим практикам Linux (без начала холивара!), и эти прекрасные помощники способствовали нашим обсуждениям в этом.

1 Мы старались, когда это возможно, включать имена участников, но в некоторых случаях не могли и, следовательно, включали их никнеймы.

Честь и слава всем людям, работающим над FreeSWITCH, YATE, SER, Kamailio, OpenSIPS, SER, sipXecs, Woomera и любыми другими проектами с открытым исходным кодом для стимулирования новых мыслей и расширения границ.

Все в сообществе Asterisk также должны поблагодарить Джима Диксона за создание первых аппаратных интерфейсов телефонии с открытым исходным кодом, начало революции и предоставление его творений всему сообществу.

Наконец, и самое главное, спасибо Марку Спенсеру, оригинальному автору Asterisk и основателю Digium, за Asterisk, за Pidgin и за то, что он внес свой вклад в сообщество с открытым исходным кодом. Asterisk - ваше наследие!

Лейф Мэдсен

Что удивляет меня, где я начал с Asterisk, и куда я пошел с ней. В 2002 году, посещая школу, мы с друзьями экспериментировали с передачей голоса через Интернет с использованием продукта MSN от Microsoft. Он работал достаточно хорошо и позволял нам играть в видеоигры, разговаривая друг с другом, до тех пор, пока мы не захотели добавить третьего участника. Итак, я отправился искать какое-то программное обеспечение, которое могло бы обрабатывать несколько голосов (слово было конференц-связь, но я даже этого тогда не знал, в то время мало интересовался платформами УАТС). Я искал в Интернете, но не нашел ничего что бы мне понравилось (или чтобы было бесплатно).[2] Я обратился в IRC и объяснил, что я ищу. Кто-то (мне жаль, что я не знал, кто) сказал, что я должен проверить какое-то программное обеспечение под названием Asterisk (он, по видимому подумал, что я искал `MeetMe()`, которое я нашел).

Имея имя, я схватил программное обеспечение и начал смотреть, что оно может сделать. Невероятно, функциональность, которую я искал, которая, как я думал, будет всей программой, была только одним компонентом в море функциональности. И запустив BBS в течение многих лет до поступления в колледж, факт, что я мог установить PCI-карту и подключить ее к телефонной сети, не закончилась на мне. Спустя пару часов взглянув на программное обеспечение и собрав его, я начал рассказывать одному из моих учителей о PCI-картах и как, может быть, мы могли бы получить некоторые лаборатории для класса и т.д. (В нашем классе было 30 компьютеров, на 10 столах по 3). Ему понравилась идея и он начал говорить с координатором программы, и в течение примерно 30 минут был сделан заказ на 20 карт. Довольно удивительно, учитывая, что это были TDM400Ps, украшенные четырьмя дочерними картами, и они услышали о них только за час до этого.

Затем началась одержимость. Я проводил каждый свободный момент этого семестра с несколькими компьютерами, посвященными использованию Asterisk. За эти два месяца я многому научился. Затем у нас был перерыв в кооперации. Я не нашел сразу никакой работы, поэтому я переехал домой и продолжил работу над Asterisk, проводя время в IRC, просматривая примеры, опубликованные Джоном Тоддом, и просто пытаясь понять, как работает программное обеспечение. К счастью, у меня была большая помощь в IRC (это были дни, предшествующие любой документации на Asterisk), и я узнал намного больше в течение этого семестра.

Увидев что люди, которые проявляли большой интерес к Asterisk в то время, имели сильное чувство общности, заставило меня также хотеть внести свой вклад. Не имея практического уровня знаний о кодинге, я решил что документация будет чем-то полезным для начала. Кроме того, я писал много работ в школе, так что это у меня получалось лучше. Однажды ночью я разместил веб-сайт под названием «Назначение документации Asterisk» (TADA) и начал записывать любую документацию, которую мог. Через пару недель мы с Джаредом Смитом начали общаться, что привело к созданию Проекта документации Asterisk с целью написания книги Asterisk для сообщества. Этот проект стал основой первого издания книги «*Asterisk: будущее телефонии*».

Спустя одиннадцать лет я все еще пишу документацию Asterisk, став основным менеджером по выпуску ошибок и менеджером релизов проекта Asterisk, выступая на каждом AstriCon с 2004 года (на котором мы с Джаредом говорили о проекте документации Asterisk; у меня все еще есть магнит AsteriskDocs, сделанный его женой), и стал консультантом, специализирующимся на интеграции баз

2 Много лет спустя, играя в MMORPG (многопользовательскую ролевую онлайн-игру), я узнал о таких приложениях, как TeamSpeak; вероятно, хорошо, что я не нашел их тогда.

данных (спасибо Тильгману за `func_odbc`) и кластеризации (спасибо Марку Спенсеру за DUNDi). Мне очень нравится Asterisk и все, что он позволил мне сделать. Сейчас я ведущий инженер по системам унифицированных коммуникаций в Thinking Phone Networks, где я продолжаю строить и расширять огромный спектр телекоммуникационных функций.

Во-первых, спасибо моим родителям Рику и Кэрол, за понимание и поддержку во всем, что я сделал в своей жизни. С первого компьютера, который они купили за слишком большие деньги, когда я был в 6 классе (я начал интересоваться компьютерами со 2-го класса благодаря Commodore 64, и они купили мне компьютер спустя несколько лет после интервью родителей и учителей) чтобы позволить мне использовать домашнюю телефонную линию для моих начинаний BBS (и в конечном итоге получить мне мою собственную телефонную линию), и все остальное что они сделали для меня, я никогда не смогу отблагодарить их. Я люблю вас обоих больше, чем вы можете себе представить.

Спасибо моей бабушке Т за то, что разрешила мне использовать ее 286 в те годы, когда у меня не было дома компьютера, и за то, что каждый год брала меня с собой по магазинам в мой день рождения в течение 15 лет. Люблю сильно!

Моей прекрасной жене, Даниэлле, за то, что каждое утро, перед уходом на работу, ставила будильник, давала мне поспать лишние 10 минут перед началом работы над этой книгой, и понимала, когда мне нужно было работать допоздна, потому что я пропустил 9 утра, когда перестал писать, спасибо, я так сильно тебя люблю. (Также нашему сыну, который скоро родится, который помог мне поставить твердую дату сдачи проекта этой книги :))

Есть очень много людей, которые помогают мне и учат меня новым вещам каждый день, но наибольшее влияние на мою жизнь в Asterisk заключается в следующем: Марк Спенсер за написание программного обеспечения, что дало мне потрясающую карьеру; Джон Тодд за его ранние примеры; Брайан К. Уэст за его раннюю помощь и энтузиазм в IRC; Стив Сокол и Олле Йоханссон за мой первый полет на AstriCon (и последующие!) и позволение мне быть частью первых учебных классов Asterisk; Джаред Смит за помощь в запуске проекта документации и выполнении всей инфраструктуры, которую я никогда не мог бы сделать; Джим Ван Меггелен за вступление в начале проекта и обучения меня новым способам смотреть на жизнь; и Расселл Брайант за то, что он отличный друг и доверенное лицо, никогда не нарушал наш FriendDA, и за то, что не держишь обиду.

Джим Ван Меггелен

Когда мы намеревались написать самое первое издание этой книги в 2004 году, мы были уверены, что Asterisk будет иметь огромный успех. Теперь, почти десятилетие спустя, мы написали это четвертое издание того, что мировое сообщество Asterisk называет “книгой Asterisk”, и мы выросли из революционеров в профессионалов Asterisk.

Asterisk доказал, что open source телекоммуникации - это долгосрочная идея, и ландшафт open source телекоммуникаций в настоящее время дополняется не только Asterisk. Проекты, как sipXecs (от SIPfoundry), OpenSER/Kamailio/OpenSIPS, и многие-многие другие (и больше) помогут завершить экосистему.

Я хочу воспользоваться этой возможностью, чтобы поблагодарить моего очень хорошего друга Лейфа Мэдсена, который был со мной на протяжении всех четырех изданий. В нашей повседневной жизни, у нас не всегда есть возможность работать друг с другом (или даже захватить пинту, в эти дни!), и всегда приятно работать с вами. Я также хочу поблагодарить Рассела Брайанта, который присоединился к нам для этого издания, и чья преданность этому проекту и проекту Asterisk в целом вдохновляет меня. Ты Человек эпохи Возрождения, Рассел. Для Джареда Смита, который помог найти проект документации Asterisk и стал соавтором первых двух изданий с Лейфом и мной (но с тех пор перешел к проекту Fedora), я могу только сказать: потеря Asterisk - это выигрыш Fedora.

Я хотел бы поблагодарить своих деловых партнеров в Core Telecom Innovations и iConverged LLC, без которых я не мог бы делать все крутые вещи, которые я делаю в своей профессиональной карьере.

Я хотел бы поблагодарить всех моих друзей в импровизированном сообществе, за то, что помогают мне продолжать смеяться над всеми проблемами, которые предоставляет жизнь.

Спасибо всей моей семье, которая привносят любовь в мою жизнь.

Наконец, спасибо вам, сообщество Asterisk. Эта книга-наш подарок вам. Мы надеемся, что вам понравится читать её так же, как нам понравилось её писать.

Рассел Брянт

Я начал работать над Asterisk в 2004 году. Я был студентом Университета Клемсона и работал в качестве кооперативного инженера в ADTRAN в Хантсвилле, штат Алабама. Моя первая работа в ADTRAN была в Отделе квалификации продукции. Я помню с Китом Морганом использовали Asterisk в качестве генератора VoIP-трафика для тестирования QoS через тестовую сеть маршрутизатора. Между тем, товарищ по кооперативу и друг, Адам Шрайбер, представил мне Марка Спенсера. В течение следующих шести месяцев я погрузился в Asterisk. Я узнал как можно больше об Asterisk, телефонии и программировании на Си. Когда Asterisk 1.0 была выпущена осенью 2004 года, я был назначен сопровождающим выпуска.

В начале 2005 года меня нанял Digium для продолжения работы над Asterisk профессионально. Я провел семь удивительных лет, работая в Digium над улучшением Asterisk.

Я работал разработчиком программного обеспечения, руководителем группы разработчиков программного обеспечения и инженером группы развития Asterisk. Я чрезвычайно благодарен за возможность внести свой вклад во многие области проекта Asterisk. Есть много людей, которые заслуживают благодарности за поддержку, которую они оказали на этом пути.

Моей жене, Джули, я не могу отблагодарить тебя за всю любовь и поддержку, которые ты дала мне. Спасибо, что поддерживаешь мою жизнь сбалансированной и счастливой. Ты самая лучшая из лучших. Я люблю тебя!

Моим родителям, спасибо за то, что дали мне так много больших возможностей в моей жизни, чтобы исследовать разные вещи и найти то, что мне действительно нравится. Вы научили меня усердно работать и никогда не отдавать верх.

Лейфу и Джиму, спасибо за приглашение внести свой вклад в эту книгу. Это был интересный проект, во многом благодаря удовольствию работать с вами двумя. Спасибо за смех и за вашу преданность этой книге как командному усилию.

Я многому научился у многих людей в Digium. Есть три человека, которые выделяются больше всего в качестве моих наставников: Марк Спенсер, Кевин Флеминг и Дэвид Дитон. Благодарю вас всех за то, что сделали все возможное, чтобы научить меня на этом пути. Я чрезвычайно благодарен.

Сообщество разработчиков Asterisk благодарит вас всех за усердную работу и преданность делу. Я многому научился у всех вас. Было удовольствием работать с Вами эти годы.

Трэвису Акстеллу спасибо за помощь в первые дни моего знакомства с Linux и за то, что был хорошим другом.

Моим собакам, Хлои и Бакстеру, спасибо, что составляете мне компанию, пока я работаю над книгой каждое утро.

Всем моим друзьям и семье, спасибо за вашу любовь, поддержку и веселье.

Спасибо всему сообществу Asterisk за использование, удовольствие и вклад в Asterisk. Надеемся, вам понравится книга!

ГЛАВА 1

Революция в телефонии

Сначала они игнорируют тебя, затем они смеются над тобой,

потом они сражаются с тобой, потом ты побеждаешь.

—Махатма Ганди

Когда мы впервые собрались в 2004 году, чтобы написать книгу об Asterisk, мы уверенно предсказали, что Asterisk кардинально изменит телекоммуникационную отрасль. Сегодня революция, которую мы предсказали, почти завершена. Asterisk - самая успешная частная телефонная система (PBX - УАТС) в мире и является принятой (хотя, возможно, не всегда любимой) технологией в телекоммуникационной отрасли.

К сожалению, за последние девять лет телекоммуникационная индустрия продолжает терять свой путь. Методы, с помощью которых мы общались, изменились. Если 20 лет назад телефонные звонки были предпочтительным способом общения на разных расстояниях, текущая тенденция - общение через текст (электронная почта, чат и т.д.). Телефонный звонок рассматривается как немного мертвая вещь, особенно грядущими поколениями.

Asterisk остается довольно устрашающей технологией, и мы полагаем, что она по-прежнему остается одной из лучших надежд на любую разумную интеграцию между телекоммуникационными компаниями и всеми другими технологиями, с которыми, возможно, захочется соединиться.

С Asterisk никто не навяжет вам, как должна работать ваша телефонная система или какими технологиями вы ограничены. Если вы этого хотите, вы можете получить это. Asterisk с любовью отражает концепцию соблюдения стандартов, а также пользуется свободой разрабатывать собственные инновации. То, что вы решите реализовать, зависит от вас - Asterisk не накладывает ограничений.

Естественно, эта невероятная гибкость поставляется с ценой: Asterisk - это не простая система для настройки. Это не потому, что она нелогична, сбивает с толку или загадочна; напротив, она очень

разумна и практична. Глаза людей загораются, когда они впервые видят диалплан Asterisk и начинают размышлять о возможностях. Но когда есть буквально тысячи способов достижения результата, этот процесс, естественно, требует дополнительных усилий. Возможно, это можно сравнить с постройкой дома: компоненты относительно легко понять, но человек, рассматривающий такую задачу, должен либо: а) привлекать компетентную помощь или б) развивать необходимые навыки посредством обучения, практики и хорошей литературы по предмету.

Asterisk and VoIP: Преодоление разрыва между традиционной и сетевой телефонией

Voice over IP (VoIP) (голос поверх IP) часто воспринимается как нечто большее, чем метод получения бесплатных междугородных звонков. Реальная ценность (и, если честно, проблема) VoIP заключается в том, что она позволяет голосу стать ни чем иным, как другим приложением в сети передачи данных.

Иногда кажется, что мы забыли, что цель телефона - позволить людям общаться. На самом деле это простая цель, и нам должно быть возможно сделать это гораздо более гибкими и творческими способами, чем в настоящее время. Такие технологии, как Asterisk, снижают барьеры для входа.

Проект телефонии Zapata

Когда проект Asterisk был запущен (в 1999 году), существовали другие проекты телефонии с открытым исходным кодом. Тем не менее, Asterisk, в сочетании с проектом телефонии Zapata, смогла предоставить интерфейсы с коммутируемой телефонной сетью общего пользования (PSTN - ТфОП), что стало важной вехой в переходе программного обеспечения от чего-то чисто сетевого к чему-то более практичному в мире телекоммуникации, который был PSTN-ориентированным.

Проект телефонии Zapata был задуман Джимом Диксоном, инженером по телекоммуникационному консалтингу, который был вдохновлен невероятными достижениями в скорости процессора, которые компьютерная индустрия стала воспринимать как нечто само собой разумеющееся. По мнению Диксона, гораздо более экономичные системы телефонии могли бы быть созданы, если бы существовала карта, на которой не было ничего больше, чем базовые электронные компоненты, необходимые для взаимодействия с телефонной линией. Вместо того, чтобы иметь дорогостоящие компоненты на карте, обработка цифрового сигнала (DSP)[1] будет обрабатываться в CPU программным обеспечением. Хотя это наложило бы огромную нагрузку на процессор, Диксон был уверен, что низкая стоимость процессоров по сравнению с их производительностью сделала их гораздо более привлекательными, чем дорогие DSP, и, что более важно, что соотношение цена/производительность будет продолжать уменьшаться, поскольку процессоры продолжают увеличиваться в мощности.

Как и многие провидцы, Диксон считал, что многие другие увидят эту возможность и что ему просто нужно ждать, пока кто-то другой создаст то, что для него было очевидным улучшением. Через несколько лет он заметил, что не только никто не создал эти карты, но вряд ли кто-нибудь когда-либо собирался.

В этот момент было ясно, что, если он хочет революции, ему придется начать её самому. Так появился проект телефонизации Zapata:

> Поскольку эта концепция была настолько революционной и, несомненно, внесла много волнений в отрасли, я обратился к мотивам революции в Мексике и назвал технологию и организацию по имени известного мексиканского революционера Эмилиано Запата. Я решил назвать карточку «tormenta», что по-испански означает «шторм» и, обычно, подразумевает большой шторм, например, ураган или что-то подобное.[2]

1 Термин DSP также означает цифровой сигнальный процессор, который является устройством (обычно чипом), сспособным интерпретировать и модифицировать сигналы различного рода. В голосовой сети DSP в первую очередь отвечают за кодирование, декодирование и транскодирование аудиоинформации. Это может потребовать много вычислительных усилий.

2 Jim Dixon, “The History of Zapata Telephony and How It Relates to the Asterisk PBX”.

Возможно, мы должны называть себя Астеристами. Несмотря на это, мы обязаны Джиму Диксону отчасти за то, что он придумал все это и за то, что дали результаты его усилий сообществу с открытым исходным кодом. В результате вклада Джима появился двигатель телефонной сети общего пользования Asterisk.

На протяжении многих лет интерфейс Zapata Telephony в Asterisk был модифицирован и улучшен. Интерфейс телефонии аппаратного устройства Digium Asterisk (Digium Asterisk Hardware Device Interface - DAHDI) , который используется сегодня, является потомком вклада Джима Диксона.

Масштабные изменения требуют гибкости технологии

У каждой АТС есть недостатки. Независимо от того, насколько полно это возможно, что-то всегда будет упущено, потому что даже самая многофункциональная АТС всегда будет не в состоянии предвидеть творчество клиента. Небольшая группа пользователей захочет получить необычную функцию, которую команда разработчиков не думала разрабатывать или она не могла оправдать затраты на создание, и, поскольку система закрыта, пользователи не смогут ее самостоятельно построить.

Если бы Интернет, таким же образом, был затруднен регулированием и коммерческими интересами, сомнительно, что он получил бы широкое признание, которым он в настоящее время пользуется. Открытость Интернета означала, что любой может позволить себе участвовать. Так все и сделали. Десятки тысяч умов, которые сотрудничали в создании Интернета, создали то, чего никогда не смогла бы создать ни одна корпорация.[3]

Как и во многих других проектах с открытым исходным кодом, таких как Linux и прочее важное программное обеспечение, работающее в Интернете, развитие Asterisk подпитывалось мечтами людей, которые знали, что должно быть что-то большее, чем то, что производят традиционные отрасли. Эти люди знали, что если можно взять лучшие части различных АТС и разделить их на взаимосвязанные компоненты, похожие на ящик кирпичей LEGO, можно было бы начать понимать вещи, которые не выдержали бы традиционного корпоративного процесса анализа рисков. Хотя никто не может всерьез утверждать, что имеет полное представление о том, как это должно выглядеть, нет недостатка в мнениях и идеях.[4]

Многие люди, знакомые с Asterisk, считают его незаконченным. Возможно, этих людей можно уподобить посетителям арт-студии, которые хотят получить подписанный, пронумерованный отпечаток. Они часто остаются разочарованными, потому что обнаруживают, что Asterisk - это пустой холст, тюбики с краской и неиспользованные кисти.[5]

Успех Asterisk можно напрямую отнести на счет воспитания большим количеством художников, чем любая другая АТС. Большинство производителей посвящают не более нескольких разработчиков одному продукту; Asterisk имеет множество. Большинство фирменных УАТС имеют всемирную группу поддержки, состоящую из нескольких десятков реальных экспертов; Asterisk имеет сотни.

Глубина и широта опыта, который окружает этот продукт, не имеет себе равных в телекоммуникационной отрасли. Asterisk пользуется большим вниманием старых парней телекоммуникационных компаний, которые помнят дисковый номеронабиратель, сотрудников телекоммуникационных компаний, которые помнят времена, когда голосовая почта была самой горячей новой технологией, и гиков и специалистов по передачи данных, которые помогали создавать

3 Мы понимаем, что технология Интернета формируется из государственных и академических институтов, но речь идет не о технологии Интернета, а о культурном феномене, который взорвался в начале 90-х годов.

4 Между версиями Asterisk 1.2 и Asterisk 1.4 было добавлено более 4000 обновлений кода в репозитории SVN. Между версиями Asterisk 1.4 и 1.8 было сделано более 10 000 обновлений.

5 Следует отметить, что эти люди не должны разочаровываться. Несколько проектов возникли, чтобы снизить барьеры для входа в Asterisk. Безусловно, самым популярным и известным является интерфейс FreePBX (и множество проектов на его основе). Эти интерфейсы - веб-сайт VoIP-Info может дать вам представление о том, что цель их - не облегчить изучение Asterisk, поскольку они отделяют вас от конфигурации платформы или диалплана, но многие из них предоставят вам рабочую АТС гораздо быстрее, чем более практический подход, который мы используем в этой книге.

Интернет. Все эти люди придерживаются общего мнения, что индустрия телекоммуникаций нуждается в революции.[6]

Астериск является катализатором.

Asterisk: хакерская АТС

Телекоммуникационные компании, которые предпочитают игнорировать Asterisk, делают это на свой страх и риск. Гибкость, которую она обеспечивает, создает возможности, о которых едва ли могут мечтать лучшие коммерческие системы. Это связано с тем, что Asterisk - это наивысшая хакерская АТС.

Разумеется, термин *хакер* был искажен средствами массовой информации в смысле «злонамеренный взломщик». Это печально, потому что этот термин фактически существовал задолго до того, как СМИ испортили его значение. Хакеры создали сетевой движок, который является Интернетом. Хакеры создали Apple Macintosh и операционную систему Unix. Хакеры также строят вашу следующую телекоммуникационную систему. Не бойтесь, это хорошие ребята, и они смогут построить систему, которая намного безопаснее, чем все, что существует сегодня. Хакеры могут быстро реагировать на изменяющиеся тенденции в области безопасности и точно настраивать телефонную систему в ответ как на корпоративную политику, так и передовую отраслевую практику. Вместо того, чтобы поддерживать сомнительную и низкую безопасность закрытых систем, хакеры могут быстро реагировать.

Как и другие системы с открытым исходным кодом, Asterisk сможет развиться в гораздо более безопасную платформу, чем любая собственная система, не несмотря на ее хакерские корни, а скорее благодаря им.

Asterisk: АТС профессионалов

Никогда в истории телекоммуникаций не было системы, которая бы соответствовала потребностям бизнеса, любой ценовой категории. Asterisk - это технология, обеспечивающая решение, и, как и в случае с Linux, становится все реже встречаться предприятие, в котором не работает какая-либо версия Asterisk, в некотором качестве в сети, решая проблемы, которые способна решить только Asterisk.

Это принятие, скорее всего, произойдет гораздо быстрее, чем в Linux, хотя бы по нескольким причинам:

- Linux уже проложил путь, который привел к принятию продуктов с открытым исходным кодом. Asterisk следует за этим путем.
- Телекоммуникационная индустрия искалечена, и руководство этой гигантской индустрии не получает лидерства. У Asterisk есть убедительное, реалистичное и захватывающее видение.
- Конечным пользователям поставляются несовместимые системы с ограниченной функциональностью и ужасной поддержкой. Asterisk решает первые две проблемы; предприниматели и сообщество обеспечат второе.

Сообщество Asterisk

Одной из сильных сторон Asterisk является страстное сообщество, которое разработало и поддерживает его. Это сообщество, возглавляемое прекрасными людьми в Digium, остро осознает культурную значимость Asterisk и имеет оптимистичный взгляд на будущее.

Одним из наиболее сильных побочных эффектов энергии сообщества Asterisk является сотрудничество, которое породило среду специалистов в области телекоммуникаций, сетей и информационных технологий, которые разделяют любовь к этому явлению. В то время как эти кадры

6 Телеком-индустрия предсказывает революцию с момента аварии; время покажет, насколько хорошо он реагирует на революцию с открытым исходным кодом

традиционно расходились друг с другом, в сообществе Asterisk они восхищаются навыками друг друга. Значение этого сотрудничества нельзя недооценивать.

Если мечта об Asterisk должна быть реализована, сообщество должно продолжать расти, но одной из ключевых проблем, с которыми сталкивается сейчас сообщество, является быстрый приток новых пользователей. Члены существующего сообщества, родившие вещь под названием Asterisk, как правило, приветствуют новых пользователей, но они стали нетерпеливыми, задаваясь вопросами, ответы на которые часто могут быть получены самостоятельно, если они готовы посвятить некоторое время исследованиям и экспериментам.

Очевидно, что новые пользователи не могут быть одинаковыми. В то время как некоторые из них будут счастливо часами экспериментировать и читать различные блоги, описывающие опыт и неудачи других людей, другие же, которые с энтузиазмом относятся к этой технологии, совершенно не заинтересованы в таких занятиях. Они хотят простое пошаговое руководство, которое поможет им запустить систему, и иметь некоторые разумные примеры, описывающие лучшие методы реализации общих функций (таких как голосовая почта, автосекретарь и т.п.).

Для членов экспертного сообщества, которые (правильно) считают, что Asterisk похож на язык веб-разработки, этот подход не имеет никакого смысла. Для них ясно, что вам нужно погрузиться в Asterisk, чтобы оценить его тонкости. Будете ли вы просить пошаговое руководство по программированию и рассчитывать извлечь из него все, что может предложить язык?

Ясно, что нет единого подхода, который подходит всем. Asterisk - совсем другое животное, и для него требуется совершенно другой разум. Однако, изучая сообщество, имейте в виду, что в него входят люди со множеством разных навыков и взглядов. Некоторые из этих людей не проявляют большого терпения к новым пользователям, но это часто связано с их страстью к теме, а не потому, что они не приветствуют ваше участие.

Списки рассылки Asterisk

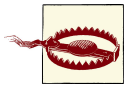
Как и в любом сообществе, есть места, где встречаются члены сообщества Asterisk для обсуждения вопросов, представляющих взаимный интерес. Списки рассылки вы найдете по адресу *http://lists.digium.com*, эти три темы в настоящее время наиболее важны:

Asterisk-Biz

> В этой рассылке содержится любая коммерческая информация об Asterisk. Если вы продаете что-то связанное с Asterisk - продайте это здесь. Если вы хотите купить услугу или продукт Asterisk - пишите сюда.

Asterisk-Dev

> Здесь обитают разработчики Asterisk. Цель рассылки - обсуждение разработки программного обеспечения Asterisk, и его участники энергично отстаивают эту цель. Ожидайте шквал негодования, если вы разместите что-нибудь в этой рассылке, не относящееся конкретно к программированию или разработке базы кода Asterisk. Общие требования к кодингу (например, запросы по взаимодействию с AGI или AMI) должны быть направлены в список *Asterisk-Users*.



> Список Asterisk-Dev не является поддержкой второго уровня! Если вы просматриваете архивы списков рассылки, вы увидите, что это строгое правило. Список рассылки Asterisk-Dev посвящен обсуждению основного развития Asterisk, и вопросы о взаимодействии внешних программ с помощью AGI или AMI должны быть опубликованы в списке *Asterisk-Users*.

Asterisk-Users

> Здесь обитают большинство пользователей Asterisk. Этот список генерирует несколько сотен сообщений в день и имеет более десяти тысяч подписчиков. В то время как вы можете обратиться за помощью, вы должны попытаться найти ответ самостоятельно, прежде чем отправлять запрос.

Wikipedia Asterisk

Digium поддерживает wiki для Asterisk на *wiki.asterisk.org*. Этот сайт постоянно обновляется командой Digium, а скрипты экспортируют документацию на основе XML из источника Asterisk в саму wiki, что делает вас уверенным в том, что данные, которые вы читаете, являются достоверными.

Более старая вики существует на www.voip-info.org, которая в наши дни представляет собой некоторое историческое любопытство и источник большого просветления и путаницы. Хотя здесь содержится огромное количество информации, большая часть ее устарела.

IRC-каналы

Сообщество Asterisk поддерживает каналы интернет-ретрансляции (IRC) на *irc.freenode.net:6667*. Два наиболее активных канала: *#asterisk* и *#asterisk-dev*.[7] Чтобы сократить вмешательство спамботов, оба этих канала требуют регистрации для присоединения. Чтобы зарегистрироваться, запустите /msg nickserv, когда вы подключитесь к службе через ваш любимый IRC-клиент.

Группы пользователей Asterisk

За последнее десятилетие, во многих городах мира, одинокие пользователи Asterisk стали понимать, что в их городах есть другие единомышленники. Группы пользователей Asterisk (AUG) начали всплывать повсюду. Хотя эти группы не имеют никакой официальной принадлежности друг к другу, они обычно ссылаются на сайты друг друга и приветствуют членов из любой точки мира. Введите «Asterisk User Group» в Google, чтобы отследить их в своем регионе.

Проект документации Asterisk

Проект документации Asterisk был начат Лейфом Мэдзеном и Джаредом Смитом, но несколько человек в сообществе внесли свой вклад.

В рамках усилий проекта Asterisk Docs эта книга была выпущена под лицензией Creative Commons.

Деловой пример

В наши дни очень редко можно найти предприятия, которому не приходилось бы регулярно перестраиваться. Так же сложно найти бизнес, который может позволить себе заменять свою инфраструктуру связи каждый раз, когда он собирается идти в новом направлении. Сегодняшние предприятия нуждаются в максимальной гибкости во всех своих технологиях, включая телекоммуникационные.

В своей книге «Пересечение пропасти» (HarperBusiness, 2002) Джеффри Мур утверждает: «Идея о том, что ценность системы будет раскрываться постепенно, а не будет известна во время установки, подразумевает, в свою очередь, гибкость и адаптивность продукта, а также обслуживание клиентов, эти факторы должны быть важными компонентами контрольного списка для оценки перед покупкой системы». Это частично означает, что истинная ценность технологии часто неизвестна до тех пор, пока она не будет развернута.

Итак, насколько убедительно иметь систему, которая в своей основе придерживается концепции открытости и ценности непрерывных инноваций.

Вывод

С чего начать? Что ж, когда дело доходит до Asterisk, говорить можно столько, что одной книги не хватит. Эта книга может только описать основы, но из этого вы сможете прийти к пониманию концепции Asterisk - и исходя из этого, кто знает, что вы будете строить?

---

7 Канал #asterisk-dev предназначен для обсуждения изменений базовой базы Asterisk и также не является поддержкой второго уровня. Обсуждения, связанные с программированием внешних приложений, которые взаимодействуют с Asterisk через AGI или AMI, предназначены для использования в #asterisk.

ГЛАВА 2

Архитектура Asterisk

Сначала с начала, но не обязательно в таком порядке.

—Доктор Кто

Asterisk сильно отличается от других, более традиционных АТС, поскольку диалплан в Asterisk рассматривает все входящие каналы практически одинаково.

В традиционной АТС существует логическое различие между станциями (телефонными аппаратами) и транками (ресурсами, которые подключаются к внешнему миру). Это означает, например, что вы не можете установить внешний шлюз на порт станции и перенаправлять внешние вызовы на него, не требуя, чтобы ваши пользователи сначала набирали внутренний номер. Кроме того, концепцию автономного ресурса (например, приемной) гораздо сложнее реализовать на традиционной АТС, поскольку система не позволит внешним ресурсам иметь доступ к внутренним функциям. Справедливости ради следует отметить, что многие традиционные АТС предлагают такую функциональность. Тем не менее, она обычно клонируется, ограничивается в функциях и требует сложного, запатентованного программного обеспечения, которое должно быть установлено в АТС (например, расширения для конкретного поставщика).

С другой стороны, у Asterisk нет внутренней концепции транков или станций. В Asterisk все, что приходит или выходит из системы, проходит через какой-то канал. Существует много разных каналов; однако диалплан Asterisk обрабатывает все каналы аналогичным образом, и это означает, что, например, внутренний пользователь может существовать на конце внешней соединительной линии (например, сотового телефона) и обрабатываться диалпланом точно так же, как это был бы обычный пользователь и находился на внутреннем номере. Если вы не работали с традиционной АТС, возможно, не сразу видно, насколько это мощно и освобождает. Рисунок 2-1 иллюстрирует различия между двумя архитектурами.

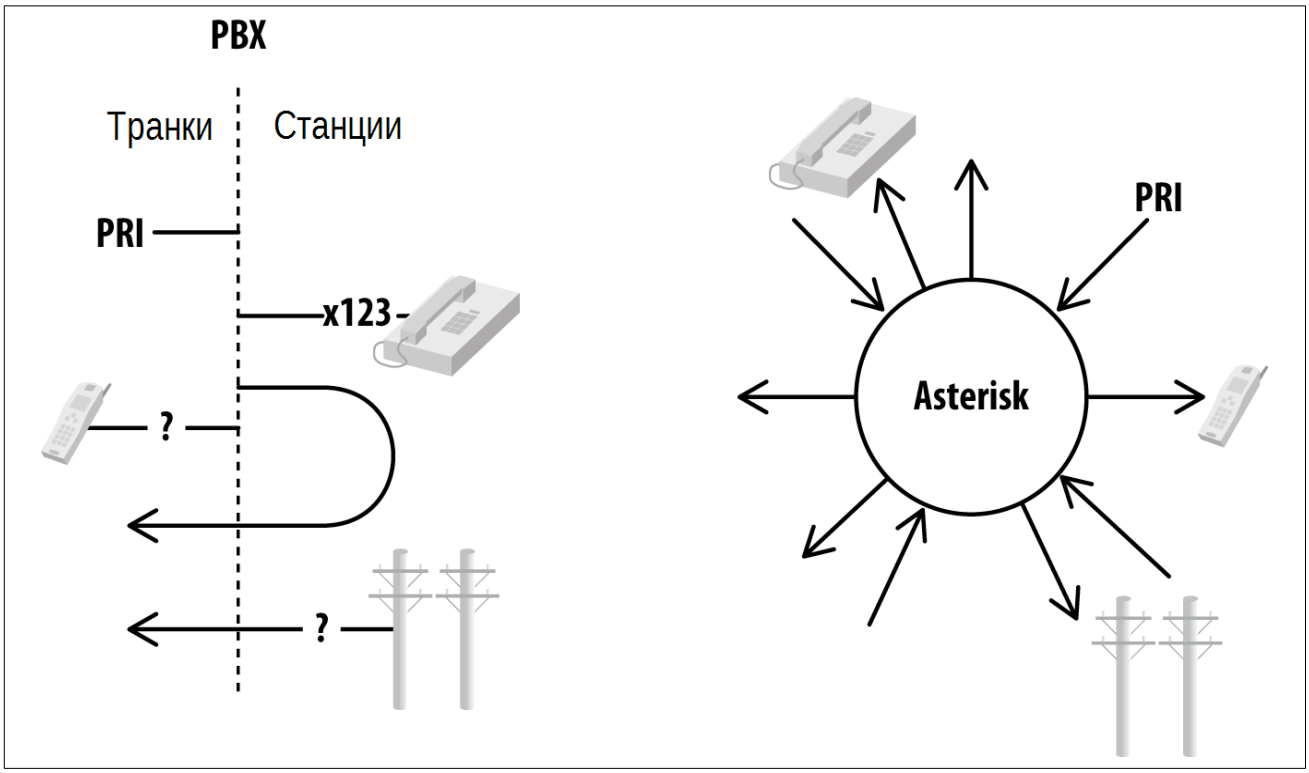


Рисунок 2-1. Архитектура Астериск и АТС

Модули

Asterisk построен на модулях. Модуль является загружаемым компонентом, который предоставляет определенные функции, такие как драйвер канала (например, *chan_sip.so*), или ресурс, который позволяет подключаться к внешней технологии (например, *func_odbc.so*). Модули Asterisk загружаются на основе файла */etc/asterisk/modules.conf*. Мы обсудим использование многих модулей в этой книге. На этом этапе мы просто хотим представить концепцию модулей и дать вам представление о типах доступных модулей.

На самом деле можно запустить Asterisk без каких-либо модулей, хотя в этом состоянии он не сможет ничего сделать. Полезно понять модульную природу Asterisk, чтобы оценить архитектуру.



Вы можете запустить Asterisk без модулей, загруженных по умолчанию, и загружать каждый желаемый модуль вручную с консоли, но это не то, что вы хотите ввести в производство; это было бы полезно только в том случае, если вы выполняли настройку производительности системы, в которой вы хотели исключить все, что не требуется вашим конкретным приложением Asterisk.

Типы модулей в Asterisk включают следующее:

- Приложения
- Модули соединений
- Модуль детальной записи вызовов (CDR)
- Модуль регистрации события канала (CEL)
- Драйверы канала
- Конверторы кодеков
- Интерпретаторы формата
- Функции диалплана
- Модули PBX
- Модули ресурсов

- Дополнительные модули
- Тестовые модули

В следующих разделах мы перечислим каждый модуль, доступный в этих категориях, кратко определим его цель и выскажем наше мнение о его относительной популярности и/или важности (в то время как некоторые модули доказанно и заслуженно популярны, другие - довольно старые, едва ли когда-либо используемые и поддерживаются только с целью обратной совместимости). Подробная информация о том, как работают определенные модули, будет рассмотрена в разных главах всей книги, в зависимости от того, что представляет собой модуль и что он делает. Некоторые модули будут рассмотрены полностью; другие не могут быть охвачены вообще.

Что касается столбца Популярность/Статус в следующих таблицах, следующий список содержит наше мнение относительно значений, которые мы выбрали (ваше мнение может отличаться):

Ничтожный

Этот модуль - древняя история. Если вы используете его, имейте в виду, что вы в основном сами по себе, когда речь заходит о какой-либо поддержке сообщества.

Ненадежный

Этот модуль является новым или экспериментальным и не подходит для продакшена.

Полезный

Этот модуль является актуальным, поддерживается, популярен и рекомендуется.

Годный

Этот модуль работает, но может быть неполным или непопулярным и/или не рекомендуется авторами.

Новый

Этот модуль является совершенно новым, и его полноту и популярность в настоящее время трудно измерить.

Устаревший

Этот модуль был заменен чем-то, что считается превосходящим.

Ограниченный

Этот модуль имеет ограничения, которые могут сделать его непригодным для ваших требований.

Существенный

Это модуль, который вам обязательно понадобится.

В дополнение к нашим мнениям о различных модулях, есть также официальный список типов статуса поддержки, включенных в *menuselect*. В каждой группе типов модулей список сортируется по подгруппам, таким как:

core (ядро)

Основные модули, которые поставляются с Asterisk, получают наибольший уровень поддержки. В дополнение к развитию сообщества, это модули, на которые команда разработчиков Digium фокусирует свои усилия. Asterisk - это обширная часть программного обеспечения, и для обеспечения ее эффективности наибольшее количество ресурсов, финансируемых Digium, применяется к модулям этого уровня поддержки.

расширенные

Расширенные модули, которые поддерживаются сообществом. Поиск проблем и отчетность о них для отслеживания по-прежнему будет сортироваться, как и все отправленные проблемы, но может не получить должного внимания.

устаревшие

Устаревшие модули не получают поддержки и обычно не поддерживаются. Модули, которые попадают в категорию устаревших, должны иметь альтернативные модули как в *расширенном*, так и в состоянии поддержки *ядро (core)*.

Вместо дублирования данных, которые могут быстро устаревать, мы укажем вам на *menuselect* для получения официального списка. Статус переменных модулей также динамичен среди основных выпусков Asterisk, по мере добавления новых функциональных возможностей, а существующие модули перемещаются в статусы *расширенный* и *устаревший*.

И теперь, без лишнего шума, давайте посмотрим на модули, сгруппированные по типу модуля.

Приложения

Приложения диалплана используются в *extensions.conf* для определения различных действий, которые могут быть применены к вызову. Например, приложение `Dial()` отвечает за выполнение исходящих подключений к внешним ресурсам и, возможно, является самым важным приложением диалплана. Доступные приложения перечислены в Таблице 2-1.

Таблица 2-1. Приложения диалплана

Имя	Назначение	Популярность/Статус
`app_adsiprog`	Загружает сценарии интерфейса аналогового отображения (ADSI) в совместимые аналоговые телефоны	Ничтожный
`app_alarmreceiver`	Поддерживает получение аварийных отчетов от оборудования	Ничтожный
`app_amd`	Обнаруживает автоответчики	Ненадежный
`app_authenticate`	Сравнивает двухтональный многочастотный (DTMF) ввод с предоставленной строкой (паролем)	Полезный
`app_cdr`	Делает специальную запись в CDR	Полезный
`app_celgenuserevent`	Создает пользовательские события для CEL	Новый
`app_chanisavail`	Проверяет состояние канала	Ничтожный
`app_channelredirect`	Перенаправляет канал в другую часть диалплана	Полезный
`app_chanspy`	Позволяет каналу прослушивать аудио на другом канале	Полезный
`app_confbridge`	Обеспечивает конференц-связь (новая версия)	Полезный в Астериск 10, Ограниченный в Астериск 1.8
`app_controlplayback`	Воспроизводит приглашение и предлагает быстрые функции перемотки вперед и назад	Полезный
`app_dahdibarge`	Позволяет подключаться к каналу DAHDI	Устаревший, смотри `app_chanspy`

Имя	Назначение	Популярность/Статус
app_dahdiras	Создает RAS (сервер удаленного доступа) по каналу DAHDI (не модемная эмуляция)	Ничтожный
app_db	Используется для добавления/изменения/удаления записей в встроенной базе данных SQLite Asterisk	Устаревший, смотри func_db
app_dial	Используется для соединения каналов (т.е. совершает телефонные звонки)	Существенный
app_dictate	Воспроизведение записи и функции запуска/остановки	Полезный
app_directed_pickup	Отвечает на вызов другого внутреннего номера	Полезный
app_directory	Представляет список имен из *voicemail.conf*	Полезный
app_disa	Обеспечивает тональный сигнал ответа станции и принимает вход DTMF	Полезный[a]
app_dumpchan	Переключает переменные канала в интерфейсе командной строки Asterisk (CLI)	Полезный
app_echo	Возвращает полученное аудио исходному каналу	Полезный
app_exec	Содержит Exec(), TryExec() и ExecIf(); выполняет приложение диалплана на основе условий	Полезный
app_externalivr	Элементы управления Asterisk как с AGI, только асинхронно	Полезный
app_fax	Предоставляет SendFax() и ReceiveFax()	Полезный[b]
app_festival	Позволяет преобразовывать текст в речь с использованием движка Festival TTS	Годный
app_flash	Выполняет флеш-переключение на каналах (в основном аналоговые)	Полезный
app_followme	Выполняет Найди-меня/Следуйте-сюда на основе *followme.conf*	Полезный
app_forkcdr	Запускает новую запись CDR при текущем вызове	Годный
app_getcpeid	Получает идентификатор CPE ADSI	Ничтожный
app_ices	Отправляет аудио на сервер Icecast	Годный
app_image	Передача изображения на поддерживаемые устройства	Ограниченный
app_ivrdemo	Пример приложения для разработчиков	Ничтожный
app_jack	Работает с JACK Audio Connection Kit для обмена звуком между совместимыми приложениями	Полезный
app_macro	Запускает макросы диалплана	Устаревший — смотри GoSub()

Имя	Назначение	Популярность/Статус
app_meetme	Обеспечивает многопользовательскую конференц-связь	Полезный — полнофункциональный, но устаревший в пользу ConfBridge() с Астериск 10, поскольку ConfBridge() не требует DAHDI
app_milliwatt	Генерирует тональный сигнал 1004 Гц для тестирования потерь на аналоговых цепях	Полезный
app_minivm	Предоставляет примитивные функции, позволяющие вам создавать собственное приложение голосовой почты в диалплане	Годный
app_mixmonitor	Записывает обе стороны вызова и сводит их вместе	Полезный
app_morsecode	Генерирует код Морзе	Годный
app_mp3	Использует *mpg123* для воспроизведения MP3	Ничтожный
app_nbscat	Потоки аудио от Network Broadcast Stream (NBS)	Ничтожный
app_originate	Позволяет инициировать вызов	Полезный
app_osplookup	Выполняет поиск Open Settlement Protocol (OSP)	Годный
app_page	Создает несколько аудиосоединений для определенных устройств для общего доступа (пейджинг)	Полезный
app_parkandannounce	Позволяет автоматическое объявление припаркованных вызовов	Годный
app_playback	Воспроизводит файл для канала (не принимает ввод)	Полезный
app_playtones	Воспроизведение пар тонов заданных частот	Полезный
app_privacy	Запрашивает ввод номера телефона вызывающего абонента, если не получен CallerID	Ничтожный
app_queue	Обеспечивает автоматическое распределение вызовов (ACD) - очереди	Полезный
app_read	Запрашивает ввод цифр от абонента и назначает переменную для присвоения ввода	Полезный
app_readexten	Запрашивает ввод цифр от абонентов и передает вызов назначенному внутр.номеру и контексту	Годный
app_readfile	Загружает содержимое текстового файла в переменную канала	Устаревший — смотри функцию FILE() в func_env
app_record	Записывает звук в файл	Полезный
app_sayunixtime	Воспроизведение времени в указанном формате	Полезный

Имя	Назначение	Популярность/Статус
app_senddtmf	Передача DTMF вызывающей стороне	Полезный
app_sendtext	Отправляет текстовую строку на совместимые каналы	Ничтожный
app_setcallerid	Устанавливает CallerID для канала	Устаревший — смотри func_callerid
app_skel	Пример приложения для разработчиков	Полезный[c]
app_sms	Отправляет SMS-сообщение в совместимых странах	Ограниченный
app_softhangup	Запрашивает завершение канала	Полезный
app_speech_utils	Обеспечивает инструменты, связанные с распознаванием речи	Полезный[d]
app_stack	Предоставляет Gosub(), GoSubIf(), Return(), StackPop(), LOCAL() и LOCAL_PEEK()	Существенный
app_system	Выполняет команды в оболочке Linux	Полезный
app_talkdetect	Подобно app_background, но позволяет любому принятому аудио прерывать воспроизведение	Полезный
app_test	Приложение для тестирования клиентов/серверов	Годный
app_transfer	Выполняет трансфер на текущем канале	Полезный
app_url	Передает URI на вызываемый канал	Ограниченный
app_userevent	Создает пользовательское событие в AMI	Полезный
app_verbose	Создает пользовательское событие в CLI Asterisk	Полезный
app_voicemail	Предоставляет голосовую почту	Существенный
app_waitforring	Ожидает событие сигнализации RING (не путать с RINGING); скорее всего, не нужно, поскольку работает только с chan_dahdi с аналоговыми каналами, в которых принимается звонок (например, порт FXO) и генерируется событие сигнализации RING	Ничтожный
app_waitforsilence	Включает функции WaitForSilence() и WaitForNoise(); прослушивает входящий канал для заданного количества миллисекунд шума/тишины	Полезный
app_waituntil	Ожидает, что текущая эпоха Linux соответствует указанной эпохе	Полезный
app_while	Включает While(), EndWhile(), ExitWhile() и ContinueWhile(); обеспечивает функциональность цикла while в диалплане	Полезный
app_zapateller	Генерирует SIT (специальный информационный тон), чтобы препятствовать телемаркетингам	Годный

[a] Использование (DISA) может представлять угрозу безопасности, если вы не будете осторожны с контролем доступа диалплана.

[b] Требуется подходящий движок DSP для обработки кодирования/декодирования факсимильной сигнализации (см. Главу 19).

[c] Если Вы разработчик.

[d] Требуется внешнее приложение распознавания речи.

Модули соединений

Модули соединений выполняют фактическое соединение каналов в новом API. Каждый из них предоставляет различные функции, которые используются в разных ситуациях в зависимости от того, что требуется соединению. Эти модули, перечисленные в Таблице 2-2, в настоящее время используются только (и имеют важное значение) для `app_confbridge`.

Таблица 2-2. Модули соединений

Имя	Назначение	Популярность/Статус
`bridge_builtin_features`	Выполняет соединение при использовании встроенных пользовательских функций (например, найденных в *features.conf*).	Новый
`bridge_multiplexed`	Выполняет сложное мультиплексирование, как это требовалось бы в большом конференц-зале (несколько участников). В настоящее время используется только `app_confbridge`.	Полезный
`bridge_simple`	Выполняет простое межканальное соединение	Полезный
`bridge_softmix`	Выполняет простое мультиплексирование, как это требовалось бы в большом конференц-зале (несколько участников). В настоящее время используется только `app_confbridge`.	Полезный

Модули записи деталей вызовов (CDR)

Модули CDR, перечисленные в Таблице 2-3, предназначены для облегчения как можно большего количества способов записи деталей вызовов. Вы можете хранить CDR в файле (по умолчанию), базе данных, на Удаленной службе аутентификации дозванивающихся пользователей (Remote Authentication Dial In User Service - RADIUS) или *syslog*.



Записи деталей вызовов не предназначены для использования в сложных биллинговых приложениях. Если вам требуется больше контроля над биллингом и отчетах о вызовах, вам нужно будет посмотреть журнал событий канала (CEL), обсужденный ниже. Преимущество CDR заключается в том, что он просто работает.

Таблица 2-3. Модули записи деталей вызовов

Имя	Назначение	Популярность/Статус
`cdr_adaptive_odbc`	Позволяет записывать CDR через структуру ODBC с возможностью добавления настраиваемых полей	Полезный
`cdr_csv`	Записывает CDR на диск как файл значений, разделенных запятыми	Годный
`cdr_custom`	Записывает CDR в CSV-файл, но позволяет добавлять настраиваемые поля	Полезный

Имя	Назначение	Популярность/Статус
cdr_manager	Выводит CDR в AMI	Полезный
cdr_odbc	Записывает CDR через структуру ODBC	Годный
cdr_pgsql	Записывает CDR в PostgreSQL	Полезный
cdr_radius	Записывает CDR в RADIUS	Годный — не поддерживает пользовательские поля
cdr_sqlite	Записывает CDR в базу данных SQLite2	Устаревший — используйте cdr_sqlite3_custom
cdr_sqlite3_custom	Записывает CDR в базу данных SQLite3 с пользовательскими полями	Полезный
cdr_syslog	Записывает CDR в *syslog*	Полезный
cdr_tds	Записывает CDR в Microsoft SQL и Sybase	Годный - требуется старая версия libtds

Мы обсудим некоторые пакеты отчетов, которые вы, возможно, захотите использовать с CDR, в Главе 25.

Модули журналирования событий канала (CEL)

Журнал событий канала (Channel Event Logging - CEL) обеспечивает гораздо более мощный контроль над сообщениями об активности вызовов. Точно так же это требует более тщательного планирования вашего диалплана и отнюдь не будет работать автоматически. Модули CEL Asterisk перечислены в Таблице 2-4.

Таблица 2-4. Модули журналирования событий канала

Имя	Назначение	Популярность/Статус
cel_custom	CEL на диск/файл	Полезный
cel_manager	CEL в AMI	Полезный
cel_odbc	CEL в ODBC	Полезный
cel_pgsql	CEL в PostgreSQL	Полезный
cel_radius	CEL в RADIUS	Годный — не поддерживает пользовательские поля
cel_sqlite3_custom	CEL в SQLite3	Полезный
cel_tds	CEL в Microsoft SQL или Sybase	Годный — требуется старая версия libtds

Драйверы канала

Без драйверов каналов Asterisk не сможет звонить. Каждый драйвер канала специфичен для протокола или типа канала, который он поддерживает (SIP, ISDN и т.д.). Канальный модуль выступает в качестве шлюза к ядру Asterisk. Драйверы канала Asterisk перечислены в Таблице 2-5.

Таблица 2-5. Драйверы канала

Имя	Назначение	Популярность/Статус
`chan_agent`	Предоставляет канал агента для `Queues()`	Популярный
`chan_alsa`	Обеспечивает подключение к Advanced Linux Sound Architecture (ALSA)	Популярный
`chan_bridge`	Используется внутри приложения `ConfBridge()`; не следует использовать напрямую	Существенный[a]
`chan_console`	Использует библиотеку portaudio для предоставления кросс-платформенного драйвера консольного канала, который использует звуковое устройство системы	Популярный
`chan_dahdi`	Обеспечивает подключение к PSTN-картам, использующим драйверы канала DAHDI	Популярный
`chan_gtalk`	Обеспечивает подключение к Google Talk	Устаревший с Asterisk 11 - см. `chan_motif`
`chan_h323`	Обеспечивает подключение к конечным устройствам H.323	Устаревший - см. `chan_ooh323` в Таблице 2-17
`chan_iax2`	Обеспечивает подключение к конечным устройствам IAX2	Полезный
`chan_jingle`	Обеспечивает подключение к конечным устройствам с поддержкой Jingle	Устаревший с Asterisk 11 - см. `chan_motif`
`chan_local`	Предоставляет механизм для обработки части диалплана в качестве канала	Полезный
`chan_mgcp`	Драйвер канала протокола Media Gateway	Годный
`chan_misdn`	Обеспечивает подключение к ISDN-картам, поддерживающим mISDN	Ограниченный
`chan_motif`	Реализует протокол Jingle, включая возможность подключения к Google Talk и Google Voice; введен в Asterisk 11	Полезный
`chan_multicast_rtp`	Обеспечивает соединение с потоками многоадресного трафика в реальном времени (RTP)	Полезный
`chan_nbs`	Драйвер канала сетевого широковещательного звука (NBS)	Ничтожный
`chan_oss`	Драйвер открытой звуковой системы (OSS)	Полезный
`chan_phone`	Драйвер интерфейса телефонии Linux (очень старый)	Ничтожный
`chan_sip`	Драйвер канала протокола установления сеанса (SIP)	Существенный
`chan_skinny`	Драйвер канала Cisco Skinny Client Control (SCCP)	Годный
`chan_unistim`	Драйвер канала протокола Nortel Unistim	Годный

[a] Если вы используете приложение `ConfBridge()`.

Конверторы кодеков

Конверторы кодеков (Таблица 2-6) позволяют Asterisk преобразовывать форматы аудиопотока между вызовами. Поэтому, если вызов поступает через линию PRI (с использованием G.711) и должен быть передан сжатым в SIP-канал (например, с использованием G.729, одного из многих кодеков, который может поддерживать SIP), соответствующий конвертор кодеков будет выполнять преобразование.[1]



Если кодек (например, G.729) использует сложный алгоритм кодирования, интенсивное использование транскодирования может наложить огромную нагрузку на процессор. Специализированное оборудование для декодирования/кодирования G.729 доступно от производителей оборудования, таких как Sangoma и Digium (и, вероятно, других).

Таблица 2-6. Конверторы кодеков

Имя	Назначение	Популярность/Статус
`codec_adpcm`	Кодек для адаптивной дифференциальной импульсно-кодовой модуляции (ADPCM)	Ничтожный
`codec_alaw`	A-law PCM кодек, используемый во всем мире (кроме Канады/США) на ТфОП	Существенный
`codec_g729`	Не поставляется с Asterisk из-за того, что он не является кодеком общего доступа, но можно приобрести у Digium; очень популярный кодек, если требуется хорошее сжатие (и использование ЦП не является проблемой)	Полезный
`codec_a_mu`	Прямой конвертор A-law - mu-law	Полезный
`codec_dahdi`	Использует проприетарную аппаратную карту кодирования Digium	Существенный[a]
`codec_g722`	Широкополосный аудиокодек	Полезный
`codec_g726`	Привкус ADPCM	Ничтожный
`codec_gsm`	Кодек глобальной системы мобильной связи (GSM)	Полезный
`codec_ilbc`	Интернет-кодек с низким битрейтом (ILBC)	Ничтожный
`codec_lpc10`	Вокодер линейного предсказательного кодирования (чрезвычайно низкая полоса пропускания)	Ничтожный
`codec_resample`	Пересемплирование между 8 и 16-битными линейными подписями	Годный
`codec_speex`	Speex-кодек	Годный
`codec_ulaw`	mu-law PCM-кодек, используемый в Канаде/США на ТфОП (PSTN)	Существенный

[a] Если вы используете карту транскодера для кодеков Digium.



Digium распространяет некоторые дополнительные полезные модули кодеков: `codec_g729`, `codec_silk`, `codec_siren7` и `codec_siren14`. Эти модули кодеков не являются открытыми по разным причинам. Вы должны приобрести лицензию на использование `codec_g729`, но остальные бесплатные. Вы можете найти их на сайте Digium.

1 Дополнительная информация о том, какие кодеки и как они работают, доступна в «Кодеки» в Приложении B.

Интерпретаторы формата

Интерпретаторы формата (Таблица 2-7) выполняют функцию кодеков-переводчиков, но выполняют свою работу не по каналам, а по файлам. Если у вас есть запись в меню, которая была сохранена как GSM, для воспроизведения этой записи необходимо использовать интерпретатор формата для любых каналов, не использующих кодек GSM.[2]

Если вы сохраняете запись в нескольких форматах (например, WAV, GSM и т.д.), Asterisk определит наименее затратный формат[3], который будет использоваться, когда канал требует этой записи.

Таблица 2-7. Интерпретаторы формата

Имя	Воспроизведение файлов, хранящихся в	Популярность/Статус
`format_g723`	G.723: *.g723*	Ничтожный
`format_g726`	G.726: *.g726*	Ничтожный
`format_g729`	G.729: *.g729*	Полезный
`format_gsm`	RPE-LTP (оригинальный кодек GSM): *.gsm*	Годный
`format_h263`	H.263—видео: *.h263*	Годный
`format_h264`	H.264—видео: *.h264*	Годный
`format_ilbc`	Интернет-кодек с низким битрейтом: *.ilbc*	Ничтожный
`format_jpeg`	Графические файлы: *.jpeg*, *.jpg*	Ничтожный
`format_ogg_vorbis`	Контейнер ogg: *.ogg*	Годный
`format_pcm`	Различные форматы импульсно-кодовой модуляции: .alaw, .al, .alw, .pcm, .ulaw, .ul, .mu, .ulw, .g722, .au	Полезный
`format_siren14`	G.722.1 Приложение C (14 кГц): .siren14	Новый
`format_siren7`	G.722.1 (7 кГц): .siren7	Новый
`format_sln16`	16-разрядная линейная подпись: .sln16	Новый
`format_sln`	8-разрядная линейная подпись: .sln, .raw	Полезный
`format_vox`	.vox	Ничтожный
`format_wav`	.wav	Полезный
`format_wav_gsm`	Звук GSM в контейнере WAV: .wav, .wav49	Годный

2 Отчасти по этой причине мы не рекомендуем использовать формат GSM по умолчанию для системных записей. Запись WAV будет лучше звучать и использовать меньше циклов процессора.

3 Некоторые кодеки могут накладывать значительную нагрузку на ЦП, так что система, которая может поддерживать несколько сотен каналов без транскодирования, может обрабатывать только несколько десятков, при использовании транскодирования.

Функции диалплана

Функции диалплана, перечисленные в Таблице 2-8, дополняют приложения диалплана (см. «Приложения»). Они предоставляют множество полезных улучшений для таких вещей, как обработка строк, переборка времени и даты и возможность подключения ODBC.

Таблица 2-8. Функции диалплана

Имя	Назначение	Популярность/Статус
`func_aes`	Шифрует/дешифрует строку AES	Полезный
`func_audiohookinherit`	Позволяет записывать вызовы после передачи	Полезный
`func_base64`	Кодирует/декодирует строку base-64	Годный
`func_blacklist`	Записывает/читает черный список в *astdb*	Полезный
`func_callcompletion`	Получает/задает параметры конфигурации завершения вызова для канала	Новый
`func_callerid`	Получает/устанавливает CallerID	Полезный
`func_cdr`	Получает/задает переменную CDR	Полезный
`func_channel`	Получает/задает информацию о канале	Полезный
`func_config`	Включает `AST_CONFIG()`; читает переменные из файла конфигурации	Годный
`func_connectedline`	Изменяет информацию о подключенной линии на поддерживаемых телефонах	Новый
`func_curl`	Использует cURL для получения данных из URI	Полезный
`func_cut`	Нарезает кубиками строки	Полезный
`func_db`	Обеспечивает функции *astdb*	Полезный
`func_devstate`	Получает состояние устройства	Полезный
`func_dialgroup`	Создает группу для одновременного набора номера	Полезный
`func_dialplan`	Подтверждает, что назначенная цель существует в диалплане	Полезный
`func_enum`	Выполняет поиск ENUM	Полезный
`func_env`	Включает `FILE()`, `STAT()` и `ENV()`; выполняет действия операционной системы	Полезный
`func_extstate`	Возвращает статус намеченного внутр.номера	Полезный
`func_global`	Получает/задает глобальные переменные	Полезный
`func_groupcount`	Получает/задает количество каналов для членов группы	Полезный
`func_hangupcause`	Получает/задает информацию о завершении канала	Полезный

Имя	Назначение	Популярность/Статус
func_iconv	Преобразует между наборами символов (кодировками)	Годный
func_jitterbuffer	Включает функцию JITTERBUFFER(), которая позволяет размещать джиттер-буфер на канале.	Полезный
func_lock	Включает функции LOCK(), UNLOCK() и TRYLOCK(); устанавливает блокировку, которая может использоваться для предотвращения условий гонки в диалплане	Полезный
func_logic	Включает ISNULL(), SET(), EXISTS(), IF(), IFTIME() и IMPORT(); выполняет различные логические функции	Полезный
func_math	Включает MATH(), INC() и DEC(); выполняет математические функции	Полезный
func_md5	Преобразует поставляемую строку в MD5-хэш	Полезный
func_module	Проверяет, загружен ли установленный модуль в память	Годный
func_odbc	Позволяет интегрировать в диалплан ресурсы ODBC	Полезный
func_pitchshift	Сдвигает высоту звукового потока	Полезный
func_presencestate	Получает/устанавливает состояние присутствия; используется в основном при интеграции Asterisk с телефонами Digium	Полезный
func_rand	Возвращает случайное число в заданном диапазоне	Полезный
func_realtime	Выполняет поиск в архитектуре Asterisk Realtime (ARA)	Полезный
func_redirecting	Предоставляет доступ к информации о том, откуда был перенаправлен этот вызов	Полезный
func_sha1	Преобразует поставляемую строку в хэш SHA1	Полезный
func_shell	Выполняет операции оболочки Linux и возвращает результаты	Полезный
func_speex	Уменьшает шум и выполняет усиление/потерю dB в аудиопотоке	Полезный
func_sprintf	Выполняет функции строкового формата, подобные функциям C с таким же именем	Полезный
func_srv	Выполняет поиск SRV в диалплане	Полезный
func_strings	Включает более дюжины функций манипуляции строкой	Полезный
func_sysinfo	Получает системную информацию, такую как оперативная память, своп, средняя загрузка и т.д.	Полезный
func_timeout	Получает/устанавливает таймауты на канале	Полезный

Имя	Назначение	Популярность/Статус
func_uri	Преобразует строки в URI-безопасное кодирование	Полезный
func_version	Возвращает информацию о версии Asterisk	Годный
func_vmcount	Возвращает количество сообщений в папке голосовой почты для определенного пользователя	Полезный
func_volume	Устанавливает громкость на канале	Полезный

Модули УАТС

Модули УАТС представляют собой периферийные модули, которые обеспечивают улучшенные механизмы управления и конфигурирования. Например, pbx_config - это модуль, который загружает традиционный диалплан Asterisk. Доступные в настоящее время модули АТС перечислены в Таблице 2-9.

Таблица 2-9. Модули PBX

Имя	Назначение	Популярность/Статус
pbx_ael	Asterisk Extension Logic (AEL) предлагает язык сценариев диалплана, который выглядит как современный язык программирования.	Годный[a]
pbx_config	Это традиционный и самый популярный язык диалплана для Asterisk. Без этого модуля, Asterisk не сможет читать *extensions.conf*.	Полезный
pbx_dundi	Выполняет поиск данных на удаленных системах Asterisk.	Полезный
pbx_loopback	Выполняет что-то подобное include диалплана, но устаревшим способом.	Ничтожный[b]
pbx_lua	Позволяет создавать диалплан с использованием языка сценариев Lua.	Полезный
pbx_realtime	Обеспечивает функциональность, связанную с архитектурой Asterisk Realtime.	Полезный
pbx_spool	Предоставляет поддержку исходящей очереди, относящуюся к файлам звонков Asterisk.	Полезный

[a] Мы не нашли большое количество людей, использующих AEL. Мы подозреваем, что это связано с тем, что большинство разработчиков склонны использовать AGI/AMI если не хотят использовать традиционный диалплан.

[b] Мы никогда не слышали о том, что это используется в продакшене.

Модули ресурсов

Модули ресурсов интегрируют Asterisk с внешними ресурсами. Эта группа модулей эффективно превратилась в универсальную для вещей, которые не подходят для других категорий. Мы разделим их на некоторые подгруппы связанных между собой модулей.

Конфигурирование

По умолчанию Asterisk настроен с использованием текстовых файлов в */etc/asterisk*. Эти модули, перечисленные в Таблице 2-10, предлагают альтернативные методы настройки. Подробную документацию по настройке конфигурации с поддержкой базы данных см. в Главе 16.

Table 2-10. Модули конфигурирования

Имя	Назначение	Популярность/Статус
res_config_curl	Выводит информацию о конфигурации с помощью cURL	Полезный
res_config_ldap	Выводит информацию о конфигурации с помощью LDAP	Годный
res_config_odbc	Выводит информацию о конфигурации с помощью ODBC	Полезный
res_config_pgsql	Выводит информацию о конфигурации с помощью PostgreSQL	Годный
res_config_sqlite	Выводит информацию о конфигурации с помощью SQLite (версия 2)	Годный
res_config_sqlite3	Выводит информацию о конфигурации с помощью SQLite (версия 3)	Годный

Интерфейсы синхронизации

Некоторые операции в Asterisk требуют источника синхронизации. Эти модули обеспечивают синхронизацию с Asterisk из разных источников. Некоторые случаи, когда Asterisk нуждается в источнике синхронизации, включают воспроизведение файлов и конференц-связь с использованием приложения `ConfBridge()`.



Общая точка путаницы предполагает, что конференц-связь с использованием приложения `MeetMe()` также требует использования одного из этих источников синхронизации. Это не относится к делу. `MeetMe()` не использует интерфейс синхронизации Asterisk и вместо этого использует DAHDI напрямую. Хотя DAHDI предоставляет интерфейс синхронизации, `MeetMe()` не использует его. DAHDI представляет собой полный механизм конференц-связи, который является ядром приложения `MeetMe()`.

Если по какой-либо причине вы не можете использовать `res_timing_dahdi`, например, запустить на компьютере, где вы не можете добавить модули ядра DAHDI, вы должны использовать `res_timing_timerfd`. Избегайте использования модуля `res_timing_pthread`, если это вообще возможно, поскольку он намного менее эффективен и повлияет на производительность Asterisk. В Таблице 2-11 перечислены модули интерфейса синхронизации.

Table 2-11. Модули интерфейсов синхронизации

Имя	Назначение	Популярность/Статус
res_timing_dahdi	Обеспечивает синхронизацию с использованием интерфейса ядра DAHDI	Полезный
res_timing_kqueue	Обеспечивает синхронизацию с использованием функции ядра в некоторых BSD, включая Mac OS X	Полезный
res_timing_pthread	Обеспечивает синхронизацию с использованием только частей стандартного API `pthread`; менее эффективен, но более переносим, чем другие модули синхронизации	Годный
res_timing_timerfd	Обеспечивает синхронизацию с использованием API-интерфейса `timerfd`, предоставляемого более новыми версиями ядра Linux	Полезный

Интеграция календаря

Asterisk включает некоторую интеграцию с системами календаря. Вы можете читать и записывать календарную информацию из диалплана. Вы также можете получать вызовы, созданные на основе записей календаря. Интеграция основного календаря обеспечивается модулем `res_calendar`.

Остальные модули обеспечивают возможность подключения к определенным типам серверов календаря. В Таблице 2-12 перечислены модули интеграции календаря.

Таблица 2-12. Модули интеграции календаря

Имя	Назначение	Популярность/Статус
`res_calendar`	Обеспечивает интеграцию баз данных с системами календаря	Полезный
`res_calendar_caldav`	Позволяет функциям, предоставляемым `res_calendar` подключаться к календарям через CalDAV	Полезный
`res_calendar_exchange`	Позволяет функциям, предоставляемым `res_calendar` подключаться к MS Exchange	Полезный
`res_calendar_icalendar`	Позволяет функциям, предоставляемым `res_calendar` подключаться к Apple/Google iCalendar	Полезный

Реализация RTP

Ядро Asterisk не включает реализацию RTP. Если вы используете один из драйверов VoIP-канала, который использует RTP, вы также должны загрузить модуль `res_rtp_asterisk`. Эта реализация RTP может быть заменена пользовательской, если Asterisk использовался в системе, которая включала настраиваемое оборудование, такое как DSP с более эффективной обработкой RTP.

Реализация многоадресного RTP используется только канальным драйвером `chan_multicast_rtp`, который является полезным для поддержки большого количества телефонов. Для получения дополнительной информации о многоадресном RTP см. Главу 11. Модули реализации RTP перечислены в Таблице 2-13.

Таблица 2-13. Модули реализации RTP

Имя	Назначение	Популярность/Статус
`res_rtp_asterisk`	Предоставляет RTP	Существенный
`res_rtp_multicast`	Предоставляет многоадресный RTP	Новый

Обработчики атрибутов формата

Одним из интерфейсов, которые может реализовать модуль Asterisk, является обработка атрибутов, связанных с определенными аудио- и видеоформатами. Группа модулей, перечисленных в Таблице 2-14, реализует этот интерфейс.

Таблица 2-14. Модули обработчиков атрибутов формата

Имя	Назначение	Популярность/Статус
`res_format_attr_celt`	Обрабатывает детали атрибутов формата для аудиоформата CELT	Полезный
`res_format_attr_h263`	Обрабатывает детали атрибутов формата для видеоформата H.263	Полезный
`res_format_attr_h264`	Обрабатывает детали атрибутов формата для видеоформата H.264	Полезный

Имя	Назначение	Популярность/Статус
res_format_attr_silk	Обрабатывает детали атрибутов формата для аудиоформата SILK	Полезный

Расширения CLI

Эта группа модулей включает в себя дополнительные функции для интерфейса командной строки Asterisk (CLI).

Таблица 2-15. Модули расширения CLI

Имя	Назначение	Популярность/Статус
res_clialiases	Создает альясы CLI	Полезный
res_clioriginate	Предоставляет команду CLI Asterisk для инициации вызовов	Годный
res_convert	Предоставляет команду CLI Asterisk для преобразования файлов/форматов	Годный
res_limit	Позволяет регулировать системные ограничения для процесса Asterisk	Годный
res_realtime	Предоставляет команды консоли для архитектуры Asterisk Realtime (ARA)	Полезный

Прочие модули ресурсов

Таблица 2-16 включает остальные модули ресурсов, которые не вписывались ни в одну из подгрупп, которые мы определили ранее в этом разделе.

Таблица 2-16. Модули ресурсов

Имя	Назначение	Популярность/Статус
res_adsi	Предоставляет ADSI	Существенный[a]
res_ael_share	Предоставляет общие процедуры для использования с pbx_ael	Существенный, если вы используете AEL
res_agi	Предоставляет интерфейс шлюза Asterisk (AGI) (см. Главу 21)	Полезный
res_corosync	Предоставляет индикацию ожидающего сообщения (MWI) и уведомления о состоянии устройства через Corosync Cluster Engine	Полезный
res_crypto	Предоставляет криптографические возможности	Полезный
res_curl	Предоставляет общие подпрограммы для других модулей cURL	Полезный
res_fax	Предоставляет общие подпрограммы для других модулей факса	Полезный
res_fax_spandsp	Плагин для факса с использованием пакета spandsp	Полезный
res_http_post	Обеспечивает поддержку загрузки POST для HTTP-сервера Asterisk	Годный
res_http_websocke	Обеспечивает поддержку WebSocket для внутреннего HTTP-	Годный

Имя	Назначение	Популярность/Статус
t	сервера Asterisk; также используется chan_sip для обеспечения SIP через соединение WebSocket, которое полезно для rtcweb	
res_jabber	Предоставляет ресурсы Jabber/XMPP	Устаревший (с Asterisk 11) - смотри res_xmpp
res_monitor	Предоставляет ресурсы для записи вызовов	Полезный
res_musiconhold	Предоставляет ресурсы музыки в режиме ожидания (MOH)	Существенный
res_mutestream	Позволяет заглушать/восстанавливать аудиопотоки	Новый
res_odbc	Предоставляет общие подпрограммы для других модулей ODBC	Полезный
res_phoneprov	Провижинг телефонов с сервера Asterisk HTTP	Новый
res_pktccops	Предоставляет ресурсы COPS PacketCable	Новый
res_security_log	Позволяет регистрировать события безопасности, созданные другими частями Астериск	Новый
res_smdi	Предоставляет уведомление о голосовой почте с использованием протокола SMDI	Ограниченный
res_snmp	Предоставляет информацию о состоянии системы в сети, управляемой SNMP	Годный
res_speech	Общий интерфейс распознавания речи	Ограниченный[b]
res_xmpp	Предоставляет ресурсы Jabber/XMPP	Полезный

[a] Хотя большинство функций ADSI в Asterisk никогда не используется, приложение голосовой почты использует этот ресурс.

[b] Для использования требуется отдельный лицензионный продукт.

Дополнительные модули

Дополнительные модули представляют собой модули, разработанные сообществом, с различными правами использования или распространения, отличными от основного кода. Они хранятся в отдельном каталоге и не компилируются и не устанавливаются по умолчанию. Чтобы включить эти модули, используйте утилиту конфигурации *menuselect*. Доступные дополнительные модули перечислены в Таблице 2-17.

Таблица 2-17. Дополнительные модули

Имя	Назначение	Популярность/Статус
app_mysql	Выполняет MySQL-запросы с помощью приложения диалплана	Устаревший - см. func_odbc
app_saycountpl	Говорит польские слова	Устаревший - теперь интегрирован в *say.conf*
cdr_mysql	Записывает CDR в базу данных MySQL	Годный - мы рекомендуем cdr_adaptive_odbc вместо этого

Имя	Назначение	Популярность/Статус
`chan_mobile`	Позволяет создавать и принимать телефонные звонки с помощью сотовых телефонов по Bluetooth	Ограниченный[a]
`chan_ooh323`	Позволяет совершать и принимать VoIP-звонки с использованием протокола H.323	Годный
`format_mp3`	Позволяет Asterisk воспроизводить файлы MP3	Годный
`res_config_mysql`	Использует базу данных MySQL как базовую конфигурацию в реальном времени	Полезный

[a] Хотя `chan_mobile` отлично работает со многими телефонами, с некоторыми моделями сообщалось о проблемах. Когда проблема возникает, разработчикам очень сложно решить проблему, если у них нет телефона той же модели для тестирования.

Тестовые модули

Тестовые модули используются командой разработчиков Asterisk для проверки нового кода. Они постоянно меняются и добавляются, и не будут полезны, если вы не разрабатываете программное обеспечение Asterisk.

Однако, если вы разработчик Asterisk, вам может быть интересен Asterisk Test Suite, так как вы можете создавать автоматические тесты для Asterisk и отправлять их обратно в проект, который работает на нескольких разных операционных системах и типах машин. Постоянно расширяя количество тестов, проект Asterisk избегает создания регрессий в коде. Предоставляя собственные тесты для проекта, вы можете чувствовать себя более уверенно в будущих обновлениях.

Более подробную информацию об установке Asterisk Test Suite можно найти в этом блоге. Более подробная информация о строительных тестах содержится в этом документе или вы можете присоединиться к каналу *#asterisk-testing* в сети IRC Freenode.

Файловая структура

Asterisk - сложная система, состоящая из множества ресурсов. Эти ресурсы используют файловую систему несколькими способами. Поскольку Linux настолько гибкий в этом отношении, полезно понять, какие данные хранятся, чтобы вы могли понять, где вы, вероятно, найдете конкретный бит хранимых данных (например, сообщения голосовой почты или файлы логов).

Файлы конфигурации

Файлы конфигурации Asterisk включают *extensions.conf*, *sip.conf*, *modules.conf* и десятки других файлов, которые определяют параметры для различных каналов, ресурсов, модулей и функций, которые могут быть использованы.

Эти файлы будут найдены в */etc/asterisk*. Вы будете много работать в этой папке при настройке и администрировании своей системы Asterisk.

Модули

Модули Asterisk обычно устанавливаются в папку */usr/lib/asterisk/modules*. Обычно вам не нужно взаимодействовать с этой папкой; однако, будет время от времени полезно знать, где находятся модули. Например, если вы обновляете Asterisk и выбираете разные модули на этапе установки *menuselect*, старые (несовместимые) модули из предыдущей версии Asterisk не будут удалены, и вы получите предупреждение из сценария установки. Эти старые файлы необходимо будет удалить из папки модулей. Это можно сделать либо вручную, либо командой make «uninstall» (*make uninstall*).

Библиотека ресурсов

Существует несколько ресурсов, которые требуют внешних источников данных. Например, музыка в режиме ожидания (MOH) не может сработать, если у вас нет музыки для воспроизведения. Системные подсказки также должны храниться где-то на жестком диске. В папке */var/lib/asterisk* хранятся системные подсказки, скрипты AGI, музыка в режиме ожидания и другие файлы ресурсов.

Spool

Spool - это то, где приложения хранят файлы в системе Linux, которые будут часто меняться или будут обрабатываться другими процессами позже. Например, задания печати Linux и ожидающие сообщения электронной почты обычно записываются в шпулю до тех пор, пока они не будут обработаны.

В Asterisk *spool* используется для хранения переходных элементов, таких как голосовые сообщения, записи вызовов,[4] файлы вызова и т.д.

Spool Asterisk будет найдена в каталоге */var/spool/asterisk*.

Журналирование (логирование)

Asterisk способен генерировать несколько разных файлов журналов. В папке */var/log/asterisk* записываются такие вещи, как записи подробной информации о вызовах (CDR), события канала из CEL, журналы отладки, журналы очереди, сообщения, ошибки и другой вывод.

Эта папка будет чрезвычайно важна для любых усилий по устранению неполадок, которые вы предпринимаете.

Мы поговорим больше о том, как использовать журналы Asterisk в Главе 24.

Диалплан

Диалплан является сердцем Asterisk. Все каналы, которые поступают в систему, будут проходить через диалплан, который содержит сценарий потока вызовов, который определяет, как обрабатываются входящие вызовы.

Диалплан можно написать одним из трех способов:

- Использование традиционного синтаксиса диалплана Asterisk в файле */etc/asterisk/extensions.conf*
- Использование логики расширения Asterisk (AEL) в */etc/asterisk/extensions.ael*
- Использование Lua в файле */etc/asterisk/extensions.lua*

Позже в этой книге мы посвящаем несколько разделов тем тому, как написать диалплан, используя традиционный синтаксис диалплана (безусловно самый популярный выбор). Как только вы изучите этот язык, довольно легко перейти на AEL или Lua, если пожелаете.

Аппаратные средства

Asterisk способен общаться с огромным количеством различных технологий. В общем, эти соединения выполняются через сетевое соединение; однако для подключения к более традиционным телекоммуникационным технологиям, таким как PSTN, требуется специальное оборудование.

Многие компании производят это оборудование, например Digium (спонсор, владелец и главный разработчик Asterisk), Sangoma, Rhino, OpenVox, Pika, Voicetronix, Junghanns, Dialogic, Xorcom, beroNet и многие другие. Авторы предпочитают карты от Digium и Sangoma, однако продукты,

4 Не записи деталей вызовов (CDR), а скорее аудиозаписи вызовов, созданных `MixMonitor()` и связанных с ними приложений.

предлагаемые другими производителями оборудования Asterisk, могут быть более подходящими для ваших требований.

Самое популярное оборудование для Asterisk, как правило, предназначено для работы через интерфейс аппаратного устройства Digium Asterisk (известный как DAHDI). Все эти карты будут иметь разные требования к установке и разные расположения файлов.

В Главе 7 мы обсудим DAHDI более подробно; однако мы ограничим наше обсуждение только DAHDI. Вам нужно будет обратиться к конкретной документации, предоставленной производителями любых карт, которые вы устанавливаете, для получения подробной информации об этих картах.

Управление версиями Asterisk

В течение последних нескольких лет методология выпуска Asterisk прошла пару итераций, и этот раздел призван помочь вам понять, что означают номера версий. Особое значение имеет изменение в версии, которое произошло с версиями выпусков версии 1.6.x, которые следуют другой логике нумерации, чем все другие выпуски Asterisk (от 1.0 до 1.8), и где мы сейчас находимся.

Предыдущие методологии выпуска



Этот раздел имеет историческую ценность, но его можно пропустить, если вы не будете работать со старыми версиями Asterisk.

Когда у нас были только Asterisk 1.2 и 1.4 вся новая разработка была выполнена в стволе (это все еще есть) и только исправления ошибок попали в ветви 1.2 и 1.4. (Все ветви до 1.8 были отмечены как EOL [End of Life] и больше не получают исправлений ошибок или обновлений безопасности.)

Поскольку вся новая разработка была выполнена в стволе, пока не была создана ветвь 1.6, люди не смогли получить доступ к новым функциям и функциональности. Это не означает, что новые функции недоступны, но со всеми изменениями, которые могут произойти в стволе, для запуска на нем производственного сервера требуется очень знакомый Asterisk (и C-код).

Чтобы попытаться облегчить давление на администраторов и обеспечить более быстрый доступ к новым функциям (в течение нескольких месяцев, а не лет), была создана новая методология. Ветви в 1.6 действительно будут отмечены как 1.6.0, 1.6.1, 1.6.2 и т. д., Причем третье число увеличивается на единицу каждый раз, когда создается новый функциональный релиз. Цель заключалась в том, чтобы предоставлять новые версии функций каждые три-четыре месяца (которые были бы разветвлены из ствола), обеспечивая более короткий и понятный путь обновления для администраторов. Если вам нужна новая функция, вам нужно будет подождать несколько месяцев и затем перейти к следующей ветви.

Теги из этих ветвей выглядят следующим образом:

- 1.6.0.1 -- 1.6.0.2 -- 1.6.0.3 -- 1.6.0.4 -- etc.
- 1.6.1.1 -- 1.6.1.2 -- 1.6.1.3 -- 1.6.1.4 -- etc.
- 1.6.2.1 -- 1.6.2.2 -- 1.6.2.3 -- 1.6.2.4 -- etc.

На рисунке 2-2 представлено визуальное представление процесса ветвления и тегирования по отношению к стволу Asterisk.

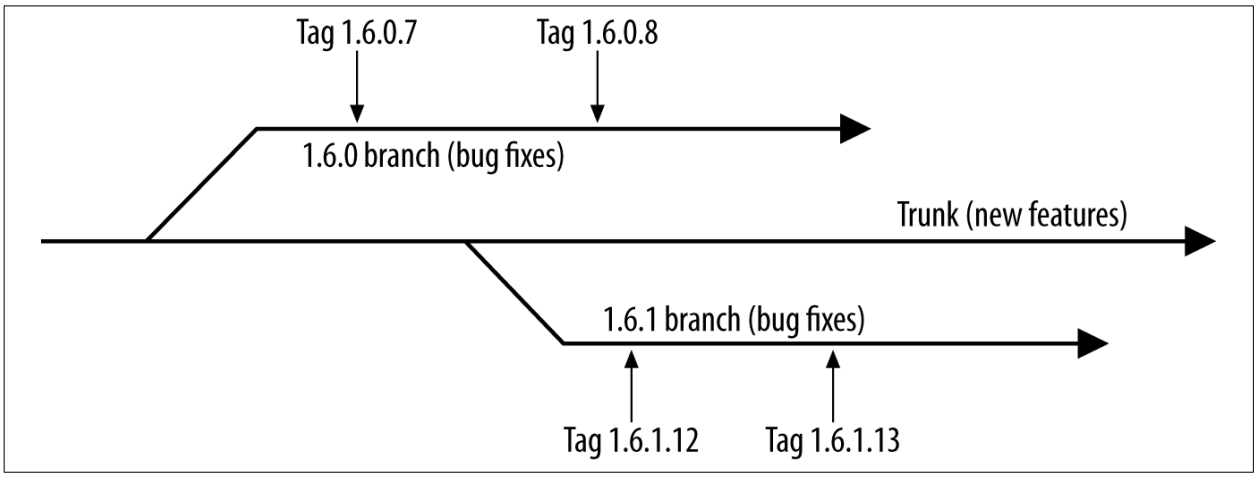


Рисунок 2-2. Процесс выпуска Asterisk 1.6.x

До сих пор у нас есть ветви: 1.2, 1.4, 1.6.0, 1.6.1 и 1.6.2 (нет ветви 1.6). В каждой из этих ветвей мы создаем *теги* (релизы), которые выглядят как 1.2.14, 1.4.30, 1.6.0.12, 1.6.1.12 и 1.6.2.15.

К сожалению, это не сработало.ветви x создавались из ствола каждые три-четыре месяца: процесс разработки привел к минимальному сроку выпуска от шести до восьми месяцев. Не только это, но 1.6.x методологию нумерации добавляет свои собственные проблемы. Люди путались относительно того, какую версию запускать, а ветви 1.6.0, 1.6.1 и 1.6.2 - это все отдельные модификации основной версии. Когда вы увеличиваете число от 1.2 до 1.4, а затем до 1.8, очевидно, что это разные ветви и основные изменения версии. С 1.6.0, 1.6.1 и 1.6.2 это менее очевидно.

К счастью, все это позади, и, как правило, для новых развертываний вы будете использовать последние LTS (долгосрочная поддержка) или последние ветви регулярного выпуска, которые имеют более разумную структуру, как мы рассмотрим в следующем разделе.

Текущая методология выпуска

Команда разработчиков многому научилась во время релизов 1.6.x. Идея, окружающая выпуски, была благородной, но реализация оказалась ошибочной, когда ее ввели в реальное использование. Итак, с Asterisk 1.8, методология вернулась, чтобы выглядеть так же, как и в версиях 1.2 и 1.4.

В то время, как команда разработчиков по-прежнему хочет обеспечить доступ к новым функциям и основным изменениям на более регулярной основе (цель - каждые 12 месяцев), есть признание, что также хорошо обеспечить долгосрочную поддержку стабильной, популярной версии Астериска. Вы можете думать о ветве Asterisk 1.4 как о версии LTS. Разделы 1.6.0, 1.6.1 и 1.6.2 можно рассматривать как текущие релизы, которые продолжают получать исправления ошибок после выпуска, но поддерживаются в течение более короткого периода времени (около года). Первой официальной версией LTS был Asterisk 1.8, получивший исправления ошибок в течение четырех лет с дополнительным годом релизов безопасности после этого. В целом команда разработчиков Digium предпочла предоставить пять лет поддержки от первоначального выпуска для Asterisk 1.8.

Asterisk 1.8 был выпущен в 21 ноября 2010 года и перейдет в режим безопасности только после четырех лет (21 ноября 2014 г.), после чего он получит исправления для проблем безопасности, но не общие исправления ошибок ещё один дополнительный год. А 21 ноября 2015 г. Asterisk 1.8 получит статус EOL, после чего в ветку 1.8 не будут вноситься изменения (Исходный код, как всегда, будет по-прежнему широко распространяться для тех, кто продолжает поддерживать Asterisk 1.8 в своих частных средах).

Эта книга основана на Asterisk 11, которая была выпущена в 25 ноября 2012 году и получит поддержку исправлений ошибок до 25 ноября 2016 года, а также поддержку безопасности в течение года.

Текущее состояние всех ветвей Asterisk - даты их выпуска, когда они войдут в режим только исправления безопасности, и когда они достигнут статуса EOL - все это задокументировано в вики Asterisk.

Упрощение номеров версий

Раньше Марк Спенсер говорил, что число 1 перед номерами версий 1.0, 1.2 и т.д. никогда не изменится, если в базе кода Asterisk не произошел крупный фундаментальный сдвиг в разработке. Теперь, когда Asterisk настолько популярен и широко используется, любой фундаментальный сдвиг в разработке будет невероятно разрушительным как для разработчиков, так и для тех, кто развертывает Asterisk на регулярной основе. В превосходной статье в блоге Кевина Флеминга объясняется, почему вместо перехода от Asterisk 1.8 до 1.10 схема нумерации Asterisk была нарушена, чтобы отбросить ведущую 1 и перейти только к Asterisk 10. По существу, это сводится к следующему: вместо версий, таких как 1.10, 1.12, 1.14 (где нечетные числа пропускаются по историческим причинам), что приводит к потенциальным номерам версий, которые выглядят как 1.10.3.1, а так как основная база Asterisk не изменится так принципиально за релиз, чтобы он оправдывал название 2.0, было бы лучше всего отказаться от 1 с самого начала версии.

Взгляд на текущую структуру ветвей показан на Рисунке 2-3.

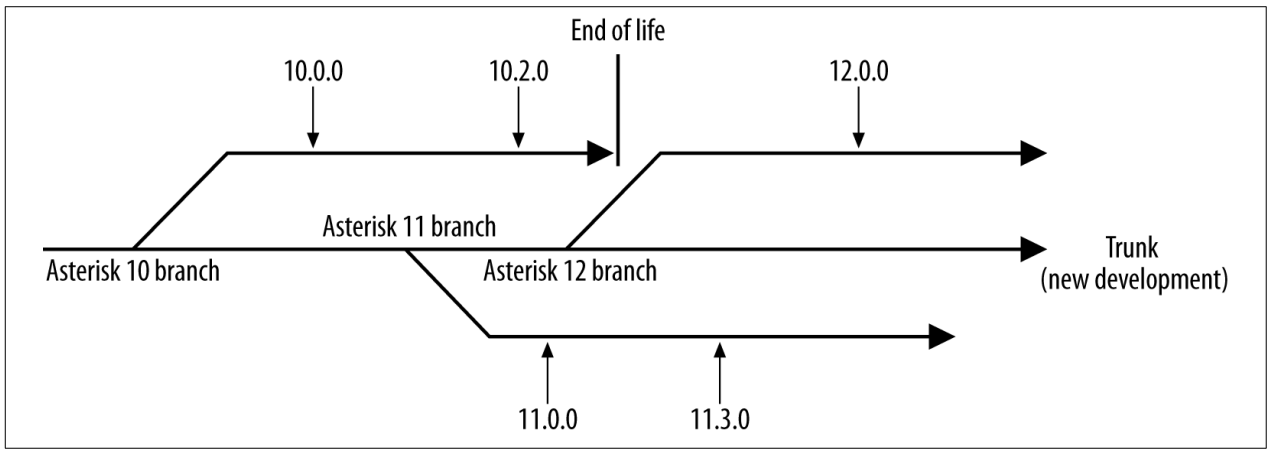


Рисунок 2-3. Текущий процесс ветвления Астериск

Версии, идущие вперёд, будут следовать той же методологии выпуска (что касается чередующихся LTS и регулярных выпусков по хорошо известному графику с предопределенным уменьшением поддержки), кроме того, теперь будет упрощена нумерация версий. Asterisk 10 была первой версией с отказом от 1 в названии, и он был регулярным релизом поддержки. Первым выпуском LTS был Asterisk 11. Таким образом, релизы будут чередоваться между регулярными и LTS.

Вывод

Asterisk состоит из множества различных технологий, большинство из которых сложны сами по себе. Таким образом, понимание архитектуры Asterisk может быть подавляющим. Тем не менее, реальность заключается в том, что Asterisk хорошо разработан для того, что он делает, и, по нашему мнению, достиг замечательного баланса между гибкостью и сложностью.

Установка Asterisk

Я долго шла к великим и благородным целям,
но мой главный долг выполнять простые задачи так,
как будто они великие и благородные. Мир движется
не только мощными импульсами, которые дают ему
герои, но также и крошечными толчками
каждого трудящегося человека.

– Хелен Келлер

В этой главе мы собираемся пройтись по установке Asterisk из исходного кода. Многие люди сторонятся от этого метода, утверждая, что это слишком сложно и отнимает много времени. Наша цель, продемонстрировать, что установка Asterisk из исходников, на самом деле, это просто. Что еще более важно, мы хотим, предоставить вам лучшую платформу Asterisk для обучения.

В этой книге мы поможем вам построить функционирующую систему Asterisk с нуля. В этой главе вы будете строить базовую платформу для системы Asterisk. Учитывая то, что мы производим установку из исходников, есть много нюансов в том, как вы можете это сделать. Процесс, который обсуждается здесь, мы использовали в течение многих лет, и готовы предоставить вам подходящую основу для Asterisk.

В рамках этого процесса мы будем объяснять различные зависимости от других программ на Linux платформе, которые будут описаны в последующих главах этой книги (например, интеграция с базами данных). Мы покажем инструкции по установке Asterisk на RHEL (Red Hat Enterprise Linux) и Ubuntu (на базе Debian), которые как мы считаем устанавливаются в большинстве случаях из Linux дистрибутивов на сегодня. Мы постараемся сделать эту инструкцию достаточно общей, что бы она была полезна на любом другом дистрибутиве предпочитаемом вами.[1]

Мы выбрали установку на RHEL и Ubuntu, потому что они являются самыми популярными, но Asterisk, как правило, не зависит от дистрибутива. Asterisk можно установить на Solaris, BSD, или OS X[2], если захотите. Мы не будем рассматривать их в этой книге, поскольку Asterisk «заточен» под Linux.

1 Если вы используете другой дистрибутив, мы готовы поспорить, что вы дружите с Linux и не должно быть никаких проблем с установкой Asterisk.

Пакеты Asterisk

Существуют также пакеты, с собранным Asterisk, которые могут быть установлены с использованием популярных программ управления пакетами, таких как *yum* или *apt-get*. Мы предлагаем вам поэкспериментировать с ними. Собранные пакеты не будут свежими, поэтому мы всегда рекомендуем установку из исходных кодов.

Если вы используете RHEL, Asterisk доступен в репозитории EPEL из проекта Fedora.

Asterisk пакеты доступны в универсальном репозитории для Ubuntu.

Некоторые команды, которые вы увидите в этой главе будут разделены на отдельные строки, для каждого дистрибутива на котором команда должна быть выполнена. Команды, для которых дистрибутив не указан общие для обоих дистрибутивов.

Проекты на основе Asterisk

Многие проекты были созданы, с использованием Asterisk в качестве базовой платформы. Некоторые из них, такие как Trixbox, стали настолько популярны, что многие люди ошибочно принимают их за отдельный продукт. Эти проекты, как правило, имеют в основе Asterisk и добавленный веб-интерфейс для администрирования, комплекс баз данных, и жесткий набор ограничений, для изменения конфигурации.

Мы решили не рассматривать эти проекты в данной книге, по нескольким причинам:

1. Эта книга - попытка, сосредоточиться на Asterisk и только Asterisk.
2. Есть книги с описанием некоторых из этих Asterisk проектов.
3. Мы считаем, что если вы узнаете Asterisk так, как мы научим вас, то эти знания пригодятся вам, независимо от того, какое решение из этих пересобранных версий Asterisk вы выберите в конце концов.
4. Для нас, сила Asterisk заключается в том, что он не пытается решить ваши проблемы за вас. Эти проекты являются прекрасным примером того, что можно создать на базе Asterisk. Они действительно удивительны. Однако, если вы хотите построить свой собственный комплекс Asterisk (это действительно то, о чем Asterisk), эти проекты будут вас ограничивать, потому что они направлены на упрощение процесса создания бизнес-PBX, а не облегчают доступ к полному потенциалу системы Asterisk.

Вот список наиболее популярных Asterisk проектов:

AsteriskNOW	Управляется Digium. Использует графический интерфейс FreePBX
Elastix	Использует графический интерфейс FreePBX
FreePBX Distro	Использует графический интерфейс FreePBX
PBX in a Flash	Использует графический интерфейс FreePBX

Мы рекомендуем вам потестировать их.[3]

2 Лейф называет это "Oh-Eh-Sex,", но Джим считает, что следует произносить "OS Тен". Мы потратили впустую несколько драгоценных минут споря по этому поводу.

3 После прочтения нашей книги, конечно.

Шпаргалка по инсталляции

Если вы хотите досконально узнать о том, как установить Asterisk и быстро запустить, выполняйте следующее в командной строке. Мы рекомендуем вам прочитать всю главу целиком, по крайней мере, один раз, хотя бы для того, чтобы лучше понять весь процесс.[4]

Данные инструкции предполагают, что вы уже установили или RHEL или Ubuntu используя шаги, описанные в разделе «Установка дистрибутива».

1. Выполните обновление системы и перезагрузку:

RHEL
```
# yum update -y && reboot
```

Ubuntu
```
sudo apt-get update && sudo apt-get upgrade && sudo reboot
```

2. Синхронзируйте время и установите NTP (Network Time Protocol) демон:

RHEL
```
# yum install -y ntp && ntpdate pool.ntp.org && \
chkconfig ntpd on && service ntpd start
```

Ubuntu
```
sudo apt-get install ntp
```



В Ubuntu потребуется дополнительные настройки в текстовом файле. Смотри главу “Включить NTP для точного времени в системе” на стр 48.

3. На RHEL, добавить в систему нового пользователя:

RHEL
```
# adduser asteriskpbx && passwd asteriskpbx && yum install sudo &&
visudo
```



Для установки на Ubuntu мы предполагаем, что созданный пользователь будет *asteriskpbx*.

4. Устанавливаем необходимое ПО:

RHEL
```
sudo yum install gcc gcc-c++ make wget subversion \
libxml2-devel ncurses-devel openssl-devel \
sqlite-devel libuuid-devel vim-enhanced
```

Ubuntu
```
sudo apt-get install build-essential subversion \
libncurses5-dev libssl-dev libxml2-dev libsqlite-dev \
uuid-dev vim-nox
```

5. Создаем структуру ваших каталогов:

```
$ mkdir -p ~/src/asterisk-complete/asterisk
$ cd ~/src/asterisk-complete/asterisk
```

6. Забираем последний релиз в серии Asterisk 11 с использованием wget:

```
$ wget \
http://downloads.asterisk.org/pub/telephony/asterisk/ \
```

4 Если у вас есть опыт работы с несколькими инсталляциями Asterisk, вы согласитесь, что это быстрый и безболезненный процесс. Тем не менее, эта глава может выглядит сложной. Это просто потому, что у нас есть обязательства — обеспечить вас сведениями, необходимыми для выполнения успешной установки.

```
asterisk-11-current.tar.gz
```

Или вы можете получить последние изменения в ветке Asterisk 11 используя Subversion:

```
$ svn co http://svn.asterisk.org/svn/asterisk/branches/11
```

Или же, вы можете проверить определенный тег (версию) используя Subversion:

```
$ svn co http://svn.asterisk.org/svn/asterisk/tags/11.3.0
```

7. Собираем и устанавливаем программу:

```
$ cd ~/src/asterisk-complete/asterisk/11
```

Если вы работаете на 64-bit RHEL, запустите скрипт *configure* с опцией `libdir`:

```
$ ./configure –libdir=/usr/lib64
```

На любой другой платформе, запустите скрипт configure без всяких аргументов.

```
$ ./configure
```

В завершение, компилируем и устанавливаем Asterisk.

```
$ make
$ sudo make install
$ sudo make config
```

8 Устанавливаем дополнительные звуковые файлы (Extras Sound Packages) из *menuselect*:

```
$ cd ~/src/asterisk-complete/asterisk/11
$ make menuselect
$ sudo make install
```

9 Изменяем владельца установленных каталогов Asterisk:

```
$ sudo chown -R asteriskpbx:asteriskpbx /var/lib/asterisk/
$ sudo chown -R asteriskpbx:asteriskpbx /var/spool/asterisk/
$ sudo chown -R asteriskpbx:asteriskpbx /var/log/asterisk/
$ sudo chown -R asteriskpbx:asteriskpbx /var/run/asterisk/
```



Примечание: Вы можете обновить все права доступа в одной строке: *sudo chown -R asteriskpbx.asteriskpbx {/usr/lib,/var/lib,/var/spool,/var/log,/var/run}/asterisk*

10 Создаем каталог */etc/asterisk/* и копируем в него файл *indications.conf* из шаблона:

```
$ sudo mkdir -p /etc/asterisk
$ sudo chown asteriskpbx:asteriskpbx /etc/asterisk
$ cd /etc/asterisk/
$ cp ~/src/asterisk-complete/asterisk/11/configs/indications.conf.sample \
./indications.conf
```

11 Копируем шаблон файла *asterisk.conf* в */etc/asterisk* и изменяем в нем *runuser* и *rungroup* на значение *asteriskpbx*:

```
$ cp ~/src/asterisk-complete/asterisk/11/configs/asterisk.conf.sample \
```

/etc/asterisk/asterisk.conf

```
$ vim /etc/asterisk/asterisk.conf
```

Смотри раздел “indications.conf и asterisk.conf” для подробной информации.

12 Создаем файл *modules.conf*. Включаем автоматически загружаемые модули, и отключаем дополнительные модули:

```
$ cat >> /etc/asterisk/modules.conf

; Файл modules.conf используется для определения модулей, которые должен
; загружать (или не загружать).
;
[modules]
autoload=yes

; Модули ресурсов в настоящее время не требуются
noload => res_speech.so
noload => res_phoneprov.so
noload => res_ael_share.so
noload => res_clialiases.so
noload => res_adsi.so

; Модули АТС в настоящее время не требуются
noload => pbx_ael.so
noload => pbx_dundi.so

; Модули канала в настоящее время не нужны
noload => chan_oss.so
noload => chan_mgcp.so
noload => chan_skinny.so
noload => chan_phone.so
noload => chan_agent.so
noload => chan_unistim.so
noload => chan_alsa.so

; Модули приложений в настоящее время не требуются
noload => app_nbscat.so
noload => app_amd.so
noload => app_minivm.so
noload => app_zapateller.so
noload => app_ices.so
noload => app_sendtext.so
noload => app_speech_utils.so
noload => app_mp3.so
noload => app_flash.so
noload => app_getcpeid.so
noload => app_setcallerid.so
noload => app_adsiprog.so
noload => app_forkcdr.so
noload => app_sms.so
noload => app_morsecode.so
noload => app_followme.so
noload => app_url.so
noload => app_alarmreceicer.so
noload => app_disa.so
noload => app_dahdiras.so
noload => app_senddtmf.so
noload => app_sayunixtime.so
```

```
noload => app_test.so
noload => app_externalivr.so
noload => app_image.so
noload => app_dictate.so
noload => app_festival.so
```

`Ctrl`+`D`

13 Настраиваем *musiconhold.conf*:

```
$ cat > musiconhold.conf
; musiconhold.conf
[default]
mode=files
directory=moh
```

`Ctrl`+`D`

14 Сохраните ваши изменения и конфигурация модуля будет выполнена. Ваша система теперь готова к настройке диалплана и каналов.

Установка дистрибутива

Поскольку Asterisk сильно нагружает процессор и имеет приоритетный доступ к нему, необходимо устанавливать Asterisk на сервер без графического интерфейса, такого как система X Windowing (Gnome, KDE и т.д.). Оба дистрибутива и RHEL и Ubuntu поставляются с конфигурацией, предназначенной для использования под сервер. Мы рассмотрим инструкции для обоих дистрибутивов.

RHEL сервер

Вы должны скачать образ RHEL Server ISO с сайта Red Hat. Инструкции по скачиванию вы найдете на том же сайте.

Запишите скачанный вами ISO файл на CD или DVD и запустите процесс инсталляции. Если вы устанавливаете в виртуальной машине (что мы не рекомендуем для промышленного использования, но это хороший путь протестировать Asterisk), вы можете смонтировать ISO файл и установить его.

Установка базовой системы

Загрузитесь с CD, выберите Установка и обновление существующей системы (Install or upgrade an existing system). Вам будет задан вопрос, вы хотите проверить носитель. Если вы запускаете установку с этого CD в первый раз, вы должны выбрать OK. После завершения проверки носителя, запуститься интерфейс установки. Выберите Next для перехода к первому шагу установки.

Выберите язык и задайте раскладку клавиатуры. Если вы находитесь в северной Америке, можете оставить установки по умолчанию. Затем вас спросят, какой тип устройства хранения нужно задействовать в установке. Если вы не уверены что выбрать, выберите `Basic Storage Devices`.

Следующим шагом вам нужно ввести имя для машины. Введите подходящее и выберите Next.

Вам нужно выбрать часовой пояс. Выбрали и нажали Next.

В этой точке вам нужно ввести пароль для *root*. Введите секретный пароль и подтвердите его повторным вводом. После ввода пароля нажмите Next.

Следующий экран спросит о типе установке. Если вы намереваетесь использовать весь носитель под эту установку, выберите *Use All Space* (использовать все пространство). Или выберите вариант

который больше вам подходит. Два checkboxes на экране в инсталяторе. Один включает шифрование на жестком диске. Другой позволяет вам просмотреть и изменить схему разделов которую создал установщик для вас. Выберите один или оба варианта, если нужно. После того как вы сделали необходимый выбор, нажмите Next для продолжения. У вас запросят дополнительное подтверждение на внесение изменений, поскольку все данные на жестком диске будут потеряны после этой точки.

Выделение точки монтирования /var в отдельный раздел

На системе, предназначенной для Asterisk, к каталогу */var* предъявляются повышенные требования. Это место где Asterisk будет хранить записи, сообщения голосовой почты, файлы журналов, подсказки и множество постоянно растущей информации. В нормальном режиме работы, маловероятно, что Asterisk заполнит весь жесткий диск. Тем не менее, если у вас включено активное логирование или идет запись всех звонков, то в теории, это может произойти. (Это может произойти через несколько месяцев после того как вы закончили установку и застать ваш персонал врасплох.)

Если диск с операционной системой методично заполняется, то есть шанс получить *kernel panic*. Отделяя */var* от остальной части жесткого диска, вы значительно снижаете риск сбоя системы.



Полностью заполненный объем диска по-прежнему является серьезной проблемой, однако, вы по крайней мере сможете войти в систему, чтобы исправить ситуацию.

На экране «Review and modify the partitioning layout», вы можете создать отдельное значение для */var*. Выбор *Yes* приведет к появлению инструмента Partitioning. Для точного создания разделов на диске, вам нужно знать размер жесткого диска; а он не соответствует маркировке на крышке диска, поскольку вы должны указать инструменту как разбивать диск. Ограничением инструмента является то, что нет возможности сказать «использовать все доступное пространство»; то есть вы не можете просто использовать 500 MB на раздел / и затем сказать «использовать оставшееся под */var*». Чтобы решить проблему запишите его размер выбрав / как весь диск, вычтите 500 MB из этого, и получите размер раздела /. Вычитаемая сумма будет зарезервирована под */var*.

Вам будет предложено выбрать набор программного обеспечения для установки. По умолчанию Basic Server достаточно, но, возможно, вам понадобится что-то еще под ваши потребности. Хотя можно легко установить дополнительные пакеты позже. После того как вы сделали выбор, нажмите Next, для продолжения. Затем установщик инсталлирует все необходимые пакеты, на основе вашего выбора. Это займет некоторое время.

На данный момент установка завершена. Вас попросят перезагрузить систему. Выберите Reboot.

Обновление базовой системы

После перезагрузки системы, вам нужно выполнить команду *yum update*, чтобы убедиться, что у вас последняя версия базовых пакетов. Чтобы сделать это, войдите, используя пользователя **`root`** и пароль созданный вами во время установки. После входа в систему, выполните следующие действия:

```
# yum update

Is this ok [y/N]: y
```

Вам будет предложено установить последние пакеты, нажмите *y* и ждите обновления пакетов. Если вас попросят принять ключи GPG, нажмите *y*. Когда закончите, перезагрузите систему, поскольку есть большая вероятность, что ядро было обновлено[5]:

```
# reboot
```

Поздравляем! Вы успешно установили и обновили базовую систему RHEL.

5 Эта перезагрузка необходима перед установкой Asterisk.

Включение NTP для точного времени в системе

Поддержка точного времени необходима в системе Asterisk, как для точности подробных записей вызов (CDR), так и для синхронизации с другими программами. Вы же не хотите что бы время ваших сообщений голосовой почты отличалось на 10 или 20 минут, так как это может привести к путанице и панике и, кто-то может подумать, что их сообщения голосовой почты доставляются слишком медленно. Команда *ntpd* может быть использована для синхронизации времени на вашем сервере Asterisk с остальным миром:

```
# yum install ntp
…
Is this ok [y/N]: y
…
# ntpdate pool.ntp.org
# chkconfig ntpd on
# service ntpd start
```

Идущая по умолчанию конфигурация с RHEL достаточна, чтобы синхронизировать время машины с остальным миром.

Добавление пользователя в систему

Процесс установки сервера Ubuntu попросит вас добавить пользователя в систему отличного от root, а RHEL нет. Для последовательного изложения в книге и для большей безопасности, мы собираемся добавить еще одного пользователя в систему и предоставить ему доступ к *sudo*[6]. Чтобы добавить нового пользователя, выполните команду *adduser*:

```
# adduser asteriskpbx
# passwd asteriskpbx
Changing password for user asteriskpbx.
New UNIX password:
Retype new UNIX password:
```

Теперь нам нужно обеспечить доступ пользователя *asteriskpbx* к *sudo*. Мы делаем это путем изменения файла *sudoers* командой *visudo*. Выполните команду *visudo* и посмотрите на строку ниже:

```
# visudo

## Allows people in group wheel to run all commands
%wheel      ALL=(ALL)         ALL
```

Снимите комментарий с строки `%wheel` как показано в нашем примере, сохраните файл нажав Esc, а затем набрав **`:wq`** и нажмите Enter. Теперь откройте файл */etc/group* в вашем любимом редакторе (*nano* прост в использовании) и найдите строку которая начинается со слова `wheel`. Измените её примерно так:

```
wheel:x:10:root,asteriskpbx
```

Сохраните файл, выйдите из под *root* набрав **`exit`**, и зайдите под созданным пользователем *asteriskpbx*. Протестируйте доступ к *sudo* запустив следующую команду:

```
$ sudo ls /root/
[sudo] password for asteriskpbx:
```

После ввода вашего пароля, вы должны получить доступ к содержимому каталога */root/*. Если этого не произошло, вернитесь и проверьте все шаги, чтобы убедиться, что вы не ничего пропустили или не сделали опечатку. При выполнении остальных инструкций в этой главе будем считать, что вы под пользователем *asteriskpbx* и что у вас есть доступ к *sudo*.

6 *sudo* - это команда, которая позволяет пользователю выполнять команды от имени иного пользователя, такого как root, или superuser.

С установкой операционной системы закончили, теперь вы готовы к установке зависимостей необходимых для Asterisk. Следующий раздел посвящен Ubuntu, так что вы можете сразу перейти к разделу с названием «Зависимости программного обеспечения», который предусматривает углубленный обзор процесса установки. Кроме того, если вы уже изучили информацию в этом разделе, вы можете обратиться к разделу "Шпаргалка по инсталляции" для поверхностного обзора, как установить Asterisk.

Ubuntu сервер

Ubuntu сервер - это популярный свободный дистрибутив Linux, созданный на базе Debian. Существует также популярная десктоп версия дистрибутива. Сервер Ubuntu не содержит GUI (графической оболочки) и идеально подходит для установки Asterisk.

Возьмите последнюю версию Ubuntu Server, посетив http://www.ubuntu.com[7] и выбрав пункт *Ubuntu Server* в выпадающем меню *Download*. Вы будите перемещены на страницу, содержащую варианты загрузки. Нажмите оранжевую кнопку *Download* и спустя несколько секунд начнется загрузка дистрибутива. Обратите внимание, что начиная с версии 18.04 32-битная редакция не поддерживается.

После того как вы скачаете ISO файл, запишите его на CD-диск (или USB-флешку) и запустите процесс установки. Если вы производите установку на виртуальную машину (мы не рекомендуем этого делать для продакшена, но это будет прекрасным способом протестировать Asterisk), вы должны примонтировать ISO-файл и устанавливать с него.

Установка базовой системы

При загрузке с CD, вам будет показан экран с предложением выбрать ваш язык. По умолчанию стоит английский (English) и, после небольшой паузы, он будет выбран автоматически. После выбора вами языка, нажмите Enter.

Следующий экран предоставит вам несколько опций, первая установить сервер - Install Ubuntu Server. Выберите её и нажмите Enter.

Затем вас спросят, какой язык вы хотите использовать при установке (да, это слегка избыточно). Выберите ваш язык (по умолчанию English), и нажмите Enter.

Вам предложат выбрать страну из списка. Найдите свою, установите на неё подсвечиваемую строку и нажмите Enter.

Следующим этапом будет определение используемой раскладки клавиатуры. Если вы знаете какой тип клавиатуры вы используете, вы можете выбрать No и затем выбрать её из списка форматов.

Если вы запустите детектор раскладки клавиатуры, вам будет предложено нажать несколько клавиш. Если детектор раскладки не сможет верно определить вашу клавиатуру (обычно когда установка происходит в виртуальной машине через удаленную консоль), вы можете вернуться и выбрать из списка вручную.[8]

После того как выбрали клавиатуру, установка будет продолжена, пытаясь автоматически настроить сеть. Если все пойдет хорошо, вам будет предложено ввести имя для системы. Вы можете ввести любое имя, если сеть не требует использовать определенное имя для вашей системы. Введите его, и нажмите Enter.

7 Конечно, проекты могут изменить свои веб-сайты, когда они хотят. Надеюсь, что инструкции представляемые здесь достаточно точны, чтобы помочь вам найти сайт, даже в случае изменений.

8 Вероятно, проще всего выбрать вручную US-keyboard, поэтому если у вас есть особые требования к клавиатуре, то пропуск автоматического определения процесса сэкономит ваше время.

Когда файл установки завершит работу, вам будет предложено ввести полное имя нового пользователя, из которого будет создано имя пользователя.[9] Система предложит имя пользователя, но вы можете изменить его на то, которое вам нравится.[10]

После ввода имени пользователя, вам будет предложено ввести пароль, а затем подтвердить его. Вы будете использовать их для входа в систему после окончания установки.[11]

Установщик спросит вас, хотите вы зашифровать ваш домашний каталог. В этом нет необходимости и это добавит нагрузку на процессор.



В дальнейших инструкциях по установке мы будем подразумевать, что имя пользователя было выбрано *asteriskpbx*

Программа установки попытается связаться с сервером Network Time Protocol (NTP) для синхронизации ваших часов. Ubuntu попытается автоматически определить ваш часовой пояс и покажет, что у неё получилось. Если она правильно определила выберите **Yes**, в противном случае выберите **No** и вы сможете в представленном списке временных зон выбрать нужную. Выберите свой часовой пояс, или выберите из мирового списка, если ваш часовой пояс не указан. После того как вы выбрали ваш часовой пояс, нажмите Enter чтобы продолжить.

Установщик задаст вам несколько вопросов о разметке вашей системы. Обычно, предложенная по умолчанию схема хорошая, она использует систему управления, занимает весь диск, и устанавливает Logical Volume Manager (LVM). Нажмите Enter когда вы сделаете свой выбор. Затем у вас спросят на какой раздел производить установку, скорее всего он будет только один в вашей системе. Нажмите Enter для продолжения, и вас еще раз попросят подтвердить изменения в таблице разделов. Выберите **Yes** и нажмите Enter для продолжения.

Теперь вас спросят, сколько места использовать (по умолчанию будет отведен весь диск). Нажмите Enter и затем введите и подтвердите размер пространства, которое вы хотите использовать. Установщик затребует последнее подтверждение до внесения изменений на диск. Выберите **Yes** для записи изменений на диск. После этого установщик отформатирует жесткий диск, запишет схему разделов, скопирует файлы, и выполнит файл установки.



Если вам удобнее использовать более сложные способы настройки файловой системы, вы можете захотеть поместить папку */var* в отдельный раздел из остальной файловой системы. Мысль об этом заключается в том, что файлы журналов и буферов (Астериск использует каталог *spool* для хранения голосовой почты и других записей) - это файлы, которые, скорее всего, заполнят ваш жесткий диск, а если вы поместите */var* в отдельный раздел, вам будет несколько легче получить доступ к файловой системе для выполнения экстренного обслуживания, если это потребуется.

Если ваша система находится за веб-прокси, введите сейчас информацию о прокси. Если вы не используете прокси, или ничего об этом не знаете, просто нажмите Enter**.**

Вам будет предложено, устанавливать обновления автоматически. Мы рекомендуем не производить никаких автоматических обновлений по умолчанию. Если система перезагрузится, при обновлении ядра, Asterisk может оказаться нестабильным, пока вы его заново не скомпилируете[12]. Лучшая практика - это устанавливать обновления регулярно и вручную, контролируя процесс. Как правило, обновление делают когда пользователи не работают или в нерабочее время (или пока работает резервная система). Выберите *No automatic updates* и нажмите Enter.

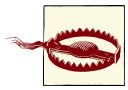
9 Мы решили использовать Asterisk PBX как полное имя для учетной записи.

10 Мы будем использовать имя пользователя *asteriskpbx*. Обратите внимание, что в Ubuntu зарезервировано имя пользователя *asterisk*, и именно поэтому мы выбрали в качестве нашего имени пользователя *asteriskpbx*.

11 Ubuntu не обеспечивает прямой доступ к root, но вместо этого использует приложение *sudo*, которое позволяет запускать команды как root, не являясь пользователем root.

12 В то время как мы говорим Asterisk, это проблема DAHDI. DAHDI представляет собой набор модулей ядра Linux используемые с Asterisk.

Мы будем устанавливать зависимые пакеты с помощью *apt-get*, а во время установки нам нужно выбрать только один пакет: OpenSSH сервер. SSH является важным пакетом, если вы хотите выполнять работу в системе удаленно. Однако, если вы будите работать только локально, согласно вашим политикам, вы можете не устанавливать сервер OpenSSH.



Предупреждение: Нажатие на клавишу Enter приведет к продолжению установки с выбранными пакетами. Вы должны использовать пробел для переключения вашего выбора.

После того как вы выбрали сервер OpenSSH, нажмите Enter.

Если это единственная операционная система на машине (что скорее всего), Ubuntu даст вам возможность установить загрузчик GRUB в вашей системе. Установщик даст вам возможность пропустить установку GRUB, поскольку это изменяет главную загрузочную запись (MBR) на вашей системе. Если есть другая операционная система, которую установщик не смог определить, и данные о которой не попадут в MBR, возможность пропустить изменение MBR в этом случае будет хорошей идеей. Если это единственная операционная система, установленная на вашем сервере, выберите **Yes**.

Когда система закончит установку, вам будет предложено удалить все носители на дисках и перезагрузить систему, выбрав **Continue**, после чего установка будет завершена, и система будет перезагружена.

Обновление базовой системы

Теперь, когда мы завершили установку Ubuntu Server, нам нужно выполнить обновление системы используя *apt-get*, чтобы убедиться, что у нас последняя версия установленных пакетов. Вам будет предложено ввести логин. Введите имя пользователя и пароль, созданные во время установки (например, *asteriskpbx*) и войдите в систему. После входа в систему, выполните следующую команду:

```
$ sudo apt-get update
[sudo] password for asteriskpbx:
…
Reading package lists... Done
$ sudo apt-get upgrade
Reading state information... Done
…
Do you want to continue [Y/n]? y
```



Заметка: Пароль *sudo* это пароль с которым вы вошли в систему.

Когда будет предложено продолжить нажмите клавишу Enter, будут установлены последние обновления пакетов на данный момент. Когда закончите, перезагрузите систему, чтобы изменения вступили в силу, так как, вероятно, ядро было обновлено.

```
$ sudo reboot
```

Поздравляем! Вы успешно установили и обновили базовую систему Ubuntu Server.

Включение NTP для точного системного времени

Поддержание точного времени необходимо в системе Asterisk, как для поддержания точных детальных записей вызова, а также для синхронизации с другими программами. Вы же не хотите, чтобы время уведомлений голосовой почты, сбивалось на 10 или 20 минут, так как это может привести к путанице и панике у тех, кто может подумать, что их уведомления о голосовой почте слишком долго доставляются:

```
$ sudo apt-get install ntp
```

Теперь перезапустите демон NTP:

```
$ sudo /etc/init.d/ntp restart
```

С установленной операционной системой, вы готовы к установке зависимостей необходимых для Asterisk. В следующем разделе приводится углубленный обзор процесса установки. Если вы уже рассмотрели информацию в разделе «Зависимости программного обеспечения» на стр 56, вы можете обратиться к разделу «Шпаргалка по установке» для обзора на детальном уровне, как установить Asterisk.

Зависимости программного обеспечения

Первое, что нужно сделать, после завершения установки операционной системы это установить зависимое программного обеспечения, требуемое для Asterisk. Команды, перечисленные в Таблице 3-1 были разделены на две колонки, для Ubuntu Server и RHEL Server. Эти пакеты позволят вам собрать базовую систему Asterisk, вместе с DAHDI и LibPRI. Не все модули можно скомпилировать с этими зависимостями, а только часто используемые. Если нужны дополнительные зависимости, для других модулей, используемых в этой книге далее, то инструкции будут предоставлены по мере необходимости.

Таблица 3-1. Зависимое программное обеспечение для Asterisk на Ubuntu Server и RHEL Server

Ubuntu	RHEL

`sudo apt-get install build-essential \`
`subversion libncurses5-dev libssl-dev \`
`libxml2-dev libsqlite3-dev uuid-dev vim-nox`
	`sudo yum install gcc gcc-c++ make wget \`
	`subversion libxml2-devel ncurses-devel \`
	`openssl-devel sqlite-devel libuuid-devel`
	`vim-enhanced`

Эти пакеты почти все, что вам нужно, чтобы начать установку Asterisk, DAHDI, и LibPRI. Обратите внимание, что пакеты которые мы указали также имеют зависимости которые нужно удовлетворить установкой соответствующих пакетов. Зависимости будут разрешены автоматически, если вы используете *yum* или *apt-get*.

Мы также включили библиотеки разработчика OpenSSL, которые не являются строго необходимыми для компиляции Asterisk, но их неплохо бы иметь: они позволяют поддерживать ключи и другие функции шифрования.

В качестве редактора мы установили *vim*, но вы можете выбрать любой, например, *nano*, *joe*, или *emacs*.



Asterisk содержит скрипт, который будет определять зависимости для всех функций Asterisk. В текущее время он полностью поддерживает Ubuntu, но у него нет полного списка необходимых для RHEL пакетов. После того как вы скачали Asterisk, используя инструкции в разделе "Загружаем то, что нам нужно", используйте следующие команды, для его запуска:

```
$ cd ~/src/asterisk-complete/asterisk/11
$ sudo ./contrib/scripts/install_prereq install
$ sudo ./contrib/scripts/install_prereq install-unpackaged
```

Сторонние репозитории

Для некоторых зависимостей пакетов нужны сторонние репозитории. Это, часто наблюдается при использовании RHEL. Вот пара репозиториев, которые, возможно, разрешат все дополнительные зависимости RPMforge и EPEL (ExtraPackages for Enterprise Linux).

Мы можем время от времени обращаться к этим сторонним репозиториям, по необходимости для разрешения зависимостей компилируемых и используемых модулей.

Загружаем то, что нам нужно

Есть несколько способов получения Asterisk: через репозиторий кода Subversion, с помощью *wget* с сайта загрузок, или через менеджер пакетов системы, такого как *apt-get* или *yum*. Мы собираемся описать только первые два метода, так как мы заинтересованы в сборке последней версии Asterisk из исходников. Как правило, менеджер пакетов системы будет иметь версии, которые старше, чем с Subversion или сайта загрузки, а мы хотим убедиться, что у нас присутствуют самые последние исправления.

Прежде чем мы скачаем файлы с исходным кодом, давайте создадим структуру каталогов под них. Мы собираемся создать структуру каталогов в домашней директории пользователя *asteriskpbx*. Как только соберем пакет, он будет установлен командой *sudo*. Затем мы вернемся и изменим права доступа и владельца установленных файлов, чтобы построить безопасную систему. Для начала выполните следующую команду:

```
$ mkdir -p ~/src/asterisk-complete/asterisk
```

Теперь, когда мы создали структуру каталогов, для хранения файлов, давайте получим исходный код. Выберите один из следующих двух методов, чтобы получить файлы:

- Subversion
- wget

Получение последней версии

Asterisk представляет собой постоянно развивающийся проект, и у него существует много различных версий, которые можно реализовать.

В Главе 2 мы говорили о версиях Asterisk. Важно понять концепцию версий Asterisk, потому что система версий системы Asterisk претерпела некоторые изменения на протяжении многих лет. Поэтому, если вы не до конца разобрались с версиями Asterisk, мы настоятельно рекомендуем вам вернуться и прочитать раздел «Управление версиями Asterisk» на стр 40.

Хотя в большинстве случаев все, что вам нужно сделать, это взять последнюю версию с веб-сайта Астериск. Мы будем устанавливать и использовать Asterisk 11 на протяжении всей книги.

Получение исходного кода из Subversion

Subversion - это система контроля версий, которая используется разработчиками для отслеживания изменений в коде за определенный период времени. Каждый раз, когда код изменен, он должен быть сначала извлечен из репозитория, а затем записан обратно, и этот момент изменения регистрируется. Таким образом, если созданное изменение регрессивное, разработчики могут вернуться к этому изменению и удалить его при необходимости. Это мощная и надежная система разработки. Иногда также это оказывается полезно и для администраторов Asterisk, стремящихся восстановить программу. Чтобы скачать исходный код последней версии Asterisk 11, используйте эти команды:

```
$ cd ~/src/asterisk-complete/asterisk
```

```
$ svn co http://svn.asterisk.org/svn/asterisk/branches/11
```

Теперь вы можете перейти к разделу «Как его устанавливать».



Приведенными командами будет получен исходный код с последними изменениями, которые были сделаны после последнего релиза. Если вы предпочитаете использовать версию релиза, пожалуйста, обратитесь к следующему разделу.

Получение исходного кода используя wget

Для получения последней выпущенной версии DAHDI, LibPRI, и Asterisk с помощью приложения *wget*, выполните следующие команды:

```
$ cd ~/src/asterisk-complete/asterisk
$ wget \
http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-
current.tar.gz
$ tar zxvf asterisk-11-current.tar.gz
$ wget \
http://downloads.asterisk.org/pub/telephony/libpri/libpri-1.4-current.tar.gz
$ tar zxvf libpri-1.4-current.tar.gz
$ wget \
http://downloads.asterisk.org/pub/telephony/dahdi-linux-complete/\
dahdi-linux-complete-current.tar.gz
$ tar zxvf dahdi-linux-complete.tar.gz
```

Следующим шагом является компиляция и установка программного обеспечения, так что вперед, к следующему разделу.

Как его устанавливать

С загруженными исходными файлами вы можете скомпилировать программу и установить ее.

Как правило, все, что нужно для запуска Asterisk - это само программное обеспечение Asterisk, поэтому если вы используете физический интерфейс PSTN (например сети PRI) в вашей системе есть два нужных вам программных пакета для того, чтобы подключить интерфейс PSTN к Asterisk. Эти два пакета называются LibPRI и DAHDI. Позже мы обсудим их более подробно. На данный момент, мы собираемся просто установить их для полноты картины. Если хотите, вы можете пропустить установку LibPRI и DAHDI, однако, поскольку нет никакого вреда или наказания от их наличия, мы рекомендуем установить их в вашу систему с самого начала.

Порядок установки такой:

1. DAHDI[13]
2. LibPRI[14]
3. Asterisk[15]

Установка в таком порядке гарантирует, что любые зависимости для DAHDI и Asterisk установленные до запуска конфигурационных скриптов, будут разрешены и модули которые зависят от LibPRI или DAHDI будет скомпилированны.

Итак, давайте начнем.

DAHDI

Digium Asterisk Hardware Device Interface, или DAHDI (ранее известный как Zaptel) - это программа Asterisk, используемая для взаимодействия с телефонным оборудованием. Мы рекомендуем вам установить её, даже если у вас нет установленного оборудования, поскольку DAHDI требуется при компиляции модулей времени `res_timing_dahdi` и используется приложениями диалплана Asterisk, такими как `MeetMe()`.

13 Этот пакет содержит драйверы ядра, позволяющие Asterisk подключаться к традиционным сетям PSTN. Он также необходим для работы конференции MeetMe(). Опять же, мы установим это для полноты картины.

14 Строго говоря, если вы не собираетесь использовать любые соединения ISDN (BRI и PRI), вы можете установить Asterisk без LibPRI. Тем не менее, мы собираемся установить его для полноты картины.

15 Если вы не установите его, ни один из примеров, приведенных в этой книге не будет работать, но книгу все еще можно читать в большой ванной комнате. Просто говорю.

DAHDI-инструменты и DAHDI-Linux

DAHDI на самом деле является комбинацией двух отдельных баз кода: *DAHDI-tools (инструменты)*, который предоставляет различные инструменты администратора, как то *dahdi_cfg*, *dahdi_scan* и т.д.; и *DAHDI-linux*, который обеспечивает драйверы ядра. Вы не должны обновлять их порознь, их нужно устанавливать оба одновременно, и называется он *DAHDI-linux-complete*. Нумерация версий для *DAHDI-linux-complete* будет выглядеть 2.6.1+2.6.1, где число слева от знака плюс - версия *DAHDI-linux*, и справа от знака плюс - номер версии *DAHDI-tools*.

Также есть драйверы DAHDI для FreeBSD, которые поддерживаются сообществом. Эти драйверы доступны на веб-сайте Астериск.

Другой зависимостью, необходимой для установки DAHDI, являются исходные коды ядра. Важно, что бы используемая версия ядра в точности совпадала с установленным исходным кодом ядра. Вы можете использовать *uname -a* для проверки, какая версия ядра в данный момент работает:

RHEL:

```
sudo yum install kernel-devel
```

Ubuntu:

```
sudo apt-get install linux-headers-`uname -r`
```

Используйте *uname -r* взяв его в кавычки (`) для заполнения текущей версии ядра для установки соответствующего пакета.

Следующие команды показывают как установить *DAHDI-linux-complete* 2.6.1+2.6.1. В то время, когда вы это читаете может быть доступна новая версия этого пакета, поэтому сперва проверьте downloads.asterisk.org.



Для списка текущих тегов (версий) DAHDI, нужно выполнить команду: *svn ls http://svn.asterisk.org/svn/dahdi/linux-complete/tags*

Если доступна новая версия, замените номер версии в командах:

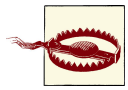
```
$ cd ~/src/asterisk-complete/
$ mkdir dahdi
$ cd dahdi/
$ svn co http://svn.asterisk.org/svn/dahdi/linux-complete/tags/2.6.1+2.6.1
$ cd 2.6.1+2.6.1
```

Если вы используете 64-bit RHEL, используйте следующие команды для компиляции и установки DAHDI:

```
$ cd tools
$ ./configure --libdir=/usr/lib64
$ make
$ sudo make install
$ cd ../linux
$ make
$ sudo make install
$ sudo make config
```

Для других платформ используйте эти команды компиляции и установки DAHDI:

```
$ make
$ sudo make install
$ sudo make config
```



При запуске этих команд, вы должны иметь доступ в интернет, поскольку они будут пытаться загрузить последнюю версию аппаратной прошивки (firmware) с серверов Digium.

После установки DAHDI, мы можем переходить к установке LibPRI. Информация о настройке DAHDI доступна в разделе “Линии PSTN (ТфОП)” в Главе 7.



Вы можете скачать исходный код используя *wget* с сайта Астериск. Проверьте сайт на наличие самой последней версии, так как эти версии наиболее вероятно устареют к тому моменту, когда вы прочтете это.

LibPRI

LibPRI - это библиотека, которая добавляет поддержку ISDN (PRI и BRI). Использование LibPRI не является обязательным, но так как он занимает очень мало времени на установку, ничто не мешает это сделать, и она пригодится, если вы захотите добавить платы в систему через некоторое время, мы рекомендуем вам установить её сейчас.

Получить последнюю версию LibPRI и собрать её можно так:

```
$ cd ~/src/asterisk-complete/
$ mkdir libpri
$ cd libpri/
$ svn co http://svn.asterisk.org/svn/libpri/tags/1.4.<номер вашей версии>
$ cd 1.4.<номер вашей версии>
$ make
```

Используйте следующую команду для установки LibPRI если вы используете 64-bit RHEL:

```
$ sudo make install libdir=/usr/lib64
```

Для других платформ используйте эту команду для установки LibPRI:

```
$ sudo make install
```



Вы также можете скачать исходный код используя wget из веб-сайта Астериск. Чтобы просмотреть текущие теги (версии) LibPRI, вы можете запустить команду: *svn ls http://svn.asterisk.org/svn/libpri/tag*

Asterisk

После установки DAHDI и LibPRI, мы сейчас можем приступить к установке Asterisk:

```
$ cd ~/src/asterisk-complete/asterisk/11
```

Если вы работаете с 64-bit RHEL, запустите скрипт configure с опцией libdir:

```
$ ./configure --libdir=/usr/lib64
```

На других платформах, запустите скрипт configure без каких либо аргументов.

```
$ ./configure
```

Завершаем компиляцию и установку Asterisk.

```
$ make
$ sudo make install
$ sudo make config
```

(*make* компилирует код, *make install* перемещает скомпилированный код в правильное размещение в файловой системе, и *make config* настраивает операционную систему для работы Asterisk как сервис, так чтобы она запускалась во время загрузки).

Еще больше доработанной документации

В Asterisk 11 полная документация для AMI (и в будущем, вероятно, еще больше статей документации) может быть получена, если Asterisk построен с *make full* чем просто *make*. Конечные файлы находятся в подкаталоге */doc* исходных кодов Asterisk и также устанавливаются в */var/lib/asterisk/documentation*.

Теперь файлы установлены в местах их расположения по умолчанию, нам нужно изменить права доступа каталогов и их содержимого.



Существует дополнительный шаг, который не является строго обязательным, но довольно распространенный (и пожалуй важный): команда *make menuselect*, которая через интерфейс псевдографики позволяет произвести детальный выбор, какие модули и функции будут скомпилированы. Мы обсудим это в разделе «make menuselect» на стр 67.

Установка разрешений для файлов

Для безопасного запуска нашей системы, мы установим Asterisk а затем и запустим его из под пользователя *asteriskpbx*. После установки файлов в их каталоги по умолчанию, нам нужно изменить разрешения для файлов, чтобы они совпадали с пользователем из под которого мы будем работать. Выполните следующие команды после запуска *make install* (что мы сделали ранее):

```
$ sudo chown -R asteriskpbx:asteriskpbx /var/lib/asterisk/
$ sudo chown -R asteriskpbx:asteriskpbx /var/spool/asterisk/
$ sudo chown -R asteriskpbx:asteriskpbx /var/log/asterisk/
$ sudo chown -R asteriskpbx:asteriskpbx /var/run/asterisk
```

Для того, чтобы использовать `MeetMe()` и DAHDI с Asterisk не из под *root*, вы должны изменить */etc/udev/rules.d/dahdi.rules*, что бы поля `OWNER` (владелец) и `GROUP` (группа) были не-*root*, а от имени кого запущен Asterisk. В нашем случае мы используем пользователя *asteriskpbx*.

Измените последнюю строку файла *dahdi.rules* на следующее:

```
SUBSYSTEM=="dahdi", OWNER="asteriskpbx", GROUP="asteriskpbx", MODE="0660"
```

После этого, мы можем перейти к выполнению базовой конфигурации, которую нужно делать после всех установок.

Базовая конфигурация

Теперь, когда мы установили Asterisk, мы можем запустить вашу систему. Наша цель - загрузить Asterisk и подготовить к работе, так как она еще ничего полезного не делает. Эти шаги, все системные администраторы должны сделать при установке новой системы. Если команды, которые должны быть выполнены отличаются в RHEL и Ubuntu, вы увидите таблицу, с указанными строками для каждого дистрибутива, в противном случае, вы увидите одну команду, которая должна выполняться независимо от дистрибутива.

Начальная настройка

Для того, чтобы Asterisk запустился, нам нужно создать некоторые конфигурационные файлы. Мы могли бы в принципе установить файлы примеров, которые идут вместе с Asterisk (выполнив команду *make samples* в исходниках Asterisk), а затем изменить эти файлы, под наши потребности, но команда *make samples* устанавливает большое количество файлов с примерами, большинство из них для модулей, которые вы никогда не будете использовать. Мы хотим ограничить загружаемые модули,

поскольку мы считаем, что легче понять конфигурацию Asterisk, если вы создаете свой конфигурационный файл с нуля. Поэтому мы собираемся создать наш собственный минимальный набор файлов конфигурации.[16]

Первое, что нужно сделать (если он еще не существует), это создать каталог */etc/asterisk/*, в котором будут жить наши конфигурационные файлы:

```
$ sudo mkdir -p /etc/asterisk/
$ sudo chown asteriskpbx:asteriskpbx /etc/asterisk/
```



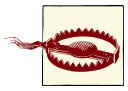
Вам нужны только права на запись в */etc/asterisk* для инструкций ниже и для некоторых дополнительных функций Asterisk, таких как файлы голосовой почты. (Пользователи меняют свой PIN-код голосовой почты к примеру)

Использование make samples для создания примеров конфигурационных файлов для дальнейшего использования

Даже если мы не собираемся использовать примеры конфигурационных файлов, которые идут с Asterisk, они являются отличным руководством. Если есть модуль, который вы не используете, но хотите запустить в работу, образцы файлов точно покажут вам, какой синтаксис использовать, и какие варианты имеются для этого модуля.

Запустите команду *sudo make samples* в каталоге исходников Asterisk[17], это безопасно на новых системах, которые только установлены, но её очень опасно запускать на системах, у которых уже есть конфигурационные файлы, так как эта команда перезапишет существующие файлы (это будет катастрофой для вас, если вы не имеете текущей резервной копии).

Если вы запустите команду *sudo make samples*, у вас может появиться желание переместить файлы, которые он создал в */etc/asterisk/* в другой каталог. Мы рекомендуем создать папку с именем */etc/asterisk/unused/* и положить туда копии всех файлов конфигурации, но вы можете хранить их где угодно.



Запуск *make samples* в системе, которая уже имеет конфигурационные файлы приведет к их перезаписи.

Теперь мы пройдем через все файлы, которые требуются, чтобы поднять простую систему Asterisk и запустить.

indications.conf и asterisk.conf

Первый файл indications.conf, содержит информацию о том, как обнаружить различные телефонные тона для разных стран. В исходниках Asterisk находится вполне годный пример файла, который мы можем использовать, так что давайте скопируйте его в наш каталог */etc/asterisk/*:

```
$ cp ~/src/asterisk-complete/asterisk/11/configs/indications.conf.sample \
/etc/asterisk/indications.conf
```

Поскольку мы запускаем Asterisk не от пользователя root, мы должны сказать Asterisk от какого пользователя запускать. Это делается в файле *asterisk.conf*. Мы можем скопировать образец из исходников Asterisk в */etc/asterisk*:

16 Если ваш каталог */etc/asterisk/* имеет файлы, переместите эти файлы в другой каталог, или удалите их если вы уверены, что они вам больше не нужны.

17 */usr/src/asterisk-complete/asterisk/asterisk-11.<ваша версия>/*

```
$ cp ~/src/asterisk-complete/asterisk/11/configs/asterisk.conf.sample \
/etc/asterisk/asterisk.conf
```

Файл *asterisk.conf* содержит множество опций, которые мы не будем здесь рассматривать (они описаны в разделе «asterisk.conf»), но мы должны сделать небольшую правку. Ближе к концу раздела `[options]`, есть две опции, которые нам нужно включить: `runuser` и `rungroup`.

Откройте файл *asterisk.conf* в редакторе *nano* или *vim*: Уберите комментарий с опций `runuser` и `rungroup`, и измените их так чтобы каждая содержала *`asteriskpbx`* как присвоенное значение. Откройте файл */etc/asterisk/asterisk.conf* в *vim*:

```
$ vim /etc/asterisk/asterisk.conf
```

Затем измените файл убрав коментарий с двух строк начинающихся с `runuser` и `rungroup` и измените езначения на *asteriskpbx*.

```
runuser=asteriskpbx
rungroup=asteriskpbx
```

Теперь у нас есть все конфигурационные файлы, необходимые для запуска самой минимальной версии Asterisk[18]. Давайте быстро запустим Asterisk на переднем плане:

```
$ /usr/sbin/asterisk -cvvv
```



Мы указали полный путь к бинарнику asterisk, но если вы измените системную переменную `PATH` включив в неё каталог */usr/sbin/* вам не нужно будет указывать полный путь. См. раздел «Добавление пользователя в систему» для получения информации об изменении переменной окружения `$PATH`.

Asterisk успешно запустится без ошибок и предупреждений (хотя это и предупредит вас, что некоторые файлы отсутствуют), и предоставит вам интерфейс командной строки Asterisk (CLI). На данный момент не существует никаких модулей, минимальная функциональность ядра и ни одного модуля канала, с которым можно соединится, но Asterisk запущен и работает.

Выполните команду *module show* и Asterisk CLI покажет, что ни какие внешние модули не загружены[19]:

```
*CLI> module show
module                  Description              Use Count
0 modules loaded
```

Мы сделали это, чтобы просто продемонстрировать, что Asterisk может работать в очень минимальном состоянии, и не требует устанавливать десятки модулей для запуска. Давайте остановим Asterisk командой core stop now в CLI:

```
*CLI> core stop now
```

18 Таким образом, минимально, на самом деле, это совершенно бесполезно в этой точке. Но мы отвлеклись.

19 Вы увидите модуль `res_adsi`, но вы можете игнорировать его, поскольку это ошибка в некоторых версиях дистрибутива, и не нужно беспокоиться.

Команды оболочки Asterisk

Asterisk может работать либо как демон или как приложение. Вы запускаете его как приложение, когда вы собираете, тестируете и отлаживаете Asterisk, и в качестве демона, когда вы запускаете его в продакшен.

Команда для запуска Asterisk будет одинаковой, независимо от того, используете вы его в качестве демона или приложения:

asterisk

Однако, без каких-либо аргументов эта команда будет принимать значения, определенные по умолчанию, и запускать Asterisk в качестве фонового приложения. Другими словами, нет смысла запускать команду *asterisk* самостоятельно, а лучше преедать ей некоторые параметры, чтобы определить поведение, которое вы ожидаете. Ниже приведены некоторые примеры общего назначения.

-h

Эта команда выводит полезный список опций, которые можно использовать. Для получения полного списка всех опций и их описания, выполните команду *man asterisk*.

-c

Эта опция запускает Asterisk как приложение (на переднем плане). Это означает, что Asterisk связан с вашей пользовательской сессией. Другими словами, если вы закроете сеанс пользователя путем выхода из системы или потеряете соединение, Asterisk умрет. Этой опцией обычно пользуются при создании, тестировании и отладке, но вы не будите её использовать в производственном решении. Если вы запустили Asterisk таким образом, наберите **`core stop now`** в строке CLI, чтобы остановить Asterisk и выйти.

-v, -vv, -vvv, -vvvv, и mд.

Эта опция может быть использована с другими опциями в целях повышения детализации вывода на консоль (напр. `-cvvv`). Она аналогична команде CLI *core set verbose n*, где *n* любое целое число от 0 до 5 (любое целое число, большее чем 5 будет принято, но не будет предоставлять больше детализации). Иногда это полезно, можно не выводить детализацию вообще. Например, если вы хотите видеть только ошибки, уведомления и предупреждения, отключите детализацию, чтобы она не мешала просмотру других сообщений.

-d, -dd, -ddd, -dddd, и mд.

Эта опция может быть использована подобно -v, но вместо нормального вывода, она указывает уровень отладочной информации (которая в первую очередь полезна для разработчиков, которые хотят устранить проблемы с кодом). Вам также необходимо включить вывод отладочной информации в файл *logger.conf* (который мы рассмотрим более подробно в Главе 22).

-r

Эта команда необходима, если вы хотите подключиться к консоли CLI, а Asterisk запущен как демон. Вы, вероятно, будите использовать эту опцию, больше, чем любую другую для систем Asterisk, которые находятся в промышленной эксплуатации. Эта опция будет работать, только если у вас Asterisk уже выполняется как демон. Для выхода из CLI, если была использована эта опция наберите **`exit`**.

-T

Эта опция добавляет временные метки в вывод CLI.

-x

Эта команда позволяет вам передавать строку в Asterisk, так если бы эта строка была введена в CLI. Например, чтобы быстро получить список всех используемых каналов без запуска консоли Asterisk, просто наберите **`asterisk -rx 'core show channels'`**, и вы получите данные, которые вы искали.

-g

Этот параметр указывает Asterisk сбросить файл ядра, если он выходит из строя.

Мы рекомендуем вам попробовать различные комбинации этих команд для понимания, что они делают.

safe_asterisk

Когда вы устанавливаете Asterisk используя директиву *make config* создается сценарий под названием *safe_asterisk*, который запускается процессом *init* Linux каждый раз при перезагрузке.

Сценарий *safe_asterisk* приносит следующую пользу:

- Автоматический перезапуск Asterisk после аварии
- Может быть настроен отсылать email администратору если произошел сбой
- Указывает где сохраняются аварийные файлы (каталог */tmp* по умолчанию)
- Выполняет указанный скрипт если произошел сбой

Вам не нужно слишком много знать об этом скрипте. Только понимать, что он должен нормально работать. В большинстве сред этот сценарий отлично работает в своем формате по умолчанию.

modules.conf

Таким образом, нам удалось получить работающий Asterisk, но он не в состоянии сделать ничего полезного для нас до сих пор. Чтобы сказать Asterisk, какие модули мы хотим загрузить, нам нужен файл *modules.conf*.

Создайте файл *modules.conf* в вашем каталоге */etc/asterisk/* следующей командой (замените >> на > если вы хотите перезаписать существующий файл):

```
$ cat >> /etc/asterisk/modules.conf
```

Наберите (или скопируйте) следующие строки, и нажмите Ctrl+D в новой строке, когда закончите:

```
; Файл modules.conf используется для определения модулей, которые должен
; загружать (или не загружать).
;
[modules]
autoload=yes
```

Ctrl+D

Использование cat для быстрого создания файлов и добавления содержимого в них

В системе Linux есть много способов, когда необходимо создать файл и затем добавить какое-то

содержимое в него . Обычно используется команда *touch* для создания файла, и затем открываем созданный файл в редакторе для добавления содержимого. Однако, есть менее известный путь сделать тоже самое. Давайте создадим файл и добавим в него содержимое одновременно:

- Используем программу *cat* для перенаправления вывода в файл (используем >> для добавления, или > для перезаписи).
- Вставить или набрать содержимое которое мы хотим добавить в файл.
- Нажмите Ctrl+D для завершения ваших изменений.

Поздравляем! Файл создан и содержимое добавлено.

Строка `autoload=yes` говорит Asterisk автоматически загружать все модули, размещенные в каталоге */usr/lib/asterisk/modules/*. Если хотите, можете оставить файл в таком состоянии и Asterisk будет просто загружать все модули, находящиеся в папке *modules*.

С началом использования файла *modules.conf*, запуск Asterisk может привести к загрузке целой кучи модулей. Вы можете это проверить, запустив Asterisk и команду *module show*:

```
$ asterisk -c
*CLI> module show

Module            Description                                Use   Count
app_adsiprlog.so        Asterisk ADSI Programming Application     0
app_alarmreceiver.so    Alarm Receiver for Asterisk               0
…
res_timing_timerfd.so   Timerfd Timing Interface                  0
192 modules loaded
```

Теперь у нас есть много загруженных модулей и множество дополнительных приложений и функций в диалплане.



Вы заметите, некоторые модули не загружаются, потому что их конфигураций файлы не были созданы. Мы разберемся с этими модулями поздней.

Нам не нужны все эти загруженные ресурсы, так давайте отфильтруем некоторые, непонятные модули, которые нам не нужны на данный момент. Измените файл *modules.conf* добавляя строки содержащие `noload`, которые скажут Asterisk пропустить загрузку этих модулей:

```
; Файл modules.conf используется для определения какие модули Asterisk должен
; загружать (или не загружать).
;
[modules]
autoload=yes

; Модули ресурсов
noload => res_speech.so
noload => res_phoneprov.so
noload => res_ael_share.so
noload => res_clialiases.so
noload => res_adsi.so

; Модули PBX
noload => pbx_ael.so
noload => pbx_dundi.so

; Модули каналов
noload => chan_oss.so
noload => chan_mgcp.so
```

```
noload => chan_skinny.so
noload => chan_phone.so
noload => chan_agent.so
noload => chan_unistim.so
noload => chan_alsa.so

; Модули приложений
noload => app_nbscat.so
noload => app_amd.so
noload => app_minivm.so
noload =>app_zapateller.so
noload => app_ices.so
noload => app_sendtext.so
noload => app_speech_utils.so
noload => app_mp3.so
noload => app_flash.so
noload => app_getcpeid.so
noload => app_setcallerid.so
noload => app_adsiprog.so
noload => app_forkcdr.so
noload => app_sms.so
noload => app_morsecode.so
noload => app_followme.so
noload => app_url.so
noload => app_alarmreceiver.so
noload => app_disa.so
noload => app_dahdiras.so
noload => app_senddtmf.so
noload => app_sayunixtime.so
noload => app_test.so
noload => app_externalivr.so
noload => app_image.so
noload => app_dictate.so
noload => app_festival.so
```

Есть, конечно, и другие модули, которые можно удалить, и модули, которые вы сочтете чрезвычайно полезными, так что не стесняйтесь изменять этот файл как захотите. В идеале, вы должны загружать только модули, необходимые для использования в вашей системе. Примеры, приведенные далее в этой книге предполагают, что файл *modules.conf* выглядит как наш пример, приведенный выше.

Дополнительная информация о файле *modules.conf* может быть найдена в разделе называемом “modules.conf” в Главе 4.

musiconhold.conf

Файл *musiconhold.conf* определяет классы для музыки на удержание в вашей системе Asterisk. Определяя различные классы, можно указать различную музыку для удержания, которая будет использоваться в различных ситуациях, таких, как различные объявления для воспроизведения, удержания в очереди, или другую музыку если у вас есть несколько АТС, размещенных в одной системе. А сейчас мы просто создадим класс музыки на удержание `default` (по умолчанию), который содержит немного музыки при переводе вызова на удержание:

```
$ cd /etc/asterisk/
$ cat >> musiconhold.conf

;musiconhold.conf
[default]

mode=files
```

```
directory=moh
```

Ctrl+D

Мы создали файл *musiconhold.conf* и определили наш класс музыки на удержание `[default]`. Мы также предполагаем, что вы установили музыку на удержание в системе через *menuselect*; по умолчанию установлен по крайней мере один музыкальны фрагмент, поэтому, если вы отключили его, то должны иметь музыку по крайней мере в одном формате.

Дополнительная информация о файле *musiconhold.conf* может быть найдена в разделе называемом “musiconhold.conf” в Главе 4.

make menuselect

menuselect текстовая утилита Asterisk с псевдографикой, которая используется для определения какие модули компилировать и устанавливать. Модули - это то, что дает Asterisk его мощь и функционалность. Новые модули создаются постоянно.



В каждой из различных категорий, список модулей разбивается на 3 части, которые представляют собой поддержку состояния модулей. Более подробная информация о состоянии поддержки модулей можно найти в разделе "Модули" и на сайте Asterisk wiki

В разделах о установке мы пропустили описание системы *menuselect* для того, чтобы изложить эти инструкции просто и понятно. Тем не менее очень важно, что мы выделили *menuselect* в свой собственный раздел.

Кроме определения, какие модули устанавливать, *menuselect* также позволяет установить флаги, которые могут помочь в отладке проблем (см. Главу 2), установить оптимизирующие флаги, выбрать другие звуковые файлы и форматы, и производить другие полезные вещи.

Использование menuselect

Нам понадобится целая глава для того, чтобы полностью изучить *menuselect*, и, по большей части, вам не придется делать больших изменений. Тем не менее, следующий пример даст вам представление о том как работает *menuselect*, и мы рекомендуем его для любой установки.

По умолчанию Asterisk устанавливает только звуковые файлы ядра (core), и только в формате GSM. Также, дерево музыкальных файлов на удержание OpSound доступных для загрузки, где отмечен только формат .wav.[20]

Мы собираемся установить дополнительные звуковые файлы, а не просто звуки ядра по умолчанию, и в формате звучащем лучше, чем GSM. Мы можем сделать это с помощью *menuselect*, запустив *make menuselect* в каталоге исходных файлов Asterisk. Прежде чем изучать что либо далее, давайте поговорим о различных интерфейсах *menuselect*.

Интерфейсы menuselect

Есть два интерфейса доступных для *menuselect*: curses и newt. Если библиотеки libnewt установлены, вы получите сине-красный интерфейс, как показано на Рисунке 3-1. В противном случае, по умолчанию *menuselect* будет использовать интерфейс curses (черно-белый), как показано на Рисунке 3-2.



Минимальный размер экрана для интерфейса curses - это 80x27 пикселей, что означает, что он может не загружаться, если вы используете стандартный размер терминала для простой установки дистрибутива. Это не проблема, если вы используете удаленный доступ по SSH, как правило, ваш терминал может быть изменен, но если вы работаете непосредственно за терминалом вам понадобится установить буфер экрана, позволяющий более высокое

20 Хороший способ поставить последний штрих в новой системе, это установить соответствующие звуковые файлы, которые будут использоваться в качестве музыки на удержании. Есть только три песни, установленные по умолчанию, и абоненты быстро устают слушать одни и теже три песни снова и снова. Мы обсудим это больше в разделе «musiconhold.conf».

разрешение, которое не рекомендуется для системы с Asterisk. Решение заключается в использовании интерфейса newt для *menuselect*.

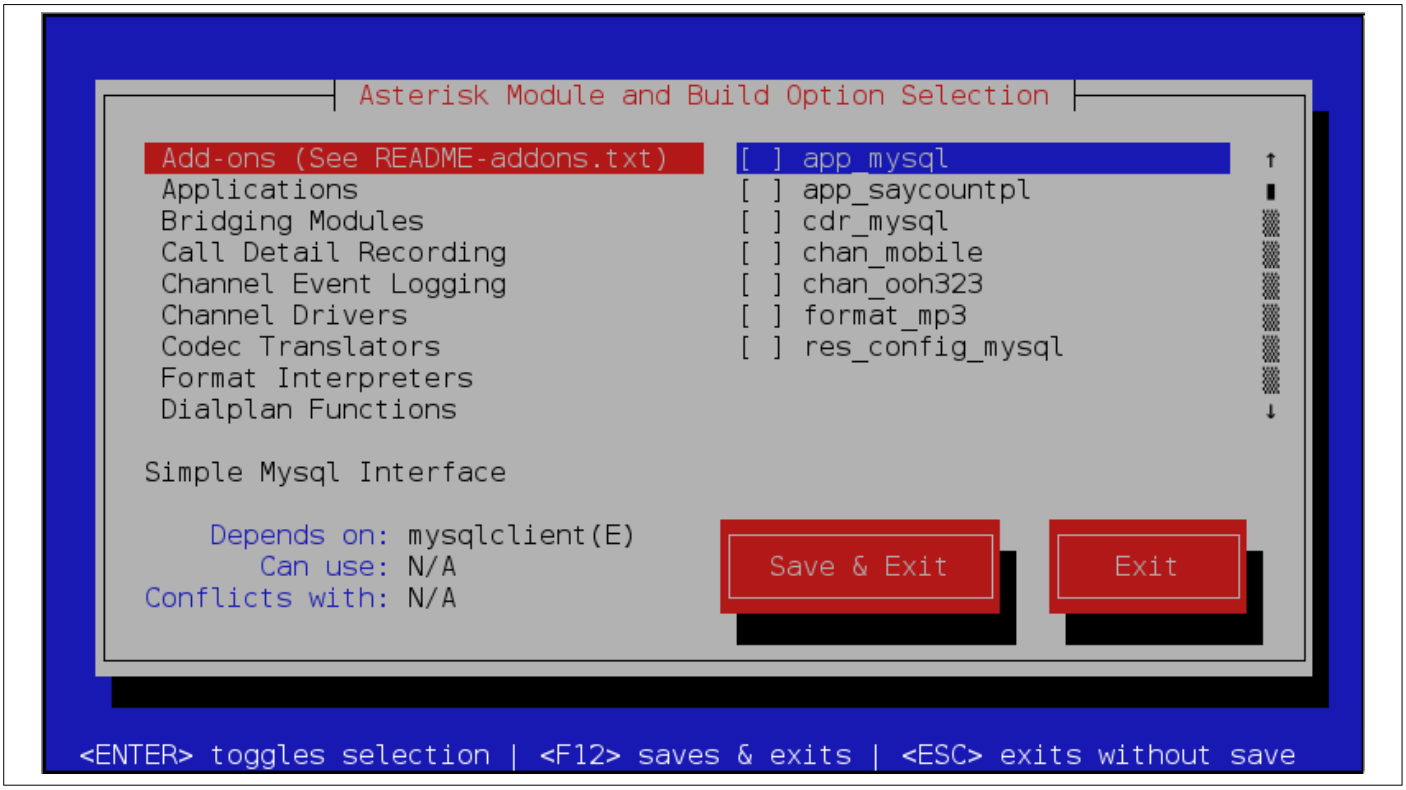


Рисунок 3-1. Использование интерфейса newt в menuselect

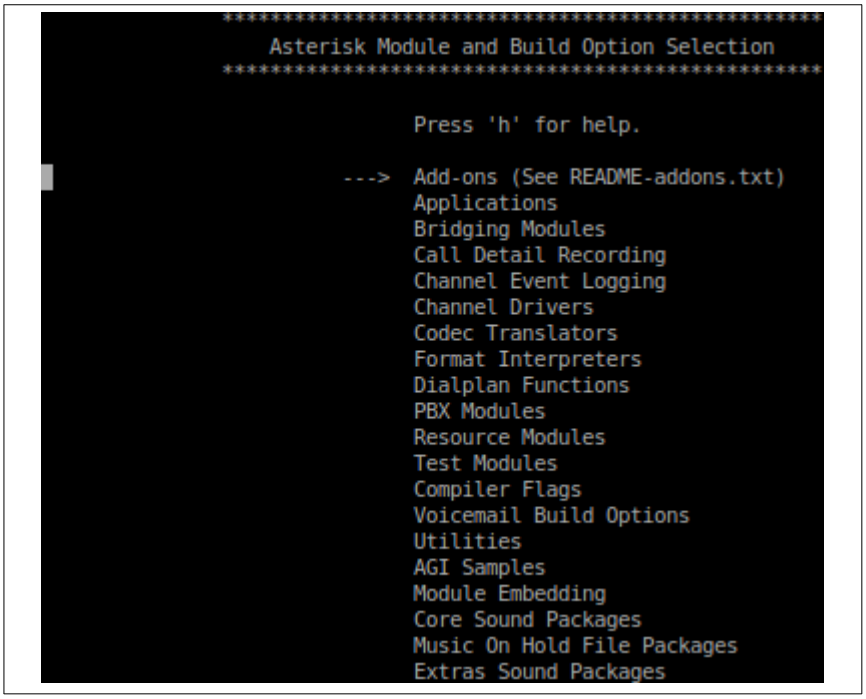


Рисунок 3-2. Использование интерфейса curses в menuselect.

Установка зависимостей для использования newt в menuselect

Для получения интерфейса newt для menuselect, вам нужно иметь установленными development библиотеки libnewt:

RHEL

```
sudo yum install libnewt-devel
```

Ubuntu

```
sudo apt-get install libnewt-dev
```

Если перед этим вы использовали *menuselect* с интерфейсом curses, вам нужно пересобрать Asterisk. Вы можете сделать это следующими командами:

```
$ cd ~/src/asterisk-complete/asterisk/11.<your version>/
$ cd menuselect
$ make clean
$ ./configure
$ cd ..
$ make menuselect
```

После чего интерфейс newt должет стать доступным.

Использование menuselect

Выполните следующие команды для запуска *menuselect*:

```
$ cd ~/src/asterisk-complete/asterisk/11.<your version>/
$ make menuselect
```

Вам будет представлен экран, подобный экрану на Рисунке 3-1 или Рисунке 3-2. Вы можете использовать клавиши со стрелками на клавиатуре для перемещения вверх и вниз. Клавиша стрелка вправо приведет вас в подменю, а клавиша стрелка влево вернет вас. Вы можете использовать пробел или клавишу Enter для выбора и отмены выбора модулей. Нажатие клавиши q прекратит работу без сохранения, а клавиша x сохранит выбранные вами параметры, а затем прекратит работу.

Зависимости модулей

Модули, которые имеют `XXX` перед собой, не могут быть скомпилированы, потому что конфигурационный скрипт не смог найти требуемых зависимостей (например, если у вас не установлен пакет разработчика u*nixODBC*, вы не сможете собрать `func_odbc`[21]). Всякий раз, когда вы устанавливаете зависимый пакет, вы всегда должны повторить *configure* до запуска *menuselect*, что бы новая зависимость была правильно отражена. Зависимый модуль станет доступена в *menuselect*. Если у модуля по-прежнему стоит `XXX`, значит скрипт *configure* по-прежнему не в состоянии найти зависимости или не все зависимости были удовлетворены.

После того как вы запустили *menuselect*, прокрутите вниз до `Core Sound Packages` и нажмите клавишу стрелка вправо (или Enter), чтобы открыть меню. Вам будет предложен список доступных опций. Эти опции представляют собой звуковые файлы ядра на различных языках и форматах. По умолчанию, выбран только набор файлов `CORE-SOUNDS-EN-GSM`, который на английском языке и в формате GSM.

Выберите `CORE-SOUNDS-EN-WAV` и `CORE-SOUNDS-EN-ULAW` (или `ALAW` если вы не в Северной Америке или Японии[22]), и другие звуковые файлы, которые могут быть применимы в вашей сети.



Причина, по которой у нас есть несколько форматов для одних и тех же файлов, заключается в том, что Asterisk может воспроизводить соответствующий формат в зависимости от того, какой кодек согласован конечной точкой. Это может значительно снизить нагрузку на процессор в системе.

21 Что мы рассмотрим в Главе 16, а также многие другие интересные вещи

22 Если вы хотите понять все о mu-law и a-law, вы можете прочесть раздел «Логарифмическая компандирование». Все, что вам нужно знать, это то, что за пределами Северной Америки и Японии, используется a-law.

После выбора соответствующих звуковых файлов, нажмите клавишу стрелка влево, чтобы вернуться в главное меню. Затем прокрутите вниз на две строки в меню `Extra Sound Packages` и нажмите клавишу стрелка вправо (или Enter). Вы увидите, что по умолчанию нет выбранных пакетов. Как и в звуковых файлах ядра, выберите нужный язык и формат для установки. Хороший выбор, для установки английских звуковых файлов это `WAV`, `ULAW`, и `ALAW` форматы.

После завершения выбора звуковых файлов, нажмите клавишу x для сохранения и выхода из *menuselect*. Вам затем нужно установить новые подсказки, скачивая их с сайта Asterisk. Это просто сделать, запустив *make install* снова:

```
$ sudo make install
$ sudo chown -R asteriskpbx:asteriskpbx /var/lib/asterisk/sounds/
```

Файлы будут скачаны, распакованы и установлены в нужном месте (по умолчанию */var/lib/asterisk/sounds/<language>/*). Вашему серверу Asterisk нужно иметь рабочее подключение к Интернету в случае получения файлов.

Скрипты menuselect

Администраторы часто создают инструменты при выполнении инсталляций на нескольких машинах, и Asterisk не является исключением. Если вам необходимо установить Asterisk на несколько машин, вы можете создать набор скриптов для автоматизации этого процесса. Система *menuselect* содержит параметры командной строки, которые можно использовать для включения или отключения модулей, которые нужно собрать и установить в Asterisk.

Если вы приступаете к новой наладке Asterisk вы должны сначала выполнить скрипт *configure* для того, чтобы определить, какие зависимости выполнены в системе. Затем вам нужно выбрать приложение *menuselect* и запустить команду *make menuselect-tree* для создания исходной древовидной структуры:

```
$ cd ~/src/asterisk-complete/asterisk/11.<your version>/
$ ./configure
$ cd menuselect
$ make menuselect
$ cd ..
$ make menuselect-tree
Generating input for menuselect ...
```

Для получения подробной информации о доступных параметрах, выполните *menuselect/menuselect --help* из корневого каталога исходников Asterisk. Вы увидите примерно такой вывод:

```
Usage: menuselect/menuselect [--enable <option>] [--disable <option>]
[--enable-category <category>] [--enable-all]
[--disable-category <category>] [--disable-all] […]
[<config-file> […]]
Usage: menuselect/menuselect { --check-deps | --list-options
|--list-category <category> | --category-list | --help }
[<config-file> […]]
```

Показанные опции можно использовать для контроля, какие модули устанавливать через приложение *menuselect*. Например, если вы хотите отключить все модули и установить базовую системы (от которой не будет большой пользы) можно использовать следующую команду:

```
$ menuselect/menuselect --disable-all menuselect.makeopts
```

Если вы затем посмотрите на файл *menuselect.makeopts*, вы увидите большое количество текста, который отображает все модули и категории, которые были отключены. Допустим, вы теперь хотите включить канал SIP и приложение `Dial()`. Включение этих модулей можно сделать с помощью следующей команды, но прежде мы посмотрим на текущий *menuselect.makeopts* (после отключения

всех модулей) и найдем `app_dial` в категории `MENUSELECT_APPS` и `chan_sip` в категории `MENUSELECT_CHANNELS`. После выполнения следующей команды, посмотрите на файл *menuselect.makeopts* снова, и вы увидите, что эти модули больше там не перечислены:

```
$ menuselect/menuselect --disable-all --enable chan_sip \
--enable app_dial menuselect.makeopts
```



Модули, перечисленные в файле *menuselect.makeopts* это те, которые не будут скомпилированы, модули которые не перечислены будут скомпилированы, когда приложение отработает *make*.

Вы можете затем создать файл *menuselect.makeopts* в любом виде с использованием других команд, которые позволят вам создавать собственные сценарии установки для вашей системы, используя любой язык сценариев.

Обновление Asterisk

Если это ваша первая установка, вы можете сразу перейти к разделу «Базовая конфигурация». Если вы находитесь в процессе обновления вашей системы, есть несколько вещей, которые вы должны знать.



Когда мы говорим *updating (обновление)* вашей системы, это довольно сильно отличается от *upgrading (модернизации)* вашей системы. Обновление системы - это процесс установки новых второстепенных версий одной и той же ветки. Например, если ваша система работает на Asterisk 11.2.0 и вам необходимо обновить до последней версии с исправленной ошибкой в ветке 11, которая будет версией 11.3.0, вы бы *обновляли* систему до 11.3.0. В противоположность этому, мы используем термин *модернизация* для обозначения изменений между ветвями Asterisk (значительное увеличение номера версии). Так, например, модернизация от Asterisk 10.0.0 до Asterisk 11.0.0.

При выполнении обновления, вы будете следовать тем же инструкциям, изложенным в разделе в разделе «Как его устанавливать» на стр 56.



Кроме того, если вы проверили новый каталог для этой версии Asterisk (запуском *svn up* на контролируемой ветке), а ранее использовали *menuselect* для выбора модулей для компиляции, вы можете скопировать файл *menuselect.makeopts* из одной директории в другую перед запуском *./configure*. При копировании *menuselect.makeopts* из старой версии в новую, вы пропускаете шаг где вы (не)выбираете все модули снова.

Базовые шаги такие:

```
$ cd ~/src/asterisk-complete/asterisk/11.<your version number>/
$ ./configure
$ make
$ sudo make install
$ sudo chown -R asteriskpbx:asteriskpbx
  {/var/lib,/var/spool,/var/log,/var/run}/asterisk
```

После установки, вы можете получить сообщение типа такого:

```
WARNING WARNING WARNING
Ваш каталог модулей Asterisk, расположенный по адресу
/usr/lib/asterisk/modules
содержит модули, которые не были установлены этой
версией . Пожалуйста убедитесь что эти модули
совместимы с данной версией перед
попыткой запустить Asterisk.

chan_mgcp.so
chan_oss.so
chan_phone.so
```

```
chan_skinny.so
codec_g729a.so

WARNING WARNING WARNING
```

Это предупреждение о том, что модули установленные в каталоге */usr/lib/asterisk/modules/* не совместимы с той версией что вы только что установили. Чаще всего это происходит, когда вы установили модули в одной из версий Asterisk, а потом установили новую версию Asterisk без компиляции этих модулей (процесс установки будет перезаписывать любые модули, которые существовали ранее, заменяя их на обновленные версии).



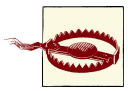
Вы *можете* выполнить безопасно *make install* поверх запущенной системы Asterisk. Попытка вручную выполнить команду *cp* над загруженным модулем, скорее всего, приведет к segfault (крах). Первый заменяет вместо перезаписи.

Чтобы обойти предупреждение, вы можете очистить каталог */usr/lib/asterisk/modules/* перед запуском *make install*. Если вы установили сторонние модули, типа коммерческих модулей от Digium (в том числе `res_fax_ digium`, `codec_g729a` и т.д.), вам придется переустановить их, если вы очистили ваш каталог модулей.

Рекомендуем держать каталог с вашими сторонними модулями в каталоге с исходниками, что бы можно было переустанавливать их при обновлении вашей системы Asterisk. Так, например, можно создать каталог */usr/src/asterisk-complete/thirdparty/11* следующим образом:

```
$ cd ~/src/asterisk-complete/
$ mkdir thirdparty/
$ mkdir thirdparty/11/
```

Загрузка сторонних модулей в этот каталог позволяет легко установить эти модули при обновлении. Просто следуйте инструкциям по установке модуля, многие из которых будут столь же простыми, как перезапуск *make install* из каталога исходников *modules* или скопируйте скомпилированные бинарные модули в каталог */usr/lib/asterisk/modules/*.



Проверьте, что права доступа к файлам совпадают с пользователем, запустившим Asterisk!

Общие проблемы

В этом разделе мы собираемся описать некоторые общие проблемы, которые могут возникнуть во время компиляции Asterisk, DAHDI, или LibPRI. Большинство вопросов, с которыми вам придется столкнуться связано с отсутствующими зависимостями. Если это так, пожалуйста, ознакомьтесь с разделом «Зависимости программного обеспечения», чтобы убедиться, что вы установили все необходимое.



Каждый раз, когда вы устанавливаете дополнительные пакеты, вам нужно будет запустить скрипт *./configure* в исходниках Asterisk для обнаружения нового пакета.

-bash: wget: command not found (команда не найдена)

Это сообщение говорит, что у вас не установлено приложение *wget*, которое требуется для скачивания пакетов с сайта Asterisk, звуковых файлов Asterisk, или для скачивания драйверов DAHDI (firmware или hardware).

Ubuntu	RHEL
`$ sudo apt-get install wget`	`$ sudo yum -y install wget`

configure: error: no acceptable C compiler found in $PATH

Это означает, что скрипт *configure* Asterisk не может найти ваш компилятор C, обычно это означает, что вы еще не установили его. Удостоверьтесь, что вы установили пакет *gcc* для вашей системы.

Ubuntu	RHEL
`$ sudo apt-get install gcc`	`$ sudo yum install gcc`

make: gcc: command not found (команда не найдена)

Это означает, что скрипт configure Asterisk не может найти ваш компилятор C, обычно это означает, что вы еще не установили его. Удостоверьтесь, что вы установили пакет *gcc* для вашей системы.

Ubuntu	RHEL
`$ sudo apt-get install gcc`	`$ sudo yum install gcc`

configure: error: C++ preprocessor “/lib/cpp” fails sanity check

Эта ошибка выдается скриптом *configure* Asterisk, когда у вас не установлены препроцессоры GCC C++.

Ubuntu	RHEL
`$ sudo apt-get install g++`	`$ sudo yum install gcc-c++`

configure: error: *** Please install GNU make. It is required to build Asterisk!

Это ошибка появляется, когда вы еще не установили приложение *make*, которое требуется для сборки Asterisk.

Ubuntu	RHEL
`$ sudo apt-get install make`	`$ sudo yum install make`

configure: *** XML documentation will not be available because the ‘libxml2’ development package is missing.

Вы встретите эту ошибку, когда библиотеки XML-парсера не установлены. Это требуется для Asterisk 1.8 и младше, так как документация в консоли (например, при запуске *core show application dial* в Asterisk CLI) формируется из XML.

Ubuntu	RHEL
`$ sudo apt-get install libxml2-dev`	`$ sudo yum install libxml2-devel`

configure: error: *** termcap support not found

Эта ошибка появляется, когда у вас не установлена библиотека разработчика `ncurses`, которая требуется для *menuselect* и для других консольных выводов в Asterisk.

Ubuntu	RHEL

```
$ sudo apt-get install ncurses-dev  $ sudo yum install ncurses-devel
```

You do not appear to have the sources for the 2.6.18-164.6.1.el5 kernel installed.

Вы получите эту ошибку, когда попытаетесь собрать DAHDI без установленных заголовков Linux, которые необходимы для создания Linux-драйверов.

Ubuntu	RHEL
`$ sudo apt-get install linux-headers-`uname -r``	`$ sudo yum install kernel-devel kernel-headers`

E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?

Если вы столкнулись с этой ошибкой то, вполне вероятно, что вы забыли перед запуском команды, которой требуются разрешения root, поставить *sudo*.

Модернизация (Upgrading) Asterisk

Обновление Asterisk между основными версиями, например, от 1.6.2 до 1.8 или от 10 до 11 сродни обновлениям операционной системы. После того, как АТС запущена в эксплуатацию, ужасно разрушительно для этой системы быть недоступной в течении любого отрезка времени, и модернизация телефонной системы должна быть хорошо продумана, спланирована, и испытана как можно сильней до развертывания. И, поскольку, каждое развертывание различается, трудно, если не невозможно, для нас, если вы пройдете через реальное обновление системы. Тем не менее, мы, безусловно, поможем направить вас в правильное русло для информации, требуемой вам для того, чтобы выполнить такие обновления, тем самым давая вам инструменты, необходимые для успеха.

Эксплуатируемая система Asterisk не должна быть модернизирована между основными версиями без предварительного развертывания её в среде разработки, где существующие файлы конфигурации могут быть проверены на возможности новых функций и изменения синтаксиса между версиями. Например, может оказаться, что ваш диалплан опирается на устаревшие команды и должен быть обновлен для использования новых команд, которые содержат больше функций, имеют лучшую базу кода, и будут обновляться на более регулярной основе. Команды, которые являются устаревшими, как правило, остаются в коде для обеспечения обратной совместимости, но вопросам об этих устаревших командах будет дан низкий приоритет, чем вопросам о новых предпочтительных методах.

Существуют два файла, которые следует читать перед любым обновлением системы: *CHANGES* и *UPGRADE.txt*, которые идут с исходным кодом Asterisk. Эти файлы содержат подробную информацию об изменениях в синтаксисе и другие вещи, информирующие об обновлении между основными версиями. Файлы разбиты на несколько разделов, которые описывают такие вещи, как синтаксические изменения диалплана, изменения синтаксиса драйвера канала, функциональные изменения, и предложения обновить файлы конфигурации, чтобы использовать новые методы.

Другим важным моментом при выполнении обновления является то, что вам действительно нужно, в первую очередь выполнить обновление. Если вы используете версию Asterisk с долгосрочной поддержкой (LTS)[23] и эта версия хорошо работает, не может быть никаких причин для обновления эксплуатируемой системы. Альтернатива модернизации всей системы - это добавление функциональности в системе, запустив две версии одновременно на отдельных системах. При запуске отдельных блоков, вы можете получить доступ к добавленным функциям в более поздней версии Asterisk без риска нарушить существующую систему. После этого можно выполнить миграцию постепенно без необходимости делать полное обновление системы за раз.

23 Более подробную информацию о релизах Asterisk и графике их поддержке можно получить на Asterisk wiki.



Мы настоятельно рекомендуем вам не использовать систему в продакшене для проверки новых функций и релизов. Вместо этого, создайте отдельную систему (виртуальная машина отлично подойдет для этого), которую можно использовать в качестве песочницы. Вы можете подключить её к эксплуатируемой системе с помощью двустороннего SIP-транка, тогда вы можете сделать свой внешний транк и внутренние номера доступными для машины в песочнице. В дополнение к этому, если вы планируете модернизацию системы в продакшене, вы можете создать новую установку Asterisk на другой виртуальной машине, и протестировать обновления перед модернизацией производственной системы. После того как вы проверили обновления и довольны результатами, у вас будет гораздо больше уверенности в процессе реального обновления.

Две части Asterisk, которые должны быть тщательно проверены при выполнении обновления между основными версиями: Asterisk Interface Manager (AMI) и Asterisk Gateway Interface (AGI).

Эти две части Asterisk полагаются на протестированный код, чтобы убедиться что любая очистка синтаксических изменений в AMI и в AGI или дополнительные функции не вмешиваются в существующий код. Выполняя аудит кода на предмет того что ваша программа ожидает получить или отправить и что происходит на самом деле, вы можете уберечь себя от головной боли в будущем.

Тестирование CDR также весьма важно, особенно если она используется как биллинг. Вся структура CDR на самом деле предназначена для простых вызовов, но она часто применяется в сложных потоках вызовов, и когда кто-то сообщает о проблеме, и она постоянна, то это может накладывать эффект на других, которые полагаются на ту же самую функциональность в различных целях. Asterisk 11 включает в себя регистрацию событий канала (CEL), которая представляет собой систему, предназначенную для обхода некоторых ограничений CDR в сложных вызовах (например, те где есть трансферы и т.д.). Более подробная информация о CEL доступна в разделе «CEL (Channel Event Logging)» в Главе 24.

Обновление Asterisk будет успешно проходить до тех пор, пока проводится хорошее планирование и тестирование до полного развертывания. В некоторых случаях переход на отдельную физическую машину, на которой вы выполняли тестирование, является лучшим вариантом, поскольку он может дать вам возможность отката в случае проблемы, которая не может быть решена немедленно. Планирование, и особенно резервное копирование, вот наиболее важный аспект обновления Asterisk.

Вывод

В этой главе мы рассмотрели, как установить операционную систему которая подходит для Asterisk (Ubuntu или RHEL), а также сам Asterisk. Мы улучшили надежность с помощью *sudo* и запускаем Asterisk от не-*root* пользователя *asteriskpbx*. Теперь у нас есть прочный фундамент, на котором строится наша система Asterisk. В следующих главах мы рассмотрим способы подключения устройств к нашей системе Asterisk, с тем, чтобы начать звонить внутри страны, и посмотрим как подключать к Asterisk внешние услуги для того, чтобы совершать и принимать телефонные звонки из ТфОП.

Задачи начальной настройки

Осторожно. Мы не требуем извлекать уроки из этого.
- Calvin & Hobbes

В предыдущей главе мы рассмотрели, как установить Asterisk. Но с чего вы должны начать работу с конфигурацией? Вот о чем рассказывает эта глава. Есть несколько общих файлов конфигурации, которые независимо от того, как вы используете Asterisk нужно настроить. В некоторых случаях они не требуют изменения, но вы должны быть осведомлены о них.

asterisk.conf

Файл конфигурации *asterisk.conf* позволяет настроить различные параметры, которые могут повлиять на работу Asterisk в целом.

Существует образец файла *asterisk.conf* в исходниках Asterisk. Не обязательно иметь этот файл в папке */etc/asterisk*, для рабочей системы, но вы можете обнаружить, что некоторые из возможных опций будут вам полезны.



Asterisk, по умолчанию, будет искать *asterisk.conf* в каталоге */etc/asterisk*. Чтобы указать другое место для *asterisk.conf*, используйте параметр *-C* в командной строке:

```
$ sudo asterisk -C /custom/path/to/asterisk.conf
```

Раздел [directories]

Для большинства установок Asterisk, нет нужды изменять расположение каталогов. Тем не менее, это может быть полезно для запуска нескольких экземпляров Asterisk одновременно, или если вы хотите хранить файлы в нетандартных местах.

В Таблице 4-1 приведены пути размещения каталогов по умолчанию, при желании их расположение можно изменить. Для получения дополнительной информации об использовании этих каталогов см. раздел «Файловая структура» в Главе 2.



Заголовок `[directories]` в примере *asterisk.conf* содержит `(!)`, который помечает его как шаблон (это значит, что изменения в этом заголовке не вступят в силу). Для того, чтобы изменения под `[directories]` давали эффект, удалите `(!)`. Смотри раздел “sip.conf” в Главе 5 для подсказок по использованию шаблонов.

Таблица 4-1. Раздел [directories] asterisk.conf

Опция	Значение/Пример	Примечание
`astetcdir`	`/etc/asterisk`	Место где находятся файлы конфигурации Asterisk.
`astmoddir`	`/usr/lib/asterisk/`	Место где хранятся подгружаемые модули.

	modules	
astvarlibdir	/var/lib/asterisk	Базовое место информации об изменении состояния используемое различными частями Asterisk. Включает элементы, записываемые Asterisk во время работы.
astdbdir	/var/lib/asterisk	Asterisk будет сохранять внешние базы данных в этом каталоге как файл с именем *astdb.*
astkeydir	/var/lib/asterisk	Asterisk, по умолчанию будет использовать подкаталог с названием keys в этом каталоге для загрузки ключей шифрования.
astdatadir	/var/lib/asterisk	Это базовый каталог для данных системы, таких как звуковые файлы, которые идут с Asterisk.
astagidir	/var/lib/asterisk/agi-bin	Asterisk, по умолчанию будет использовать подкаталог *agi-bin* для запуска скриптов AGI.
astspooldir	/var/spool/asterisk	Asterisk spool каталог, где будут храниться voicemail, записанные звонки, и spool создаваемых звонков.
astrundir	/var/run/asterisk	Место куда Asterisk будет записыватьсвои UNIX control socket, а также ID (PID) процесса.
astlogdir	/var/log/asterisk	Каталог где Asterisk будет сохранять свои log файлы.

Раздел [options]

В этом разделе файла *asterisk.conf* устанавливаются значения по умолчанию для глобальных параметров среды выполнения. Доступные параметры приведены в Таблице 4-2. Большинством из них можно еще управлять через командную строку приложения asterisk. Полный список параметров командной строки, в том числе и эти опции, можно изучить через страницу руководства Asterisk:

```
$ man asterisk
```

Таблица 4-2. Раздел [options] asterisk.conf

Опция	Значение/Пример	Примечание
verbose	3	Устанавливает значение степени детализации Asterisk. Это значение также устанавливается параметром *-v* командной строки. Уровень детализации по умолчанию равен 0.
debug	3	Устанавливает значение степени отладки Asterisk. Это значение также устанавливается параметром *-d* командной строки. Уровень отладки, по умолчанию равен 0.
alwaysfork	yes	Раздвоение заставляет Asterisk всегда работать в фоновом режиме. Эта опция устанавливается по умолчанию в значение no.
nofork	yes	Заставляет Asterisk всегда работать в активном режиме. Эта опция устанавливается по умолчанию в значение no.
quiet	yes	Тихий режим, уменьшает количество вывода в консоль, когда Asterisk выполняется в активном режиме. Эта опция устанавливается по умолчанию в значение no.
timestamp	yes	Добавляет временные метки на все выводы, кроме вывода из командной строки. Эта опция устанавливается по умолчанию в no.
execincludes	yes	Включает использование #exec в конфигурационных файлах Asterisk. Эта опция устанавливается по умолчанию в no.
console	yes	Запускает Asterisk в консольном режиме. Asterisk будет запущен в активном режиме и предоставит приглашение для команд CLI. Эта опция устанавливается по умолчанию в значение no.
highpriority	yes	Запускает приложение Asterisk с приоритетом реального времени. Эта опция устанавливается по умолчанию в no

Опция	Значение/Пример	Примечание
initcrypto	yes	Загружает ключи из astkeydir при запуске. Эта опция устанавливается по умолчанию в no.[a]
nocolor	yes	Подавляет цветной вывод в консоли Asterisk. Это полезно когда вывод консоли перенаправляется в файл. Эта опция устанавливается по умолчанию в no.
dontwarn	yes	Отключает некоторые предупреждающие (warning) сообщения. Эта опция была введена для подавления предупреждений, которые в целом правильны, но могут быть настолько очевидны, что сильно раздражают. Эта опция устанавливается по умолчанию в no.
dumpcore	yes	Говорит Asterisk генерировать дамп ядра в случае аварии. Эта опция устанавливается по умолчанию в no.[b]
languageprefix	yes	Определяет, как язык используется при построении пути к звуковому файлу. По умолчанию это yes, который помещает язык перед любыми подкаталогами, такими как *en/digits/1.gsm*. Установка этого параметра в значение no приводит к тому, что Asterisk ведет себя так же, как и в предыдущих версиях, помещая язык в качестве последнего каталога в пути (например, *digits/en/1.gsm*)
internal_timing	yes	Использует источник синхронизации для синхронизации звука, который будет отправлен на канал в случае воспроизведения файлов или музыки на удержание. Эта опция, по умолчанию, устанавливается в yes, и должна оставаться такой, её полезность значительно снизилась за последние несколько основных версий Asterisk.
systemname	my_system_name	Дает этому экземпляру Asterisk уникальное имя. Когда опция установлена, имя системы будет использоваться как часть поля uniqueid для каналов. Это крайне полезно, если несколько экземпляров Asterisk будут писать лог CDR в таблицы базы данных. По умолчанию эта опция не установлена.
autosystemname	yes	Автоматически устанавливает имя системы используя hostname системы. Эта опция устанавливается по умолчанию no.
mindtmfduration	80	Устанавливает минимальную длительность сообщений DTMF. Если Asterisk получает сообщение DTMF с длительностью менее этого значения, значение продолжительности сообщения DTMF будет изменено на указанное в этом параметре.
maxcalls	100	Устанавливает максимальное количество одновременно входящих каналов. По умолчанию не ограничено.
maxload	0.9	Устанавливает максимум средней нагрузки. Если средняя нагрузка равна или превышает этот порог, Asterisk не будет принимать новые звонки. Порог не устанавливается по умолчанию.
maxfiles	1000	Устанавливает максимальное количество дескрипторов файлов, которое разрешено открыть Asterisk. По умолчанию предел, установленный в системе обычно равен 1024, этого недостаточно для высоконагруженных систем. Опция является общей для установки этого предела до очень высокого значения. Значение лимита, установленного в системе, обычно используется по умолчанию.[c]
minmemfree	1	Устанавливает минимальное количество мегабайт свободной памяти, необходимой Asterisk, чтобы продолжать принимать звонки. Если Asterisk обнаруживает, что свободной памяти меньше, чем это значение, новые звонки приниматься не будут. Эта опция не устанавливается по умолчанию.
cache_record_files	yes	Когда идет запись, файл сохраняется в record_cache_dir

Опция	Значение/Пример	Примечание
		пока запись не завершится. После завершения, она будет перемещена в изначально указанное размещение. По умолчанию значение опции no.
record_cache_dir	/tmp	Устанавливает используемый каталог, когда cache_record_files установлена в yes. По умолчанию каталог astspooldir располагается в *tmp*.
transmit_silence	yes	Передает тишину абоненту в случае, когда нет других источников звука. Это включает в себя запись звонка и семейство приложений диалплана Wait(), а также другие вещи. По умолчанию значение no.[d]
transcode_via_sln	yes	При создании пути кодека транскодирования, sln будет одним из частей в пути. По умолчанию значение опции yes.
runuser	asterisk	Устанавливает пользователя системы от которого будет запущено приложение Asterisk. Эта опция не установлена по умолчанию и означает что приложение будет запущенно с правами пользователя, запустившим его.
rungroup	asterisk	Устанавливает системную группу под которой приложение Asterisk будет запущено. Эта опция не установлена по умолчанию.
lightbackground	yes	Когда используется цвет в консоле Asterisk, эта опция выводит цвета совместимые со светлым фоном. Эта опция, по умолчанию, установлена в no, в этом случае Asterisk использует цвета которые хорошо смотрятся на черном фоне.
forceblackbackground	yes	На терминалах со светлым фоном, вы можете заставить Asterisk, установить черный цвет фона для того, чтобы цвета в консоли, выглядели правильно.
defaultlanguage	en	Установить язык по умолчанию.
documentation_language	en_US	С приложением Asterisk идет встроенная документация, функции, и другие вещи включая внешние XML документы. Эта опция определяет предпочитаемый язык для документации. Если он недоступен, по умолчанию будет использоваться en_US.
hideconnect	yes	Установка этой опци в yes позволяет Asterisk не показывать уведомления от подключенных удаленных консолей и отключениях Asterisk CLI. Это полезно на системах где проблематично использовать скрипты с доступом к удаленным консолям. По умолчанию установлена в no.
lockconfdir	yes	Когда эта опция включена, каталог с конфигурационными файлами Asterisk будет защишен блокировкой. Это помогает защитить от одновременной записи в одни файлы несколькими приложениями. Значение по умолчанию no.
stdexten	gosub	При использовании *users.conf*, опция hasvoicemail=yes вызывает подпрограмму stdexten. В Asterisk 1.4, это будет сделано с помощью макроса, но в более поздних версиях Asterisk использует GoSub (предпочтительный метод для Asterisk 1.6.0). Эта опция используется для обратной совместимости.

[a] Если какие-то ключи требуют ввода пароля, это заблокирует запуск Asterisk. В качестве альтернативы можно запустить *key init* в Asterisk из командной строки.

[b] Это критично для отладки аварий. Однако, Asterisk должен быть скомпилирован с включенной опцией DONT_OPTIMIZE в *menuselect* для использования дампа ядра.

[c] Чтобы определить текущее значение, вы можете запустить:

```
$ sudo cat /proc/`pidof asterisk`/limits
```

Это покажет вам различные системные ограничения, включая максимальное количество открытых файлов. Хорошее значение для максимальных открытых файлов где-то вокруг «пиковых вызовов», умноженных на 5 (при условии, что 2 канала RTP и RTCP на канал, плюс накладные расходы). Так что если у вас будет 300 одновременные вызовы, вы должны получить не менее 1500 открытых файлов. Вы можете установить это в файле *asterisk.conf* с помощью параметра `maxfiles`, либо в файле */etc/security/limits.conf* для вашей операционной системы.

[d] Важно отметить, когда эта опция включена тишина генерируется в несжатом подписаном линейном формате (slin), это означает, что она должна быть перекодирована в формат вызывающего канала. В результате может оказаться, что для вызова потребуется транскодирование, которое фактически не требуется.

Раздел [files]

Этот раздел *asterisk.conf* включает параметры, связанные с сокетом управления Asterisk. Он в основном используется на удаленных консолях (*asterisk -r*). Доступные опции перечислены в Таблице 4-3.

Таблица 4-3. Раздел asterisk.conf [files]

Опция	Значение/Пример	Примечание
`astctlpermissions`	`0660`	Устанавливает права доступа для управляющего сокета Asterisk.
`astctlowner`	`root`	Устанавливает владельца для управляющего сокета Asterisk.
`astctlgroup`	`apache`	Устанавливает группу для управляющего сокета Asterisk.
`astctl`	*asterisk.ctl*	Устанавливает имя файла для управляющего сокета Asterisk. По умолчанию *asterisk.ctl*.

Раздел [compat]

Иногда команда разработчиков Asterisk решает, что лучший путь развития предполагает внесение изменений, которые не совместимы с предыдущей версией. Этот раздел содержит некоторые опции (перечисленные в Таблице 4-4), которые позволяют вернуть поведение некоторых модулей к предыдущей версии.

Таблица 4-4. Раздел asterisk.conf [compat]

Опция	Значение/Пример	Примечание
`pbx_realtime`	1.6	В версиях Asterisk до 1.6.x, модуль `pbx_realtime` автоматически конвертировал символы канала в запятые для аргументов приложений Asterisk. Это больше не делается по умолчанию. Для включения предыдущего поведения, установите эту опцию в 1.4.
`res_agi`	1.6	В версиях Asterisk до 1.6.x, команда *EXEC* AGI автоматически конвертировала символы канала в запятые для аргументов приложений Asterisk. Это больше не делается по умолчанию. Для включения предыдущего поведения, установите эту опцию в 1.4.
`app_set`	1.6	Начиная с версии Asterisk 1.6.x, приложение `Set()` позволяет устанавливать значения только одной переменной. Раньше `Set()` позволяла устанавливать значения нескольких переменных разделяя их &. Это было сделано, что бы позволить использовать любые символы в значении переменной, включая символ &, который ранее использовался как разделитель. `MSet()` - это новое приложение, которое подобно `Set()`. Однако, установка этой опции в 1.4 делает поведение `Set()` аналогичным `MSet()`.

modules.conf

Этот файл не является строго обязательным при установке Asterisk, однако, без модулей Asterisk не сможет делать ничего полезного, файл *modules.conf* должен находиться в каталоге */etc/asterisk*. Если вы просто укажите `autoload=yes` в файле *modules.conf*, Asterisk будет искать все модули в каталоге */usr/lib/asterisk/modules* и загружать их при запуске.

Хотя большинство модулей не являются ресурсоемкими, и все они загружаются очень быстро, разумным было бы загружать только те модули, которые вы планируете использовать в вашей системе. Кроме того, по соображениям безопасности, не стоит загружать все модули, которые поддерживают соединение по сети.

В прошлом мы чувствовали, что явная загрузка каждого нужного модуля была лучшим способом, но с тех пор мы обнаружили, что эта практика создает дополнительную работу. После каждого обновления нам приходилось редактировать файл *modules.conf*, для исправления различий в модулях между релизами, и каждый раз процесс становился сложнее. Сейчас мы предпочитаем разрешать в Asterisk автоматическую загрузку модулей, которые он находит, но те модули, которые мы не хотим загружать в Asterisk, мы указываем директивой `noload`. Пример файла *modules.conf* можно найти в разделе «modules.conf».

Использование menuselect для контроля какие модули компилировать и устанавливать

Другой путь контроля какие модули загружать в Asterisk это не компилировать и не устанавливать их изначально. В течении процесса установки Asterisk, выполнение команды *make menuselect* запускает интерфейс с меню, который позволяет вам определить много различных директив для компиляции, включая какие модули компилировать и устанавливать. Если вы никогда не будите компилировать и устанавливать модуль, то получите эффект от этого уже во время загрузки, поскольку модуль не будет существовать, и, следовательно, не будет загружен. Если вы новичок в Linux и в Asterisk, это может создать трудность для вас, если позже вы захотите использовать модуль и обнаружите, что он не существует в вашей системе.

Больше информации о menuselect доступно в разделе называемом “make menuselect”.

Секция [modules]

Файл *modules.conf* содержит один раздел. Параметры, доступные в этом разделе, перечислены в Таблице 4-5. За исключением `autoload`, все параметры могут быть указаны более одного раза.



Список всех загружаемых модулей доступен в Главе 2 с примечаниями о наших мнениях относительно популярности/статуса каждого из них.

Таблица 4-5 Раздел modules.conf [modules]

Параметр	Значение/Пример	Примечание
`autoload`	`yes`	Вместо того, чтобы явно перечислять, какие модули загружать, вы можете использовать эту директиву, чтобы сказать Asterisk загружать все модули, которые он находит в каталоге модулей. Исключение составляет список модулей с директивой `noload`, которые не будут загружены. Значение по умолчанию `yes` и мы рекомендуем это значение
`preload`	`res_odbc.so`	Указывает, что модуль должен быть загружен в самом начале порядка загрузки модулей. Эта директива является гораздо менее актуальной, чем это было раньше. Модули теперь имеют приоритет загрузки, встроенные в них. Это решает проблемы, которые ранее приходилось решать этой директивой.
`load`	`chan_sip.so`	Определяет модуль, который должен быть загружен. Эта директива актуальна только если `autoload` установлена в `no`.
`noload`	`chan_alsa.so`	Определяет модуль, который не должен загружаться. Эта директива актуальна только если `autoload` установлена в `yes`.
`require`	`chan_sip.so`	Аналогична директиве `load`; кроме того, Asterisk выйдет, если по какой-то причине этот модуль не загрузится.
`preload-require`	`res_odbc.so`	Аналогична директиве `preload`; кроме того, Asterisk выйдет, если по какой-

indications.conf

Звуки, которые люди ожидают от телефонной сети разные в разных странах. Различные страны и регионы используют различные звуки для таких событий, как гудок, сигнал занято, обратный звонок, окончание соединения и так далее.

В файле *indications.conf* определены параметры для различных звуков, которые телефонная система может производить, и вам позволено настроить их. В первые дни Asterisk в этом файле содержал звуки только для ограниченного числа стран, но на данный момент описание звуков исчерпывающее.

Чтобы присвоить характерный региональный тон для ваших каналов, вы просто можете присвоить тоновую зону используя функцию `CHANNEL()`, и она будет применяться на протяжении всего разговора (если не будет изменена позже):

```
Set(CHANNEL(tonezone)=[вашастрана]) ; напр. ua, ru, uk, de, и так далее.
```

Однако, поскольку сигналы из вызова могут исходить из разных мест (из несущей, из Asterisk, или даже от самого себя), следует отметить, что просто установка tonezone в вашем диалплане не гарантирует, что эти сигналы будут представлены во всех ситуациях.

Взлом indications.conf для развлечения и прибыли

Если у вас слишком много времени и шаловливые ручки, вы можете делать всякие бессмысленные, но занимательные вещи с вашей индикацией. Например, фанаты *Звездных Войн* могут сделать следующие изменения в конце своего файла *indications.conf*:

```
[starwars](us)
description = Star Wars Theme Song
ring = 262/400,392/500,0/100,349/400,330/400,294/400,524/400,392/500,0/100,
       349/400,330/400,294/400,524/400,392/500,0/100,349/400,330/400,349/400,
       294/500,0/2000
```

Если вы затем используете имя 'starwars' в ваших файлах настройки или диалплане, любой звонок принимаемый вами будет отличаться от стандартного звонка. Попробуйте следующий код в диалплане для проверки звучания вашего нового звонка:

```
exten => 500,1,Answer()
    same => n,Set(CHANNEL(tonezone)=starwars)
    same => n,Dial(SIP/0000FFFF0002) ; или как называется ваш канал в sip.conf
```



В зависимости от типа устройства, используемого для вызова в этом примере, вы можете задаться вопросом, будет ли оно работать. Например, SIP-телефоны обычно генерируют свои собственные тоны, а не Asterisk генерирует их. Этот пример был тщательно разработан, чтобы гарантировать, что Asterisk будет генерировать тональный сигнал вызова вызывающему абоненту. Ключ - это `Answer()`, который выполняется первым. Позже, когда будет исходящий вызов на другое устройство, единственный способ Asterisk передать звон индикацией звонка путем генерации unband audio, так как телефон абонента занят, ведь на этот вызов уже ответили.

Если вдруг Asterisk будет работать без файла *indications.conf*, настоятельно рекомендуется: скопировать образец через **`sudo cp ~/src/asterisk-complete/11/configs/indications.conf.sample`**, изменить параметр страны в разделе `[general]`, для соответствия вашему региону, и перезапустить Asterisk.

chan_dahdi игнорирует indications.conf

DAHDI не использует файл *indications.conf* из Asterisk, а скорее будет создавать тоны сам.

Для подробной информации смотри Главу 7.

Если ваша система поддерживает несколько стран (например, если у вас есть централизованная система Asterisk, и есть пользователи из разных регионов), вы просто не в состоянии определить страну по умолчанию. В этом случае, у вас есть несколько вариантов:

1. Определить страну в файле определения канала для пользователя.
2. Определить страну в диалплане используя функцию `CHANNEL(tonezone)`.

Для получения дополнительной информации об использовании Asterisk в различных странах, см. Главу 9.

musiconhold.conf

Если вы планируете продавать телефонные системы основанные на Asterisk и не измените значение по умолчанию музыки на удержание, которая поставляется с Asterisk, вы отправляете сообщение, громко и ясно, что вы действительно не знаете, что делаете.



Иногда это почти невозможно найти музыку на удержание, которая может удовлетворить требования ваших клиентов. Как альтернатива, вы можете просто играть звуковые сигналы вместо музыки. Сайт, который имеет бесплатные копии звуковых сигналов находится на http://www.university-music-on-hold.com/

Часть проблемы с музыкой на удержании (Music On Hold - MoH) является то, что в прошлом просто подключалось радио или CD-плеер в телефонную систему, правовая реальность такова, что большинство лицензий на музыку не позволяют этого сделать. Такая штука: если вы хотите, проигрывать музыку при удержании, то кто-то, где-то, как правило, хочет, чтобы вы платили им за эту привилегию.

Так как бороться с этим? Есть два правовых способа: 1) платить лицензионные отчисления за музыку на удержании владельцу авторских прав, или 2) найти музыку, которая выпущена под лицензией подходящей для Asterisk.

Мы здесь не для того, чтобы дать вам юридическую консультацию, Вы несете ответственность за понимание того, что требуется от вас, при использовании определенного фрагметра музыки как музыки на удержание. Мы только покажем вам, как получить музыку и как заставить её работать с Asterisk.

Получение свободной музыки:

Есть несколько сайтов, которые предлагают музыку, выпущенную под лицензией Creative Commons или другими лицензиями. В последнее время мы получили большое удовольствие от музыки Jamendo . Каждая песня может иметь свои собственные требования к лицензированию, и только потому, что вы можете скачать песни бесплатно не означает, что вы имеете право использовать их в качестве музыки на удержании. Всегда проверяйте условия лицензирования музыки, которую вы планируете использовать при удержании.

Конвертирование музыки в формат, который лучше работает с Asterisk

В наши дни иметь музыку в формате MP3 - это обычное дело. И хотя Asterisk может использовать MP3-файлы в качестве источника музыки, этот метод вовсе не идеален. MP3 файлы сильно сжаты, и для того, чтобы проиграть их, процессор должен выполнить серьезную работу, по распаковке их в режиме реального времени. Хорошо, когда вы играете только одну песню и хотите сэкономить место на вашем iPod, но для музыки на удержании, нужно, конвертировать MP3 в формат более легкий для ЦП.

RHEL необходимые условия

Поскольку RHEL не имеет установленной поддержки MP3 с *sox*, вы должны будете установить *mpg123* прежде чем вы сможете конвертировать MP3 файлы для использования с Asterisk.

Сначала вам необходимо установить репозиторий *rpmforge*. Чтобы узнать, какая версия вам нужна, откройте веб-браузер и перейдите к http://repoforge.org/use/, выделите текст для версии/архитектуры в которую вы хотите установить и вставьте его в свою консоль:

```
$ rpm -Uhv http://pkgs.repoforge.org/rpmforge-release/\
                                        rpmforge-release-....rpm
```

После добавления репозитория вы можете перейти к установке *mpg123*:

```
$ yum install mpg123
```

Как только это будет сделано, ваша система RHEL готова конвертировать файлы MP3 для использования с Asterisk.

Если вы знакомы с форматами файлов и имеете некоторый опыт работы с программами обработки аудио, такими как Audacity, вы можете конвертировать файлы на вашем компьютере и загружать их на Asterisk. Мы поступаем проще - загружаем исходные MP3 файлы на сервер Asterisk (скажем, в папку */tmp*), а затем конвертируем их в командной строке.

Чтобы конвертировать MP3 файлы в формат, который понимает Asterisk, вам необходимо выполнить команды изложенные здесь (в этом примере мы будем использовать файл с именем *SilentCity.mp3*).

RHEL

Сначала конвертируем MP3 файл в WAV файл:

```
$ mpg123 -w SilentCity.wav SilentCity.mp3
```

Затем, декодируем полученный WAV файл в частоту дискретизации, понимаемую Asterisk:

```
$ sox SilentCity.wav -t raw -r 8000 -s -w -c 1 SilentCity.sln
```



Если у вас ещё не установлен *sox*, вы можете это сделать выполнив `yum install sox`.

Ubuntu

Если у вас не установлены пакеты *sox* и `libsox-fmt-all`:

```
$ sudo apt-get install sox libsox-fmt-all
```

Затем, конвертируем ваш MP3 файл непосредственно в несжатый формат SLN:

```
$ sox SilentCity.mp3 -t raw -r 8000 -s -w -c 1 SilentCity
```



В новой версии *sox* (т.е., версии 14.3.0), опцию `-w` нужно заменить на `-2`.

Завершение преобразования файлов

Полученный файл будет существовать в каталоге */tmp* (или куда вы его загрузили) и нужно скопировать его в каталог */var/lib/asterisk/moh*:

```
$ cp *.sln /var/lib/asterisk/moh
```

Вам нужно перегрузить *musiconhold* в Asterisk чтобы он распознал ваши новые файлы:

```
$ asterisk -rx "module unload res_musiconhold.so"
$ asterisk -rx "module load res_musiconhold.so"
```

Чтобы проверить, что ваша музыка работает корректно, добавьте следующие строки в контекст `[UserServices]` в вашем диалплане:

```
exten => 664,1,NoOp()
    same => n,Progress()
    same => n,MusicOnHold()
```

Наберите `664` и случайным образом должен прозвучать один файл из вашего списка в каталоге *moh*.

Файл по-умолчанию musiconhold.conf

Давайте рассмотрим основы файла *musiconhold.conf*, который идет в файлах примеров с Asterisk. Мы немного почистили его, но теперь файл можно поместить в каталог */etc/asterisk/*, чтобы включить основную функциональность музыки на удержание.



Вам нужно установить несколько файлов с музыкой на удержание, которая включается по умолчанию в *menuselect*, прежде чем вы получите хоть какую музыку.

Это то, что по умолчанию содержит файл:

```
[general]

[default]
mode=files
directory=moh
```

В следующем разделе мы рассмотрим, какие еще опции доступны в файле *musiconhold.conf*.

Раздел [general]

Раздел `[general]` имеет одну опцию (как показано в Таблице 4-6) и она полезна только при загрузке музыки на удержание из realtime (см. Динамический Realtime в Главе 16 для получения информации о *extconfig.conf* и функции динамического realtime).

Таблица 4-6. Раздел musiconhold.conf [general]

Опция	Значение/Пример	Примечание
`cachertclasses`	`yes`	Эта опция используется, чтобы указать Asterisk кешировать realtime музыку на удержание в памяти, когда она загружена. По умолчанию значение `no`. Доступны значения `yes` или `no`.

Разделы класов музыки на удержание

Ниже раздела `[general]`, вы определяете классы музыки на удержании с именем между квадратными скобками; вот так `[jazz]`. Доступны следующие опции, перечисленные в Таблице 4-7.

Таблица 4-7. Раздел классов musiconhold.conf

Опция	Значение/Пример	Примечание
`mode`	`files`	Режим определяет, как класс музыки на удержании должен себя вести. Когда режим установлен в `files`, то файлы воспроизводятся из каталога в файловой системе (см. описание опции directory ниже). Если установлен режим `custom`, то приложение может использоваться для потоковой передачи музыки, а не играть её из файловой системы напрямую

Опция	Значение/Пример	Примечание

		(см.описание опции application ниже). Дополнительные значения:
		`quietmp3`
		Проигрывает MP3 файлы. Должно использовать модуль *format_mp3*, который может быть включен в Add-Ons меню в *menuselect*.
		`mp3`
		То же самое, но более громкая версия.
		`mp3nb`
		То же самое, но без буферизации.
		`quietmp3nb`
		Подобно `quietmp3`, но без буферизации

`directory`	`moh`	Имя каталога, который определяет где находятся файлы. Путь относительно */var/lib/asterisk/* (или путь определен в `astvarlibdir` в *asterisk.conf*).
`digit`	`#`	Если опция `digit` определена для класса MoH то, когда вызывающий слушает MoH и нажмет эту digit (цифру), класс музыки на удержание будет переключен на класс, который указывает эта цифра.
`announcement`	`queue-thankyou`	Если определено для класса музыки на удержании, то до воспроизводимой музыки, вызывающему будет представлен анонс (announcement). Он также будет представлен при переключении между музыкальными файлами. Наиболее распространенные сценарии использования этого, вероятно, с очередями. См. Воспроизведение объявлений между файлами музыки на удержание для получения дополнительной информации в Главе 13.
`sort`	`alpha`	Позволяет буквенно-цифровую сортировку воспроизводимых файлов. Если этот параметр не установлен, то порядок сортировки будет неопределенным.
`application`	`/usr/bin/stream player 172.16.0.100 888`	Когда используется сочетание `mode=custom`, вы можете определить приложение, которое будет поставлять звук в канал при вызове MoH.[a]
`format`	`ulaw`	При использовании аудиопотока из удаленного источника в сочетании с `mode=custom` и `application` можно определить формат, который может обрабатывать Asterisk.

[a] Это может быть довольно затратно, если приложение живет локально и выполняет перекодирование файла для каждого канала. Если это так, то лучше всего конвертировать файлы MoH в собственный формат, например, `ulaw` и использовать `mode=files`.

Дополнительные файлы конфигурации

На основе модулей, которые вы установили на этапе установки (те, которые вы выбрали в *menuselect*, скомпилировали и впоследствии установили), у вас будут дополнительные файлы конфигурации, которые вам нужно будет добавить в каталог */etc/asterisk*, чтобы получить чистый запуск. Чтобы определить, какие файлы могут отсутствовать, вы можете запустить Asterisk в активном режиме командой *asterisk -c*.

После того, как вы запустили Asterisk в активном режиме, вы увидите вывод, похожий на следующий:

```
[ Initializing Custom Configuration Options ]
Unable to load config file 'acl.conf'
Unable to open AMI configuration manager.conf, or configuration is invalid.
Asterisk management interface (AMI) disabled.
Unable to load config file 'udptl.conf'
Could not reload udptl config
Could not load features.conf
Could not find valid ccss.conf file. Using cc_max_requests default
Could not find valid ccss.conf file. Using cc_[state]_devstate defaults
130 modules will be loaded.
...Unable to load config res_stun_monitor.conf
Unable to load config smdi.conf: SMDI disabled
```

```
No SMDI interfaces are available to listen on, not starting SMDI listener.
```

Если мы посмотрим на наш результат, то увидим, что у нас отсутствуют следующие файлы конфигурации: *acl.conf*, *manager.conf*, *udptl.conf*, *features.conf*, *ccss.conf*, *res_stun_monitor.conf* и *smdi.conf*.

Чтобы начать работу с полным набором файлов конфигурации мы скопируем из каталога образцов базовый набор файлов, а затем вернемся и, при необходимости, изменим их. Как правило это самый быстрый способ получить полный набор конфигурационных данных, но без основной части использования чего-то вроде *make samples*. Например командой, которую мы могли бы запустить в bash, может быть:

```
$ cd ~ /src/asterisk-complete/asterisk/11
$ for f in acl manager udptl features ccss res_stun_monitor smdi; do
cp configs/$f.conf.sample /etc/asterisk/$f.conf;
done
```

Теперь мы можем запустить Asterisk и подтвердить, что мы начинаем с чистого листа.

Вывод

Эта глава помогла вам выполнить некоторые начальные настройки Asterisk. Отсюда вы можете перейти к настройке некоторых телефонов и воспользоваться многими функциями Asterisk, которые она может предложить.

ГЛАВА 5

Конфигурация пользовательских устройств

Я не всегда знаю, о чем говорю,
но я знаю, что я прав.
—Мухаммед Али

В этой главе мы рассмотрим пользовательские устройства, которые вы можете подключить к Asterisk; как правило это будут VoIP-телефоны. Настройка канала в Asterisk для подключения устройства относительно проста, но вам также необходимо настроить само устройство, чтобы оно знало, куда отправлять свои вызовы.[1] Другими словами, для настройки устройства необходимо выполнить две отдельные задачи с Asterisk: 1) сообщить Asterisk об устройстве и 2) сообщить устройству о Asterisk.

Некоторые мысли о протоколе SIP

Протокол инициации сеанса (SIP) является одноранговым протоколом, и, хотя обычно существует настройка, в которой конечные точки действуют как клиенты, а какой-то шлюз действует как сервер, сам протокол думает в терминах одноранговых протоколов, отношениями пир-к-пиру. Это означает, что SIP-телефон вполне способен осуществлять прямое подключение к другому SIP-телефону без промежуточной АТС.

Реальность такова, что большинство транзакций SIP происходят через какой-то сервер, а в случае с Asterisk, как правило, АТС оказывается посередине, перекрывая все соединения. Когда SIP-вызов выполняется с телефона на другой телефон через Asterisk, на самом деле происходит два вызова: звонок от исходного аппарата до Asterisk и другой отдельный вызов от Asterisk к целевому комплекту (этот второй этап вызова может даже использовать не SIP). Asterisk соединяет их вместе.

Использование SIP-телефона с Asterisk означает, что вы хотите настроить SIP-телефон для совершения всех своих вызовов через Asterisk, даже если устройство может напрямую подключиться к другой конечной точке SIP без сервера Asterisk. Телефон будет рассматривать Asterisk как его прокси-сервер (хотя Asterisk фактически является Back to Back User Agent или B2BUA), и будет искать Asterisk для принятия решений о маршрутизации для всех вызовов.

В то время как большинство устройств будут иметь веб-интерфейс для определения параметров, если вы создаете более одного или двух телефонов в продакшене, мы рекомендуем использовать процесс конфигурации на основе сервера, в котором в наборе должно быть указано только местоположение сервера конфигурации. Аппарат будет подключаться к серверу, идентифицировать себя и загружать настроенные файлы, которые определяют требуемые параметры для этого телефона (очень просто использовать MAC-адрес телефона в качестве идентификатора для именования каждого уникального конфигурационного файла). Например, файлы конфигурации могут быть в формате XML,

1 Это не имеет ничего общего с конфигурацией Asterisk, и каждый производитель оборудования будет иметь свои собственные инструменты, которые позволят вам настроить его устройства. Большинство SIP-телефонов имеют веб-интерфейс, и большинство софт-телефонов имеют меню конфигурации, встроенное в их графический интерфейс.

расположенном на сервере HTTPS. Точный процесс загрузки, протокол и синтаксис этих файлов будут отличаться от производителя к производителю (большинство производителей предлагают более одного способа создания файлов конфигурации). Существуют десятки разных производителей SIP-телефонов, каждый из которых имеет несколько иной способ обработки серверной конфигурации, и попытаться охватить все из них (и постоянно обновлять процессы) просто невозможно. Большинство производителей предлагают свободно загружаемые и подробные руководства по настройке своих телефонов, поэтому, если вы знакомы с настройкой Linux и проведете немного исследований, то найдете множество информации по этому вопросу в Интернете. По нашему опыту, документация, предоставляемая каждым производителем, превосходна и будет представлять самую последнюю информацию об инициализации своих устройств.

Это книга об Asterisk, и в этой главе мы сосредоточимся на конфигурации аппаратов с точки зрения Asterisk.

Концепции именования телефонов

Прежде чем мы начнем с настройки Asterisk для наших телефонов, мы предложим некоторые рекомендации по наименованию телефонов: абстрагирование понятий пользователей, внутренних номеров и телефонов.

В Asterisk вся система заботится о названии канала. В действительности нет понятия пользователя вообще[2], и внутренние номера - это не что иное, как триггеры, которые инициируют последовательность инструкций. Например, вы можете написать кусок диалплана, указав, что при запросе внутреннего номера 100 он должен позвонить телефону на моем столе. Тем не менее, номер 100 может так же легко вызвать окно голосовой почты компании, воспроизвести приглашение, присоединиться к конференц-залу или любое другое действие. Мы даже можем указать, что номер 100 должен звонить на устройство на моем столе с понедельника по пятницу между 9:00 и 17:00, но в остальное время на устройство на чужом столе. И наоборот, когда звонок совершается с устройства в рабочее время, идентификатор вызывающего абонента может показывать дневной номер, а оставшееся время может отображать номер нерабочего времени (многие стойки регистрации становятся столами безопасности ночью).

Внутренние номера Asterisk

Концепция внутренних номеров в Asterisk имеет решающее значение. В большинстве АТС внутр.номер - это номер, который вы набираете, чтобы позвонить по телефону или услуге. В Asterisk внутр.номер является именем группы инструкций в диалплане. Подумайте о внутр.номере Asterisk как имени сценария, и вы на правильном пути. Да, внутр.номер Asterisk может быть числом (например `100`), которое звонит на телефон, но так же легко может быть имя (например, `voicemail`), которое выполняет последовательность приложений диалплана.

Мы продолжим работу над внутр.номерами Asterisk более подробно в этой книге, но прежде чем мы это сделаем, мы хотим настроить несколько телефонов.

Абстракция между именем внутр.номера и тем что он делает является мощной концепцией в Asterisk, поскольку расширение 100 может выполнять множество функций в зависимости от любого числа переменных, которые запрограммированы в системе. Это особенно актуально в контексте таких функций, как hot-desking.

Hot-desking - это функция, которая позволяет кому-то войти в устройство и получать вызовы на этом устройстве. Допустим, у нас есть три агента по продажам, которые обычно работают за пределами офиса, но проводят несколько дней в офисе, чтобы оформить документы. Поскольку они вряд ли будут находиться на месте одновременно, вместо того, чтобы иметь отдельный телефон для каждого из этих трех агентов по продажам, они могут разделить один офисный телефон (или в более крупном

2 Фактически, Asterisk пытается реализовать и абстрагировать понятия пользователей и устройств внутри, используя файл *users.conf*; однако он обычно используется только графическим интерфейсом Asterisk. Абстрагирование понятий логически с использованием диалплана легче понять и гораздо более гибко.

масштабе, дюжина человек может делить пул, скажем, в три телефона). Этот сценарий иллюстрирует удобство (и необходимость), позволяющее системе отделить концепцию пользователя и внутреннего номера от физического телефона.

Итак, каковы примеры плохих имен для телефонных устройств? Что-то вроде имени человека, такого как `[SimonLeBon]`, было бы плохим именем для телефона, так как телефон также может использоваться Джоан Джетт и Рик Эстли. Те же рассуждения можно применить к тому, почему вы не хотите называть телефон на основе добавочного номера: имя телефона `[100]` было бы плохим выбором, поскольку вы можете захотеть повторно завести устройство для номера 160 в будущем или может использоваться несколькими людьми с разными номерами в решении с hot-desking. Использование числовых имен учетных записей также очень плохо с точки зрения безопасности и более подробно рассматривается в Главе 26.

Популярным способом именования телефонов является использование MAC-адреса устройства. Это уникальный идентификатор, характерный для телефона, который закреплен за ним, где он находится, и напрямую не связан с пользователем, работающим с телефоном, или с добавочным номером, который в настоящее время привязан к аппарату.

У некоторых корпораций есть наклейки, которые они размещают на своем оборудовании со штрих-кодом и другой информацией, которая позволяет им хранить запас готового оборудования; эти уникальные коды также будут приемлемым выбором для использования в именовании телефонов, поскольку они не обеспечивают никакого логического отношения к конкретному человеку, но предоставляют конкретную информацию о самих устройствах.

Выбор заключается в том, как вы хотите назвать свои телефоны, но мы в первую очередь хотим отвлечь любую концепцию телефона, принадлежащего человеку, или даже его местоположение в сети, поскольку эти концепции находятся за пределами Asterisk и могут измениться в любое время.

В этой книге вы увидите, что мы используем имена телефонов, которые выглядят как MAC-адреса (например, `0000FFFF0001` и `0000FFFF0002`), чтобы различать устройства. Вам понадобятся названия телефонов, которые соответствуют используемому оборудованию (или другая строка, которая уникальна для регистрируемого устройства).

В качестве окончательного утверждения мы должны четко указать что то, что мы предлагаем в отношении имен устройств, не является техническим требованием. Вы можете называть свои устройства как угодно, если ваши имена отвечают требованиям назначений имен Asterisk для устройств (оставаясь буквенно-цифровыми без пробелов, то все будет хорошо).

Телефоны, софтфоны и телефонные адаптеры

Существует три типа конечных точек, которые вы обычно предоставляете своим пользователям, которые могут служить в качестве телефона. Они в большинстве случаев называются телефонами, программными телефонами (софтвонами) и аналоговыми терминальными адаптерами (Analog Terminal Adapters - ATA).

Аппаратный телефон - это физическое устройство. Он похож на офисный телефон: у него есть телефонная трубка, кнопки с цифрами, экран и т. д. Он подключается непосредственно к сети, и это то, о чем люди говорят, когда говорят о VoIP-телефоне (или SIP-телефоне).

Программный телефон (софтфон) - это программное приложение, которое работает на ноутбуке, настольном компьютере, смартфоне или другом вычислительном устройстве. Звук должен проходить через звуковую систему устройства, поэтому вам нужна гарнитура, которая будет хорошо работать с приложениями телефонии. Совсем недавно приложения для софтфона были написаны для смартфонов, которые позволяют вам подключаться к сетям, отличным от сети сотовой связи. Интерфейс софтфона часто оформлен так, чтобы выглядеть как физический телефон, но это необязательно.

ATA предназначен для использования традиционными аналоговыми телефонами (и другими аналоговыми устройствами, такими как факсимильные аппараты, беспроводные телефоны, пейджинговые усилители и т.д.) для подключения к сети SIP[3], и обычно это коробка размером с ~~сэндвич~~ бутерброд, содержащая RJ11 разъем для телефона (обычно называемый внешней электронной станцией (Foreign eXchange Station) или FXS-портом), разъем RJ45 для сети и разъем питания. Некоторые ATA могут поддерживать более одного телефона. Другие ATA могут иметь расширенные функции, такие как брандмауэр или порт Foreign eXchange Office (FXO) (аналоговый порт, который может подключаться к схеме ТфОП - PSTN).

У телефонов есть преимущество в том, что, как правило, они имеют хорошие акустические свойства для голосовой связи. Любой телефон хорошего качества спроектирован для восприятия частот человеческого голоса, фильтрации нежелательных фоновых шумов и нормализации результирующего сигнала. Люди будут пользоваться телефонами до тех пор, пока существует телефонная сеть, и мы склонны любить то, что знакомо, поэтому наличие устройства, которое сочетается с Asterisk с использованием знакомого интерфейса, будет привлекательным для многих пользователей. Кроме того, для аппаратного телефона не требуется компьютер.

Недостатки аппаратных телефонов заключаются в том, что они не являются мобильными и как правило дороги по сравнению со многими качественными программными телефонами, имеющимися на рынке сегодня, которые доступны бесплатно. Кроме того, лишний беспорядок на вашем столе может быть нежелательным, если у вас ограниченное рабочее пространство, и если вы много передвигаетесь и, как правило, не находитесь в одном и том же месте, ваш телефон вряд ли подходит вашим потребностям (хотя бы по одному аппарату на каждое место, в котором вы часто бываете, может быть правильным решением).

Софтфоны решают проблему переносимости, установкой на устройство, которое, скорее всего, уже перемещается вместе с вами, например, ваш ноутбук или смартфон. Кроме того, их минимальная стоимость (как правило, бесплатно или около 30 долларов США за полнофункциональную) привлекательна. Поскольку многие программные телефоны бесплатны, вполне вероятно, что первым телефоном, который вы подключите к Asterisk, станет программный телефон. Кроме того, поскольку софтфоны - это просто программное обеспечение, их легко установить и обновить, и у них обычно есть иные функции, которые используют остальные периферийные устройства, такие как веб-камера для видеовызовов или возможность загружать файлы со своего рабочего стола для отправки факсов.

Некоторые из недостатков софтфонов - это не всегда включенное устройство, необходимость надевать гарнитуру каждый раз когда вы звоните, и тот факт, что многие ПК в течение дня будут заняты другими делами в зависимости от того, чем пользователь будет заниматься, что может привести к тому, что софтфон перестанет работать в то время как некоторые фоновые задачи загружают CPU.

Преимущество ATA заключается в том, что вы можете подключать аналоговые устройства[4] к вашей сети SIP, такие как беспроводные телефоны (которые во многих случаях превосходят более продвинутые типы радиотелефонов[5]), пейджинговые усилители и звонки. ATA также иногда можно использовать для подключения к старой проводке, где сетевое соединение может работать неправильно.

Основным недостатком ATA является то, что вы не получите те же функции через аналоговую линию, как на SIP-телефоне. Это технология, которой больше века.

В Asterisk нам необязательно делать выбор между софтфоном, аппаратным телефоном или ATA; вполне возможно и довольно просто иметь один добавочный номер, который одновременно вызывает

3 Или любая другая сеть, если на то пошло. ATA более формально можно назвать аналого-цифровыми шлюзами, где характер цифрового протокола может изменяться (например, патентованные ATA на традиционных АТС). Дело в том, что ATA не обязательно является устройством SIP.

4 ATA - это не единственный способ подключения аналоговых телефонов. Поставляемое оборудования, такое как карты Digium, которые входят в сервер Asterisk и предоставляют порты аналоговой телефонии.

5 Для действительно удивительного беспроводного аналогового телефона вы должны проверить устройства от EnGenius DuraFon, которые дороги, но впечатляют.

несколько устройств, такие как настольный телефон, софтфон на ноутбуке, мобильный телефон и, возможно, стробоскоп на задней стороне фабрики (где слишком много шума для прослушивания звонка).

Asterisk с радостью позволит вам взаимодействовать с внешним миром способами, о которых едва ли мечтали всего несколько лет назад. Поскольку мы видим больше унификаций коммуникационных приложений с популярными социальными сетями, такими сообществами, как Skype, и больше внимания к сетевым сервисам, таким какие предоставляет Google, гибкость и популярность программного обеспечения конечного оборудования будут продолжать расти. Размывание линий между голосом и приложениями постоянно развивается, а софтфоны имеют хорошие возможности для быстрого реагирования на эти изменения.

Тем не менее, нам все равно нравится настольный телефон.

Настройка Asterisk

В этом разделе мы расскажем, как создать файлы конфигурации *sip.conf* и *iax.conf* в каталоге */etc/asterisk*, которые используются для определения параметров, с помощью которых устройства SIP и IAX2 могут связываться с вашей системой.



Asterisk позволяет устройствам, использующим множество разных протоколов, общаться с ним (и, следовательно, друг с другом). Однако протоколы SIP и IAX2 являются наиболее популярными и зрелыми модулями VoIP, поэтому мы сосредоточим наше внимание на них. Для вашей первой сборки Asterisk вам, вероятно, лучше не беспокоиться о других протоколах (таких как Skinny/SCCP, Unistim, H.323 и MGCP). Если вы заинтересованы в одном из этих протоколов, сосредоточьтесь на том, чтобы сначала работать с SIP и IAX2. Конфигурация для других протоколов аналогична, и образцы конфигурационных файлов полны информации и примеров, поэтому, как только у вас есть основы, другие протоколы будут относительно просты в работе.

Файлы конфигурации канала, такие как *sip.conf* и *iax.conf*, содержат информацию о конфигурации, относящуюся к этому канальному драйверу (например, *chan_iax2.so* или *chan_sip.so*), а также параметры и учетные данные, характерные для телефонных устройств, которые вы хотите подключить к Asterisk через этот тип технологии.

Общая информация о драйвере канала содержится в верхней части файла конфигурации в разделе `[general]`. Все имена разделов заключены в квадратные скобки, включая имена устройств. Все, что следует за именем раздела (или определением устройства, которое для наших целей является, по сути, одним и тем же), применяется к этому разделу. Раздел `[general]` также может содержать информацию для определения значений по умолчанию для конфигурации устройств. Всё из раздела `[general]` можно переопределить в разделе, посвященном каждому устройству (или шаблону, назначенному этому разделу). Asterisk также поставляется с жесткими настройками по умолчанию, поэтому, хотя некоторые параметры являются обязательными, многие другие можно просто игнорировать если вы довольны настройками по умолчанию.



Asterisk назначит параметры, используя следующий рейтинг:

1. Конкретный раздел для соответствующего канала
2. Шаблон для раздела
3. Раздел `[general]`
4. Жестко заданные значения по умолчанию

Это означает, что даже если вы не указали значение для определенного параметра в разделе канала, ваш канал будет по-прежнему иметь настройку для этого параметра. Если вы не уверены, то всегда можете установить параметр явно в разделе файла конфигурации, который имеет дело с этим конкретным каналом (или в соответствующем шаблоне). Учитывая это сказанное, многие из параметров по умолчанию - это параметры, которые не

нужно менять, поэтому не беспокойтесь слишком о параметрах, которые не соответствуют вашим потребностям; значения по умолчанию должны быть точными.
Эта концепция должна иметь больше смысла, когда вы будете читать дальше.

Как файлы конфигурации работают с диалпланом

Хотя мы еще не обсуждали диалпланы Asterisk, полезно иметь возможность визуализировать связь между файлами конфигурации канала (*sip.conf*, *iax.conf*) и диалпланом (*extensions.conf*). Диалплан является сердцем системы Asterisk: он контролирует, как логика вызовов применяется к любому соединению с любого канала, например, что происходит, когда устройство набирает номер 101 или как проходит входящий вызов от внешнего провайдера. Как основной файл конфигурации канала, так и файл *extensions.conf* играет роль в большинстве вызовов, направляемых через систему. На Рисунке 5-1 отображено графическое представление отношений между файлами *sip.conf* и *extensions.conf*.[6]

Когда вызов поступает в Asterisk, идентификатор входящего вызова сопоставляется в файле конфигурации канала для используемого протокола (например, *sip.conf*). Файл конфигурации канала также обрабатывает аутентификацию и *определяет, где этот канал войдет в диалплан*.

Как только Asterisk определит, как обрабатывать канал, он передаст управление вызовом в правильный контекст в диалплане. Параметр `context` в файле конфигурации канала сообщает каналу, где он будет входить в диалплан (который содержит всю информацию о том, как обрабатывать и маршрутизировать вызов)

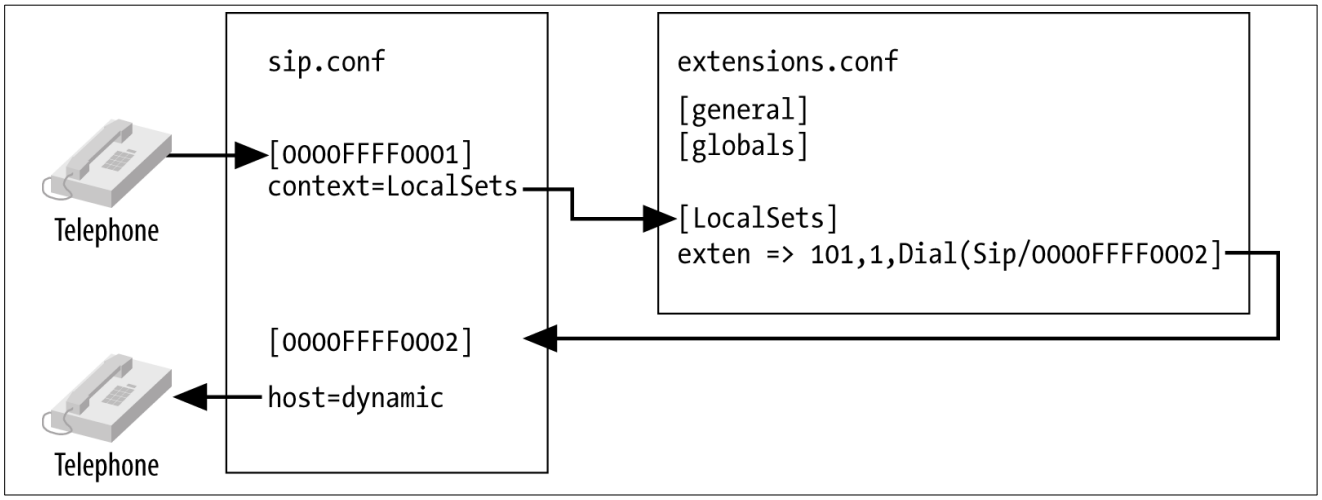


Рисунок 5-1. Связь sip.conf с extensions.conf

И наоборот, если диалплан запрограммирован для набора другого устройства (в приведенном выше примере это произойдет в том месте, где номер 101 запускает приложение `Dial()`), запрос на набор телефонного устройства `0000FFFF0002` будет использовать файл конфигурации канала чтобы определить, как передать вызов из диалплана на целевой телефон в сети (включая такие данные, как аутентификация, кодек и т.д.). Пункт назначения не обязательно должен быть телефоном. Это может быть любой тип канала, о котором знает Asterisk.

Ключевым моментом для запоминания является то, что файлы конфигурации канала управляют не только тем, как вызовы входят в систему, но и как они покидают систему. Например, если один из аппаратов вызывает другой, файл конфигурации канала используется не только для передачи вызова в диалплан, но и для направления вызова из диалплана в пункт назначения.

6 Концепция контекста диалплана подробно обсуждается в главах диалплана. Именно контекст определяет, где в диалплане будет обрабатываться вызов, поступающий на канал.

sip.conf

Модуль канала SIP,[7] возможно, является самым зрелым и многофункциональным из всех канальных модулей в Asterisk. Это связано с огромной популярностью протокола SIP, который занял сектор VoIP/ телеком индустрии и реализован в тысячах устройств и АТС. Если вы просмотрите файл *sip.conf.sample* в подкаталоге *./configs* исходников Asterisk, вы увидите множество доступных вариантов. К счастью, параметры по умолчанию, как правило содержат все, что вам нужно для начала работы, и поэтому вы можете создать очень простой файл конфигурации, который позволит большинству стандартных SIP-телефонов подключаться к Asterisk.

Первое, что вам нужно сделать, это создать файл конфигурации в каталоге */etc/asterisk* с именем *sip.conf*.

Вставьте или введите следующую информацию в файл:

```
[general]
context=unauthenticated ; контекст по умолчанию для входящих вызовов
allowguest=no           ; отключаем неаутентифицированные (гостевые) вызовы
srvlookup=no            ; отключить поиск DNS SRV на исходящих вызовах
                        ; (если у вас нет надежного подключения к DNS,
                        ; в ином случае - yes)
udpbindaddr=0.0.0.0     ; прослушивать UDP запросы на всех интерфейсах
tcpenable=no            ; отключить поддержку TCP

[office-phone](!)       ; создаем шаблон для ваших устройств
type=friend       ; драйвер канала будет искать сперва имя пользователя,
                  ; а затем IP
context=LocalSets ; здесь вызовы с устройства входят в диалплан
host=dynamic      ; устройства будут регистрироваться в Астериск
nat=force_rport,comedia ; предположим устройство находится за NAT
                  ; *** NAT - это преобразование сетевых адресов,
                  ; которое позволяет нескольким внутренним устройствам
                  ; совместно использовать внешний IP-адрес.
dtmfmode=auto     ; принимать тоновые сигналы от устройств, согласовывать
                  ; автоматически
disallow=all      ; установите какие голосовые кодеки это устройство примет
                  ; или предложит
allow=g722        ; аудио кодеки, которые примет или запросит устройство
allow=ulaw        ; в порядке приоритета
allow=alaw

; определим имя устройства и используем office-phone как шаблон
[0000FFFF0001](office-phone)
secret=4VQ96sg6ROc      ; уникальный пароль для этого устройства –
                        ; НЕ ИСПОЛЬЗУЙТЕ ПАРОЛЬ, УКАЗАННЫЙ В ПРИМЕРЕ!

; определим другое имя устройства с использованием того же шаблона
[0000FFFF0002](office-phone)
secret=sKAw7GCTtcA      ; уникальный пароль для этого устройства –
                        ; НЕ ИСПОЛЬЗУЙТЕ ПАРОЛЬ, УКАЗАННЫЙ В ПРИМЕРЕ!
```

Откройте файл *sip.conf*, который вы только что создали, и мы рассмотрим каждый элемент.

Мы создали четыре раздела, первый из которых - раздел `[general]`. Это стандартный раздел, который отображается в верхней части файла конфигурации для всех канальных модулей и всегда должен быть назван таким образом. Раздел `[general]` содержит общие параметры настройки того,

7 SIP RFC долго читается, но в первых 25 страницах - хорошее введение.

как этот протокол относится к вашей системе, и может также использоваться для определения параметров по умолчанию.

Например, мы определили контекст по умолчанию как `unauthenticated`, чтобы убедиться, что мы явно объявили, где неаутентифицированные гостевые звонки будут входить в диалплан (скорее иметь больше шансов). Мы назвали его `unauthenticated` чтобы было очевидно, что вызовам, обработанным в этом контексте, не доверяют и, следовательно, они не могут выполнять такие действия, как выполнение исходящих вызовов в PSTN (что потенциально может стоить денег или представлять кражу личных данных). Вы должны знать, что мы могли бы использовать любое имя, которое хотели бы, а также, чтобы в *extensions.conf* был указан идентичный контекст, чтобы определить поток вызовов для неаутентифицированных вызовов.

Следующий параметр - `allowguest`, который мы отключили, так как в настоящее время мы не хотим принимать какие-либо неаутентифицированные вызовы. Имейте в виду, что для некоторых каналов вы действительно можете принимать неаутентифицированные вызовы. Обычное использование разрешения неавторизованных вызовов для компаний, которые разрешают набор номера с помощью унифицированных идентификаторов ресурсов (URI), например адресов электронной почты. Если мы хотим разрешить клиентам звонить нам со своих телефонов без проверки подлинности, мы могли бы разрешить гостевые вызовы и обрабатывать их в контексте `unauthenticated`, для непрошедших проверку подлинности, определенным предыдущим вариантом.

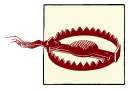


Вы можете задаться вопросом, почему вы, возможно, захотите разрешить несанкционированные звонки. Причина в том, что если вы публикуете свой SIP URI на своих визитных карточках (например, `sip:leif.madsen@shifteight.org`), вызовы этого URI будут неудачны, если ваш контекст для неаутентифицированных просто сбрасывает вызов. Вместо этого вы хотите, чтобы ваш неаутентифицированный контекст помещал входящие вызовы в контролируемую среду. Возможно, вы захотите разрешить вызовы, так как это позволит людям звонить вам, но вы не можете доверять этим вызовам (с точки зрения предоставления им доступа к системным ресурсам, таким как исходящие линии). Концепция здесь похожа на сетевую DMZ.

Вся концепция безопасности и доверия в сети VoIP - это то, что может стать довольно сложным. Спамеры уже усердно работают над этой технологией, и вам нужно знать концепцию. Мы рассмотрим это более подробно позже в книге, в Главах 7 и 26.

Параметр `srvlookup` используется для указания Asterisk выполнять поиск с помощью записи SRV DNS, которая обычно используется для исходящих подключений к поставщикам услуг. Если DNS недоступен для вашей системы, то он должен быть установлен на `no`[8]. Мы поговорим больше об Asterisk и DNS в Главе 12.

Параметр `udpbindaddr`[9] принимает значение IP-адреса или 0.0.0.0, чтобы сообщить Asterisk, какой сетевой интерфейс он должен прослушивать для запросов, переносимых сетевым транспортным протоколом UDP (который является протоколом фактически передающим голосовые каналы при SIP-вызове). Определив 0.0.0.0, мы инструктируем драйвер канала прослушивать все доступные интерфейсы. В качестве альтернативы, мы могли бы ограничить VoIP-соединения для этого протокола одним интерфейсом, указав IP-адрес определенного сетевого интерфейса в нашей системе.



В настоящее время в Asterisk параметры `udpbindaddr` и `tcpbindaddr` - это предложение всё или ничего. Другими словами, если у вас есть три сетевых адаптера в вашей системе, вы не можете ограничить VoIP-трафик двумя из них: это либо один, либо все.

IPv6 в sip.conf

8 Если у вас установлено значение `srvlookup - yes`, а ваше подключение к Интернету или DNS отключено, это может вызвать всевозможные проблемы с производительностью.

9 Дополнением к этому варианту является `tcpbindaddr`, используемый для прослушивания запросов, передаваемых через сетевой протокол TCP.

Asterisk поддерживает IPv6 для трафика SIP и RTP. Все параметры конфигурации в файле */etc/asterisk/sip.conf*, относящиеся к IP-адресам, могут принимать либо IPv4 либо IPv6-адрес. В качестве примера рассмотрим различные значения параметра udpbindaddr:

Значение udpbindaddr	Описание
`192.168.100.50`	Привязать к определенному адресу IPv4
`2001:db8::1`	Привязать к определенному адресу IPv6
`0.0.0.0`	Привязка ко всем адресам IPv4 в системе
`::`	Привязка ко всем адресам IPv4 и IPv6

Параметр `tcpenable` разрешает принимать запросы через сетевой протокол TCP. На данный момент мы отключили его, так как метод UDP в настоящее время более зрелый (и более популярный), и мы пытаемся устранить как можно больше барьеров. Тем не менее, не стесняйтесь тестировать поддержку TCP, как только вы освоите настройку своих устройств.



Существуют также параметры `tlsenable` и `tlsbindaddr` для включения SIP через Transport Layer Security (TLS или зашифрованного SIP). Мы рассмотрим конфигурацию SIP с TLS в Главе 7.

Следующий раздел, который мы определили, является шаблоном, который мы выбрали и назвали его `[office-phone](!)`. Мы создали его как шаблон, чтобы использовать значения внутри него для всех наших устройств.



Следущий за именем раздела `(!)` в файле конфигурации указывает Asterisk что этот раздел рассматривается как шаблон. Делая это, мы устраняем необходимость повторного добавления и изменения параметров конфигурации для каждого устройства, которое мы выбираем для определения. Шаблоны чрезвычайно полезны и доступны во всех конфигурационных файлах Asterisk. Если вы хотите что-то изменить для отдельного устройства, что ранее было определено в шаблоне для этого устройства, вы можете сделать это под заголовком раздела только для этого устройства и переопределить то, что было определено шаблоном. Нет необходимости использовать шаблоны, но они чрезвычайно полезны, и мы используем их широко.

В шаблоне `[office-phone](!)` мы определили несколько параметров, необходимых для проверки подлинности и управления вызовами устройств, использующих этот шаблон. Первый параметр, который мы настроили, - это `type`, который мы установили как `friend`. Это говорит о том, что драйвер канала пытается сначала совместить имя, а затем IP-адрес.

Согласование конфигурации SIP и опции type

В приведенном примере конфигурация для телефонов SIP устанавливается `type=friend`. Существуют два других типа определений, которые вы можете использовать: `user` и `peer`. Различия между ними связаны с тем, как Asterisk интерпретирует входящие запросы SIP. Правила приведены в этой таблице:

type=	Описание
`peer`	Сопоставьте входящие запросы с записью конфигурации, используя IP-адрес источника и номер порта.
`user`	Сопоставляет входящие запросы с записью конфигурации, используя имя пользователя в заголовке From запроса SIP. Это имя соответствует разделу в *sip.conf* с тем же именем в квадратных скобках.

`friend`	Это позволяет сопоставлять правила как для `peer`, так и для `user`. Эта настройка, наиболее часто используется для телефонов SIP.

Когда запрос от телефона получен и аутентифицирован Asterisk, запрошенный добавочный номер обрабатывается диалпланом в контексте, определенном в конфигурации устройства; в нашем случае - контекст с именем `LocalSets`. Другими словами, именно так мы определяем, как канал входит в диалплан.

Параметр `host` определяет IP-адрес конечной точки этого канала (необходимо, когда мы хотим отправить ему вызов). Если мы определим значение как `dynamic`, мы дадим Asterisk знать, что телефон сообщит нам, где он находится в сети, вместо того, чтобы определить его местоположение статически[10]. Если бы мы хотели определить адрес статически, то могли бы заменить `dynamic` IP-адресом, как например `192.168.128.30`. Имейте в виду, что если вы определяете статический адрес для канала SIP, соединения на этом канале больше не будут работать, если адрес устройства изменится. Преимущество определения статического IP-адреса заключается в том, что устройству не нужно регистрироваться, чтобы Asterisk знал, где он находится.

Строго говоря, параметр `nat` необходим только тогда, когда удаленное устройство (телефон) находится за брандмауэром с трансляцией сетевых адресов (NAT). Это важно, потому что протокол SIP включает в себя IP-адреса в сообщениях. Если телефон находится в частной сети, он может поместить в сообщения SIP частные адреса, которые часто бесполезны.



Более подробную информацию о настройке Asterisk для взаимодействия с устройствами за NAT и Asterisk, расположенными за NAT, находится в разделе «Как справиться с преобразованием сетевых адресов (NAT)» в Главе 7.

Опция `dtmfmode` используется для определения формата DTMF (тонального) Asterisk, который должен быть отправлен по телефону. Четыре варианта: `info`, `inband`, `rfc2833` и `auto`. Значение `info` означает использовать метод SIP INFO, `inband` - для внутриполосных аудио тонов, а `rfc2833` - для внеполосного метода, определенного этим RFC. Использование `auto` позволяет Asterisk автоматически определять, какой режим DTMF использовать (он предпочитает rfc2833, если тот доступен).

Последние два параметра, `disallow` и `allow` (*sip.conf*), используются для управления кодеками, какие принимаются и предлагаются на телефоне. Определив `disallow=all` вначале, мы сообщаем Asterisk об изменении любых ранее установленных определенных кодеках в разделе `[general]` (или внутренних значениях по умолчанию); далее мы явно объявляем, какие кодеки мы будем принимать (в порядке, который мы предпочитаем). В нашем примере мы включили как `ulaw`, так и `alaw`, причем наиболее предпочтительным является `ulaw` (если вы находитесь за пределами Канады или США, вы, скорее всего, захотите сначала объявить `alaw`).

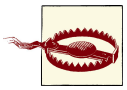
Теперь, когда мы закончили с нашим шаблоном, мы можем определить имена наших устройств и, используя шаблон `office-phone`, значительно упростить объем деталей, требуемый для каждого раздела устройства. Имя устройства определено в квадратных скобках, а шаблон, который будет применяться, определяется в круглых скобках, следующих за именем устройства. Мы можем добавить дополнительные опции к имени устройства, указав их ниже имени устройства:

```
[0000FFFF0003](office-phone)  ; шаблон должен быть на той же строке и не
                              ; содержать пробелов
secret=o3dbMtrRV6U            ; НЕ ИСПОЛЬЗУЙТЕ ПАРОЛЬ ИЗ ЭТОГО ПРИМЕРА!
allow=gsm
```

Пароль для устройства определяется параметром `secret`. Протокол SIP вполне позволяет разрешить практически любой пароль (в том числе никакой); однако, следует отметить, что довольно часто неприятные ребята запускают сценарии фишинга, которые ищут открытые учетные записи VoIP с

10 Этот процесс известен как регистрация. См. «Тестирование для обеспечения регистрации устройств» для получения дополнительной информации о концепциях регистрации устройств.

простыми именами устройств и небезопасными паролями (например, имя устройства `100` с паролем `1234`). Используя необычное имя устройства, такое как MAC-адрес, и пароль, который содержит строку, замедляющую атаку перебором, мы можем значительно снизить риск для нашей системы, если нам нужно будет открывать ее для внешнего мира.



Не используйте пароль, который мы определили в этом примере. Эта книга будет доступна в Интернете и может быть загружена кем угодно, и этот конкретный пароль почти наверняка станет одним из первых пропусков, добавленных в список, используемый скриптами фишинга VoIP, при атаках паролей с перебором по словарю. Если ваши SIP-порты открыты в Интернете и вы используете простые пароли, будьте уверены, что вас в конечном итоге выманят.



Вы можете создать безопасный пароль, используя один из нескольких генераторов паролей, доступных в Интернете и в вашей операционной системе. Вот простой скрипт, который вы можете запустить на консоли, чтобы создать такой:

```
$ dd if=/dev/random count=1 bs=8 2>/dev/null | base64 | sed -e 's/=*$//'
```

Когда модуль SIP считывает файл конфигурации *sip.conf*, он будет интерпретировать каждый раздел, включая шаблон. Поэтому предыдущий пример канала будет настроен так, как если бы он был записан таким образом:

```
[0000FFFF0003]
type=friend
context=LocalSets
host=dynamic
nat=force_rport,comedia
dtmfmode=auto
disallow=all
allow=g722
allow=ulaw
allow=alaw
; Вышеуказанные параметры взяты из шаблона
secret=o3dbMtrRV6U      ; Переопределяет пароль из шаблона
allow=gsm               ; Добавляем gsm в список разрешенных кодеков
;Остальные параметры были определены только для этого канала.
```

Обратите внимание: если вы укажете опцию, которая также была определена в шаблоне, модуль дважды увидит эту опцию. Для большинства параметров значение в секции канала будет переопределять значение, полученное из шаблона, однако для некоторых, например `type`, `allow` и `disallow` может быть иначе.

Фаерволл и преобразование сетевых адресов (NAT)

Телефонный вызов с использованием протокола SIP фактически состоит из трех отдельных сетевых подключений: 1) SIP-соединение (сигнализация для вызова), 2) исходящий аудиопоток (RTP), 3) входящий аудиопоток (RTP).

Если межсетевой экран между двумя концами соединения не распознает входящий поток RTP (т.е. не связывает его с SIP-соединением), то может его отбросить. Когда это произойдет, пользователи не смогут услышать одну из сторон телефонного разговора.

В «VoIP» в Главе 7 мы более подробно обсуждим эту концепцию.

Нахождение последовательного и надежного решения для этого - постоянная боль для сообщества SIP, учитывая множество различных типов брандмауэров и множество различных способов реализации протокола SIP (не только Asterisk, но и всех других SIP -блокируемых устройств и систем, используемых сегодня).

iax.conf

IAX2[11] - это протокол Inter-Asterisk eXchange, версия 2. IAX2 был разработан для упрощения процесса VoIP-вызовов через брандмауэры (путем передачи сигнализации и мультимедиа через одно и то же соединение) и для удобства прохождения сетей, в которых использовались устройства NAT (которые исторически были проблематичны для протокола SIP).

По мере развития Asterisk на протяжении многих лет, протокол IAX2 созрел. Информационный RFC (RFC 5456 - IAX: Inter-Asterisk eXchange Version 2) был опубликован в 2010 году. Однако IAX2 не стал популярным среди производителей аппаратного обеспечения, возможно, из-за относительной новизны IAX2 RFC, но, конечно, в значительной степени из-за того, что SIP является далеко идущим и самым признанным протоколом VoIP с точки зрения разумности (т.е. нетехнические люди чаще слышали о SIP, чем любом другом протоколе VoIP). Однако у IAX2 есть преимущества, которые стоит обсудить.

Одним из основных преимуществ IAX является однопортовое проникновение брандмауэра. Весь трафик, включая сигнализацию и аудиоданные, передается по одному порту UDP (по умолчанию, через порт 4569), что может значительно упростить настройку и устранение неполадок внешних подключений к Asterisk.[12]

Другим преимуществом IAX2 является возможность его транкинга, которая инкапсулирует пакеты для нескольких голосовых кадров в одну и ту же дейтаграмму с использованием одного заголовка IAX2. Преимущество этого заключается в уменьшении объема служебной информации пропускной способности, необходимой для отправки множества одновременных вызовов между двумя конечными точками. Объем пропускной способности, сохраненный с транкингом IAX2 при отправке только нескольких вызовов между локациями незначителен, но когда вы начинаете масштабирование до размеров десятков или сотен вызовов, экономия может быть существенной.

Пока нас интересует минимальная конфигурация, необходимая для того, чтобы наши конечные точки IAX2 говорили друг с другом, поэтому давайте рассмотрим, что нам нужно настроить в *iax.conf* для этого.

Так как IAX2 чаще используется для создания транков между системами Asterisk, мы будем приводить пример этого (а не пример использования IAX2 для пользовательских устройств). Если у вас есть пользовательское устройство IAX2 (например, софтфон), вы обнаружите, что знакомство с файлом *sip.conf* поможет вам настроить *iax.conf* для обработки внутренних номеров.

Во-первых, нам нужно создать наш файл *iax.conf*. Обратите внимание, что, поскольку в этом примере обсуждается IAX-транк, для его настройки вам понадобятся две системы Asterisk, каждая из которых имеет рабочий диалплан. На каждом конце вам понадобится соответствующий файл *iax.conf*. В каталоге конфигурации */etc/asterisk* создайте файл с именем *iax.conf* и добавьте следующую конфигурационную информацию:

```
[general]
; не останавливаться на длительное время, если другая конечная точка не
отвечает
autokill=yes

; отключить DNS SRV lookups для исходящих вызовов
srvlookup=no

; ----------------------------------------------------
```

11 Произносится «EEKS» (как в «weeks»). Не нужно говорить «eeks two», так как больше нет такой версии, как 1.

12 SIP, который имеет отдельные протоколы и порты передачи сигналов и голосовых данных, использует один порт для сигнализации (SIP-сообщения), но также требует, по меньшей мере, двух RTP-соединений для каждого активного вызова для передачи звука. По умолчанию Asterisk использует порт 5060 для SIP и порты от 10 000 до 20 000 для RTP. Протоколы SIP и RTP позволяют использовать разные порты, и, таким образом, эти вещи могут быть более четко определены в файле *rtp.conf*.

```
; шаблон для соединений офисов на основе IAX
[inter-office-trunk](!)

; Asterisk разрешает вызовы с и на этот телефон
type=friend

; контекст, с которого входящие запросы заходят в диалплан
context=LocalSets  ❶
delayreject=yes   ; задержка отклонения аутентификации(ограничит атаки
                  ; брутфорсом)
disallow=all      ; сбросить все доступные голосовые кодеки
allow=ulaw        ; предпочтительный кодек ulaw
allow=alaw        ; но также разрешить alaw

; ------------------------------------------------

;определить транк до нашего первого местонахождения
[head-office](inter-office-trunk)

; безопасный пароль — это значит НЕ ИСПОЛЬЗУЙТЕ ЭТОТ!
secret=NuFuYhg4iHI

; Определите IP-адрес системы Астериск на другом конце этого соединения
host=[ip-адрес дальнего конца]

; ------------------------------------------------

; определите другой транк используя этот шаблон
[branch-office](inter-office-trunk)

; безопасный пароль — это значит НЕ ИСПОЛЬЗУЙТЕ ЭТОТ!
secret=o8XBEf2DfQI

; IP-адрес системы Астериск на другом конце
host=[ip-адрес другого конца]
```

❶ В этом примере контекст LocalSets использовался для упрощения. В производственной среде вы, как правило, создаете отдельные контексты для работы с вашими транковыми устройствами. Это особенно важно для безопасности, так как вы всегда должны минимизировать риск несанкционированного вызова.

Перейдем к вариантам, которые мы добавили в этот файл, начиная сначала - с раздела `[general]`. Здесь мы определяем нашу конфигурацию по умолчанию, наши глобальные параметры и общую настройку драйвера канала. Здесь мы можем определить множество опций, и мы заставим вас проверить файл *iax.conf.sample* в каталоге *configs* ваших исходников Asterisk[13], но поскольку мы ищем простую конфигурацию, мы собираемся разрешить использование многих параметров по умолчанию.

Следующий раздел, который мы определили, называется `[inter-office-trunk](!)`, Который является шаблоном, содержащим параметры, общие для всех наших линий IAX.



Как упоминалось в предыдущем разделе, с пометкой `(!)` после имени раздела Asterisk будет рассматривать этот раздел как шаблон. Шаблоны полезны, поэтому используйте их(!).

13 Мы также рекомендуем вам прочитать раздел «sip.conf» в этой главе, поскольку он содержит много концепций, которые одинаково применимы к другим файлам конфигурации Asterisk.

Первый вариант, который мы настроили в нашем шаблоне, - это `type`. Мы определили `type` как `friend`, который сообщает Asterisk, что мы планируем как устанавливать вызовы с сетью, так и получать запросы (вызовы) из сети.



Остальные два типа являются `user` и `peer`. В IAX2 `friend` представляет собой комбинацию из обоих как `user`, так и `peer`, что является обычным явлением, поскольку межсетевые транки обычно имеют вызовы, идущие в обоих направлениях. Мы могли бы альтернативно определить два отдельных раздела с тем же именем с типами `user` и `peer`, а затем определить только необходимую информацию для каждого из этих типов; или если мы планируем только совершать вызовы или принимать вызовы с помощью раздела (например, в случае поставщика услуг входящего или исходящего доступа), мы будем определять его как `user` или `peer`. Однако, в подавляющем большинстве случаев, просто использование типа `friend` является наиболее логичным выбором.
Обратите внимание на разницу между значением параметра типа в *iax.conf* и *sip.conf* (см. врезку «Согласование конфигурации SIP и опции type»). В *iax.conf* смысл намного проще: он имеет отношение только к направлениям телефонных звонков.

Следуя опции `type`, мы установили для хоста `dynamic`, что означает, что телефон зарегистрируется у нас, чтобы определить его местонахождение в сети (чтобы мы знали, куда отправлять свои вызовы). В качестве альтернативы мы могли бы статически определить IP-адрес, например `192.168.128.50`, где все вызовы будут приниматься и совершаться. Это будет работать, только если устройство всегда будет иметь один и тот же IP-адрес.

Опция `context` определяет контекст, в котором этот канал будет входить в диалплан. Когда вызов совершается телефоном, и запрос получен Asterisk, он обрабатывается логикой, определенной в контексте, сконфигурированном здесь в диалплане (*extensions.conf*).

Параметр `delayreject=yes` указывает Asterisk на небольшую задержку на любые сообщения об отказе из-за неудачной аутентификации. Целью это является повышением уровня защиты от атак брутфорсом (которые в конфигурации по умолчанию могут запускаться по сотни попыток за считанные секунды).

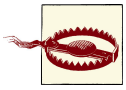
Ниже приведены параметры `disallow` и `allow`. Они определяют кодеки, которые будут разрешены для этого канала (в порядке предпочтения). Директива `disallow=all` сбрасывает любые кодеки по умолчанию, которые могут быть разрешены в разделе `[general]` (или как часть значений по умолчанию для канала). Затем мы определяем конкретные кодеки, которые хотим разрешить с помощью `allow`. В нашем примере мы явно разрешили только `ulaw` и `alaw`. Это рекомендуется для целей тестирования.[14]

Теперь, когда у нас есть наш шаблон, мы можем создать определенные описания каналов для каждой конечной точки, которую мы хотим поддерживать. Нам нужно указать только те параметры, которые не определены в шаблоне.

Мы назвали два канала, `head-office` и `branch-office`. Вы можете назвать свои каналы как пожелаете, однако, имейте в виду, что, поскольку мы используем `type=friend`, канал должен иметь одно и то же имя на обоих концах сети, и поэтому ему должно быть присвоено имя, которое имеет смысл с обеих сторон.

Мы назначили наш шаблон `inter-office-trunk` для обоих каналов, который автоматически назначит все параметры в шаблоне для этого канала.

Параметр `secret` определяет пароль.



Вы должны убедиться, что используете безопасный пароль, особенно если планируете открыть свою систему для внешнего мира. Не используйте что-то глупое, например `1234`, иначе пожалеете об этом. Нет, серьезно. Мы не шутим. Даже в лаборатории.

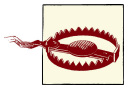
14 Если вы хотите использовать более сложные кодеки сжатия, мы рекомендуем вам попробовать их, но сначала запустите канал с основными кодеками, а затем поэкспериментируйте с другими кодеками позже.

Число успешных атак на телефонные системы с поддержкой VoIP (и не только Asterisk) растет, и будет продолжать увеличиваться. Обычно успешные взломы происходят из-за слабых паролей. Если у вас появится привычка использовать надежные пароли, у вас будет гораздо больше защиты в будущем.



Вы можете сгенерировать сложный пароль с использованием генератора паролей, как в Интернете, так и в вашей операционной системе. Вот простой скрипт, который можно запустить в командной строке Linux, чтобы создать строку, которая будет подходящей в качестве пароля:

```
$ dd if=/dev/random count=1 bs=8 2>/dev/null | base64 | sed -e 's/=*$//'
```



Не используйте пароли, которые мы определили в примере. Вы можете быть уверены, что вскоре после публикации этой книги это будет один из первых паролей, который бот попытается использовать в вашей системе.

Наконец, мы определяем IP-адрес дальнего конца этой сети. В этом случае мы решили жестко настроить IP-адрес, однако вы также можете определить хост как `dynamic`, что потребует регистрации дальнего конца у вас. И это для IAX.

Изменение файлов конфигурации канала для вашей среды

Наши примеры до сих пор были основаны на гипотетических именах устройств. Чтобы создавать фактические каналы на основе того, что у вас есть в вашей среде, вы захотите изменить имена устройств в файлах *sip.conf* и *iax.conf* на то, что имеет больше смысла.

Например, если у вас есть Polycom IP 430 с MAC-адресом `0004f2119698`, вам нужно определить идентификатор устройства в *sip.conf* для этого устройства:

```
[0004f2119698](office-phone)
```



В *sip.conf* вы также можете добавить значение `description` (описание), которое затем можно просмотреть в консоли Asterisk при запуске *sip show peers*. Например:

```
[0004f2119698](office-phone)
description=Polycom IP670 Phone
```

Имейте в виду, что, хотя вы можете описать такие вещи, как «Телефон в офисе Джона» или «Телефон в кабинете 213», реальность заключается в том, что все перемещается, поэтому наилучшей практикой было бы упростить описание, на то, что вряд ли изменится (например, модель телефона никогда не изменится). Некоторые компании назначают теги отслеживания активов для оборудования. Это может быть полезным, если отметить такие детали.
Вы также можете рассмотреть параметр `description` как часть вашего шаблона (например, `description=office-phone`), который позволит вам видеть в консоли Asterisk какой шаблон используется устройством.

Если у вас есть софтфон IAX на ПК, который вы хотите использовать, вашему файлу *iax.conf* может потребоваться что-то вроде этого:

```
[001b63a28ccc](office-phone)
secret=n4AgD7LURvg
```

Помните, что вы можете называть свои устройства как угодно (Asterisk об этом не заботится), но для простоты управления убедитесь, что вы выбрали для себя соглашение об именах, которое является логичным, масштабируемым, устойчивым и безопасным (под которым мы подразумеваем: не используйте дополнительные номера в качестве имен каналов).

Телефоны Digium с Астериск

Digium предлагает серию SIP-телефонов, предназначенных для работы с Asterisk. Несмотря на то, что это стандартные SIP-телефоны, добавление модуля DPMA (Digium Phone Module for Asterisk) позволяет этим устройствам работать более тесно с Asterisk чем любой другой модели SIP-телефона. DPMA также позволяет значительно упростить автоматическую настройку этих аппаратов.
Модуль DPMA - это проприетарная технология, обеспечивающая безопасное соединение между системой Asterisk и телефоном Digium. Преимущество DPMA заключается в том, что конфигурация пользовательских устройств значительно упрощена в сочетании с тем фактом, что комплекты с поддержкой DPMA могут быть более тесно интегрированы с функциями Asterisk. Конкретная интеграция Asterisk обеспеченная DPMA, включает:

- Упрощенный процесс провижинга. Телефоны автоматически обнаруживают серверы Asterisk и извлекают их конфигурацию.
- Телефонные контакты могут быть сохранены на сервере Asterisk.
- Интеграция со следующими функциями Asterisk: голосовая почта, каталог, парковка, записи вызовов, очереди вызовов и многое другое.
- Расширенная поддержка присутствия на основе пользователей.
- Возможность писать пользовательские приложения в Javascript, которые работают на телефоне. Эта возможность по-прежнему находится в стадии бета-тестирования на момент написания, но наверняка станет невероятно мощным дополнением.

Подробную документацию по настройке DPMA можно найти в вики Digium.

Загрузка конфигураций ваших новых каналов

Чтобы сообщить Asterisk о новых конфигурациях, вам необходимо передать ему команду, которая заставляет перезагрузить соответствующий файл конфигурации. В CLI Asterisk вы можете передавать различные команды в действующую систему Asterisk.

Asterisk CLI

Лучший способ увидеть, что происходит с вашей системой Asterisk, - это использовать CLI Asterisk. Этот интерфейс обеспечивает различные уровни вывода, чтобы знать, что происходит в системе, и предлагает множество полезных утилит, которые позволят повлиять на работу системы. Начните с вызова CLI Asterisk и перезагрузки файлов конфигурации для модулей канала:

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

Убедитесь, что ваши новые каналы загружены:

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



На этом этапе ваша система Asterisk должна быть сконфигурирована для обработки регистраций с определенных устройств. Вся регистрация - это определение местоположения устройства. Регистрация сама по себе не имеет никакого отношения к тому, разрешено ли комплекту совершение вызовов или нет (хотя успешная регистрация позволяет предположить, что комплект использует правильные учетные данные и, следовательно, также может успешно совершать вызовы).
Вызовы к и от не будут работать до тех пор, пока конфигурация устройств не будет завершена. Поскольку каждое устройство отличается в этом отношении, подробные

инструкции по настройке для каждой модели выходят за рамки этой книги. Большинство производителей устройств обеспечивают отличные инструкции по настройке и, во многих случаях, предлагают примеры, характерные для Asterisk.

Тестирование для обеспечения регистрации устройств

Как только ваше устройство зарегистрировалось на Asterisk, вы сможете запросить местоположение и состояние устройства из Asterisk CLI.



Распространенным заблуждением является то, что регистрация - это то, как устройство аутентифицируется для получения разрешения на совершение звонков. Это неверно. Единственная цель регистрации - позволить устройству идентифицировать свое местоположение в сети, так что Asterisk (или любой другой сервер регистратора SIP, если на то пошло) знает, куда отправлять вызовы, предназначенные для этого устройства.
Аутентификация исходящих вызовов является полностью отдельным процессом и всегда выполняется для каждого вызова независимо от того, зарегистрирован ли аппарат. Это означает, что ваш аппарат может выполнять вызовы, но не принимать их. Так обычно происходит, когда устройство не зарегистрировано успешно (поэтому Asterisk не знает, где оно), и все же имеет правильные учетные данные для совершения вызовов (поэтому Asterisk готов принять вызовы от него).

Чтобы проверить статус регистрации устройства, просто вызовите CLI Asterisk:

```
$ sudo asterisk -r
```

Ввод следующей команды возвращает список всех одноранговых узлов (peer’ов), о которых знает Asterisk (независимо от их состояния):

```
$ CLI> sip show peers
Name/username                Host                Dyn NAT ACL Port  Status
0000FFFF0001/0000FFFF0001    192.168.1.100        D    A     5060  Unmonitored
0000FFFF0002/0000FFFF0002    192.168.1.101        D    A     5060    Unmonitored
```



Вы можете заметить, что поле Name/username не всегда показывает полное имя устройства. Это связано с тем, что это поле ограничено 25 символами.

Обратите внимание, что в нашем примере в статусе `Unmonitored`. Это связано с тем, что мы не используем параметр `qualify=yes` в нашем файле *sip.conf*.

Аналоговые телефоны

Существует два популярных способа подключения аналоговых телефонов к Asterisk. Первый заключается в использовании ATA, который чаще всего подключается к Asterisk с использованием протокола SIP. Конфигурация Asterisk для ATA такая же, как и для любого другого телефона на базе SIP. Другой способ - напрямую подключить телефоны к серверу Asterisk с использованием аппаратного обеспечения телефонии от поставщика, такого как Digium. Digium продает телефонные карты, которые могут быть установлены на ваш сервер для предоставления портов FXS для подключения аналоговых телефонов (или факсимильных аппаратов). Для демонстрации мы собираемся показать требуемую конфигурацию, если у вас есть карта Digium AEX440E, которая представляет собой AX410 с половинной длиной PCI Express с четырьмя модулями FXS и аппаратным эхоподавлением.



Независимо от того, какое оборудование вы используете, ознакомтесь с документацией вашего поставщика о любых требованиях к конфигурации оборудования.

Во-первых, убедитесь, что установлены как Asterisk, так и DAHDI (см. «Как его устанавливать»). Обратите внимание, что DAHDI должен быть установлен перед установкой Asterisk.[15] Когда вы устанавливаете DAHDI, обязательно установите скрипт инициализации (*init*) (который в большинстве случаев устанавливается автоматически, в противном случае используйте *make config* из каталога tools). Это обеспечит правильную инициализацию вашего оборудования при загрузке системы. Скрипт *init* устанавливается из пакета *DAHDI-tools*.

Скрипт *init* использует файл */etc/dahdi/modules* для определения того, какие модули должны быть загружены для поддержки оборудования в системе. Установка этого скрипта попытается автоматически настроить этот файл для вас, но вы должны проверить и убедиться что он настроен верно:

```
# Автогенерация от tools/xpp/dahdi_genconf (Dahdi::Config::Gen::Modules) на
# Tue Jul 27 10:31:46 2010
# Если вы отредактируете этот файл и запустите tools/xpp/dahdi_genconf снова,
# ваши изменения вручную будут ПОТЕРЯНЫ.
wctdm24xxp
```

Для DAHDI требуется еще один файл конфигурации: */etc/dahdi/system.conf*. Он выглядит так:

```
; Укажите, что хотели чтобы DAHDI генерировал тоны, которые используются
; в Соединенных Штатах.
loadzone = us
defaultzone = us

; У нас есть 4 порта FXS; настроить их для сигнализации FXO.
fxoks = 1-4
```



Эта конфигурация предполагает что карта используется в Соединенных Штатах. Некоторые советы по интернационализации см. в Главе 9.

Если карта, которую вы настраиваете, не имеет аппаратной эхоподавления, в */etc/dahdi/system.conf* необходимо добавить еще одну строку, чтобы включить эхоподавление на основе программного обеспечения:

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



MG2 - это эхоподавитель, который поставляется с официальным пакетом DAHDI. Еще один эхоподавитель с открытым исходным кодом, совместимый с DAHDI - OSLEC (Open Source Line Echo Canceller). Большинство людей сообщают о превосходных результатах с OSLEC (он включен в ядро Linux). Дополнительную информацию об установке OSLEC в вашей системе см. на этом веб-сайте.

Теперь используйте скрипт *init* для загрузки соответствующих модулей и инициализации оборудования:

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

Теперь, когда настроен DAHDI, пришло время перейти к соответствующей конфигурации Asterisk. После установки Asterisk убедитесь, что модуль `chan_dahdi` установлен. Если он не загружен в

15 Обратитесь к Главе 3 за дополнительной информацией об переустановке Asterisk после установки DAHDI.

Asterisk, проверьте, существует ли он в */usr/lib/asterisk/modules*. Если он есть, отредактируйте */etc/asterisk/modules.conf*, чтобы загрузить *chan_dahdi.so*. Если модуль отсутствует на диске, DAHDI не был установлен перед установкой Asterisk; вернитесь и установите его сейчас (подробнее см. «DAHDI»). Вы можете проверить его наличие, используя следующую команду:

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

Затем вы должны настроить */etc/asterisk/chan_dahdi.conf*. Это файл конфигурации для модуля `chan_dahdi`, который является интерфейсом между Asterisk и DAHDI. Он должен выглядеть так:

```
[trunkgroups]

; В этой конфигурации нет групп транков.

[channels]

; Контекст channels используется при определении каналов с использованием
; старого устаревшего метода. Не используйте его как имя раздела.

[phone](!)
; Шаблон для размещения опций для всех телефонов.

usecallerid = yes
hidecallerid = no
callwaiting = no
threewaycalling = yes
transfer = yes
echocancel = yes
echotraining = yes
immediate = no
context = LocalSets
signalling = fxo_ks ; Использовать FXO сигнализацию для канала FXS

[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
```

Вы можете убедиться, что Asterisk загрузил вашу конфигурацию, запустив команду CLI *dahdi show channels*:

```
*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
```

Для получения подробной информации о конкретном канале вы можете запустить *dahdi show channel 1*.

Базовый диалплан для проверки ваших устройств

Мы пока не будем слишком глубоко погружаться в диалплан, но будет полезен начальный диалплан, который вы можете использовать для тестирования ваших недавно зарегистрированных устройств. Поместите следующее содержимое в */etc/asterisk/extensions.conf*:

```
[LocalSets]

exten => 101,1,Dial(SIP/0000FFFF0001) ; Замените 0000FFFF0001 на имя вашего
                                      ; устройства

exten => 102,1,Dial(SIP/0000FFFF0002) ; Замените 0000FFFF0002 на имя вашего
                                      ; устройства

exten => 103,1,Dial(SIP/0000FFFF0003) ; Замените 0000FFFF0003 на имя вашего
                                      ; устройства

exten => 104,1,Dial(IAX2/0000FFFF0004) ; Замените 0000FFFF0004 на имя вашего
                                       ; устройства

exten => 105,1,Dial(IAX2/0000FFFF0005) ; Замените 0000FFFF0005 на имя вашего
                                       ; устройства

;
; Это позволит Вам набирать каждый из 4 аналоговых телефонов настроенных
; в предыдущем разделе.
;
exten => 106,1,Dial(DAHDI/1)
exten => 107,1,Dial(DAHDI/2)
exten => 108,1,Dial(DAHDI/3)
exten => 109,1,Dial(DAHDI/4)

exten => 200,1,Answer()
    same => n,Playback(hello-world)
    same => n,Hangup()
exten => 3101,1,Dial(IAX2/head-office/101) ; Dial extension 101 at head office
exten => 3102,1,Dial(IAX2/head-office/102) ; Dial extension 102 at head office
exten => _3XXX,1,Dial(IAX2/head-office/${EXTEN})
```

Этот базовый диалплан позволит вам набирать ваши SIP-устройства с помощью добавочных номеров 100, 101 и 102. Устройства на основе IAX2 могут набираться на внутренних линиях 104 и 105. Четыре линии аналоговой карты можно набирать через номера от 106 до 109 соответственно, Вы также можете прослушать приглашение `hello-world`, которое было создано для этой книги, путем набора номера 200. Набор 3 плюс любой трехзначный номер попытается отправить этот вызов по вашему IAX-транку в головной офис. Все эти номера являются произвольными числами и могут быть любыми какие захотите. Кроме того, это далеко не полный диалплан; мы разработаем его далее в последующих главах.

Вам нужно будет перезагрузить диалплан, прежде чем изменения вступят в силу в Asterisk. Вы можете перезагрузить его из оболочки Linux:

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

или из Asterisk CLI:

```
*CLI> dialplan reload
```

Теперь вы можете набирать между двумя новыми внутр.номерами. Откройте CLI, чтобы просмотреть ход вызова. Вы должны увидеть что-то вроде этого (и аппарат, который вы вызываете, должен звонить):

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

Если этого не произойдет, вам нужно будет просмотреть свою конфигурацию и убедиться, что вы не сделали никаких опечаток.

Под капотом: Ваш первый звонок

Чтобы вы могли подумать о том, что происходит под капотом, мы кратко рассмотрим кое-что из того, что на самом деле происходит с протоколом SIP, когда два аппарата в одной системе Asterisk вызывают друг друга.



Asterisk как B2BUA

Имейте в виду, что здесь на самом деле два вызова: один от исходного аппарата до Asterisk, а другой - от Asterisk до целевого аппарата. SIP является одноранговым протоколом, и, с точки зрения протокола, происходит два вызова. Протокол SIP не знает, что Asterisk соединяет вызовы; каждый аппарат понимает его связь с Asterisk не имея реального знания об устройстве с другой стороны. Именно по этой причине Asterisk часто упоминается как B2BUA (Back to Back User Agent). Вот почему так легко объединить разные протоколы вместе с Asterisk.

Для вызова, который вы только что сделали, будут иметь место диалоги, показанные на Рисунке 5-2. Подробнее о том, как работает обмен сообщениями SIP, см. Приложение B и SIP RFC.

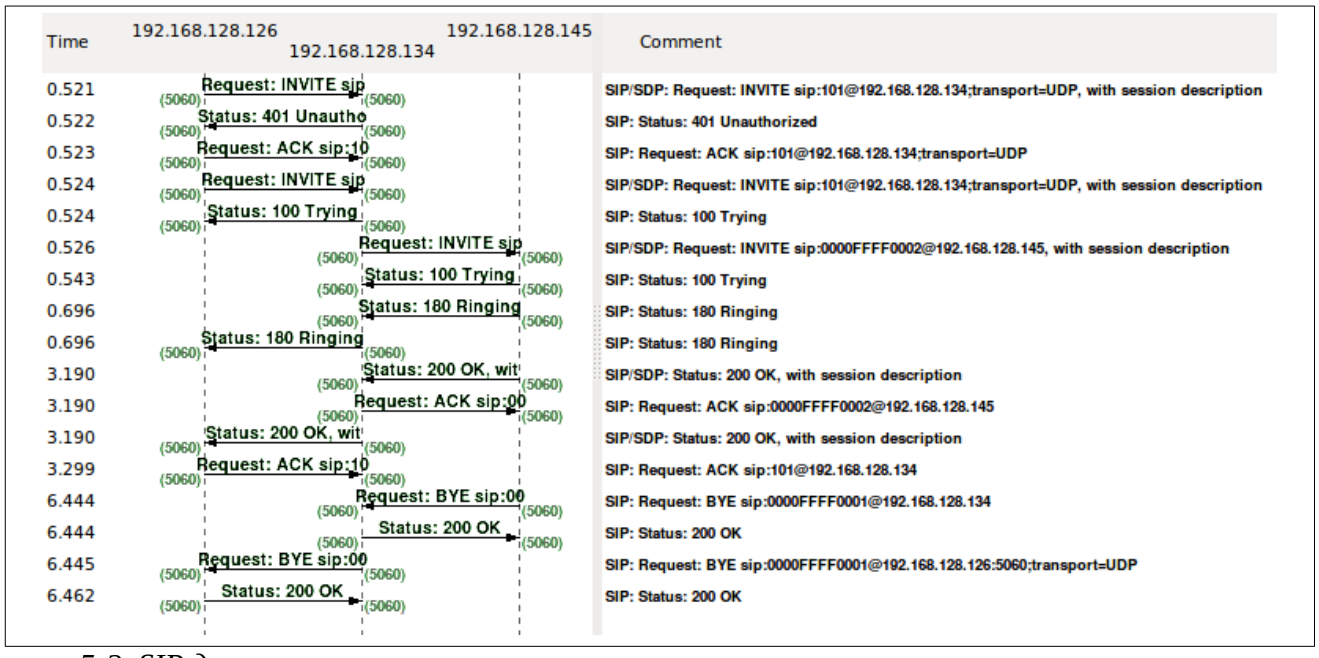


Рисунок 5-2. SIP диалоги

Вывод

В этой главе мы узнали о лучших методах для именования устройств, абстрагируя понятия пользователей, внутренних номеров и устройств, а также о том, как определять параметры

конфигурации устройства и аутентификации в файлах конфигурации канала. Затем мы рассмотрим магию Asterisk, которая называется диалпланом, и посмотрим, как простые вещи могут создавать отличные результаты.

Основы диалплана

Всё должно быть изложено так просто, как только возможно,
но не более того.

— Альберт Эйнштейн.

Диалплан — это сердце вашей системы Asterisk. Он определяет как обрабатываются вызовы в вашей системе. Диалплан, как скриптовый язык содержит инструкции, которым следует Астериск, реагируя на внешние триггеры. В отличии от традиционных телефонных систем диалплан Астериска может полностью перенастраиваться.

Эта глава расскажет самую суть концепции диалплана. Изложенная здесь информация является критичной для понимания вами кода диалплана и является основой для любого диалплана, который вы напишите сами. Примеры, описанные ниже, следуют связанно друг за другом, и поэтому мы не рекомендуем пропускать слишком большую часть этой главы, потому что это является фундаментально важным для понимания Астериск. Просим не рассматривать также данную главу как исчерпывающий обзор всех возможностей, которые можно реализовать посредством диалплана; наша цель состоит в том, чтобы охватить только самое необходимое. Мы рассмотрим более расширенные возможности диалплана в последующих главах. Вам предлагается поэкспериментировать.

Синтаксис диалплана

Диалплан Астериска задаётся в конфигурационном файле *extensions.conf*.



Файл *extensions.conf* обычно расположен в директории */etc/asterisk*, но его расположение может быть совершенно различным в зависимости от того как вы инсталлировали Астериск. Другие возможные места структуры Asterisk - */usr/local/etc/asterisk* и */opt/etc/asterisk*.

Диалплан состоит из четырёх основных концепций: контекст (context), расширения (extensions), приоритеты и приложения (applications). После объяснения роли каждого из этих четырёх элементов, работающих в диалплане, мы построим простой, но вполне функционирующий диалплан.

Примерный конфигурационный файл

Если вы инсталлировали примеры конфигурационных файлов, то скорее всего у вас уже существует файл *extensions.conf*. Вместо того, чтобы начинать разбираться с этим примером мы рекомендуем создать свой собственный *extensions.conf* с нуля. Начинать с изучения примерного файла конфигурации не самый хороший или лёгкий путь изучать построение диалплана.

Но хочется всё же заметить, что пример *extensions.conf* содержит фантастические ресурсы, он

полон примеров и идей, которые вы можете использовать после того, когда изучите основные концепции. Если вы следовали инструкциям при инсталляции, то найдёте файл *extensions.conf.sample* в каталоге *~/src/asterisk-complete/asterisk/11/configs* (там же, где много и других примеров конфигурации).

Контекст

Диалплан разбит на секции, называемые контекстами. Контексты содержат различные части диалплана, сообщающиеся между собой. Расширение, определяемое в одном контексте полностью изолировано от других расширений в других контекстах, кроме разрешённых специфических соединений. (Мы рассмотрим как сообщаются контексты между собой в конце этой главы. См. «Вложения (инклюды)» для информации).

Как простой пример, представьте, что у нас две компании используют один разделяемый сервер Asterisk. Если мы помещаем автосекретаря с приветствием[1] компании в соответствующий ему контекст, то они полностью отделены друг от друга. Это также позволит независимо указывать что произойдёт при нажатии, скажем, на 0: позвонившие в интерактивное меню компании А нажавшие 0 соединятся с секретарём компании А, в то время как позвонившие и нажавшие 0 в интерактивном меню компании Б соединятся с секретарём компании Б. Здесь мы подразумеваем, разумеется, что мы сообщили Asterisk делать трансфер на секретаря если позвонивший нажимает 0[2].)

Контекст определяется именем/названием, которое помещается в квадратные скобки (`[ ]`). Имя может содержать буквы от A до Z (большие и маленькие), цифры от 0 до 9, дефис и подчёркивание[3]. Контекст для входящих вызовов может быть таким:

```
[incoming]
```



Названия/имя контекста должно быть не более 79 символов (80 символов минус один, нуль).

Все инструкции, расположенные под названием контекста являются частью этого контекста, пока ниже этого не будет определено название следующего контекста. В начале диалплана определены два специальных контекста: `[general]` и `[globals]`. Первый - `[general]` содержит список общих установок диалплана (о которых, возможно, вам вообще не придётся беспокоиться), и мы позже обсудим контекст `[globals]` в разделе «Глобальные переменные». Сейчас просто важно знать про эти две метки не являются контекстами. Избегайте использования `[general]`, `[default]` и `[globals]` в качестве имен контекстов, но в противном случае называйте свои контексты как хотите.

Когда вы определяете канал (что не делается в файле *extensions.conf*, но для этого используются такие файлы, как *sip.conf*, *iax.conf*, *chan_dahdi.conf*, и тд), одним из обязательных параметров в каждом определении канала является `context`. *Контекст - это точка в диалплане, где будут начинаться соединения с этим каналом.* Настройка контекста для канала - это способ подключения канала к диалплану. На Рисунке 6-1 показана связь между файлами конфигурации канала и контекстами в диалплане.

1 Автоответчики рассматриваются в Главе 15.

2 Это очень важное соображение. По сравнению с традиционными УАТС, где, как правило, существует набор значений по умолчанию для таких отделов, как приёмная (что означает - если вы забудете определить эти значения, то они, вероятно, будут работать в любом случае). В Астериск верно обратное: если вы не укажете Астериск как обрабатывать каждую ситуацию и он столкнется с чем-то, что он не сможет обработать, вызов, как правило, будет отклонен. Мы рассмотрим некоторые лучшие примеры из практики, которые помогут убедиться, что этого не произойдет, позже. Дополнительные сведения см. в разделе «Обработка неверных записей и таймаутов».

3 Заметим, что пробелы не входят в список допустимых символов. Не используйте имена контекстов с пробелами — ибо результат этого вам совсем не понравится!

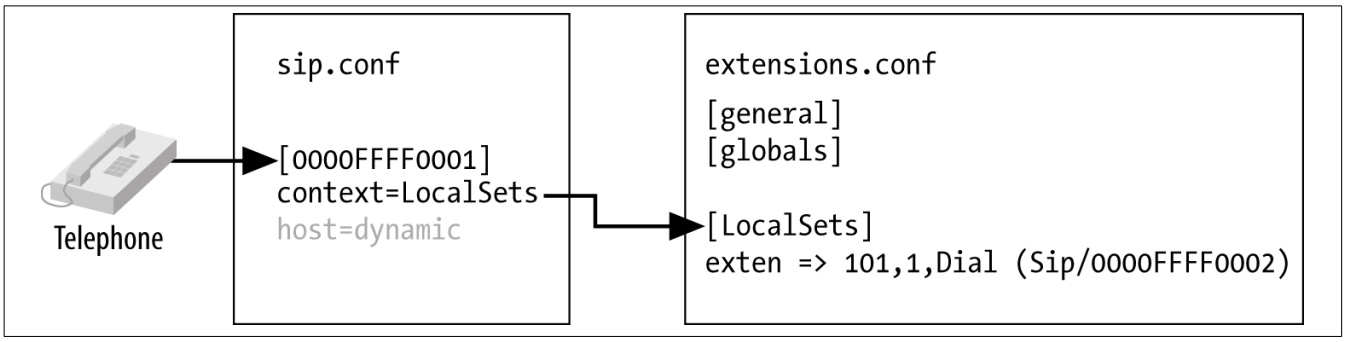
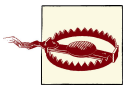


Рисунок 6-1. Связь между файлом конфигурации канала и контекстом в диал-плане



Это является одной из важнейших концепций для понимания как соотносятся между собой каналы и диал-план. Когда вы поймёте связь в определении контекста для канала, чтобы находить соответствие ему в диалплане, вы найдёте это лёгким и удобным для поиска проблем в прохождении вызовов через Asterisk-систему.

Наиболее важный смысл использования контекстов (а возможно и самый важный) — обеспечить безопасность. При правильном использовании контекстов вы можете дать определённым пользователям доступ к особым функциональным возможностям (таким как возможность совершать международные вызовы), которые недоступны простым пользователям. Если вы небрежно отнесётесь к построению своего диалплана, то можете несознательно дать возможность посторонним лицам использовать вашу систему в мошеннических целях. Пожалуйста, помните об этом при построении вашей Asterisk-системы; в Интернете уже существует множество специально написанных ботов для обнаружения и внедрения в небрежно настроенные Asterisk-системы.



Asterisk wiki содержит несколько шагов, которые необходимо предпринять для обеспечения безопасности Asterisk-системы. (Глава 26 этой книги также посвящена вопросам безопасности.) Жизненно важно прочесть и понять эту страницу. Если вы проигнорируете меры безопасности, изложенные там, вы можете в конечном итоге позволить кому-либо, и вообще всем, совершать междугородние или платные звонки за ваш счет!
Если Вы не воспринимаете безопасность вашей системы Asterisk всерьез, вы можете в конечном итоге дорого заплатить — в буквальном смысле. *Пожалуйста*, потратьте время и усилия, чтобы защитить вашу систему от мошенничества.

Расширения (Extensions)

В мире телекоммуникаций слово *extension* относится к внутреннему номеру абонента, набрав который услышим вызов, звонок телефона, (или системный ресурс типа голосовой почты или очереди). В Aterisk же понятие расширения (extension) намного мощнее, поскольку оно определяет уникальную серию шагов (каждый шаг содержит приложение), через которую Asterisk будет принимать этот вызов.

Внутри каждого контекста мы можем использовать множество (или несколько) расширений, сколько нам потребуется. Когда запрашивается обычное расширение (на входящем вызове или по набору цифр в канале), Asterisk выполняет последовательно шаги, указанные для этого расширения. Это расширение, таким образом, описывает то, что должно произойти с вызовом, который попадает по своему пути в соответствующий диалплан. Расширения могут, разумеется, использоваться как специфические телефонные внутренние номера в традиционном смысле (например — расширение 153 будет вызывать SIP-телефон на столе у Ивана), но в диалплане Asterisk они могут использоваться для гораздо большего.

Синтаксис процедур для расширений начинается со слова `exten`, затем следуют символы — знаки равно и больше, в виде стрелки вправо, как в примере ниже:

```
exten =>
```

Далее следует имя (или номер) расширения. Если набирается в/из традиционных телефонных систем, мы предполагаем обработку вызываемого расширения как номера, который набирается чтобы послать вызов на другой телефонный аппарат. В Asterisk вы получаете несколько больше возможностей; например, имя в качестве расширения, которое может содержать как буквы, так и цифры. В рамках краткого курса в этой главе мы рассмотрим как цифровые, так и буквенно-цифровые расширения.



Назначение имён для расширений может быть выглядит слишком революционной концепцией, но если вы реализуете поддержку нескольки видов транспортных VoIP протоколов (даже активно используемых) набор имени или адреса электронной почты вместо цифр даст великолепное ощущение. Только одно это свойство уже делает Астериск гибким и мощным.

Каждый шаг в расширении имеет три компонента:

- Имя (или номер) расширения
- Приоритет (каждое расширение может содержать несколько шагов; номер шага и называется «приоритет»)
- Приложение (или команда), которое выполняется на этом шаге

Эти три компонента разделяются запятыми, как например показано ниже:

```
exten => name,priority,application()
```

Вот самый простейший пример как обрабатывается реальный внутренний номер:

```
exten => 123,1,Answer()
```

В этом примере имя расширения (или внутренний номер) `123`, приоритет `1`, и приложение `Answer()`.

Приоритеты

Каждое расширение может иметь несколько шагов, называемых *приоритетами*. Приоритеты нумеруются последовательно, начиная с 1, и каждый шаг выполняет одно специфическое действие. Например, следующее ниже расширение выполнит ответ на звонок (поднимет трубку на приоритете 1) и затем положит трубку (на приоритете 2):

```
exten => 123,1,Answer()
exten => 123,2,Hangup()
```

Довольно очевидно, что этот код не делает ничего полезного. Мы ещё доберемся до него. Ключевым моментом здесь является то, что для конкретного расширения Asterisk следует приоритетам по порядку. Этот стиль синтаксиса диалплана все еще виден время от времени, хотя (как вы увидите скоро) он обычно больше не используется в новом коде:

```
exten => 123,1,Answer()
exten => 123,2,делаем что-то
exten => 123,3,делаем что-то ещё
exten => 123,4,сделаем ещё одну вещь
exten => 123,5,Hangup()
```

Ненумерованные приоритеты

В старых релизах Asterisk нумерация приоритетов вызывала множество проблем. Представьте себе расширение, которое имеет 15 приоритетов, а затем нужно добавить что-то на шаге 2: все последующие приоритеты должны быть перенумерованы вручную. Asterisk не обрабатывает пропущенные шаги или неправильно пронумерованные приоритеты и отладка этих типов ошибок была удручающей.

Начиная с версии 1.2, разработчики Asterisk решили эту проблему: ввели использование приоритета `n`, который означает "next" (следующий). Каждый раз, когда Астериск встречает приоритет с именем `n`, он берет номер предыдущего приоритета и добавляет 1. Это упрощает внесение изменений в диалплан, так как вам не нужно изменять нумерацию всех шагов. Например, ваш диалплан может выглядеть примерно так:

```
exten => 123,1,Answer()
exten => 123,n,делаем что-то
exten => 123,n,делаем что-то ещё
exten => 123,n,сделаем ещё одну вещь
exten => 123,n,Hangup()
```

Внутри себя Asterisk будет вычислять следующий номер приоритета каждый раз, когда он встречает `n`[4]. Имейте в виду, что вы всегда должны указать вначале приоритет номер 1. Если вы случайно поставили `n` вместо `1` для первого приоритета (распространенная ошибка даже среди опытных кодеров диалплана), то после перезагрузки диалплана вы обнаружите, что расширение не будет существовать.

Оператор same =>

В нескончаемых усилиях по упрощению кодинга была придумана новая конструкция, чтобы сделать создание расширений и управление ими еще проще. До тех пор, пока расширение остается тем же, вместо того, чтобы вводить полное расширение в каждой строке, вы можете просто ввести `same =>`, указывая далее приоритет и приложение:

```
exten => 123,1,Answer()
   same => n,делаем что-то
   same => n,делаем что-то ещё
   same => n,сделаем ещё одну вещь
   same => n,Hangup()
```

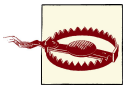
Отступ не требуется, но это может облегчить чтение. Этот стиль диалплана также облегчит копирование кода из одного расширения в другое. Мы предпочитаем этот стиль сами, и очень рекомендуем его.

Метки приоритетов

Метки на приоритетах дают возможность назначить имя к определённому номеру приоритета для расширения. Таким образом вы можете перейти к определённому приоритету другим образом, чем просто его номер (который, как правило, неизвестен, учитывая что диалпланы теперь обычно используют ненумерованные приоритеты). Причина, по которой важно обратиться к определенному приоритету в расширении, состоит в том, что вы захотите передавать вызовы из других частей диалплана определенному приоритету в конкретном расширении. Мы поговорим об этом позже. Чтобы назначить текстовую метку приоритету, просто добавьте метку в круглых скобках после приоритета, как здесь:

```
exten => 123,n(label),application()
```

Позже мы рассмотрим, как переключаться между различными приоритетами на основе логики диалплана. Вы увидите гораздо больше возможностей для меток приоритетов, и будете использовать их часто в своих диалпланах.



Наиболее распространённая ошибка при написании диалплана с метками — это указание метки после запятой, между n и скобкой (, как здесь:

```
exten => 123,n,(label),application() ;<-- ЭТО НЕ БУДЕТ РАБОТАТЬ
```

4 Астериск допускает простую арифметику в пределах приоритета, например `n+200` и приоритет `s` (для same), но их использование несколько устарело из-за существования меток приоритета. Обратите внимание, что *расширение* `s` и *приоритет* `s` - это два разных понятия.

Эта ошибка разрушит часть вашего диалплана и вы получите сообщение об ошибке, что приложение не найдено.

Приложения

Приложения — это рабочие лошадки диалплана. Каждое приложение выполняет определённое действие в текущем канале, типа — воспроизвести звуки, принять набор сигналов DTMF, найти что-то в базе данных, сделать вызов в канал, повесить трубку или что-то иное. В предыдущем примере мы показали два простых приложения: `Answer()` и `Hangup()`. Вы узнаете больше о том как это моментально работает.

Некоторые приложения, включая `Answer()` и `Hangup()` не требуют дополнительных инструкций для выполнения своей задачи. Но большинству приложений, однако, требуется дополнительная информация. Эти дополнительные элементы или *аргументы* передаются в приложения для того, чтобы они смогли произвести определённые действия. Чтобы передать аргументы приложению, поместите их в скобки после приложения, разделяя запятыми, если их несколько.



Иногда вместо запятой в качестве разделителя аргументов может использоваться символ пайп (|). Начиная с Asterisk 1.6.0 поддержка пайпа в качестве разделителя была удалена (кроме некоторых разделов в *voicemail.conf*).

Приложения Answer(), Playback() и Hangup()

Приложение `Answer()` используется для ответа на вызывающий канал. Оно производит начальную настройку на канале, который получает входящий вызов. Как мы упоминали ранее, `Answer()` не принимает аргументов. `Answer()` не всегда вообще требуется (фактически, в некоторых случаях он вообще нежелателен), но в остальных случаях он вполне годен, чтобы подключить канал прежде чем начать последующие действия.

Приложение Progress()

Иногда полезно иметь возможность передавать информацию обратно в сеть перед ответом на вызов. Приложение `Progress()` пытается предоставить исходному каналу информацию о ходе выполнения вызова. Некоторые операторы ожидают этого и таким образом Вы можете быть в состоянии решить странные проблемы сигнализации, вставив `Progress()` в диалплан куда поступают ваши входящие вызовы. С точки зрения биллинга, использование `Progress()` позволяет оператору знать, что вы обрабатываете вызов без запуска тарификации.

Приложение `Playback()` используется для воспроизведения предварительно записанных звуковых файлов. При этом аудиовход от пользователя игнорируется, подразумевается что вы не хотите использовать `Playback()` в автоответчиках (автосекретаре), например — просто не хотите принимать аудио данные в этой точке[5].



Астериск поставляется с большим количеством профессионально записанных звуковых файлов, которые находятся в каталоге звуков по умолчанию (обычно */var/lib/asterisk/sounds*). При компиляции Asterisk можно выбрать установку различных наборов звуковых файлов, записанных на различных языках и в различных форматах. Мы будем использовать эти файлы во многих наших примерах. Некоторые из файлов в наших примерах взяты из дополнительного звукового пакета, поэтому, пожалуйста, найдите время, чтобы установить его (см. Главу 3). Вы также можете иметь свои собственные звуковые подсказки, записанные в тех же голосах, что и стоковые подсказки, посетив http://www.theivrvoice.com. Далее в книге мы поговорим о том, как использовать телефонный аппарат и диалплан для создания собственных системных записей и управления ими.

5 Это будет другое приложение называемое `Background()`, которое почти идентично `Playback()` кроме того, что разрешает входящий аудиопоток от абонента. Вы можете более подробно прочесть об этом приложении в Главах 15 и 17.

Чтобы использовать `Playback()` укажите имя файла (без расширения) в качестве аргумента. Например, `Playback(filename)` воспроизведёт файл *filename.wav*, предполагая, что он расположен в дефолтной для звуковых файлов директории. Заметим, что можно также включать полный путь к файлу, если необходимо, как указано ниже:

```
Playback(/home/john/sounds/filename)
```

В приведённом примере мы воспроизвели *filename.wav* из директории */home/john/sounds*.

Можно также использовать относительные пути к именам файлов относительно к основной директории звуковых файлов, как например ниже:

```
Playback(custom/filename)
```

В этом примере будет проигран файл *filename.wav* из поддиректории */custom* в директории где обычно хранятся звуковые файлы (то есть */var/lib/asterisk/sounds/custom/filename.wav*). Имейте ввиду, что если указанная директория содержит более, чем один файл с одинаковым именем, но разными расширениями, Asterisk автоматически воспроизведет лучший по качеству файл[6].

Приложение `Hangup()` делает именно то, что отражено в его названии: закрывает активный канал. Необходимо использовать его в конце контекста, если хотите завершить текущий канал, чтобы быть уверенным, что действия, которые, возможно, и не предполагались, в активном канале не продолжатся. Приложение `Hangup()` не требует аргументов, но, если хотите, в него можно передать ISDN код завершения, например `Hangup(16)`.

В процессе работы над книгой мы познакомим вас со многими приложениями Астериск.

Простой диал-план

OK, хватит теории. Откройте в редакторе файл */etc/asterisk/extensions.conf*, и посмотрите на ваш первый диалплан (который был создан в предыдущей Главе 5). Мы будем дополнять его.

Hello World

Наиболее типичным примером для многих книг по технологиям (точнее — книг по программированию) является первый пример под названием «Привет, мир!» (“Hello World!”).

Первым приоритетом нашего расширения мы отвечаем на вызов. Затем, на втором шаге мы воспроизводим файл с названием *hello-world* и затем, третьим шагом мы просто кладём трубку завершая вызов. Интересующий нас код для этого примера выглядит вот так:

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

Если вы прошли Главу 5 вы уже получили сконфигурированный один или два канала, а также пример диалплана, содержащий наш код. Если нет, то всё, что вам нужно — просто добавить в файл *extensions.conf* в директории */etc/asterisk* следующий код:

```
[LocalSets] ; это название контекста
exten => 100,1,Dial(SIP/0000FFFF0001) ; замените 0000FFFF0001 на имя вашего
                                      ; устройства
exten => 101,1,Dial(SIP/0000FFFF0002) ; замените 0000FFFF0002 на имя вашего
                                      ; устройства
```

6 Asterisk выбирает лучший файл с точки зрения издержек трансляции кодеков (транскодинга) — таким образом затрачиваются наименьшие ресурсы по загрузке ЦПУ при конвертации в родной аудио-формат. Когда запускается Asterisk, вычисляются издержки трансляции между различными форматами аудио (они часто различаются в различных системах). Вы можете увидеть это набрав в командной строке своего Asterisk *core show translation*. Цифры в таблице показывают скорость перекодирования (в миллисекундах) односекундного аудио. Мы поговорим больше о различных аудио форматах (более известных как *кодеки*) в разделе "Кодеки" в Приложении B.

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



Если у вас ещё нет сконфигурированных к этому моменту каналов, то сейчас самое время сделать это. Существует реальное удовлетворение, которое приходит от вашего первого вызова в диалплан в Asterisk-системе, которую вы построили с нуля. Люди обычно ощущают забавную ухмылку на своих лицах, поскольку они понимают, что только что создали телефонную систему. Это удовольствие может быть получено и вами, поэтому, пожалуйста, не двигайтесь дальше, пока не сделаете это небольшое задание по диалплану. Если при этом вы столкнётесь с какими-то проблемами, вернитесь к Главе 5 и проработайте примеры оттуда.

Если у вас ещё нет этого кода в диалплане, пора его вставить и применить через перезагрузку диал-плана командой CLI:

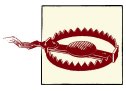
```
*CLI> dialplan reload
```

или из командной оболочки:

```
$ sudo /usr/sbin/asterisk -rx "dialplan reload"
```

Вызов расширения 200 с преднастроенного телефона даст возможность услышать голос Элиссон Смит, которая скажет “Hello, world!”

Если это не работает, смотрите в консоль Астериска на предмет сообщений об ошибках, и убедитесь в том, что ваш канал привязан к контексту `LocalSets`.



Мы не рекомендуем двигаться дальше по этой главе, прежде чем вы не проверите следующее:

1. Вызовы между внутренними номерами 100 и 101 работают
2. Вызов на внутренний номер 200 воспроизводит “Hello World”

Несмотря на то, что этот пример очень короткий и простой, он подчеркивает основные понятия контекстов, расширений, приоритетов и приложений. Теперь у вас есть фундаментальные знания, на которых строятся все диалпланы.

Построение интерактивного диал-плана

Диал-план, который мы только что построили является статическим; его результат исполнения будет тот же при каждом звонке. Тем не менее множество диал-планов требуют логически совершать разные действия в зависимости от введённых пользователем данных, которые мы сейчас и рассмотрим.

Приложения Goto(), Background() и WaitExten()

Как понятно из названия, приложение `Goto()` используется для отсылки вызова в другой участок диалплана. Синтаксис приложения `Goto()` потребует от нас указать данные: контекст назначения, расширение и приоритет в качестве аргументов, как указано ниже:

```
same => n,Goto(context,extension,priority)
```

Давайте создадим новый контекст `TestMenu` и укажем ещё одному расширению в нашем контексте `LocalSets` перенаправить вызовы туда через `Goto()`:

```
exten => 201,1,Goto(TestMenu,start,1)     ; добавим эту строку в конце
                                          ; контекста [LocalSets]
```

```
[TestMenu]
exten => start,1,Answer()
```

Теперь, если любое устройство в контексте `LocalSets` наберёт `201` вызов перенаправится на расширение `start` в контексте `TestMenu` (в котором пока ещё нет ничего интересного, ибо его код мы напишем позже).



Мы используем расширение `start` в этом примере, хотя можно использовать вообще любое имя в качестве расширения, включая цифры и буквы. Мы предпочитаем использовать буквенные расширения, хотя их и не набрать с цифровой панели телефона, но зато это даёт более читаемый диалплан. В нашем примере мы могли бы использовать `123` или `xyz123`, или же `99luftballons` или вообще что угодно вместо выбранного нами `start`. Само слово “start” ничего специально не означает в диалплане; это просто произвольное название расширения.

Одним из наиболее полезных приложений в интерактивном диалплане Asterisk является приложение `Background()`[7]. Как и `Playback()` оно воспроизводит записанный звуковой файл. Однако, в отличие от `Playback()` при нажатии клавиши (или серии клавиш) на клавиатуре телефона, оно прерывает воспроизведение и передает вызов на добавочный номер, соответствующий набранным цифрам. Если абонент нажимает `5`, к примеру, Asterisk сразу передаст вызов на первый приоритет расширения 5 (предполагая, что расширение 5 существует для обработки вызова).

Чаще всего приложение `Background()` используется для создания голосовых меню (часто называемых *автосекретарями*[8] или *телефонными деревьями*). Многие компании используют голосовые меню, чтобы направить абонентов в соответствующие отделы, тем самым избавляя операторов от необходимости отвечать на каждый вызов.

`Background()` имеет такой же самый синтаксис как и `Playback()`:

```
[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
```

Если хотите, чтобы Asterisk ожидал донабор от позвонившего после приветствия, то можно использовать `WaitExten()`. Приложение `WaitExten()` ожидает от позвонившего сигналы DTMF (нажатие цифр) и указывается сразу за приложением `Background()` как указано ниже:

```
[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
   same => n,WaitExten()
```

Если необходимо, чтобы приложение `WaitExten()` ожидало ввода только в течение определённого количества секунд (вместо дефолтного значения таймаута)[9] просто укажите эту цифру в скобках в качестве аргумента в `WaitExten()` как ниже:

```
same => n,WaitExten(5) ; Мы всегда передаём аргумент в приложение WaitExten()
```

Оба приложения - `Background()` и `WaitExten()` позволяют вводить цифры в виде донабора DTMF. После чего Астериск пытается найти расширение в текущем контексте, соответствующее введённым цифрам. Если такое совпадение найдено, Asterisk посылает вызов согласно этому расширению. Давайте продемонстрируем это добавлением нескольких строк в нашем примере диалплана:

---

7 Следует отметить, что некоторые люди ожидают что `Background()` из-за его имени, продолжит движение вперед, следуя шагам в диалплане во время воспроизведения звука. На самом деле его имя относится к тому, что он играет звук в фоновом режиме, ожидая DTMF на переднем плане.

8 Дополнительную информацию об автосекретаре вы найдёте в Главе 15.

9 Загляните в функцию диалплана `TIMEOUT()` чтобы узнать как можно изменить значение таймаутов по умолчанию. См. Главу 10 по списку всех функций диалплана.

```
[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
   same => n,WaitExten(5)
exten => 1,1,Playback(digits/1)

exten => 2,1,Playback(digits/2)
```

После выполнения этих изменений нужно сохранить и перезагрузить диалплан:

```
*CLI> dialplan reload
```

Если вы звоните на номер 201 вы должны услышать приветствие “Введите внутренний номер сотрудника чтобы соединиться с ним.” Система будет ожидать ввода цифр в течение 5 секунд. Если вы нажмёте `1` или `2`, то Астериск найдёт соответствующее расширение и прочитает его вам. Так как мы не прописали там никаких других инструкций к выполнению, то наш вызов на этом завершится. Вы также можете попробовать вводить другие цифры (например `3`), которых нет в диалплане и, потому, он не будет выполняться.

Давайте немного приукрасим эти вещи. Мы сделаем возврат на начало при помощи приложения `Goto()` чтобы наше приветствие повторялось после воспроизведения цифр донабора:

```
[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
   same => n,WaitExten(5)

exten => 1,1,Playback(digits/1)
   same => n,Goto(TestMenu,start,1)

exten => 2,1,Playback(digits/2)
   same => n,Goto(TestMenu,start,1)
```

Эти новые строки в диалплане передают управление на расширение `start` после воспроизведения выбранной цифры. В общем случае это выглядит более дружественно, чем просто завершение вызова.



Если вы рассмотрите внимательно возможности приложения `Goto()`, то увидите возможность передавать вызов в это приложение с одним, двумя и тремя аргументами. Если передаётся один аргумент, то Asterisk предположит что это приоритет назначения в текущем расширении. Если передаётся два аргумента, то Астериск будет рассматривать их как расширение и приоритет для перехода в текущем контексте.
В этом примере мы передали все три аргумента для ясности, но передача только расширения и приоритета имела бы тот же эффект, поскольку контекст назначения совпадает с исходным контекстом.

Обработка неверных значений и тайм-аутов

После того, как мы создали вместе наше первое голосовое интерактивное меню, давайте добавим в него некоторые дополнительные специальные расширения. Для начала нам нужно расширение для обработки неверных, несуществующих расширений. Asterisk получив в контексте запрос на обработку несуществующего в текущем контексте расширения (например – нажатие `9` в примере выше), перенаправит вызов в расширение `i`. Нам также понадобится специальное расширение для обработки ситуаций, когда позвонивший вообще ничего не нажал в течение определённого времени (дефолтный таймаут 10 секунд). Вызов так же будет перенаправлен в расширение `t` если абонент слишком поздно нажал что-то после `WaitExten()` при вызове. Посмотрите как будет выглядеть наш диалплан после того, как мы добавим эти два расширения:

```
[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
   same => n,WaitExten(5)

exten => 1,1,Playback(digits/1)
   same => n,Goto(TestMenu,start,1)

exten => 2,1,Playback(digits/2)
   same => n,Goto(TestMenu,start,1)

exten => i,1,Playback(pbx-invalid)
   same => n,Goto(TestMenu,start,1)

exten => t,1,Playback(vm-goodbye)
   same => n,Hangup()
```

Использование расширений `i`[10] и `t` делает наше меню более надежным и дружественным по отношению к пользователям. Тем не менее, оно все еще довольно ограничено, потому что внешние абоненты по-прежнему не имеют возможности связаться с живым человеком. Для этого нам нужно узнать о другом приложении, называемом `Dial()`.

Использование приложения Dial()

Одной из наиболее ценных функций Asterisk является возможность соединения разных абонентов друг с другом. Это особенно полезно, когда абоненты используют различные методы связи. Например, абонент А может общаться по традиционной аналоговой телефонной сети, в то время как пользователь Б может сидеть в кафе на другом конце света и говорить по IP-телефону. К счастью, Asterisk берёт на себя большую часть тяжелой работы по соединению и трансляции вызовов между разрозненными сетями. Все что вам нужно сделать, это научиться использовать приложение `Dial()`.

Синтаксис приложения `Dial()` более сложен, чем другие приложения, которые мы использовали до сих пор, но это не должно вас пугать. В приложение `Dial()` передаётся до четырёх аргументов, которые мы рассмотрим ниже.

Аргумент 1: назначение

Первый аргумент – это назначение: т.е. куда вы хотите позвонить, который (в простой форме) содержит технологию (или транспорт), посредством которой выполняется вызов, далее знак слэш `-` / и адрес удалённой конечной точки или ресурса. Наиболее типичные примеры технологий включают в себя DAHDI (для аналоговых и цифровых T1/E1/J1, PRI & BRI каналов), SIP и IAX2.

Например, если мы хотим совершить вызов на конечное устройство DAHDI, идентифицируемого как `DAHDI/1`, которое является фактически аналоговым FXS каналом с подключенным к нему аналоговым телефоном. Технология вызова будет `DAHDI` и ресурс (или идентификатор канала) будет `1`. Точно так же вызов на SIP устройство (которое описано в *sip.conf*) может иметь назначение вида `SIP/0004F2001122`, а вызов IAX устройства (описанного в *iax.conf*) может иметь назначение вида `IAX2/Softphone`[11]. Если мы собираемся позвонить через Asterisk в канал `DAHDI/1` за которым закреплён в диал-плане внутренний номер `105`, мы добавим следующее расширение:

```
exten => 105,1,Dial(DAHDI/1)
```

10 Расширение `i` для обработки несуществующих назначений поддерживается в основном для таких приложений как `Background()`. Оно не используется для обработки неверно набранных расширений или не попадающих в шаблон расширений.

11 Если это рабочая конфигурация (а не тестовый пример) то не очень удачное название для этого устройства. Если вы добавите больше чем один софтфон (или планируете это в будущем) – как вы будете их различать?

Можно также делать одновременный вызов нескольких каналов, разделяя назначения знаком амперсанд (&), как показано ниже:

```
exten => 105,1,Dial(DAHDI/1&SIP/0004F2001122&IAX2/Softphone)
```

Приложение `Dial()` позвонит всем указанным назначениям одновременно, и соединит входящий вызов с первым ответившим каналом назначения (остальные назначения при этом прекращают звонить). Если приложение `Dial()` не может соединиться ни с одним назначением, Asterisk установит переменную вызова `DIALSTATUS` в значение с причиной невозможности набрать назначение, и продолжит выполнение со следующего приоритета в расширении[12].

Приложение `Dial()` также даёт возможность установить соединение с удалённым конечным VoIP-устройством, даже не указанным ранее в конфигурационных файлах соответствующего канала. Полный синтаксис команды имеет вид:

```
Dial(technology/user[:password]@remote_host[:port][/remote_extension])
```

В качестве примера вы можете позвонить на демонстрационный сервер компании Digium используя протокол IAX2 и следующее расширение:

```
exten => 500,1,Dial(IAX2/guest@misery.digium.com/s)
```

Полный синтаксис приложения `Dial()` несколько отличается для каналов DAHDI:

```
Dial(DAHDI/[gGrR]channel_or_group[/remote_extension])
```

Например, если вы набираете номер `1-800-555-1212` через DAHDI канал под номером 4[13]:

```
exten => 501,1,Dial(DAHDI/4/18005551212)
```

Аргумент 2: тайм-аут

Второй аргумент приложения `Dial()` - это тайм-аут, указываемый в секундах. Если тайм-аут задан, приложение `Dial()` будет пытаться вызывать назначение указанное количество секунд до поднятия трубки, после чего перейдёт на следующий приоритет набранного расширения.

Если тайм-аут не указан, `Dial()` будет продолжать звонить на указанные каналы до тех пор, пока кто-либо не поднимет трубку. Давайте добавим 10-ти секундный тайм-аут в наше расширение:

```
exten => 502,1,Dial(DAHDI/1,10)
```

Если абонент ответит до истечения таймаута, то каналы соединяются и диалплан на этом закончится. Если же абонент просто не отвечает, занят, или даже недоступен по каким-то причинам, Asterisk устанавливает это значение в переменной `DIALSTATUS` и продолжает выполнение следующего приоритета в текущем расширении.

Давайте соорудим сейчас то, что мы изучили в следующем примере:

```
exten => 502,1,Dial(DAHDI/1,10)
   same => n,Playback(vm-nobodyavail)
   same => n,Hangup()
```

Как мы видим в этом примере, звуковой файл vm-nobodyavail.gsm будет воспроизведён в случае, если вызов не отвечен.

12 Мы рассмотрим переменные в разделе Использование переменных. В следующей главе мы обсудим как задавать выполнение диалплана основываясь на значениях `DIALSTATUS`.

13 Подразумевается некоторые понимание того, что выбранный канал знает как добраться до внешних номеров.

Аргумент 3: опции

Третий аргумент в `Dial()` - строка опций. Она может содержать один или больше знаков, которые модифицируют поведение приложения `Dial()`. Список всех возможных опций слишком огромен, чтобы рассматривать его в рамках этого раздела. Например, одна из популярных опций – `m`. Если букву `m` указать третьим аргументом, вызывающая сторона будет слышать музыку вместо гудков вызова, который совершается в этот момент вызываемой стороне (разумеется мы предполагаем, что музыка на удержание у нас уже правильно сконфигурирована). Добавим опцию `m` в наш последний пример, чуть изменив первую строчку:

```
exten => 502,1,Dial(DAHDI/1,10,m)
   same => n,Playback(vm-nobodyavail)
   same => n,Hangup()
```

Аргумент 4: URI

Четвёртый и последний аргумент в приложении `Dial()` - это URI. Если канал назначения поддерживает приём вызовов по URI, то указанный URI будет передаваться (например, если у вас есть IP-телефон, поддерживающий приём URI, то он появится на экране телефона; в случае использования софтфона URI может появиться в виде всплывающего окна на экране компьютера). Этот аргумент очень редко используется.



Некоторые телефоны (если у вас такие есть) поддерживают информацию URI, которая к ним передаётся. Если вы ищете что-то типа всплывающих окон, рекомендуем обратиться к Главе 18, и более детально к разделу о Jabber в "Использование XMPP (Jabber) с Asterisk" на странице .

Обновление диалплана

Давайте изменим расширения 1 и 2 в нашем меню, где используем приложение `Dial()`:

```
[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
   same => n,WaitExten(5)

exten => 1,1,Dial(SIP/0000FFFF0001,10) ; Замените 0000FFFF0001 на имя своего
                                       ; устройства
   same => n,Playback(vm-nobodyavail)
   same => n,Hangup()

exten => 2,1,Dial(SIP/0000FFFF0002,10) ; Замените 0000FFFF0002 на имя своего
                                       ; устройства
   same => n,Playback(vm-nobodyavail)
   same => n,Hangup()

exten => i,1,Playback(pbx-invalid)
   same => n,Goto(TestMenu,start,1)

exten => t,1,Playback(vm-goodbye)
   same => n,Hangup()
```

Пустые аргументы

Заметим, что второй, третий и четвёртый аргументы могут быть пустыми; обязательным является только первый аргумент. Например, если требуется указать строковую опцию, но без таймаута, просто оставьте аргумент тайм-аут незаполненным, как указано ниже:

```
exten => 1,1,Dial(DAHDI/1,,m)
```

Использование переменных

Переменные могут использоваться в диалплане Asterisk чтобы уменьшить набор строк кода, увеличить понимание или добавить логику. Если у вас есть опыт по программированию, то вы уже понимаете что такое переменные. Если нет – мы кратко расскажем что такое переменные и как они используются. Переменные являются жизненноважной концепцией диалплана Asterisk (и вы не найдете их в диалплане любой проприетарной АТС).

Переменная - это именованный контейнер, который может содержать некоторое значение. Преимущество переменной заключается в том, что её содержимое может изменяться, но её имя не меняется, то есть можно написать код, который ссылается на имя переменной и не беспокоиться о её содержимом. Например, мы можем создать переменную `JOHN` и присвоить ей значение `DAHDI/1`. И, когда мы пишем наш диалплан, мы можем указать канал John по имени, вместо того, чтобы помнить, что John использует канал с именем `DAHDI/1`. Если в какой-то момент мы изменим канал John на что-то другое, нам не придется менять код, ссылающийся на переменную `JOHN`, нам нужно только изменить значение, присвоенное переменной.

Имеется два способа ссылаться на переменную. Чтобы сослаться на имя переменной, просто напишите имя переменной, например `LEIF`. Другим способом можно сослаться на значение переменной указав знак доллара, открывающую фигурную скобку, имя переменной и закрывающую фигурную скобку (в случае если мы будем ссылаться на значение переменной с `${LEIF}`). Вот как мы можем использовать переменную внутри приложения Dial:

```
exten => 301,1,Set(LEIF=SIP/0000FFFF0001)
   same => n,Dial(${LEIF})
```

В нашем примере диалплана, каждый раз, когда происходит обращение к `${LEIF}`, Астериск автоматически изменяет её на значение, которое было перед этим присвоено переменной с именем `LEIF`.



Помните, что названия переменных чувствительны к регистру. Переменная `LEIF` совсем не та же, что с названием `Leif`. Для большей читабельности все наши переменные в примерах мы будем писать в верхнем регистре. Вам надо также в дальнейшем знать, что все переменные в Asterisk будут также в верхнем регистре. Некоторые переменные, такие как `CHANNEL` и `EXTEN`, зарезервированы в Asterisk. Не пытайтесь назначить какие-то значения этим переменным. Распространённым также является запись глобальных переменных в верхнем регистре, а канальных переменных в формате языков Pascal/Camel.

Можно использовать три типа переменных в нашем диал-плане: глобальные переменные, канальные переменные и переменные окружения. Рассмотрим каждый из них.

Глобальные переменные

Как следует из самого названия, *глобальные* переменные всегда доступны всем каналам. Глобальные переменные полезны тем, что их можно использовать в любом месте диалплана для повышения удобочитаемости и управляемости. Представьте на мгновение, что у вас есть большой диалплан и несколько сотен обращений к каналу `SIP/0000FFFF0001`. Теперь вообразите, что вы передвигаетесь по диалплану и меняете все ссылки на `SIP/0000FFFF0002`. Это будет долгим процессом, и не без ошибок, как минимум.

Другим образом, вы могли бы определить глобальную переменную, содержащую значение `SIP/0000FFFF0001` в начале диалплана и ссылаться на неё, вместо того чтобы изменять значения по всему диалплану теперь достаточно изменить всего одну строку кода, чтобы изменить действие во всех местах диалплана, где используется этот канал.

Глобальные переменные декларируются в контексте `[globals]` в самом начале файла *extensions.conf*. Например, мы создадим глобальную переменную с именем `LEIF` и значением

SIP/0000FFFF0001. Эта переменная устанавливается каждый раз, когда Asterisk перечитывает диалплан:

```
[globals]
LEIF=SIP/0000FFFF0001
```

Канальные переменные

Переменная *канала*, или канальная переменная – это переменная, ассоциированная только с обычными вызовами. В отличии от глобальных переменных эта переменная действует только в течение текущего звонка, и доступна только для использования в этом звонке.

Имеется множество предопределённых переменных канала для использования их в диалплане, они описаны подробно в Asterisk wiki. Канальные переменные устанавливаются через приложение Set():

```
exten => 202,1,Set(MagicNumber=42)
    same => n,SayNumber(${MagicNumber})
```

Вы можете увидеть и другие различные канальные переменные. Почитайте!

Переменные среды

Переменные среды – это способ получить переменные Unix среды из среды Asterisk. Это делается при помощи функции диалплана ENV()[14]. Синтаксис ${ENV(*var*)}, где *var* - переменная среды Unix на которую вы ссылаетесь. В большинстве случаев переменные среды не используются в диалпланах Asterisk, но то, что они доступны, вам необходимо знать.

Добавление переменных в ваш диал-план

Теперь, когда мы изучили переменные, давайте мы заставим их работать в нашем диалплане. Мы начнём с добавления трёх глобальных переменных, ассоциированных с их канальными именами:

```
[globals]
LEIF=SIP/0000FFFF0001
JIM=SIP/0000FFFF0002
RUSSELL=SIP/0000FFFF0003

[LocalSets]
exten => 101,1,Dial(${LEIF})
exten => leif,1,Dial(${LEIF})

exten => 102,1,Dial(${JIM})
exten => jim,1,Dial(${JIM})

exten => 103,1,Dial(${RUSSELL})
exten => russell,1,Dial(${RUSSELL})

exten => 201,1,Goto(TestMenu,start,1) ; доступ к контексту TestMenu

[TestMenu]
exten => start,1,Answer()
   same => n,Background(enter-ext-of-person)
   same => n,WaitExten()

exten => 1,1,Dial(DAHDI/1,10)
   same => n,Playback(vm-nobodyavail)
   same => n,Hangup()
```

14 Мы рассмотрим функции диалплана позже. Не беспокойтесь о слишком многих переменных среды прямо сейчас. Это не так важно для понимания диалплана.

```
exten => 2,1,Dial(SIP/Jane,10)
   same => n,Playback(vm-nobodyavail)
   same => n,Hangup()

exten => i,1,Playback(pbx-invalid)
   same => n,Goto(TestMenu,start,1)

exten => t,1,Playback(vm-goodbye)
   same => n,Hangup()
```

Вы заметили, что мы добавили имена-псевдонимы в качестве внутренних номеров. В разделе "Расширения", мы поясняли уже, что для Asterisk нет никакой разницы по какой схеме имён идентифицируются внутренние номера. Мы просто добавляем оба – номерные и именные расширения для связи с каждым конечным устройством; внутренний номер `101` и `leif` оба приведут к устройству `SIP/0000FFFF0001`; внутренний номер `102` и `jim` оба приведут к устройству `SIP/0000FFFF0002`; а `103` и `russell` попадают на `SIP/0000FFFF0003`. Устройства, нами определены при помощи глобальных переменных `${LEIF}`, `${JIM}`, и `${RUSSELL}`, соответственно, и мы можем набирать их используя приложение `Dial()`.

В нашем тестовом меню мы просто укажем несколько различных конечных устройство для набора, как `DAHDI/1` и `SIP/Jane`. Они могут быть заменены на любые конечные устройства, какие мы пожелаем. В нашем контексте `TestMenu`, который мы создаем, чтобы иметь представление как выглядит диалплан Asterisk, может дать вам идеи как это лучше сделать.

Совпадение по шаблонам

Если мы хотим, чтобы люди могли звонить *через* Asterisk, который соединяет их с внешними ресурсами, нам нужен способ к распознаванию любого возможного номера телефона, который может набрать абонент. В подобных ситуациях Asterisk предлагает *сопоставление по шаблонам*. Сравнение/сопоставление с шаблоном позволяет создать в вашем диалплане всего одно расширение, соответствующее множеству различных номеров. Это чрезвычайно полезно.

Синтаксис сравнения по шаблонам

При использовании сопоставления, определённые конкретные буквы и символы представляют то, с чем мы сравниваем. *Шаблон всегда начинается с подчёркивания* (_). Это сообщает Asterisk, что мы собираемся сравнивать с шаблоном, а не ожидаем точное имя расширения.



Если вы забыли поставить знак подчёркивания в начале шаблона, то Asterisk предполагает, что это просто именное расширение, и не пытается сравнивать с ним набор как с шаблоном. Это одна из наиболее общих ошибок людей, которые начинают изучать Астериск.

После подчёркивания можно использовать следующие одну или несколько букв:

`X`
: Совпадение одной цифры от 0 до 9.

`Z`
: Совпадение одной цифры от 1 до 9.

`N`
: Совпадение одной цифры от 2 до 9.



Другая распространённая ошибка – использование букв `X`, `Z`, и `N` в словах, составляющих шаблон совпадения; чтобы это обойти – используйте эти буквы в квадратных скобках (чувствительны к регистру), например `_[n]o[x]ious-XXX`.

`[15-7]`

Совпадение одного знака из указанного диапазона цифр. В данном случае совпадение будет найдено при цифре 1, а также других цифрах в диапазоне 5, 6, 7.

`.` (*точка*)

Свободное совпадение по шаблону; совпадёт *один или более* символов, независимо от того, что они представляют.



Если вы не очень заботитесь о точности, свободный шаблон с точкой даст непредсказуемое поведение в вашем диалплане (например – совпадение встроенных специальных расширений, таких как `i` или `h`). Нужно использовать точку в шаблонах после одной или нескольких цифр, которые найдены, если возможно, сравнением набранного номера с шаблоном. Например, вот такой шаблон никогда не должен использоваться:

```
_.
```

По факту использования такого шаблона, если всё же вы пытаетесь использовать его, Астериск будет каждый раз предупреждать вас. Если по каким-то причинам всё же вам нужно использовать шаблон для всех цифр, например, используйте следующий шаблон, по которому будут совпадать все наборы, начинающиеся с любой цифры (рассмотрите использование знака `!` если нужно улавливать по отсутствию любых знаков в шаблоне или нескольких):

```
_X.
```

Или такой пример, отвечает на совпадение любой цифры или буквы:

```
_[0-9a-zA-Z].
```

`!` (*восклицание*)

Подстановочный знак соответствия; соответствует *нулю или более* символов, независимо от того, что они представляют.

Для использования шаблонов в вашем диалплане просто поместите шаблон в качестве имени (или номера) расширения:

```
exten => _NXX,1,Playback(silence/1&auth-thankyou)
```

В этом примере с шаблоном сравнивается любой трёхзначный внутренний номер от 200 до 999 (буква `N` определяет цифры от 2 до 9, а каждая `X` определяет цифры от 0 до 9). Тем самым мы говорим, что любой, кто наберёт трёхзначный номер от 200 до 999 в этом контексте, услышит аудио файл *auth-thankyou.gsm*.

Ещё одно важное дополнение для понимания как набранный номер сопоставляется с шаблоном, если Asterisk нашёл более чем один шаблон для набранного номера. В таком случае будет использоваться *наиболее конкретный* (просчитывая сравнение слева направо). Скажем, вы указали следующие два шаблона, и абонент набрал `555-1212`:

```
exten => _555XXXX,1,Playback(silence/1&digits/1)
exten => _55512XX,1,Playback(silence/1&digits/2)
```

В этом случае будет выбрано второе расширение , потому что оно точнее подходит.

Примеры совпадения по шаблонам

Следующий шаблон соответствует любому семизначному номеру, если первая цифра равна 2 или больше:

```
_NXXXXXX
```

Приведённый выше шаблон описывает все семизначные локальные номерные планы в Северной Америке.

В региональных 10-значных кодах этот шаблон будет выглядеть следующим образом:

```
_NXXNXXXXXX
```

Обратите внимание, что ни один из этих двух шаблонов не будет обрабатывать междугородние вызовы. Мы рассмотрим их в ближайшее время.

NANP и мошенничество с тарифами

Североамериканский номерной план (NANP) является открытой схемой телефонной нумерации, охватывающей 19 стран Северной Америки и Карибского архипелага. Все эти страны имеют международный код 1.

В Соединённых Штатах и Канаде правила регуляции в области телекоммуникаций схожи (и рациональны) и достаточны для совершения международных вызовов в страны с кодом 1 и ожидаемой адекватной стоимостью звонка. Однако большинство людей в 17 странах (кроме США и Канады) не реализуют возможности, заложенные правилами регуляции, которые сильно отличаются в этих странах. Смотрите правила NANP.

Одна из популярных афер использования NANP - попытка обмануть наивных Североамериканцев для вызовов по дорогим поминутным тарифам на номера в Карибские страны; абоненты считают, что, поскольку они набрали номер 1-NPA-NXX-XXXX, то они платят по стандартному национальному международному тарифу. Поскольку данная страна может иметь правила, которые позволяют эту форму вымогательства, абонент сам несет ответственность за стоимость звонка.

Единственный способ предотвратить подобные действия - блокировать вызовы определенных региональных кодов (например, 809) и снимать ограничения только по мере необходимости.

Рассмотрим ещё один:

```
_1NXXNXXXXXX
```

Этот пример шаблона описывает номер, начинающийся с 1, региональный код от 200 до 999, и затем любой семизначный номер. В зоне действия регуляции NANP вы будете использовать этот шаблон для международных вызовов[15].

И наконец – такой:

```
_011.
```

Обратите внимание на точку в конце шаблона. Он описывает все номера, которые начинаются с 011 и далее одну или более цифр. В зоне действия NANP будет определяться как иностранный номер. (Мы будем использовать этот шаблон в следующем разделе при добавлении возможностей исходящих вызовов в нашем диалплане.)

Шаблоны, используемые в других странах

В этой секции показаны шаблоны ориентированные на NANP, но базирующиеся на логике применения в любой стране. Ниже даны примеры для некоторых стран (обратите внимание, что мы не проверяли их, и они почти наверняка неполные):

```
; UK, Германия, Италия, Китай и т.д.
_00. ; международный телефонный код
_0.  ; национальный префикс набора

Австралия
```

15 Если вы выросли в Северной Америке, то можете поверить, что 1, которую вы набираете перед междугородным звонком, - это «междугородный код». Это неверно. Цифра 1 - это международный код страны для NANP. Имейте это в виду, если Вы пришлете свой номер телефона кому-то в другой стране, получатель может не знать ваш код страны и, таким образом, не сможет позвонить вам только с вашим кодом города и номером телефона. Ваш полный номер телефона с кодом страны +1 NPA NXX XXXX (где NPA-ваш код города) - например, +1 416 555 1212.

```
_0011. ; международный телефонный код
_0. ;  национальный префикс набора
```

Это никоим образом не является всеобъемлющим, но должно дать вам общее представление о шаблонах, которые вы можете применять для своей собственной страны.

Использование канальной переменной ${EXTEN}

Что произойдёт, если вы хотите использовать шаблон для сравнения набранных номеров, но при этом хотите знать какие именно цифры были при этом набраны? Укажите переменную канала `${EXTEN}`. Каждый раз, когда набран какой либо номер, Asterisk установит переменную канала `${EXTEN}` в значение набранных цифр. Мы можем использовать приложение `SayDigits()` чтобы протестировать это:

```
exten => _XXX,1,Answer()
   same => n,SayDigits(${EXTEN})
```

В этом примере приложение `SayDigits()` прочитает нам набранный трёхзначный номер.

Часто могут быть полезными манипуляции с `${EXTEN}` отрезая определённое количество цифр впереди внутреннего номера. Это достигается синтаксисом `${EXTEN:`*x*`}`, где *x* - количество цифр, которые нужно пропустить в переменной слева направо. Например, если значение переменной `${EXTEN}` равно `95551212`, то `${EXTEN:1}` будет возвращать `5551212`. Давайте попробуем такой пример:

```
exten => _XXX,1,Answer()
   same => n,SayDigits(${EXTEN:1})
```

В этом примере приложение `SayDigits()` стартует со второй цифры и проговорит только две последние цифры из набранного внутреннего номера.

Продвинутые возможности манипуляций с цифрами

Переменная `${EXTEN}` в общем случае имеет полный синтаксис `${EXTEN:`*x*`:`*y*`}`, где *x* – начальная позиция и *y* – количество возвращаемых цифр. Например, при набранном:

```
94169671111
```

мы можем выделить следующие цифры используя конструкцию `${EXTEN:`*x*`:`*y*`}`:

- `${EXTEN:1:3}` вернёт нам `416`
- `${EXTEN:4:7}` вернёт нам `9671111`
- `${EXTEN:-4:4}` отсчитает 4 цифры с конца и вернёт четыре цифры, мы получим `1111`
- `${EXTEN:2:-4}` отсчитает две цифры и отнимет последние четыре цифры, мы получим `16967`
- `${EXTEN:-6:-4}` отсчитает шесть цифр с конца и отнимет последние четыре цифры, мы получим `67`
- `${EXTEN:1}` вернёт нам все цифры после первой, то есть `4169671111` (если количество возвращаемых цифр будет пустым, будет возвращена вся строка)

Это очень мощная конструкция, но большинство этих вариаций не нужны при обычном использовании. Большей частью вы будете использовать просто `${EXTEN}` (или, возможно - `${EXTEN:1}` если потребуется отрезать внешний префикс, такой как приставка 9 перед набором).

Включения (инклюды)

Asterisk содержит важную возможность добавления расширений, доступных в одном контексте, в другой контекст. Это доступно через директиву `include`. Директива `include` даёт нам возможность контролировать доступ к другой секции диалплана.

Оператор `include` имеет следующую форму, где *`context`* – название другого контекста, который мы включаем в текущий контекст:

```
include => context
```

Включение одного контекста в другой даёт возможность набора расширений включенного контекста в текущем.

Когда мы включаем другие контексты в текущий, мы должны помнить о порядке, в котором мы их включаем. Asterisk сначала попытается найти совпадение набранного номера с расширениями в текущем контексте. В случае неудачи он попробует первый включенный контекст (включая любые контексты, включенные в этот контекст), а затем продолжит поиск в других включенных контекстах в том порядке, в котором они были включены.

Мы продолжим разговор о директиве `include` дальше в Главе 7.

Вывод

И наконец у вас есть он — простой, но вполне функционирующий диалплан. Есть ещё многие вещи, которые мы не рассмотрели пока, но у вас уже есть все фундаментальные основы. В следующих главах мы продолжим строить на этих основах.

Если отдельные части этого диалплана недостаточно осмыслены, можно вернуться назад и перечитать раздел ещё раз или два, прежде чем перейти к следующей главе. Крайне важно, чтобы вы понимали эти принципы и как их применять, поскольку следующие главы основываются на этой информации.

Внешние подключения

Вы не всегда можете контролировать что происходит снаружи.
Но вы всегда можете контролировать что присходит внутри.

— Уэйн Дьер

В предыдущих главах мы рассмотрели много важной информации, которая необходима для рабочей системы Asterisk. Тем не менее, нам еще предстоит выполнить одно дело, которое жизненно важно для любой полезной АТС, а именно - подключение к внешнему миру. В этой главе мы исправим эту ситуацию.

Архитектура Asterisk значительна во многом благодаря тому, что она рассматривает все типы каналов как равные. Это контрастирует с традиционной АТС, где транки (которые подключаются к внешнему миру) и внутренние номера (которые подключаются к пользователям и ресурсам) сильно отличаются. Тот факт, что диалплан Asterisk относится ко всем каналам похожим образом, означает, что в системе Asterisk вы можете легко выполнять вещи, которые намного сложнее (или невозможно) достичь на традиционной АТС.

Однако эта гибкость имеет цену. Поскольку система не знает разницы между внутренним ресурсом (например, телефонным аппаратом) и внешним (например, телефонные линии), вам нужно убедиться что ваш диалплан правильно обрабатывает каждый тип ресурса.

Основы транкинга

Целью *транкинга* является предоставление общего соединения между двумя объектами. Например, транк - это шоссе, которое соединяет два города вместе. Железные дороги широко использовали термин «trunk» (транк, с англ. - магистраль), чтобы ссылаться на основную линию, соединяющую фидерные линии вместе.

Аналогичным образом, в телекоммуникационной сети транкинг используется для соединения двух систем вместе. Провайдеры используют соединительные линии связи (транки) для соединения своих сетей вместе, а в АТС линии, соединяющие АТС с внешним миром обычно называются транками (хотя сами поставщики услуг обычно не считают их транками). С технической точки зрения определение транка (магистрали) не так ясно как раньше (транки (соединительные линии) АТС использовали совершенно другую технологию для линий станций), но в качестве концепции все еще очень важны транки. Например, с VoIP все на самом деле является одноранговым (так что с точки зрения технологии больше не существует такой вещи, как транк), но все же полезно иметь возможность различать ресурсы VoIP, которые подключаются к внешнему миру и VoIP-ресурсы, которые подключаются к конечным точкам пользователя (например, SIP-телефонам).

Вероятно, проще всего подумать о транке как о наборе линий, обслуживающих маршрут. Таким образом, на УАТС Asterisk у Вас могут быть транки, которые идут к вашему VoIP-провайдеру для междугородних звонков, транки для ваших сетей ТфОП и транки, которые соединяют ваши офисы вместе. Эти транки могут фактически работать через одно и то же сетевое соединение, но в вашем диалплане можете относиться к ним совершенно по-разному.

Хотя мы считаем, что VoIP в конечном итоге полностью заменит ТфОП, многие из концепций, которые используются в VoIP-линиях (например, «номер телефона»), обязаны своим существованием больше истории, чем любые технические требования, и поэтому мы считаем, что будет полезно обсудить использование традиционных линий ТфОП с Asterisk, прежде чем мы перейдем к VoIP.

Если система, которую вы устанавливаете, будет использовать только VoIP-линии - это не проблема. Идите прямо в раздел VoIP этой главы[1], и мы рассмотрим что вам нужно сделать. Мы рекомендуем читать разделы о ТфОП по вашему усмотрению, так как в них могут быть общие знания, которые могут вам пригодиться, но это не является строго необходимым для понимания и использования Asterisk.

Фундаментальный диалплан для исходящих соединений

В традиционной УАТС внешние линии обычно доступны посредством кода доступа, который должен быть набран до номера.[2] Обычно для этой цели используется цифра 9.

В Asterisk аналогичным образом можно назначить 9 для маршрутизации внешних вызовов, но поскольку диалплан Asterisk намного более интеллектуальный, на самом деле не нужно заставлять ваших пользователей набирать 9 перед тем, как совершить вызов. Как правило, у вас будет диапазон номеров для вашей системы (скажем, 100-199) и диапазон кодов функций (от *00 до *99). Все, что находится за пределами диапазона, который соответствует шаблону набора номера для вашей страны или региона, можно рассматривать как внешний вызов.

Если у вас есть один оператор, предоставляющий всю вашу внешнюю маршрутизацию, вы можете обрабатывать свой внешний набор через несколько простых совпадений шаблонов. Пример в этом разделе действителен для Североамериканского плана нумерации (NANP). Если ваша страна не находится в пределах NANP (который обслуживает Канаду, США и несколько стран Карибского бассейна), вам понадобится другой шаблон.

Раздел `[globals]` содержит две переменные, называемые `LOCAL` и `TOLL`.[3] Целью этих переменных является упрощение управления вашим диалпланом, если вам когда-либо понадобится изменить провайдера. Они позволяют сделать одно изменение в диалплане, которое повлияет на все места, где указан данный канал:

```
[globals]
LOCAL=DAHDI/G0                ; если у Вас в системе есть PSTN карта
TOLL=SIP/YourVoipCarrier      ; как определено в sip.conf
```

Секция `[external]` содержит фактический код диалплана, который распознает набранные номера и передает их в приложение `Dial()`:[4]

```
[external]
exten => _NXXNXXXXXX,1,Dial(${LOCAL}/${EXTEN})  ; 10-значный шаблон для NANP
exten => _NXXXXXX,1,Dial(${LOCAL}/${EXTEN})     ; 7-значный шаблон для NANP
exten => _1NXXNXXXXXX,1,Dial(${TOLL}/${EXTEN})  ; шаблон международного
                                                 ; направления для NANP
exten => _011.,1,Dial(${TOLL}/${EXTEN})   ; Шаблон международного вызова,
                                          ; сделанного из NANP
; Этот раздел функционально совпадает с приведенным выше разделом.
; Это для людей, которым нравится набирать «9» для их звонков
exten => _9NXXNXXXXXX,1,Dial(${LOCAL}/${EXTEN:1})
exten => _9NXXXXXX,1,Dial(${LOCAL}/${EXTEN:1})
exten => _91NXXNXXXXXX,1,Dial(${TOLL}/${EXTEN:1})
```

1 Но не требуйте 200 долларов.

2 В ключевой системе каждая линия имеет соответствующую кнопку на каждом телефоне, а линии доступны по нажатию нужной клавиши.

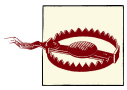
3 Вы можете назвать как пожелаете. Слова «local» и «toll» не имеют никакого встроенного значения для диалплана Астериска.

4 Для получения дополнительной информации о шаблонах см. Главу 6.

```
exten => _9011.,1,Dial(${TOLL}/${EXTEN:1})
```

В любом контексте, который будет использоваться комплектами или пользовательскими устройствами, вы должны использовать директиву `include =>`, чтобы разрешить доступ к контексту `external`:

```
[LocalSets]
include => external
```



Крайне важно, чтобы вы не включали доступ к внешним линиям в любом контексте, который мог бы обрабатывать входящий вызов. Риск здесь заключается в том, что фишинг-бот может в конечном итоге получить доступ к вашим исходящим транкам (вы будете удивлены тому, насколько распространены эти фишинг-боты).
Мы не можем не подчеркнуть, насколько важно, чтобы вы гарантировали, что никакой внешний ресурс не сможет получить доступ к вашим платным линиям.

Линии PSTN (ТфОП)

Общественная коммутируемая телефонная сеть (PSTN) (или же телефонная сеть общего пользования - ТфОП) существует уже более века. Это предшественник многих технологий, которые формируют наш мир сегодня, от Интернета до MP3-плееров.

Традиционные транки ТфОП

Существует два типа фундаментальных технологий, которые используются операторами телефонной связи для доставки телефонных линий: аналоговых и цифровых.

Аналоговая телефония

Первые телефонные линии были аналоговыми. Звуковой сигнал, который вы создаете с помощью вашего голоса, использовался для генерации электрического сигнала, который переносился на другой конец соединения. Электрический сигнал имел те же характеристики, что и создаваемый звук.

Аналоговые линии имеют несколько характеристик, которые отличают их от других линий, которые вы, возможно, захотите подключить к Asterisk:

- Нет канала сигнализации - большинство сигналов являются электромеханическими.
- Контроль отключения обычно задерживается на несколько секунд и не является полностью надежным.
- Дальнейшее наблюдение минимально (например, контроль ответа отсутствует).
- Различия в линиях означают, что звуковые характеристики будут варьироваться от линии к линии и потребуют настройки.

Аналоговые линии, которые вы захотите подключить к вашей системе Asterisk, должны подключаться к порту Foreign eXchange Office (FXO). Поскольку на любом стандартном компьютере нет порта для FXO, в систему необходимо приобрести и установить плату FXO для подключения традиционных аналоговых линий.[5]

FXO и FXS

Для любой аналоговой линии есть два конца: офис (обычно центральный офис ТфОП) и станция (обычно это телефон, но также может быть карта, такая как модем или линейная карта в АТС).

5 Вы бы использовали ту же самую карту, если бы хотели подключить традиционную домашнюю телефонную линию к вашей системе Asterisk.

Центральный офис отвечает за:

- Мощность на линии (номинально 48 Вольт постоянного тока)
- Напряжение звонка (номинально 90 Вольт переменного тока)
- Предоставление гудка набора
- Обнаружение состояния трубки (снятие трубки и повешение)
- Отправка дополнительной сигнализации, такой как идентификатор вызывающего абонента

Станция несет ответственность за:

- Предоставление звонка (или, по крайней мере, способности обрабатывать сигнальное напряжение каким-либо образом)
- Предоставление панели набора номера (или способ отправки DTMF)
- Предоставление рычажного переключателя трубки для указания статуса линии

Порт Foreign eXchange (FX) называется так в зависимости от того, к чему он подключается, а не тем, что он делает. Так, например, порт Foreign eXchange Office (FXO) фактически является станцией: он подключается к центральному офису. Порт External eXchange Station (FXS) фактически является портом, предоставляющим услуги центрального офиса (другими словами, вы подключаете аналоговый аппарат к порту FXS).

Вот почему настройки сигнализации в конфигурационных файлах Asterisk кажутся обратными: порты FXO используют сигнализацию FXS; порты FXS используют сигнализацию FXO. Когда вы понимаете, что имя физического типа порта основано на том, к чему оно подключается, имена сигналов в Asterisk имеют немного больше смысла: если порт FXO подключается к центральному офису, он должен вести себя как станция , и поэтому требуется сигнализация FXS.

Обратите внимание, что переход от FXO к FXS не так то просто сделать с помощью изменения настроек. Для портов FXO и FXS требуется совершенно разная электроника. Большинство аналоговых карт, доступных для Астериска, используют какую-то дочернюю карту, которая соединяется с основной и предоставляет правильный тип канала, а это означает, что у вас есть определенная гибкость в определении того, какие типы портов у вас есть на вашей карте.

Аналоговые порты обычно не используются в средних и больших системах. Их чаще всего используют в небольших офисах (менее 10 линий, менее 30 телефонов). Ваше решение использовать аналог может основываться на следующих факторах:

- Доступность цифровых соединительных линий в вашем районе
- Стоимость (аналог дешевле при меньших плотностях, но более дорог при более высоких плотностях)
- Логистика (если у вас уже установлены аналоговые линии, вы можете их сохранить)

С технической точки зрения, вы, возможно, хотели бы иметь цифровые, а не аналоговые линии. Реальность не всегда приспосабливается, однако, аналог, вероятно, будет вокруг еще несколько лет.

Цифровая телефония

Цифровая телефония была разработана для преодоления многих ограничений аналоговой.
Некоторые из преимуществ цифровых линий включают в себя:

- Отсутствие потери амплитуды на больших расстояниях
- Снижение шума в сети (особенно междугородных линий)
- Возможность переносить более одного вызова на линию
- Более быстрая настройка и отключение вызова
- Более богатая сигнальная информация (особенно, если используется ISDN)
- Более низкая стоимость для провайдеров

- Более низкая стоимость для клиентов (при более высокой плотности)

В системе Asterisk (или любой АТС, если на то пошло) существует несколько типов цифровых линий, которые вы можете подключить:

T1 (24 канала)

Используется в Канаде и США (в основном для ISDN-PRI)

E1 (32 канала)

Используется в остальном мире (ISDN-PRI или MFC/R2)

BRI (2 канала)

Используется для схем ISDN-BRI (Euro-ISDN)

Обратите внимание, что физическая линия может быть дополнительно определена протоколом, запущенным на линии. Например, T1 может использоваться для ISDN-PRI или CAS, а E1 может использоваться для ISDN-PRI, CAS или MFC/R2. Мы обсудим различные протоколы в следующем разделе.

Установка ТфОП транков

В зависимости от установленного оборудования процесс установки ваших ТфОП-карт будет отличаться. Мы обсудим установку в общих чертах, которая будет применяться ко всем картам Digium PSTN. Другие производители, как правило, предоставляют инсталляционные скрипты с их аппаратными средствами, которые будут автоматизировать большую часть этого процесса.

Загрузка и установка DAHDI

Интерфейс аппаратного устройства Digium Asterisk, a.k.a. DAHDI (*DAW-dee*),[6] - это программная среда, необходимая для связи между ТфОП-картами и Asterisk. Даже если у вас нет оборудования ТфОП, мы рекомендуем установить DAHDI, поскольку это простой и надежный способ получения правильного источника синхронизации.[7] Полные инструкции по установке DAHDI можно найти в Главе 3.

Отключение загрузки дополнительных модулей DAHDI

По умолчанию DAHDI загрузит все скомпилированные модули в память. Поскольку это необязательно, давайте отключим загрузку любых аппаратных модулей. Если в конфигурационные файлы не загружены модули, DAHDI загрузит драйвер `dahdi_dummy`, который предоставит интерфейс для Asterisk, чтобы получать синхронизацию с ядром для корректной работы таких модулей как MeetMe и IAX2.



Начиная с DAHDI 2.3.0, требования загрузки `dahdi_dummy` для интерфейса синхронизации больше не существует. Та же функциональность теперь интегрирована в основной модуль ядра `dahdi`.

Файл конфигурации, определяющий, какие модули DAHDI будут загружаться, находится в */etc/dahdi/modules*. Чтобы отключить загрузку дополнительных модулей, все, что нам нужно сделать, это отредактировать файл *модулей* и закомментировать все модули, разместив хеш (#) в начале каждой строки. Когда все будет готово, файл конфигурации модулей должен выглядеть примерно так:

```
# Содержит список модулей для загрузки/выгрузки из /etc/init.d/dahdi.
#
# ПРИМЕЧАНИЕ: Пожалуйста добавьте/отредактируйте /etc/modprobe.d/dahdi или
# /etc/modprobe.conf если хотите добавить какие-либо параметры модуля.
```

6 Не спрашивайте.

7 Существуют и другие способы получения источника синхронизации, и если вам нужна действительно плотная система, можно запустить Asterisk без DAHDI, но это не то, что мы рассмотрим здесь.

```
#
# Формат этих файлов: список модулей, каждый на новой строке.
# Всё, что следует за '#' будет игнорироваться, как начальные и конечные
# пробелы и пустые строки.
# 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
```



Вы также можете использовать *dahdi_genconf modules* для создания правильного пустого файла конфигурации. Приложение *dahdi_genconf* будет искать вашу систему для аппаратного обеспечения, и если ничего не найдено создаст файл *modules*, который не загружает какие-либо аппаратные модули.

Затем вы можете перезапустить свой DAHDI-процесс, чтобы выгрузить все существующие драйверы, которые были загружены, и загрузить только модуль `dahdi_dummy` с помощью скрипта инициализации:

```
$ 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 ]
```

Однако, прежде чем вы сможете начать использовать свое оборудование, вам необходимо настроить файл */etc/dahdi/system.conf*; этот процесс описан в разделе «Настройка цифровых линий» и «Настройка аналоговых линий».

Настройка цифровых линий

Цифровая телефония была разработана провайдерами как способ снизить стоимость междугородных связей, а также улучшить качество передачи. Вся магистраль ТфОП была полностью цифровой уже много лет. Суть цифровой линии заключается в оцифровке звука, но цифровые линии связи также обеспечивают более сложную и надежную сигнализацию. Несколько стандартов были разработаны и развернуты, и для каждого стандарта могут быть и региональные различия.



Вы можете использовать *dahdi_hardware* и *lsdahdi*, чтобы помочь определить какое аппаратное обеспечение телефонии содержит ваша система. Вы также можете использовать *dahdi_genconf modules* для создания файла */etc/asterisk/modules* для вас на основе найденного оборудования.

PRI ISDN. Primary Rate Interface ISDN (широко известный как PRI) - это протокол, предназначенный для работы в основном на линии DS1 (T1 или E1, в зависимости от того, где вы находитесь) между провайдером и клиентом. PRI использует один из каналов DS0 в качестве канала сигнализации (называемый D-каналом). Таким образом, типичная схема PRI разбивается на группу B-каналов (каналы-носители, которые фактически несут вызовы) и D-канал для сигнализации. Хотя наиболее часто обнаруживается, что линия PRI переносится по одной физической линии (такой как T1 или E1), возможно иметь несколько каналов DS1 в цепи PRI и даже иметь несколько D-каналов.[8]

Хотя существует множество различных способов настройки линии PRI, мы надеемся избежать путаницы с вами со всеми параметрами (многие из которых устарели или, по крайней мере, больше не используются), и вместо этого предоставим примеры более общих конфигураций.



При установке аппаратного обеспечения телефонии обязательно обновите файл */etc/dahdi/modules*, чтобы включить соответствующие модули для вашего устройства, а затем перезагрузите DAHDI с помощью скрипта инициализации (*/etc/init.d/dahdi*). Вы можете использовать команду *dahdi_genconf modules* для создания файла модулей для вашей системы.

Большинство линий PRI в Северной Америке будут использовать T1 со следующими характеристиками:

- Код линии: B8ZS (биполярный с замещением 8-нулей)
- Кадрирование: ESF (расширенный суперкадр)

Вам нужно будет настроить два файла. Файл */etc/dahdi/system.conf* должен выглядеть примерно так:

```
loadzone = us
defaultzone = us

span = 1,1,0,esf,b8zs
bchan = 1-23
echocanceller = mg2,1-23
hardhdlc = 24
```

И файл */etc/asterisk/chan_dahdi.conf* должен выглядеть так:

```
[trunkgroups]

[channels]

usecallerid = yes
hidecallerid = no
callwaiting = yes
usecallingpres = yes
callwaitingcallerid = yes
threewaycalling = yes
transfer = yes
canpark = yes
cancallforward = yes
```

8 Иногда схемы ссылаются на количество B- и D-каналов, которые они содержат, поэтому один T1, использующий протокол PRI в Северной Америке, может называться 23B + D, а двойная схема T1 с резервным D-каналом будет а 46B + 2D. Мы даже видели PRI, на который ссылается как nB + nD, хотя это может стать немного педантичным.

```
callreturn = yes
echocancel = yes
echocancelwhenbridged = yes
relaxdtmf = yes
rxgain = 0.0
txgain = 0.0
group = 1
callgroup = 1
pickupgroup = 1
immediate = no

switchtype = national ; обычно называемый NI2
context = from-pstn
group = 0
echocancel = yes
signalling = pri_cpe
channel => 1-23
```

Некоторые операторы будут использовать DMS-коммутатор Nortel, который обычно использует протокол DMS100 вместо национального ISDN 2. В этом случае вы установите для параметра `switchtype` значение `DMS100`:

```
switchtype = dms100
```

Вне Канады и США линии PRI будут передаваться по стандарту E1.

В Европе стандарт E1, используемый для PRI, обычно имеет следующие характеристики:

- Код линии: CCS
- Кадрирование: HDB3 (биполярный с высокой плотностью)

Файл */etc/dahdi/system.conf* может выглядеть примерно так:

```
span = 1,0,0,ccs,hdb3,crc4
bchan = 1-15,17-31
hardhdlc = 16
```

И файл */etc/asterisk/chan_dahdi.conf* должен выглядеть примерно так:

```
[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
```

BRI ISDN. Basic Rate Interface ISDN (иногда называемый BRI, а иногда даже ISDN) должен был быть меньшим братом для PRI. BRI предоставляет только два 64-бкитных B-каналов и 16-кбитный D-канал. Использование BRI было несколько ограничено в Северной Америке (мы не рекомендуем использовать его по какой-либо причине), но в некоторых странах Европы он широко используется и почти полностью заменил аналоговый.

Поддержка BRI для Asterisk будет отличаться в зависимости от карты BRI, которую вы устанавливаете. Производитель вашей карты BRI предоставит конкретные инструкции по установке для своего оборудования.



При установке аппаратного обеспечения телефонии обязательно обновите файл */etc/dahdi/modules*, чтобы включить соответствующие модули для вашего оборудования, а затем перезагрузите DAHDI с помощью скрипта инициализации (*/etc/init.d/dahdi*). Вы можете использовать команду *dahdi_genconf modules* для создания файла *modules* для вашей системы.

MFC/R2. Протокол MFC/R2 можно рассматривать как предшественника ISDN. Сначала он использовался на аналоговых линиях, но теперь в основном используется в тех же линиях E1, которые также имеют ISDN-PRI. Этот протокол обычно не встречается в Канаде, США или Западной Европе, но он очень популярен в некоторых других частях мира (особенно в Латинской Америке и Азии), главным образом потому, что он, как правило, является менее дорогостоящим предложением от провайдеров.

Существует много разных вариантов этого протокола, каждая страна имеет свой региональный вариант.

Проект OpenR2 предоставляет библиотеку `libopenr2`, которая должна быть установлена в вашей системе, чтобы Asterisk поддерживал ваши линии R2. Однако перед установкой `libopenr2` вам необходимо установить DAHDI.

Таким образом, порядок компиляции и установки:

1. DAHDI
2. `libopenr2`
3. Asterisk

Как только OpenR2 будет установлен, вы можете использовать приложение *r2test*, чтобы просмотреть список поддерживаемых вариантов:

```
$ 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
```

Дополнительную информацию о настройке поддержки R2 в Asterisk см. В файле *configs/chan_dahdi.conf.sample*, включенном в дерево исходников Asterisk (поиск по «mfcr2»). Кроме того, OpenR2 содержит некоторые примеры файлов конфигурации для подключения Asterisk к линиям в разных странах. Чтобы прочитать информацию о некоторых вариантах стран, выполните поиск в папке */doc/asterisk* и обратитесь к документам в соответствующем подкаталоге:

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

В качестве примера OpenR2 предоставляет пример конфигурации для подключения к Telmex или Axtel в Мексике. Мы проведем вас через все шаги чтобы дать представление о процессе. Во-первых, вы должны настроить DAHDI, изменив */etc/dahdi/system.conf*, как показано здесь:

```
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
```

Затем вы должны настроить Asterisk, изменив */etc/asterisk/chan_dahdi.conf* следующим образом:

```
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
; это для целей отладки
mfcr2_logdir = log
mfcr2_logging = all
; конец конфигурации отладки
channel => 1-15
channel => 17-31
```

Настройка аналоговых линий

Есть много компаний, производящих PSTN-карты для Asterisk. Для карты должны быть установлены драйверы, чтобы Linux мог ее распознать (DAHDI поставляется с этими драйверами для карт Digium). С этого момента конфигурация обрабатывается модулем Астериска `chan_dahdi`.



Вы можете использовать *dahdi_hardware* и *lsdahdi* для определения того, какое аппаратное обеспечение телефонии содержит ваша система.
При установке аппаратного обеспечения телефонии обязательно обновите файл */etc/dahdi/modules*, чтобы включить соответствующие модули для вашего оборудования, а затем перезагрузите DAHDI с помощью скрипта инициализации (*/etc/init.d/dahdi*). Вы можете использовать команду *dahdi_genconf modules* для создания файла модулей для вашей системы.

Чтобы настроить FXO-карту для работы с Asterisk, необходимы два файла: первый не является файлом конфигурации Asterisk и поэтому находится в папке */etc/dahdi* в вашей системе.[9] Этот файл *system.conf* позволяет вам определить некоторые основные параметры, а также указать каналы, которые будут доступны вашей системе. В нашем примере используется четырехпортовая FXO-карта, но в зависимости от вашего оборудования возможно множество различных комбинаций:

```
loadzone = us     ; определяет звуки, который должен произвести интерфейс
                  ; (гудок, сигнал занятости, обратный звонок и тд.)
defaultzone = us  ; определяет tonezone по умолчанию
fxsks = 1-4       ; какие каналы на карте будут иметь эти параметры
```

Как только ваша карта и каналы известны операционной системе, вы должны настроить их для Asterisk с помощью файла */etc/asterisk/chan_dahdi.conf*:

```
[channels]

;
; Чтобы применить другие опции для этих каналов, поместите их перед "channel"
;
signalling = fxs_ks     ; в Asterisk FXO каналы используют FXS сигнализацию
                        ; (и наоборот FXS каналы используют FXO сигнализацию)
channel => 1-4          ; применяем все указанные настройки к этим каналам
```

В этом примере мы сказали Asterisk, что первые четыре канала DAHDI в системе являются портами FXO.

Расширение s. Если вы подключаетесь к ТфОП с использованием аналоговых каналов, нам нужно объяснить расширение (добавочный номер) `s`. Когда вызовы входят в контекст без специального назначения (например, вызывающая линия FXO из ТфОП), они передаются на добавочный номер `s` (`s` означает «start», так как здесь начинается вызов, если с вызовом не передавалась никакая информация о добавочном номере). Этот номер также может быть полезен для принятия вызовов, которые были перенаправлены из других частей диалплана. Например, если бы у нас был список номеров прямого входящего набора (DID), которые все были в одном месте, мы могли бы указать каждый DID на добавочный номер `s`, вместо того, чтобы иметь код для дублирования логики диалплана для каждого DID.

Поскольку это именно то, что нам нужно для нашего диалплана, давайте начнем заполнять его. Мы будем выполнять три действия по вызову (ответить на него, воспроизвести звуковой файл и повесить трубку), поэтому нашему добавочному номеру потребуется три приоритета. Мы поставим три приоритета ниже `[incoming]`, потому что мы решили, что все входящие вызовы должны начинаться в этом контексте:[10]

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

Очевидно, вы не хотели бы отвечать на звонок, а затем сразу вешать трубку. Как правило, на входящий вызов отвечает либо автоответчик, либо звонит напрямую на телефон (или группу телефонов).

---

9 Теоретически эти карты могут использоваться для любого программного обеспечения, поддерживающего DAHDI; поэтому основная конфигурация карты не входит в Asterisk.

10 В любом контексте нет ничего особенного. Мы могли бы назвать этот контекст `[stuff_that_comes_in]`, и до тех пор, пока это был контекст, назначенный в определении канала в *sip.conf*, *iax.conf*, *chan_dahdi.conf* и др., канал войдет в диалплан в этом контексте. Сказав это, настоятельно рекомендуется указывать имена ваших контекстов, которые помогут вам понять их цель. Некоторые хорошие имена контекстов могут включать в себя `[incoming]`, `[local_calls]`, `[long_distance]`, `[sip_telephones]`, `[user_services]`, `[experimental]`, `[remote_locations]` и т. д. Всегда помните, что контекст определяет, как канал входит в диалплан, поэтому укажите его соответствующим образом.

VoIP

По сравнению с обширной историей телекоммуникаций, VoIP по-прежнему является относительно молодой концепцией. В течение столетия или около того до VoIP, единственный способ подключить вас к ТфОП - это использование линий, предназначенных для этой цели вашей местной телефонной компании. В настоящее время VoIP позволяет устанавливать соединения между конечными точками без участия ТфОП (хотя в большинстве сценариев VoIP в какой-то момент все еще будет компонент ТфОП, особенно если используется традиционный телефонный номер E.164).

Как справиться с преобразованием сетевых адресов (NAT)

Если вы собираетесь использовать VoIP через любую широкополосную сеть (например Интернет), вы будете иметь дело с брандмауэрами и преобразованием сетевых адресов (NAT).[11] Основное понимание того, как протоколы SIP и RTP работают вместе для создания VoIP-вызова, может помочь в понимании и отладке функциональных проблем (таких как очень распространенная проблема «одностороннего звука», часто возникающая при настройке конфигурации NAT). NAT позволяет использовать один внешний IP-адрес несколькими устройствами за маршрутизатором.

Поскольку NAT обычно обрабатывается в брандмауэре он также является частью уровня безопасности между частной сетью и Интернетом.

VoIP-вызов с использованием SIP состоит не только из сообщений сигнализации для настройки вызова (SIP-часть соединения). Он также требует потоков RTP (медиапотоков), которые несут фактическое аудиосоединение[12], как показано на Рисунке 7-1.

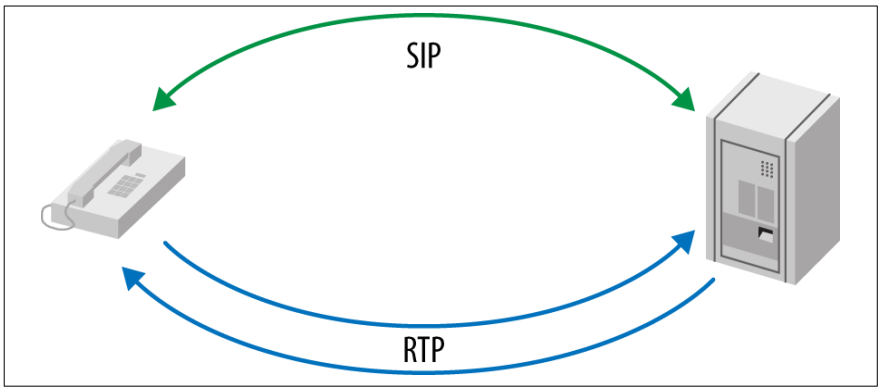


Рисунок 7-1. SIP и RTP

Использование отдельных протоколов для передачи звука - это то, что может сделать проблематичным обход NAT для VoIP-соединений, особенно если удаленные телефоны находятся за одним NAT, а АТС - за другим. Проблема связана с тем, что, хотя сигнализация SIP, как правило, может проходить через межсетевые экраны с обоих концов, потоки RTP могут не распознаваться как часть сеанса SIP, и, поэтому, они будут игнорироваться или блокироваться, как на Рисунке 7-2. Эффект блокирования одного или обоих потоков RTP отобразится в том, что пользователи будут жаловаться на то, что их звонки происходят, и можно ответить на них, но не слышно собеседника (или он не слышит).

11 Страница Википедии по трансляции сетевых адресов на самом деле неплоха. Узнайте больше о разных типах NAT и о том, как работает NAT в целом, проверьте его!

12 SIP не является единственным протоколом VoIP с использованием RTP для переноса медиапотоков.

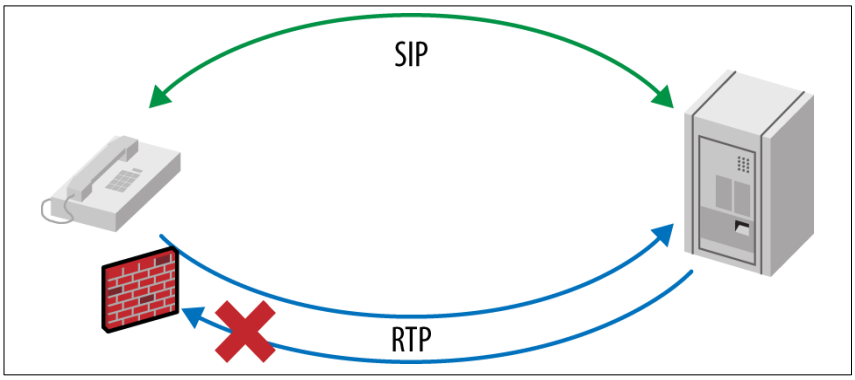


Рисунок 7-2. RTP заблокирован брандмауэром

В этом разделе мы обсудим некоторые из методов, которые вы можете использовать для устранения проблем, вызванных NAT. Существует два разных сценария, которые необходимо учитывать; каждый из которых требует определения параметров в файле *sip.conf*. Когда Вы начнете понимать, с каким сценарием вы сталкиваетесь,[13] проблемы с NAT останутся в прошлом.

Устройства за NAT

Во-первых, давайте рассмотрим устройства, расположенные за удаленным NAT, подключенным к вашему Астериск, как показано на Рисунке 7-3.

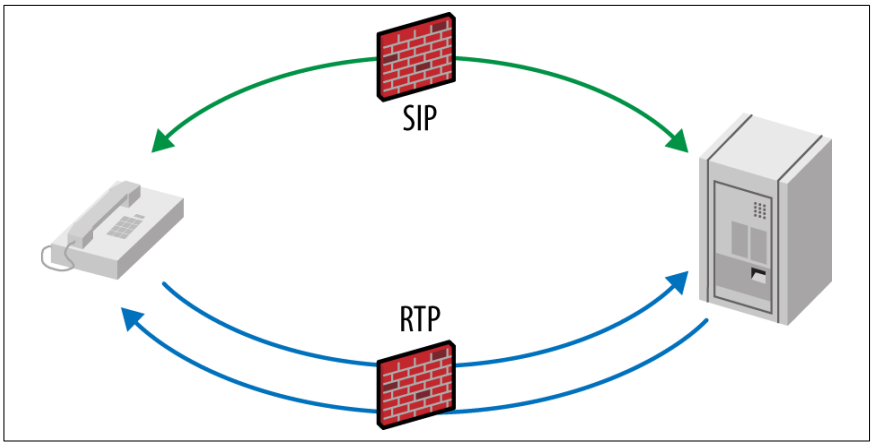


Рисунок 7-3. Удаленные устройства за NAT

Когда устройство пытается инициировать сеанс, оно составляет сообщение SIP, которое содержит его IP-адрес и некоторую дополнительную информацию. Когда Asterisk видит эту информацию, он использует ее, чтобы определить как реагировать. Поскольку устройство находится за NAT, SIP-сообщение будет иметь адрес ответа, который не будет маршрутизироваться (например, 192.168.1.104). Тем не менее, мы можем сказать Asterisk игнорировать этот адрес сообщения SIP и вместо этого использовать то, что предоставляется сетевым стеком. Мы включаем это через опцию `nat` в *sip.conf*. В Таблице 7-1 перечислены аргументы, которые мы можем установить с помощью опции `nat`.

Таблица 7-1. Аргументы доступные для nat в sip.conf

Аргумент	Описание
`no`	Не выполнять особую обработку NAT, кроме тех, что указаны в RFC 3581.[a]
`force_rport`	Даже если параметр `rport` не указан, действовать так, как если бы он был.
`comedia`	Отправляйте медиапоток обратно на порт, из которого он был получен и игнорировать запрошенный порт в заголовке SDP.
`auto_force_rport`	Если Asterisk может определить, что устройство находится за NAT, установить параметр `force_rport`. Это значение по умолчанию.
`auto_comedia`	Если Asterisk может определить, что устройство находится за NAT, установить параметр

13 Да, вы могли бы одновременно использовать оба сценария, которые в народе называют «двойным NAT».

comedia.

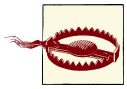
[a] http://www.ietf.org/rfc/rfc3581.txt

RFC 3581 позволяет устройству использовать параметр `rport` для передачи сигнала на дальний конец, чтобы он отвечал на IP-адрес и порт источника и отправлял запрос на него вместо адреса, указанного в заголовке SIP. Подстановка параметра `rport` может произойти, когда устройство знает, что оно находится за NAT и не может записать информацию, которая потребуется для двусторонней связи, в заголовке SIP. Asterisk всегда будет соблюдать параметр `rport`, если он установлен, но поскольку это не так часто как хотелось бы, мы можем заставить Asterisk предположить, что устройство предоставило бы параметр `rport`, если бы оно знало лучше. Делая это, мы поручаем Asterisk всегда отвечать на исходный IP-адрес и порт от которого он получил запрос. Если никакая настройка `nat` не определена явно, Asterisk по умолчанию будет устанавливать `auto_force_rport` в качестве значения параметра `nat`. Вы можете заставить делать это принудительно установив `nat=`*`force_rport`*.

Опция `comedia` (connection-oriented media) может использоваться для указания Asterisk отправки медиафайлов (RTP) на адрес и порт, из которых они поступают. Она используется, когда устройство находится за NAT и не может указать Asterisk правильное местоположение для отправки медиа.

Вы также можете указать несколько параметров для параметра `nat`, разделив аргументы запятой. Например, принято устанавливать как `force_rport`, так и `comedia` в качестве метода обработки NAT (с которого мы рекомендуем начать):

```
nat=force_rport,comedia
```



Важно отметить, что если параметр `nat` в общем разделе отличается от того, который находится в секции пира, тогда становится возможным обнаружить действительные пиры в системе. Это проблема безопасности, и очень рекомендуется указывать значения для `nat` в общем разделе. Обнаружение пиров осуществляется путем проверки системы, и эти пиры с разными настройками `nat` будут реагировать по-разному, тем самым позволяя потенциальному злоумышленнику обнаружить, какие пиры действительны в системе.

Еще один трюк при работе с устройствами за NAT - это возможность выбора опции `qualify`. Во многих системах NAT, если диалог не поддерживается периодически, то устройство, предоставляющее NAT, может закрыть соединение. Используя сигнал готовности (qualify), Астериск отправит запрос в дальний конец, на который следует ответ. По умолчанию, если вы используете `qualify=yes` для пира, время для этих транзакций составляет каждые 2000 миллисекунд (2 секунды). Вы также можете указать время в миллисекундах вместо использования `yes`:

```
qualifyfreq=60 ; зондировать конечное устройство пира каждые 60 секунд
qualify=120000 ; разрешить 10 секунд для ответа (qualify)
```

Сохранение открытости удаленного брандмауэра

Иногда возникает проблема с SIP-телефоном, при которой телефон регистрируется и функционирует при первой загрузке, но затем внезапно становится недоступным. Причина этого заключается в том, что удаленный брандмауэр, не видя активности закрывает внешнее соединение с телефоном, и, таким образом, АТС теряет возможность отправки пакетов. Эффект заключается в том, что, если АТС пытается отправить вызов на телефон, то не сможет подключиться (удаленный брандмауэр отклонит соединение). Если, с другой стороны, пользователь делает вызов, в течение нескольких минут телефон снова сможет принимать входящие вызовы. Естественно, это может вызвать путаницу у пользователей.

Относительно простое решение этой проблемы[14] включает установку таймера регистрации на удаленном телефоне на достаточно низкое значение, которое будет стимулировать подключение

14 Является ли это лучшим решением, мы еще не обсудили.

каждую минуту или около того, и таким образом убедить брандмауэр, что этому соединению может быть разрешено существовать еще немного. Это немного похоже на взлом, но вариант оказался успешным. Проблема с предложением универсального решения заключается в том, что существует множество различных моделей брандмауэров, от недорогих единиц потребительского класса до сложных пограничных контроллеров сеанса, и это одно из немногих решений, которое, по-видимому, почти полностью устраняет проблему.

Этот подход лучше всего подходит для небольших систем (менее 100 телефонов). Большая система с сотнями или тысячами телефонов не будет хорошо обслуживаться этим решением, так как будет увеличена нагрузка на систему из-за почти постоянного потока регистраций с удаленных телефонов. В таком случае потребуется более продуманное решение об общей конструкции (например, вместо Asterisk можно использовать выделенный сервер регистратора для обработки трафика регистрации).

В идеальном мире вы могли бы указать конкретную модель брандмауэра и разработать конфигурацию для тех брандмауэров, которые гарантировали бы, что ваш SIP-трафик был надлежащим образом обработан. На самом деле вы столкнетесь не только с разными моделями брандмауэров, но и разными версиями прошивки для одной и той же модели брандмауэра.

Asterisk за NAT

Второй сценарий заключается в том, что Asterisk находится за NAT, как показано на Рисунке 7-4.

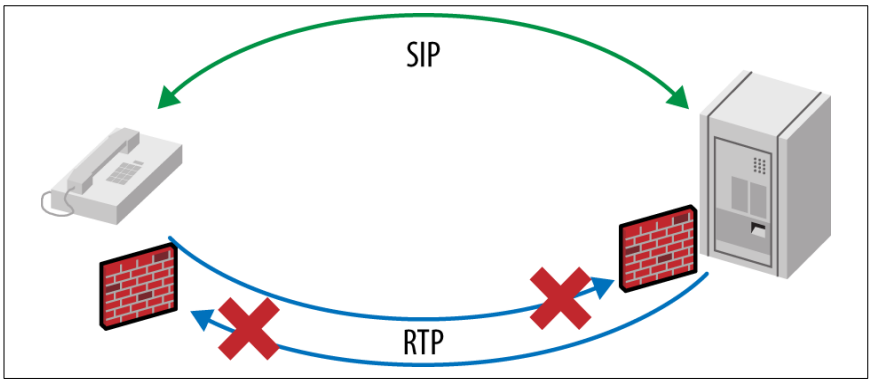


Рисунок 7-4. Астериск за NAT

В этом случае у нас есть несколько способов написать заголовки SIP в дружеской манере к другому концу.

Существуют два основных параметра, когда Asterisk находится за NAT: `externip` и `externhost`. Использование этих параметров - является решением либо-одно-либо-другое, поскольку они эффективно выполняют ту же функциональность: составляют заголовок SIP, используя внешний адрес интерфейса. Поскольку у нашего Asterisk будет немаршрутизируемый через Интернет IP-адрес (мы за NAT), мы можем указать Asterisk, каков его внешний адрес маршрутизации, чтобы разместить его в заголовках нашего SIP-сообщения.

Если наш внешний IP-адрес был чем-то вроде 98.139.183.24, мы могли бы установить `externip` следующим образом:

```
externip=98.139.183.24
```

Мы также можем указать номер порта, на который следует ответить:

```
externip=98.139.183.24:9999
```

Если номер порта не указан, будет использоваться номер порта, указанный с помощью параметра `udpbindaddr`.

В качестве альтернативы вы можете использовать опцию `externhost`, которая похожа на `externip`, но *IP-адрес будет установлен каждый раз, когда* `chan_sip.so` *загружается в память (или при*

последующих перезагрузках).[15] Формат похож на `externip`, за исключением того, что вы используете имя хоста. Вы также можете указать номер порта. Например:

```
externhost=pbx.shifteight.org:9999
```

Если вы хотите периодически обновлять внешнее имя хоста, вы можете использовать опцию `externrefresh`. Значение указано в секундах:

```
externhost=pbx.shifteight.org:5060
externrefresh=300
```

Если вы развернули сервер Asterisk за NAT вполне вероятно что к нему также будут подключены локальные телефонные аппараты. Мы не хотим искажать заголовки SIP-сообщений, которые отправляем нашим локальным аппаратам, поэтому мы можем указать, какие IP-адреса считаются частью нашей локальной сети. Таким образом, Asterisk будет отвечать просто с заголовками, как они были бы во внутренней сети. Мы используем параметр `localnet`, чтобы указать, какие сети считаются локальными для нашей системы Asterisk.



Также важно установить параметр `localnet`, даже если никакие локальные устройства не будут взаимодействовать с Asterisk, поскольку сама система Asterisk является частью локальной сети. Использование опции `localnet` в сочетании с `externip` или `externhost` поможет Asterisk понять, какие сетевые адреса нужно искать в заголовках SIP, чтобы переписать.

Опция `localnet` принимает формат IP-адреса и маски подсети. Вы можете использовать нотацию CIDR или точечную нотацию. Допускается несколько записей:

```
externip=98.139.183.24
localnet=172.16.0.0/24
localnet=192.168.100.0/255.255.255.0
```

Обработка медиапотоков (RTP)

В этом разделе мы рассмотрим, как Asterisk обрабатывает медиапотоки и наметим некоторые доступные вам параметры. Если у вас простая сетевая топология, где Asterisk подключается напрямую к ТфОП через традиционное оборудование (аналоговые или цифровые соединения), а ваши пиры находятся в одной локальной сети где и Asterisk, конфигурация по умолчанию, вероятно, прекрасна, и вы можете перейти к в следующему разделу. Если вы подключаетесь к провайдеру интернет-телефонии (ITSP) через SIP вы можете начать с установки `directmedia=no` в файле *sip.conf*. Если у вас нет причины перенаправлять медиаданные вне Asterisk, то использование `directmedia=no` обычно упрощает конфигурацию:

- Asterisk всегда будет использовать симметричный режим RTP, как определено в RFC 4961,[16] что означает, что Asterisk всегда будет отправлять пакеты из того же порта, на который он первоначально принимал их. Текст в RFC 4961 довольно короткий и полезно прочитать о том, как Asterisk будет обрабатывать RTP между конечными точками. Эти знания должны упростить ваши усилия по созданию сети, если вы планируете обрабатывать пиров вне локальной сети в любой момент в будущем.
- Если вам нужно переопределить IP-адрес, используемый для медиапотока, вы можете сделать это с помощью параметра `media_address`. Эта опция позволит вам игнорировать медиа-адрес, указанный в заголовках SDP[17], и направлять медиапоток в другое место. Использование

15 Причина, по которой мы подчеркиваем этот факт, заключается в том, что если у вас нет надежной службы DNS, вы будете испытывать всевозможные неприятные действия в Asterisk, так как он терпеливо ждет, пока имена хостов будут разрешены, в то время как вы неустанно ожидаете от драйвера канала SIP запуска работы (проблемы разрешения DNS могут вызвать всевозможные странные действия в Asterisk).

16 http://www.ietf.org/rfc/rfc4961.txt

17 Session Description Protocol - компонент сообщения SIP, который определяет параметры мультимедиа.

параметра `media_address` не может быть установлено для пира: это только общая (глобальная) опция.

- По умолчанию Asterisk будет пытаться сделать re-INVITE медиапотока непосредственно между конечными точками. Перенаправление медиа происходит, когда Asterisk определяет, что ему не нужно оставаться на пути к медиапотокам. (Времена, когда Asterisk должен находиться на медиа-пути, включают запись вызова и прослушивание DTMF.) Когда Asterisk находится за пределами сети, где конечные точки находятся за NAT, перенаправление медиа работает не очень хорошо (или вообще). В этом случае вы должны использовать опцию `directmedia=no`, чтобы предотвратить перенаправление медиапотока.

  Значение по умолчанию - `directmedia=yes`, поэтому, если у вас есть конечные точки за NAT, которые не являются частью Asterisk, вы должны установить опцию `directmedia=no`.



Изменение параметра `directmedia` влияет только на re-`INVITE` медиапотоков, а не на другие случаи re-`INVITE`, такие, как во время переговоров T.38.

Существует опция, позволяющая Asterisk перенаправлять медиапотоки между конечными точками в той же сети, что и Asterisk, по крайней мере, как это определено в ядре RTP. Включение перенаправления медиапотоков в локальной сети может осуществляться с помощью `directmedia=nonat`. Если вы хотите выполнить `UPDATE` вместо re-`INVITE` при перенаправлении медиапотока, вы можете сделать это с помощью `directmedia=update`. Если вы комбинируете `update` с `nonat` (например, `directmedia=nonat,update`), вы фактически выполняете `directmedia=yes`.

Если у вас есть пир, который, как вы знаете, сам собирается отправить Asterisk re-`INVITE` при входящем вызове, вы можете установить `directmedia=outgoing`, чтобы дать указание Asterisk не беспокоиться о попытке re-`INVITE` для медиа от этого пира (поскольку ожидается что это попытается сделать дальний конец). Установка этой опции помогает избежать ситуаций, когда оба конца одновременно пытаются перенаправить медиапоток после начальной настройки вызова (см. Рисунки 7-5 и 7-6).

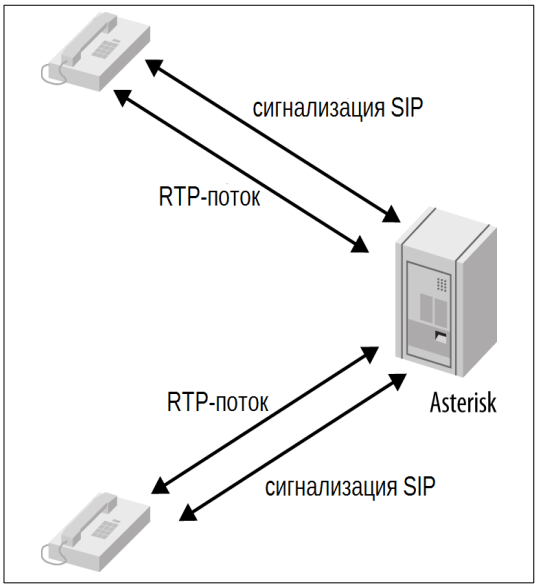


Рисунок 7-5. RTP без re-INVITE

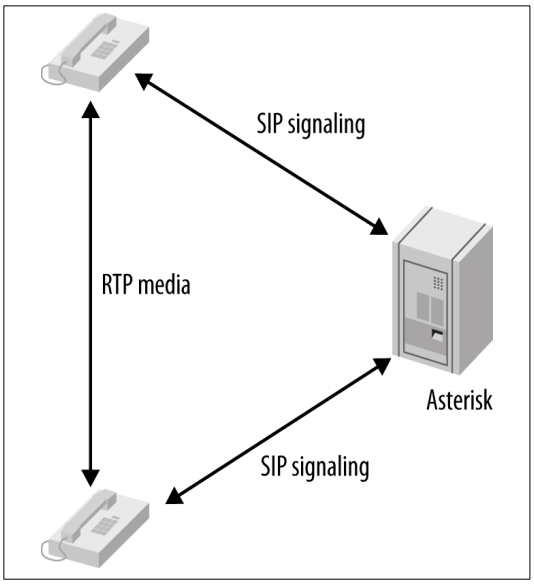


Рисунок 7-6. RTP с re-INVITE

Протокол SIP был разработан, чтобы позволить конечным точкам согласовывать возможности мультимедиа и впоследствии устанавливать прямые соединения друг с другом. Концепция заключалась в том, что серверам не нужно было обрабатывать медианагрузку для клиентов, которые имеют возможность напрямую обмениваться медиа непосредственно друг с другом (например, два телефона, участвующих в телефонном разговоре). Однако из-за проблем, присущих брандмауэрам с поддержкой NAT и другим сложным сетевым топологиям, в сочетании с невероятной вычислительной мощностью современных серверов, часто проще передавать все медиапотоки через сервер, а не справляться с проблемой re-INVITE во всех видах сетей и топологий.

Asterisk также может попытаться настроить медиапотоки непосредственно между пирами без re-INVITE, включив `directrtpsetup=yes`. Однако использование `directrtpmedia` связано с некоторыми предостережениями; он считается экспериментальным и не будет работать для видео или случаев, когда вызываемый отправляет RTP-сообщения и заголовки fmtp в `200 OK`, которые не соответствуют `INVITE` вызывающего. Кроме того, эта опция не может использоваться, когда узлы находятся за NAT.

Вы можете контролировать, какие конечные точки могут пытаться отправлять `directmedia` друг другу с помощью `directmediadeny` и `directmediapermit`. Это полезно в ситуациях, когда ваша сетевая топология не позволяет определенным сетям передавать медиа друг другу, вместо этого требуется, чтобы медиапоток протекал через Asterisk. Примером использования было бы запретить все медиа-потоки между сетями, а затем явно разрешить определенным сетям отправлять `directmedia`:

```
directmediadeny=0.0.0.0/0
directmediapermit=192.168.101.0/24
directmediapermit=172.16.1.0/24
```

Как вы можете видеть, существует несколько параметров, которые позволяют вам контролировать, как медиапотоки обрабатываются Asterisk. Такая гибкость делает Asterisk настолько привлекательным для использования во многих сетях. С правильным набором опций (и большим количеством терпения) вы должны иметь возможность обрабатывать практически любые сетевые препятствия.

Терминация ТфОП

Пока VoIP полностью не заменит ТфОП будет существовать необходимость подключать вызовы от VoIP-сетей к телефонной сети общего пользования. Этот процесс называется *терминацией (завершением)*. Это означает, что в какой-то момент шлюз, подключенный к ТфОП, должен

принимать вызовы от сети VoIP и подключать их к сети ТфОП. С точки зрения ТфОП, вызов инициирован в точке терминации.

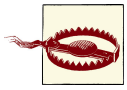
Asterisk может использоваться в качестве точки терминации ТфОП. Фактически, учитывая что Asterisk легко справляется с преобразованием протоколов, это может быть отличным использованием для системы Asterisk.

Чтобы обеспечить терминацию Asterisk должен будет обрабатывать все протоколы, которые вы хотите подключить к ТфОП. В общем, это означает, что вашему Asterisk потребуется схема PRI для обработки вызовов между ТфОП и SIP-каналами, поступающими из сети VoIP. Основополагающий принцип тот же, независимо от того, используете ли вы небольшую систему, предоставляющую соединительные линии ТфОП в офис, полный VoIP-телефонов, или сложную сеть шлюзовых машин, развернутых в стратегических сетях, предлагая терминацию тысячам абонентов.

Вызовы из сети VoIP будут поступать в диалплан в любом контексте, который вы назначили входящим SIP-каналам, а диалплан будет передавать вызовы через интерфейс ТфОП. В самом простейшем случае часть диалплана, поддерживающая терминацию, может выглядеть так:

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

На самом деле вам, скорее всего, придется выполнять более сложный план маршрутизации, который учитывает такие вещи, как география, корпоративная политика, стоимость, доступные ресурсы и т. д.



Учитывая, что большинство линий ТфОП позволят вам набирать любой номер в любом месте в мире и учитывая, что вы будете оплачивать все расходы, мы не можем подчеркнуть важность безопасности на любом шлюзовом компьютере, предоставляющем терминацию ТфОП. Преступники приложили немало усилий для взлома телефонных систем (особенно плохо защищенных систем Asterisk), и если вы не будете уделять пристальное внимание всем аспектам безопасности, вы станете жертвой мошенничества с платной подпиской. Это только вопрос времени.
Не допускайте никаких незащищенных VoIP-соединений в любой контекст, который содержит терминацию ТфОП.

Инициирование ТфОП

Очевидно, что если вы хотите передавать вызовы из вашей VoIP-сети в ТфОП, вы также можете иметь возможность принимать вызовы из ТфОП в вашу VoIP-сеть. Процесс выполнения этого обычно называют *инициированием* (*оригинацией)*. Это просто означает, что вызов инициирован в ТфОП.

Для обеспечения инициирования требуется номер телефона. Поэтому вам нужно будет получить линию от вашей местной телефонной компании, которую вы подключите к вашей системе Asterisk. В зависимости от того, где вы находитесь в мире, существует несколько различных типов линии, которые могут обеспечить эту функциональность, от базовой аналоговой линии POTS до конвейера несущей SS7.



Телефонные номера, используемые для целей инициирования, обычно называются номерами прямого входящего набора (DID). Это не относится строго ко всем ситуациям (например, номер телефона на традиционной аналоговой линии не будет считаться DID), но термин достаточно полезен, чтобы прижиться. Исторически DID ссылался на номер телефона, связанный с транком, подключенным к оборудованию клиента (CPE).

Поскольку номера телефонов контролируются традиционной телекоммуникационной отраслью, вам нужно будет получить номер либо от поставщика напрямую, либо от одной из многих компаний, которые покупают номера оптом и перепродают их в меньших блоках. Если вы получите линию, такую как PRI, вы, как правило, сможете заказать номера DID, которые будут подключены по этой линии.

Чтобы принять вызов из сети, которую вы используете для инициирования, вам, как правило, придется обрабатывать передачу номера телефона, который был вызван. Это связано с тем, что соединительные линии ТфОП обычно могут обрабатывать более одного номера телефона, и поэтому поставщику необходимо определить, какой номер был вызван, чтобы ваша система Asterisk знала, как маршрутизировать вызов. Номер, который был набран, обычно называется номером сервиса предоставления информации о набранном номере (Dialed Number Information Service - DNIS). Номер DNIS и DID не должны совпадать,[18] но как правило будут. Если вы заказываете линию у поставщика, вам нужно попросить, чтобы он отправил DNIS (если он этого не понимает, вы можете рассмотреть другого поставщика).

В диалплане вы связываете входящую линию с контекстом, который будет знать, как обрабатывать входящие номера. Например, он может выглядеть примерно так:

```
[from-pstn]
; Этот контекст будет указан в файле конфигурации для схемы
; (например chan_dahdi.conf)

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

[number-mapping]
; Этот контекст не является строго обязательным, но он облегчит
; отслеживание ваших DID в одном месте в вашем диалплане.
; Отсюда Вы можете передать вызов другой части диалплана,
; где будет выполняться фактическая обработка в диалплане

exten => 4165551234,1,Dial(SIP/0000FFFF0001)
exten => 4165554321,1,Goto(autoattendant-context,start,1)
exten => 4165559876,1,VoiceMailMain()     ; удобный черный ход для
                                          ; прослушки голосовых сообщений
exten => i,1,Verbose(2,Incoming call to invalid number)
```

В контексте `number-mapping` вы явно перечисляете все DID, которые вы планируете обрабатывать, а также недопустимый обработчик для любых DID, которые не указаны (вы можете отправить недопустимые числа на ресепшен или автосекретарю или в какой-то контекст который воспроизводит оповещение о неверном назначении).

VoIP в VoIP

В конце концов, потребность в ТфОП, скорее всего, исчезнет и большинство голосовых коммуникаций будет происходить по сетевым соединениям.

Первоначальное представление, лежащее в основе протокола SIP, состояло в том, что он должен был быть протоколом одноранговой сети. Технически это все еще так. Тем не менее, такие вопросы, как безопасность, конфиденциальность, корпоративные политики, интеграция, централизация и т.д. сделали вещи немного более сложными, чем просто размещение URI в SIP-телефон и возможность иметь телефон SIP где-то в другом месте.

18 В традиционных АТС целью DID было разрешить подключение непосредственно к внутреннему номеру в офисе. Многие АТС не могли поддерживать такие понятия, как перевод чисел или гибкие длины цифр, и, таким образом, поставщику пришлось передавать внутренний номер в виде номера DID, а не номера, который был набран (номер DNIS). Например, номер телефона 416-555-1234, возможно, был отображен на добавочный номер 100, и, таким образом, оператор отправил цифры 100 в АТС вместо DNIS 4165551234. Если вы когда-либо заменяли старую АТС системой Астериск , вы можете найти этот перевод на месте, и вам нужно будет получить список сопоставлений между номерами, которые набираются абонентом, и номерами, которые отправляются в АТС. Также широко распространено мнение, что носитель передает только последние четыре цифры номера DNIS, который затем АТС преобразует во внутренний номер.

Неавторизованные вызовы через SIP

Как мы отметили в Главе 5, вы можете настроить контекст `unauthenticated` в разделе `general` файла *sip.conf* и установить `allowguest=yes`. Делая это, вы разрешаете неаутентифицированные звонки в ваш диалплан. Это требование справедливо и когда вы хотите принимать вызовы так же, как и по электронной почте. (SIP URI будет похож на *sip:leif@shifteight.org*, который похож на *mail:leif@shifteight.org*.) После того, как вы настроили свой неаутентифицированный контекст в *extensions.conf*, просто включите все, что вы хотели бы сделать общедоступным из внешнего мира; или создайте новый контекст для включения в ваш неаутентифицированный контекст. Например, вы можете создать `GoSub()`, который может искать электронные адреса и внутренние номера пользователей в вашей базе данных или сервере LDAP и маршрутизировать вызовы на основе SIP URI, которые соответствуют их адресам электронной почты.

Протокол SIP стал раздутым и сложным. Внедрение систем и сетей на базе SIP, возможно, станет еще более сложным, чем внедрение традиционных телефонных АТС и сетей.[19]

Мы не собираемся заниматься сложностями проектирования и реализации VoIP-сетей в этой книге, но мы обсудим некоторые из способов, которыми вы можете настроить Asterisk для поддержки VoIP-подключения к другим системам VoIP.

Настройка транков VoIP

В Asterisk нет необходимости явно устанавливать ваши VoIP-модули (если по какой-то причине вы не скомпилировали Asterisk с необходимыми модулями). Существует несколько протоколов VoIP, которые вы можете использовать с Asterisk, но мы сосредоточимся на двух самых популярных: SIP и IAX.

Настройка SIP-транков между Asterisk-системами

SIP - это самый популярный VoIP-протокол - настолько, что многие люди считают, что другие протоколы VoIP устарели (это не так, но нельзя отрицать, что SIP доминирует в VoIP уже много лет).

Протокол SIP является одноранговым и на самом деле не имеет формальной спецификации транка. Это означает, что если вы подключаете один телефон к вашему серверу или соединяете два сервера вместе, SIP-соединения будут похожи.

Соединение двух систем Asterisk посредством SIP. Возможность одновременного соединения двух систем Asterisk, позволяющих осуществлять вызовы между ними, является довольно распространенным требованием. Возможно, у вас есть компания с двумя офисами и вы хотите иметь АТС в каждом из них, или, может быть, вы являетесь администратором УАТС компании, и вам нравится Asterisk так сильно, что вы также хотели бы установить его дома. В этом разделе приводится краткое руководство по настройке двух серверов Asterisk для передачи вызовов друг другу по протоколу SIP. В нашем примере мы будем ссылаться на два сервера как `serverA` и `serverB`.

Первый файл, который необходимо изменить, - */etc/asterisk/sip.conf*. Это основной конфигурационный файл для настройки учетных записей SIP. Во-первых, эта запись должна быть добавлена в *sip.conf* на `serverA`. Он определяет SIP-пиров для другого сервера:

```
[serverB]

;
; Укажите тип учетной записи SIP как 'peer'. Это означает, что входящие
; вызовы будут сопоставляться по IP-адресу и номеру порта. Таким образом,
```

19 На рынке существует много проприетарных систем АТС, которые имеют базовую конфигурацию, которая будет работать прямо из коробки. Развертывание Asterisk является гораздо более гибким, но редко бывает простым.

```
; когда Астериск получает вызов от 192.168.1.102 и стандартный SIP порт 5060,
; он будет соответствовать этой записи в sip.conf. Затем он попросит
; аутентификацию и ожидает что пароль будет соответствовать, указанному
; здесь в поле 'secret'.
;
type = peer
;
; Это IP-адрес удаленного сервера (serverB). Эта опция также может быть
; представлена как имя хоста.
;
host = 192.168.1.102
;
; Когда мы отправляем вызовы этому SIP-пиру и предоставляем аутентификацию
; мы используем 'serverA' как наше имя пользователя по умолчанию.
;
defaultuser = serverA
;
; Это общий с serverB пароль. Он будет использоваться в качестве
; пароля при принятии вызова от serverB, либо при отправке вызова на serverB.
;
secret = apples
;
; При получении вызова от serverB сопоставляйте его с внутренними номерами
; в контексте «incoming» из extensions.conf.
;
context = incoming
;
; Начните с запрета списка разрешенных кодеков.
;
disallow = all
;
; Разрешите только кодек ulaw.
;
allow = ulaw
```



Обязательно измените параметр `host`, чтобы он соответствовал IP-адресу для вашей собственной настройки.

Теперь добавьте следующую запись в */etc/asterisk/sip.conf* на `serverB`. Она почти идентична содержимому записи, помещенной на `serverA`, но имя пира и IP-адрес будут изменены:

```
[serverA]

type = peer
host = 192.168.1.101
defaultuser = serverB
secret = apples
context = incoming
disallow = all
allow = ulaw
```

На этом этапе вы должны убедиться, что конфигурация была успешно загружена в Астериск с использованием некоторых команд CLI. Первая команда, которую нужно попробовать, это sip show peers. Как следует из названия, в нем будут показаны все настроенные SIP-узлы:

```
*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]
```



Вы также можете попробовать *sip show peer serverB*. Эта команда покажет гораздо больше деталей.

Последним шагом в настройке SIP-вызовов между двумя серверами Asterisk является изменение диалплана в файле */etc/asterisk/extensions.conf*. Например, если вы хотите, чтобы любые вызовы, сделанные с `serverA` на внутренние номера с 6000 по 6999 поступали на `serverB`, вы должны использовать эту строку в диалплане:

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

Подключение системы Asterisk к SIP провайдеру. Когда вы подключаетесь к SIP-провайдеру, у вас может быть служба для отправки и/или приема телефонных звонков. Конфигурация будет немного отличаться в зависимости от вашего использования SIP-провайдера. Кроме того, конфигурация будет отличаться для каждого провайдера. В идеале SIP-провайдер, к которому вы подключены, предоставит вам примеры конфигурации Asterisk, которые помогут вам подключиться как можно быстрее. В случае, если он этого не сделает, мы попытаемся дать вам общую настройку, которая поможет вам начать работу.

Если вы будете получать звонки от провайдера, то он, скорее всего, потребует, чтобы ваш сервер зарегистрировался на одном из его серверов. Для этого вы должны добавить строку регистрации в раздел `[general]` в */etc/asterisk/sip.conf*:

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

Затем вам нужно будет создать запись пира в *sip.conf* для вашего провайдера. Вот пример записи для пира:

```
[myprovider]

type = peer
host = your.provider.tld
defaultuser = username
secret = password
; Большинство провайдеров не будут аутентифицироваться, когда отправляют вам
; звонки, поэтому вам нужна эта строка, чтобы просто принять их вызовы.
insecure = invite
dtmfmode = rfc2833
disallow = all
allow = ulaw
```

Теперь, когда учетная запись определена, вы должны добавить некоторые расширения в диалплан, чтобы отправлять звонки своему провайдеру:

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

Шифрование SIP-вызовов. Asterisk поддерживает TLS для шифрования SIP-сигнализации и Secure Realtime Transport Protocol (SRTP) для шифрования медиапотоков телефонного вызова. В этом разделе мы будем настраивать вызовы с использованием SIP TLS и SRTP между двумя серверами Asterisk. Первым шагом является обеспечение правильных зависимостей. Убедитесь, что установлены как OpenSSL, так и LibSRTP. Если один из них не был установлен, переустановите Asterisk после установки этих зависимостей чтобы включить поддержку TLS и SRTP. После завершения убедитесь, что модуль `res_srtp` был скомпилирован и установлен. Чтобы установить

OpenSSL, необходим пакет `openssl-devel` на RHEL и `libssl-dev` на Ubuntu. Для установки LibSRTP пакет будет `libsrtp-devel` на RHEL и `libsrtp0-dev` на Ubuntu.

Затем мы настроим SIP TLS. Вы должны включить TLS, используя глобальную опцию `tlsenable` в разделе `[general]` в */etc/asterisk/sip.conf* на обоих серверах. Вы можете указать адрес для привязки, если хотите ограничить прослушивание TLS-соединений одним IP-адресом в системе. В этом примере у нас есть адрес подкаталога IPv6, указанный для разрешения соединений TLS на всех адресах IPv4 и IPv6 в системе:

```
[general]

tlsenable = yes
tlsbindaddr = ::
```

Следующий шаг - получить сертификаты. В целях демонстрации конфигурации и функциональности мы собираемся создавать самоподписанные сертификаты, используя вспомогательный скрипт, распространяемый вместе с Asterisk. Если вы устанавливаете его в производственной среде, вы можете не захотеть использовать самоподписанные сертификаты. Однако, если вы это сделаете, есть несколько приложений, которые помогают упростить управление вашим собственным центром сертификации (Certificate Authority - CA), например TinyCA.

Сценарий, который мы собираемся использовать, - *ast_tls_cert*, который находится в каталоге */contrib/scripts* дерева исходников Asterisk. Нам нужно создать сертификат CA и два сертификата сервера. Первый вызов *ast_tls_cert* будет генерировать сертификат CA и серверный сертификат для `serverA`. Второй вызов *ast_tls_cert* будет генерировать серверный сертификат для `serverB`:

```
$ 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
```

Теперь, когда сертификаты созданы, их необходимо перенести в соответствующие места на `serverA` и `serverB`. Мы будем использовать каталог */var/lib/asterisk/keys* для хранения сертификатов. Переместите следующие файлы на `serverA`:

- *ca.crt*
- *serverA.pem*

И переместите эти файлы на `serverB`:

- *ca.crt*
- *serverB.pem*

Имея сертификаты, мы можем завершить настройку Asterisk. Нам нужно указать Asterisk на сертификат сервера, который мы только что создали. Поскольку мы используем самоподписанные сертификаты, нам также нужно указать сертификат CA. В разделе `[general]` файла */etc/asterisk/sip.conf* на `serverA` добавьте следующие параметры:

```
[general]

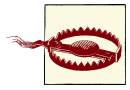
tlscertfile = /var/lib/asterisk/keys/serverA.pem
tlscafile = /var/lib/asterisk/keys/ca.crt
```

Внесите те же изменения в sip.conf на serverB:

```
[general]

tlscertfile = /var/lib/asterisk/keys/serverB.pem
```

```
tlscafile = /var/lib/asterisk/keys/ca.crt
```



Когда вы создаете сертификат сервера, поле «Common Name» должно соответствовать имени хоста сервера. Если вы используете скрипт *ast_tls_cert*, это значение задается с параметром `-C`. Если при выполнении вызова возникает проблема с проверкой сертификата сервера, вам может потребоваться исправить поле «Common Name». В качестве альтернативы, для тестирования вы можете установить для параметра `tlsdontverifyserver` значение `yes` в разделе `[general]` в файле */etc/asterisk/sip.conf*, и Asterisk разрешит вызов даже если он не сможет проверить сертификат сервера.

В разделе «Соединение двух систем Asterisk посредством SIP» мы создали конфигурацию, необходимую для передачи вызовов между серверами. Теперь мы собираемся изменить эту конфигурацию, чтобы Asterisk знал, что вызовы между двумя серверами должны быть зашифрованы. Единственное изменение, которое необходимо сделать, это добавить параметр `transport=tls` в запись пира для другого сервера.

На `serverA`:

```
[serverB]

type = peer
host = 192.168.1.102
defaultuser = serverA
secret = apples
context = incoming
disallow = all
allow = ulaw
transport = tls
```

На `serverB`:

```
[serverA]

type = peer
host = 192.168.1.101
defaultuser = serverB
secret = apples
context = incoming
disallow = all
allow = ulaw
transport = tls
```

Теперь, когда вы совершаете вызов с использованием `Dial(SIP/serverA)` или `Dial(SIP/serverB)`, сигнализация SIP будет зашифрована. Вы можете изменить диалплан, чтобы заставить исходящие вызовы иметь зашифрованную сигнализацию, установив для функции `CHANNEL(secure_bridge_signaling)` значение `1`:

```
[default]

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

На стороне принимающей вызов, вы можете проверить зашифрована ли сигнализация входящего вызова с помощью функции диалплана `CHANNEL(secure_signaling)`. Рассмотрим следующий пример диалплана:

```
[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()
```

Когда вызов отправляется с `serverA` на `serverB` с помощью этой конфигурации, вы можете видеть из вывода в консоли Asterisk, что диалплан определяет, что передача сигнала входящего вызова шифруется:

```
   -- 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
```

Теперь, когда SIP TLS настроен для вызовов между `serverA` и `serverB`, мы настроим SRTP так, чтобы медиапотоки, связанные с вызовом, также были зашифрованы. К счастью, это довольно легко настроить, по сравнению с тем, что требовалось для работы SIP TLS. Во-первых, убедитесь, что у вас есть модуль `res_srtp`, загруженный в Asterisk:

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

Чтобы включить SRTP, установите для функции CHANNEL(secure_bridge_media) значение `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)
```

Это означает, что для исходящего вызова требуется зашифрованный медиапоток. Когда вызов отправляется через SIP, Asterisk потребует использования SRTP или вызов завершится с ошибкой.

Используя все эти инструменты, вы можете гарантировать, что вызовы между двумя серверами Asterisk будут полностью зашифрованы. Те же методы должны применяться для шифрования вызовов между Asterisk и SIP-телефоном.

Функции диалплана предоставляют механизм для проверки состояния шифрования входящего вызова и принудительного шифрования исходящего вызова. Однако, имейте в виду, что эти инструменты предоставляют только средства для управления шифрованием для одного хопа пути вызова. Если вызов проходит через несколько серверов, эти инструменты не гарантируют, что вызов зашифрован по всему пути вызова. Важно внимательно рассмотреть какие требования необходимы для безопасных вызовов, и предпринять все необходимые шаги для обеспечения соблюдения этих требований на протяжении всего пути вызова. Безопасность сложная и тяжелая работа.

Настройка IAX-транков между Asterisk-системами

Протокол Inter-Asterisk eXchange 2 версии (наиболее известный как IAX), является собственным VoIP-протоколом Asterisk. Он отличается от SIP тем, что сигнализация и медиапотоки переносятся в одном и том же соединении. Это различие является одним из преимуществ протокола IAX, поскольку он значительно упрощает работу IAX для работы с NAT-соединениями.

IAX транкинг. Одной из наиболее уникальных особенностей протокола IAX является IAX-транкинг. Транкинг соединения IAX может быть полезен на любом сетевом соединении, которое часто будет выполнять несколько одновременных VoIP-вызовов между двумя системами. Инкапсулируя несколько аудиопотоков в один пакет, IAX-транкинг сокращает накладные расходы при подключении к данным, что позволяет сэкономить полосу пропускания на сильно загруженном соединении.

Шифрование IAX. Основным преимуществом шифрования IAX является то, что для файла */etc/asterisk/iax.conf* требуется одно простое изменение:

```
[general]

encryption = yes
```

Для дополнительной защиты вы можете установить следующий параметр, чтобы гарантировать отсутствие соединения IAX без шифрования:

```
forceencryption = yes
```

Оба эти параметра можно указать в разделе `[general]`, а также в разделах `peer/user/friend` в *iax.conf*.

Экстренный набор

В Северной Америке люди привыкли к тому, чтобы набирать номер 911 для получения экстренной помощи. За пределами Северной Америки общеизвестные номера службы экстренной помощи - 112 и 999. Если вы предоставляете свою систему Asterisk людям, вы обязаны (во многих случаях регламентируется) обеспечить, чтобы звонки могли идти в службы экстренной помощи с любого подключенного телефона (даже с телефонов, которые ограничиваются выполнением вызовов).

Одна из существенных частей информации, которую должна знать организация реагирования на чрезвычайные ситуации - это место чрезвычайной ситуации (например, куда отправлять пожарные машины). В традиционном ТфОП транке эта информация уже известна поставщику услуг и затем передается вместе с Public Safety Answering Point (PSAP). С помощью VoIP-схем все может усложниться, поскольку VoIP-линии физически не привязаны к географическому положению.

Вам необходимо убедиться что ваша система будет правильно обрабатывать вызовы 911 с любого телефона, подключенного к ней, и вам нужно сообщить, что доступно вашим пользователям. Например, если вы разрешаете пользователям регистрироваться в системе с софтфонов на своих ноутбуках; что происходит, если они находясь в гостиничном номере в другой стране, набирают номер 911?[20]

Диалплан для обработки экстренных вызовов не должен быть сложным. На самом деле, будет гораздо лучше, если сохранить его простым. Люди часто испытывают соблазн внедрить всевозможные причудливые функции в частях своих диалпланов для аварийных служб, но если ошибка в одной из ваших причудливых функций приводит к сбою в экстренном вызове, жизнь может быть подвержена риску. Это не место для игры. Раздел `[emergency-services]` вашего диалплана может выглядеть примерно так:

```
[emergency-services]
exten => 911,1,Goto(dialpsap,1)
exten => 9911,1,Goto(dialpsap,1)    ; некоторые люди наберут '9', потому что
                                    ; привыкли так делать с УАТС
exten => 999,1,Goto(dialpsap,1)
exten => 112,1,Goto(dialpsap,1)

exten => dialpsap,1,Verbose(1,Call initiated to PSAP!)
```

20 Не думайте, что этого не может случиться. Когда кто-то звонит 911, это потому, что у них случилась чрезвычайная ситуация, и небезопасно предположить, что они будут в адеквтном состоянии.

```
    same => n,Dial(${LOCAL}/911)    ; ЗАМЕНИТЕ 911 НА ДРУГОЙ
                                    ; СООТВЕТСТВУЮЩИЙ ВАШЕЙ МЕСТНОСТИ

[internal]
include => emergency-services ; это должно быть в любом контексте
                              ; в котором есть пользователи
```

В тех контекстах, где вы знаете, что пользователи не находятся на месте (например, удаленные пользователи со своими ноутбуками), что-то вроде этого может быть лучше:

```
[no-emergency-services]
exten => 911,1,Goto(nopsap,1)
exten => 9911,1,Goto(nopsap,1) ; для людей, набирающих '9' для внешних вызовов
exten => 999,1,Goto(nopsap,1)
exten => 112,1,Goto(nopsap,1)

exten => nopsap,1,Verbose(1,Call initiated to PSAP!)
    same => n,Playback(no-emerg-service) ; вам нужно записать это приглашение

[remote-users]
include => no-emergency-services
```

В Северной Америке правила обязывали многих VoIP-операторов предлагать то, что широко известно как *E911*.[21] Когда вы подписываетесь на эти услуги, они потребуют адресную информацию для каждого DID, который вы хотите связать с исходящими вызовами. Затем эта адресная информация будет отправлена в PSAP, соответствующий этому адресу, и ваши вызовы с чрезвычайной ситуацией должны обрабатываться так же, как если бы они были набраны на традиционной линии ТфОП.

Суть в том, что вам нужно убедиться, что созданная вами телефонная система позволяет совершать экстренные вызовы.

Вывод

Мы считаем, что ТфОП, в конце концов, полностью исчезнет. Однако, прежде чем это произойдет, потребуется широко распространенный и надежный механизм распределения, позволяющий организациям и частным лицам публиковать адресную информацию, чтобы их можно было найти. Мы рассмотрим некоторые из способов, которые уже возможны в Главе 12.

21 На самом деле это предлагает не оператор, а скорее это возможность PSAP. E911 также используется на ТфОП транках, но поскольку она происходит без какого-либо участия с вашей стороны (поставщики ТфОП делают бумажную работу за вас), вы, как правило, не знаете, что у вас есть E911 на ваших местных линиях.

ГЛАВА 8

Голосовая почта

Просто оставьте сообщение, может быть, я позвоню.
—Джо Уолш

До того, как электронная почта и обмен мгновенными сообщениями стали повсеместными, голосовая почта была популярным методом электронных сообщений. Несмотря на то, что большинство людей предпочитают текстовые системы обмена сообщениями, голосовая почта остается важным компонентом любой АТС.

Комедийная почта

Одной из самых популярных (или, возможно, непопулярных) функций любой современной телефонной системы является голосовая почта. Asterisk имеет достаточно гибкую систему голосовой почты с именем Comedian Mail.[1] Голосовая почта в Asterisk предоставляется в диалплане модулем `app_voice mail.so`.

Некоторые из функций системы голосовой почты Asterisk включают в себя:

- Неограниченные, защищенные паролем ящики голосовой почты, каждый из которых содержит папки почтовых ящиков для организации голосовой почты
- Различные приветствия для состояния «занято» и «недоступен»
- Встроенные и пользовательские приветствия
- Возможность связывать телефоны с несколькими почтовыми ящиками и почтовые ящики с несколькими телефонами
- e-mail уведомления о голосовой почте, с голосовой почтой, необязательно вложенной как звуковой файл[2]
- Переадресация и трансляция голосовой почты
- Индикатор ожидающего сообщения (мигающий идникатор или зашифрованный тон) на многих типах телефонов
- Каталог сотрудников компании на основе ящиков голосовой почты

И это лишь верхушка айсберга!

Для версии конфигурации файла по умолчанию */etc/asterisk/voicemail.conf* требуется несколько настроек, чтобы обеспечить конфигурацию, подходящую для большинства ситуаций.

1 Это название - больше игра слов, частично вдохновлено системой голосовой почты Nortel Meridian Mail.
2 Нет, вам действительно не нужно платить за это - и да, это действительно работает.

Сначала мы рассмотрим различные параметры, которые вы можете определить в файле *voicemail.conf*, а затем мы предоставим образец файла конфигурации с настройками, которые мы рекомендуем для большинства развертываний.

Файл *voicemail.conf* содержит несколько разделов, в которых параметры могут быть определены. В следующих разделах описаны все доступные опции.

Раздел [general]

Первый раздел - [general], позволяет вам определять глобальные настройки для вашей системы голосовой почты. Доступные параметры перечислены в Таблице 8-1.

Таблица 8-1. раздел опций [general] для for voicemail.conf

Опция	Значение/Пример	Примечание
`format`	`wav49|gsm|wav`	Для каждого из перечисленных форматов Asterisk создает отдельную запись в этом формате всякий раз, когда оставляется сообщение. Преимущество состоит в том, что некоторые шаги транскодирования могут быть пропущены, если сохраненный формат совпадает с кодеком, используемым на канале. Нам нравится WAV, потому что это самое высокое качество, и WAV49, потому что он славно сжат и легко отправляется по электронной почте. Нам не нравится GSM из-за его колющего звука, но он пользуется некоторой популярностью.[a]
`serveremail`	`user@domain`	Когда письмо отправляется из Asterisk, это адрес электронной почты, который, по-видимому, будет в поле «От:».[b]
`attach`	`yes,no`	Если для почтового ящика указан адрес электронной почты, это определяет, прикреплено ли голосовое сообщение к электронной почте (если нет, отправляется сообщение с простым оповещением).
`maxmsg`	`9999`	По умолчанию Asterisk позволяет хранить не более 100 сообщений
`maxsecs`	`0`	Этот тип настроек был полезен, когда большая система голосовой почты могла иметь только 40 МБ[c] памяти: необходимо было ограничить систему, потому что было легко заполнить жесткий диск. Этот параметр может раздражать вызывающих абонентов (хотя это заставляет их дойти до сути, поэтому некоторым людям он нравится). В настоящее время, когда терабайтные приводы распространены, нет никаких технических причин для ограничения длины сообщения. Два соображения: 1) если канал попадает в почтовый ящик, полезно установить какую-то ценность, чтобы он не оставался там в течение нескольких дней, но 2) если пользователь хочет использовать свой почтовый ящик для записи заметок себе, он не будет оценить, если вы отключите её через 3 минуты. Вероятно, настройка будет составлять от 600 секунд (10 минут) до 3600 секунд (1 час).
`minsecs`	`4`	Многие люди будут класть трубку, а не оставлять сообщение, когда они звонят кому-то и получают голосовую почту. Иногда это происходит после начала записи, поэтому владелец почтового ящика получает раздражающее 2-секундное сообщение о том, что кто-то положил трубку. Этот параметр гарантирует, что Asterisk будет игнорировать сообщения, которые короче заданной минимальной длины. Вы должны позаботиться о том, чтобы не устанавливать слишком высокое значение, потому что тогда сообщение «Эй, это я, позвоните мне» (что можно сказать менее чем за 1 секунду) потеряется, а вы будете получать жалобы на исчезновение сообщений. Кажется, что три секунды — верный результат. Чтобы препятствовать людям оставлять ультракороткие сообщения, которые могут быть отброшены, ваше приветствие может потребовать от абонентов идентифицировать себя и

Опция	Значение/Пример	Примечание
		оставить некоторую информацию о том, что они хотели.
maxgreet	1800	Вы можете определить максимальную длину приветствия, если хотите. Опять же, поскольку хранение не является проблемой, и установка слишком низкого значения будет раздражать ваших более многословных пользователей, мы предлагаем установить его на максимум и позволить пользователям определить подходящую длину для себя.
skipms	3000	При прослушивании сообщений пользователи могут пропустить вперед или назад, нажав (по умолчанию) * и #. Этот параметр указывает длину скачка (в миллисекундах).
maxsilence	5	Этот параметр определяет максимальное время, в течение которого вызывающий абонент может оставаться молчащим до остановки записи. Нам нравится устанавливать этот параметр на 1 секунду дольше, чем `minsecs` (если вы установите его равным или больше `minsecs`, вы получите предупреждение о том, что «maxsilence должно быть меньше minsecs или вы можете получать пустые сообщения»). Это значение, наиболее полезно, когда у вас есть аналоговые транки, поскольку разъединение дальнего конца может быть хитрым на таких линиях. При использовании любой цифровой линии (PRI или VoIP) сообщение о разъединении на дальнем конце будет довольно аккуратно обрабатывать конец сообщения, и на самом деле вам может потребоваться увеличить это значение до 10 секунд, чтобы уменьшить вероятность того, что тихие люди случайно отключатся в середине сообщения.
silencethreshold	128	Этот параметр позволяет точно настроить чувствительность к тишине предыдущего параметра, `maxsilence`. Допустимые значения от `0` до `32767`. Значение по умолчанию - `128`. Значение помогает `app_voicemail` решить, какую амплитуду использовать в качестве контрольной точки относительно того, что она будет рассматривать как «тишину». Поскольку это линейное значение, а амплитуда должна быть надлежащим образом рассматриваема в децибелах (которые являются логарифмическими), мы не рекомендуем настраивать этот параметр, если у вас нет четкого понимания амплитуды, децибел, логарифмических масштабов, программирования на C, обработки цифрового сигнала и т. д. Если у вас проблемы со звуком в вашей системе, это не первое место чтобы попытаться их решить.
maxlogins	3	Эта небольшая функция безопасности предназначена для того, чтобы атаки брут-форсом на ваши пароли почтовых ящиков были более трудоемкими. Если неверный пароль получен много раз, голосовая почта будет отключаться, и вам придется перезвонить, чтобы повторить попытку. Обратите внимание, что это не заблокирует почтовый ящик. Терпеливые детективы могут продолжать пытаться войти в ваш почтовый ящик столько раз, сколько захотят, им просто нужно будет перезвонить каждую третью попытку. Если у вас много пользователей с толстыми пальцами, вы можете установить например `5`.
moveheard	yes	Этот параметр переместит прослушиваемые сообщения в папку *Old*. Мы рекомендуем оставить по умолчанию.
forward_urgent_auto	no	Установка этого параметра в `yes` будет сохранять исходную настройку срочности любых сообщений, которые пользователь получает, а затем пересылает далее. Если вы оставите его без ответа, пользователи могут сами установить уровень срочности в сообщениях, которые они пересылают.
userscontext	default	Если вы используете файл *users.conf* (мы этого не делаем), вы можете определить здесь контекст, где регистрируются записи.
externnotify	/путь/к/скрипту	Если вы хотите запускать внешнее приложение, когда сообщение

Опция	Значение/Пример	Примечание
		остается, вы можете определить его здесь.
smdienable	no	Если вы используете Астериск в качестве сервера голосовой почты на АТС, которая поддерживает SMDI, вы можете включить его здесь.
smdiport	/dev/ttyS0	Здесь вы должны определить порт SMDI, через который будут передаваться сообщения между Asterisk и внешней АТС.
externpass	/путь/к/скрипту	Каждый раз, когда пароль в почтовом ящике изменяется, скрипт, который вы определяете здесь, будет уведомлен о `context`, `mailbox` и `new password`. Затем скрипт будет отвечать за обновление *voicemail.conf* (приложение голосовой почты Астериска не будет обновлять пароль, если этот параметр определен).
externpassnotify	/path/to/script	Каждый раз, когда пароль в почтовом ящике изменяется, скрипт, который вы определяете здесь, будет уведомлен о `context`, `mailbox` и `new password`. Астериск будет обрабатывать обновление пароля в *voicemail.conf*. Если вы определили `externpass`, этот параметр будет проигнорирован.
externpasscheck	/usr/local/bin/voicemailpwcheck.py	См. врезку в этой таблице для описания этой опции.
directoryintro	dir-intro	Приложение диалплана `Directory()` использует файл *voicemail.conf* для поиска по имени из автосекретаря. Существует приглашение воспроизводимое по умолчанию, которое называется `dir-intro`. Если вы хотите, то можете указать вместо этого другой файл.
charset	ISO-8859-1	Если вам нужен набор символов, отличный от ISO-8859-1 (a.k.a Latin 1), вы можете указать его здесь.
adsifdn	0000000F	Используйте этот параметр для настройки Feature Descriptor Number.[d]
adsisec	9BDBF7AC	Используйте этот параметр для настройки кода блокировки безопасности.
adsiver	1	Указывает номер версии приложения голосовой почты ADSI.
pbxskip	yes	Если вы не хотите, чтобы в тему сообщения электронной почты с вашей голосовой почтой добавлялась строка [PBX], вы можете установить ее как `yes`.
fromstring	The Asterisk PBX	Этот параметр можно использовать для настройки имени `From:`, которое будет отображаться в электронной почте с вашей АТС.
usedirectory	yes	Эта опция позволяет пользователям, составляющим сообщения из своих почтовых ящиков, использовать справочник.
odbcstorage	<item from res_odbc.conf>	Если вы хотите хранить голосовые сообщения в базе данных, то можете сделать это, используя коннектор `res_odbc`. Здесь вы должны указать имя элемента в файле *res_odbc*. Подробнее см. в Главе 22.
odbctable	<table name>	Этот параметр указывает имя таблицы в базе данных, к которой относится параметр `odbcstorage`. Подробнее см. в Главе 22.
emailsubject	[PBX]: New message ${VM_MSGNUM} in mailbox ${VM_MAILBOX}	Когда Asterisk отправляет электронное письмо, вы можете использовать этот параметр, чтобы определить, как будет выглядеть строка `Subject:` Подробнее см. файл *voicemail.conf.sample*.
emailbody	Dear ${VM_NAME}:\n\n\tjust wanted to let you know you were just left a	Когда Asterisk отправляет электронное письмо, вы можете использовать этот параметр, чтобы определить, как будет выглядеть тело письма. Подробнее см. файл *voicemail.conf.sample*.

Опция	Значение/Пример	Примечание
	${VM_DUR} long message (number ${VM_MSGNUM}) \nin mailbox ${VM_MAILBOX} umight\nwant to check it when you get a chance. Thanks!\n\n\t\t\t\t--Asterisk\n	
pagerfromstring	The Asterisk PBX	Мы не знаем никого, кто ещё использует пейджеры (и мы не можем вспомнить, чтобы видели его за многие годы), но если у вас есть одна из этих исторических странностей и вы хотите настроить чтобы Asterisk отправлял уведомления на пейджер, вы можете сделать это. Очень практичное применение этой функции для уведомлений о голосовой почте - это отправить короткое сообщение на шлюз электронной почты-SMS.
pagersubject	New VM	Как и выше.
pagerbody	New ${VM_DUR} long msg in box ${VM_MAILBOX}\nfrom ${VM_CALLERID}, on ${VM_DATE}	Форматирование для этого использует те же правила, что и emailbody.
emaildateformat	%A, %d %B %Y at %H:%M:%S	Этот параметр позволяет указать формат даты в письмах. Использует те же правила, что и функция C STRFTIME.
pagerdateformat	%A, %d %B %Y at %H:%M:%S	Этот параметр позволяет указать формат даты в pagerbody. Использует те же правила, что и функция C STRFTIME.
mailcmd	/usr/sbin/sendmail -t	Если вы хотите переопределить приложение операционной системы по умолчанию для отправки почты, вы можете указать его здесь.
pollmailboxes	no, yes	Если содержимое почтовых ящиков не изменено ничем, кроме app_voicemail (например, внешними приложениями или другой системой Asterisk), установка этого параметра yes приведет к тому, что app_voicemail опросит все почтовые ящики на изменения, что вызовет правильные обновления ожидающего сообщения (MWI),
pollfreq	30	Используется совместно с pollmailboxes, эта опция указывает количество секунд ожидания между опросами почтовых ящиков.
imapgreetings	no, yes	Включает/отключает удаленное хранение приветствий в папке IMAP. Подробнее см. Главу 18.
greetingsfolder	INBOX	Если вы включили imapgreetings, этот параметр позволяет вам определить папку, в которой будут сохранены ваши приветствия (по умолчанию - INBOX).
imapparentfolder	INBOX	Серверы IMAP могут обрабатывать родительские папки по-разному. Это поле позволяет указать родительскую папку для ваших почтовых ящиков. Подробнее см. Главу 7.
imapserver	localhost	Определяет сервер IMAP, к которому должен подключаться сервер Asterisk.
imapport	143	Определяет порт сервера IMAP для подключения.
imapflags	ssl	Серверы IMAP обычно имеют разные флаги, которые могут быть переданы вместе с именем почтового ящика. Общий флаг для передачи - ssl, который позволяет использовать OpenSSL для шифрования сообщений, если библиотеки IMAP были

Опция	Значение/Пример	Примечание
		скомпилированы с поддержкой OpenSSL.
imapfolder	INBOX	Папка для хранения сообщений голосовой почты на сервере IMAP. По умолчанию используется INBOX.
authuser	user	Если ваш сервер IMAP был определен с учетной записью, которая может обращаться ко всем почтовым ящикам, вы можете определить, какой пользователь от Asterisk должен подключиться к серверу.
authpassword	password	Этот параметр определяет пароль, который будет использоваться с атрибутом authuser.
imapopentimeout	60	Тайм-аут открытия TCP в секундах.
imapclosetimeout	60	Тайм-аут закрытия TCP в секундах.
imapreadtimeout	60	Тайм-аут чтения TCP в секундах.
imapwritetimeout	60	Тайм-аут записи TCP в секундах.

[a] Сепаратор, который используется для каждого параметра формата, должен быть символом pipe (|).

[b] Отправка электронной почты от Asterisk может потребовать некоторой тщательной настройки, так как многие спам-фильтры обнаруживают сообщения от Asterisk подозрительными и просто игнорируют их. Мы больше поговорим о том, как настроить электронную почту для Asterisk в Главе 18.

[c] Да, вы правильно это прочитали: мегабайты.

[d] Интерфейс аналоговых дисплеев - это стандарт, который позволяет более сложные взаимодействия функций с помощью дисплея телефона и меню. С появлением VoIP-телефонов популярность ADSI снизилась за последние годы.

Внешняя проверка паролей голосовой почты

По умолчанию Asterisk не проверяет пароли пользователей, чтобы убедиться, что они хотя бы несколько безопасны. Любой, кто поддерживает системы голосовой почты, скажет вам, что большой процент пользователей почтовых ящиков устанавливает свои пароли примерно на 1234 или 1111 или какую-то другую строку, которую легко угадать. Это представляет собой огромную дыру в системе голосовой почты.

Поскольку модуль *app_voicemail.so* не имеет встроенной возможности проверки паролей, настройки `externpass`, `externpassnotify` и `externpasscheck` позволяют проверить их с помощью внешней программы. Asterisk вызовет программу на основе указанного вами пути и передаст ей следующие аргументы:

```
mailbox context oldpass newpass
```

Затем скрипт будет оценивать аргументы на основе правил, определенных вами во внешнем скрипте, и, основываясь на ваших правилах, он должен вернуть Asterisk значение `VALID` при успехе или `INVALID` при отказе (на самом деле, возвращаемое значение для неудачного пароля может быть любым, кроме слов `VALID` или `FAILURE`). Это значение обычно печатается в стандартный вывод - *stdout*. Если скрипт возвращает `INVALID`, Asterisk будет воспроизводить уведомление о недействительном пароле, и пользователю нужно будет попробовать другой пароль.

В идеале вы бы хотели реализовать такие правила, как:

- Пароли должны иметь длину не менее шести цифр
- Пароли не должны быть строками повторяющихся цифр (например, `111111`)
- Пароли не должны быть строками смежных цифр (например, `123456` или `987654`)

Asterisk поставляется с простым скриптом, который значительно улучшит безопасность вашей системы голосовой почты. Он находится в исходном коде в папке: */contrib/scripts/voice/mailpwcheck.py*.

Мы настоятельно рекомендуем вам скопировать его в папку */usr/local/bin* (или где бы вы хранили такие вещи), а затем раскомментировать параметр `externpasscheck=` в файле *voicemail.conf*. Затем ваша система голосовой почты будет применять правила безопасности паролей, которые вы установили.

Часть раздела `[general]` - это область, называемая расширенными опциями. Эти параметры (перечисленные в Таблице 8-2) определяются так же, как и другие параметры в разделе `[general]`, но они также могут быть определены для каждого почтового ящика для конкретной настройки, чтобы переопределить все, что определено в `[general]`.

Таблица 8-2. Расширенные опции для voicemail.conf

Опция	Значение/Пример	Примечание
`tz`	`eastern,european, etc`	Задает имя `zonemessages`, как определено в `[zonemessages]` (обсуждается в следующем разделе).
`locale`	`de_DE.utf8, es_US.utf8, etc`	Используется для определения того, как Asterisk генерирует строки даты/времени в разных локалях. Чтобы определить локали, действующие в вашей системе Linux, введите **`locale -a`** в shell.
`attach`	`yes,no`	Если для почтового ящика указан адрес электронной почты, это определяет, прикреплены ли сообщения к уведомлениям по электронной почте (в противном случае отправляется сообщение с простым оповещением).
`attachfmt`	`wav49,wav,etc`	Если `attach` включено и сообщения хранятся в разных форматах, это определяет, какой формат отправляется с уведомлением по электронной почте. Часто `wav49` является хорошим выбором, поскольку он использует лучший алгоритм сжатия и, следовательно, будет использовать меньшую пропускную способность.
`saycid`	`yes,no`	Эта команда укажет callerID вызывающего абонента, который оставил сообщение.
`cidinternalcontexts`	`<context>, <another context>`	Любые контексты диалплана, перечисленные здесь, будут найдены при попытке найти контекст почтового ящика, чтобы можно было указать имя, связанное с номером почтового ящика. Номер ящика голосовой почты должен соответствовать внутреннему номеру, из которого пришел вызов, а контекст голосовой почты должен соответствовать контексту диалплана.[a]
`sayduration`	`yes,no`	Команда укажет длину сообщения.
`saydurationm`	`2`	Используйте это, чтобы указать минимальную продолжительность сообщения, чтобы соответствовать его длине воспроизведения. Например, если вы установите значение `2`, любое сообщение длиной менее 2 минут не будет указано. (Опция действительно является `saydurationm` и не является опечаткой здесь)
`dialout`	`<context>`	Если разрешено, пользователи могут набирать номер из своих почтовых ящиков. Это считается очень опасной особенностью в телефонной системе (главным образом потому, что многие пользователи голосовой почты любят использовать `1234` в качестве пароля) и поэтому не рекомендуется. Если вы настаиваете на разрешении этого, убедитесь, что у вас есть второй уровень в диалплане, где указан другой пароль. Тем не менее, это небезопасная практика.
`sendvoicemail`	`yes,no`	Позволяет пользователям составлять сообщения другим пользователям из своих почтовых ящиков.
`searchcontexts`	`yes,no`	Позволяет приложениям голосовой почты в диалплане не указывать контекст голосовой почты, так как поиск будет выполняться во всех контекстах. Это не рекомендуется.
`callback`	`<context>`	Указывает какой контекст диалплана использовать для обращения

Опция	Значение/Пример	Примечание
		к отправителю сообщения. Указанный контекст должен иметь возможность обрабатывать набор номеров в том формате, в котором они получены (например, код страны не может быть получен с CallerID, но может потребоваться для исходящего вызова).
exitcontext	<context>	Существуют опции, позволяющие вызывающим абонентам выйти из системы голосовой почты, когда они находятся в процессе оставления сообщения (например, нажав 0, чтобы перейти на оператора). По умолчанию контекст, который вызвал звонящий, будет использоваться в качестве контекста выхода. При желании этот параметр определяет другой контекст для вызывающих абонентов, выходящих из системы голосовой почты.
review	yes,no	Почти всегда должен быть настроен на yes (хотя по умолчанию равно no). Люди расстраиваются, если ваша система голосовой почты не позволяет им просматривать свои сообщения до их доставки.
operator	yes,no	Лучшая практика диктует, что вы должны позволить своим абонентам «обнулить» почтовый ящик, если они не захотят оставлять сообщение. Обратите внимание, что в контексте exitcontext требуется расширение o (не «ноль», а буква «o») чтобы обрабатывать эти вызовы.
envelope	no,yes	Вы можете воспроизводить детали сообщения голосовой почты прежде, чем оно воспроизводится. Поскольку эту информацию можно также получить нажав 5, мы обычно устанавливаем как no.
delete	no,yes	После отправки сообщения электронной почты (которое может включать само сообщение) оно будет удалено. Этот вариант является рискованным, потому что тот факт, что сообщение отправлено по электронной почте, не является гарантией того, что оно было получено (фильтры спама часто любят удалять сообщения голосовой почты Asterisk). В новой системе оставьте это без изменений, пока не убедитесь, что сообщения не теряются из-за спам-фильтров.
volgain	0.0	Параметр позволяет увеличить объем полученных сообщений. В предыдущих версиях Asterisk это было проблемой, но не в течение последних лет. Мы рекомендуем оставить по умолчанию. Для нормальной работы требуется утилита *sox*.
nextaftercmd	yes,no	Эта удобная небольшая настройка позволит вам сэкономить некоторое время, так как она сразу переходит к следующему сообщению, когда вы закончите работу с текущим.
forcename	yes,no	Этот странный маленький параметр будет проверять, совпадает ли пароль почтового ящика с номером почтового ящика. Если это так, то заставит пользователя изменить пароль голосовой почты и записать своё имя.
forcegreetings	yes,no	Как и выше, но для приветствий.
hidefromdir	no,yes	Если хотите, то можете скрыть определенные почтовые ящики из приложения Directory(), используя этот параметр.
tempgreetwarn	yes,no	Установка этого параметра yes будет предупреждать владельца почтового ящика, что у него есть временное приветствие. Это может быть полезным напоминанием, когда люди возвращаются из поездок или каникул.

Опция	Значение/Пример	Примечание
passwordlocation	spooldir	Если хотите, то можете иметь пароли почтового ящика, хранящиеся в папке *spool* для каждого почтового ящика.[b] Одним из преимуществ использования опции `spooldir` является то, что он позволит вам определять операторы `#include` в файле *voicemail.conf* (это означает, что можете хранить ссылки на почтовые ящики в нескольких файлах, как в случае, например, с диалплан-кодом). Это невозможно в противном случае, поскольку `app_voicemail` обычно записывает изменение пароля в файловую систему и не может обновлять пароль почтового ящика, хранящийся за пределами *voicemail.conf* или */spool*. Если вы не используете `passwordlocation`, то не сможете определять почтовые ящики вне *voicemail.conf*, так как обновление пароля не произойдет. Хранение паролей в файле в конкретной папке почтового ящика в */spool* решает эту проблему.
messagewrap	no,yes	Если для этого параметра установлено значение `yes`, то, когда пользователь прослушал последнее сообщение, выбор следующего (6) приведет к первому сообщению. Также, выбор предыдущего (4), находясь в первом сообщении пользователя перейдет к последнему сообщению.
minpassword	6	Этот параметр обеспечивает минимальную длину пароля. Обратите внимание, что он не мешает пользователям устанавливать свои пароли в виде чего-то легкомысленного (например, `123456`).
vm-password	custom_sound	Если хотите, можете указать здесь пользовательский звук, чтобы использовать для подсказки пароля в голосовой почте.
vm-newpassword	custom_sound	Если хотите, можете указать здесь пользовательский звук, чтобы использовать для подсказки «Пожалуйста, введите новый пароль, а затем клавишу решетка» в голосовой почте.
vm-passchanged	custom_sound	Если хотите, можете указать здесь пользовательский звук, чтобы использовать для подсказки «Ваш пароль был изменен» в голосовой почте.
vm-reenterpassword	custom_sound	Если хотите, можете указать здесь пользовательский звук, чтобы использовать для подсказки «Пожалуйста, введите пароль, за которым следует символ решетка».
vm-mismatch	custom_sound	Если хотите, то можете указать пользовательский звук, чтобы использовать для подсказки «Введенные вами пароли и подтверждение пароля не совпадают».
vm-invalid-password	custom_sound	Если хотите, вы можете указать пользовательский звук здесь, чтобы использовать для подсказки «Это недействительный пароль. Повторите попытку».
vm-pls-try-again	custom_sound	Если хотите, то можете указать здесь пользовательский звук, чтобы использовать подсказку «Попробуйте еще раз».
vm-prepend-timeout	custom_sound	Если хотите, то можете указать здесь пользовательский звук, чтобы использовать его, когда пользователь отключается во время добавления записи сообщения. Подсказка по умолчанию: «затем нажмите решетку» и следуйте подсказке `vm-pls-try-again`.
listen-control-forward-key	#	Вы можете использовать этот параметр для настройки клавиши быстрой перемотки вперед.
listen-control-reverse-key	*	Вы можете использовать этот параметр для настройки клавиши перемотки назад.
listen-control-pause-key	0	Вы можете использовать этот параметр для настройки клавиши паузы/продолжения.
listen-control-restart-key	2	Вы можете использовать этот параметр для настройки клавиши повтора.

Опция	Значение/Пример	Примечание
`listen-control-stop-key`	`123456789`	Вы можете использовать этот параметр для настройки клавиши прерывания воспроизведения.
`backupdeleted`	`0`	Этот параметр позволит вам указать, сколько удаленных сообщений автоматически сохраняется системой. Это похоже на корзину. Установка этого параметра на `0` отключает эту функцию. Можно сохранить до 9999 сообщений, после чего самое старое сообщение будет удалено при каждом удалении другого сообщения.

[a] Да, мы нашли это немного запутанным.

[b] Типичная папка spool */var/spool/asterisk*, и ее можно переопределить в файле */etc/asterisk/asterisk.conf*.

Раздел [zonemessages]

Следующий раздел файла *voicemail.conf* - это раздел `[zonemessages]`. Цель этого раздела - разрешить обработку сообщений по часовому поясу, чтобы вы могли воспроизводить сообщения пользователей с правильными отметками времени. Вы можете установить имя зоны в нужное вам значение. Следуя имени зоны, вы можете определить, на какой часовой пояс будет ссылаться это название, а также некоторые параметры, определяющие, как воспроизводятся временные метки. Вы можете посмотреть файл *~/src/asterisk-complete/asterisk/11/configs/voicemail.conf.sample* для деталей синтаксиса. Asterisk включает примеры, показанные в Таблице 8-3.

Таблица 8-3. Секция опций [zonemessages] для voicemail.conf

Имя зоны	Значение/Пример	Примечание
`eastern`	`America/New_York|'vm-received' Q 'digits/at' IMp`	Это значение подходит для восточного часового пояса (EST/EDT).
`central`	`America/Chicago|'vm-received' Q 'digits/at' IMp`	Это значение подходит для центрального часового пояса (CST/CDT).
`central24`	`America/Chicago|'vm-received' q 'digits/at' H N 'hours'`	Это значение также подходит для CST/CDT, но будет воспроизводить время в 24-часовом формате.
`military`	`Zulu|'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'`	Это значение подходит для универсального временного координирования (время Зулу, ранее GMT).
`european`	`Europe/Copenhagen|'vm-received' a d b 'digits/at' HM`	Это значение подходит для среднеевропейского времени (CEST).

Раздел контекстов

Все остальные разделы в файле *voicemail.conf* будут контекстами голосовой почты, которые позволят вам разделить группы почтовых ящиков.

Во многих случаях вам понадобится только один контекст голосовой почты, обычно называемый `[default]`. Его стоит отметить, так как он упростит диалплан: все приложения, связанные с голосовой почтой, принимают контекст `default`, если другой не задан. Другими словами, если вы не требуете разделения ваших пользователей голосовой почты, используйте `default` в качестве своего единственного контекста голосовой почты.

Формат почтовых ящиков следующий (вы должны ввести все это в одну строку):

```
mailbox => password[,FirstName LastName[,email addr[,pager addr [,options[|
options]]]]]
```



Пайп (вертикальная черта) (`|`) был очень популярен в Asterisk. В течение первых нескольких лет он использовался в качестве стандартного разделителя. Совсем недавно он почти полностью был заменен запятой; однако все еще есть несколько мест, где используется пайп.

Один из них находится в файле *voicemail.conf*: например, в качестве разделителя для любых параметров, специфичных для почтового ящика, а также как символ разделителя в описании `format=`. Вы увидите это в нашем следующем примере, а также в файле *voicemail.conf.sample*.

Части для определения почтового ящика:

`mailbox`

Это номер почтового ящика. Обычно соответствует добавочному номеру.

`password`

Это числовой пароль, который владелец почтового ящика будет использовать для доступа к голосовой почте. Если пользователь изменит свой пароль, система обновит это поле в файле *voicemail.conf*.

Если паролю предшествует символ дефиса (-), пользователь не может изменить пароль своего почтового ящика.

Если вы храните пароли в *spool* (используя параметр `passwordlocation`) то это поле игнорируется. Тем не менее, анализатор синтаксиса по-прежнему требует чтобы здесь было поле, поэтому, если вы укажете какие-либо другие параметры для этого почтового ящика, запятая будет необходима в качестве заполнителя для поля пароля.

`FirstName LastName`

Это имя владельца почтового ящика. В каталоге компании используется текст в этом поле, позволяющий вызывающим пользователям задавать имена пользователей.

`email address`

Это адрес электронной почты владельца почтового ящика. Asterisk может отправлять уведомления голосовой почты (включая само сообщение голосовой почты в виде вложения) на указанный почтовый ящик.

`pager address`

Это адрес электронной почты пейджера или мобильного телефона владельца почтового ящика. Asterisk может отправить короткое сообщение с уведомлением о голосовой почте на указанный адрес электронной почты.

`options`

Это поле представляет собой список параметров для настройки часового пояса владельца почтового ящика и переопределения глобальных настроек голосовой почты. Существует немало действительных вариантов:

`tz, locale, attach, attachfmt, saycid, cidinternalcontexts, sayduration, saydurationm, dialout, sendvoicemail, searchcontexts, callback, exitcontext, review, operator, envelope, delete, volgain, nextaftercmd, forcename, forcegreeting, hidefromdir, tempgreetwarn, passwordlocation, messagewrap, minpassword.`

Эти параметры должны быть в парах *`option = value`*, разделенных символом pipe (|). См. Таблицу 8-4 для получения более подробной информации о том, что делает каждый из этих параметров.

Опция `tz` устанавливает часовой пояс пользователя в ранее определенный в разделе `[zonemessages]` в файле *voicemail.conf*. Другие параметры переопределяют глобальные настройки голосовой почты с одинаковыми именами.

Таблица 8-4. Опции почтового ящика

Опция	Описание
`attach`	Прилагать ли голосовую почту к уведомлению по электронной почте. Если установлено `yes`, она будет прикреплена к письму, заданному в поле адреса электронной почты.
`attachfmt`	Устанавливает формат для прикрепления к письму. Обычно это первое значение,

Опция	Описание
	определяемое параметром формата, но вы можете переопределить это для каждого почтового ящика с помощью этой опции. Опция может быть установлена только для каждого почтового ящика.
`callback`	Если этот параметр определен, то он позволит получателю электронной почты перезвонить отправителю голосовой почты непосредственно из приложения `Voicemail()`. Этот параметр определяет, из какого контекста будет отправлен вызов. Если не установлено, вызов отправителя не будет разрешен.
`cidinternalcontexts`	Это очень старый вариант с 2004 года, но по существу вы можете определить несколько контекстов (разделенных запятой), которые передадут Asterisk что надо проверить, пришел ли вызов из внутреннего контекста. Если это так, он будет воспроизводить запись имени человека вместо того, чтобы указывать свой добавочный номер. Неясно, остается ли этот вариант действительным или функциональным. Вероятно, лучше всего использовать в разделе голосовой почты `[general]`, а не в почтовом ящике.
`delete`	После отправки голосовой почты по электронной почте она удаляется с сервера. Эта опция полезна для пользователей, которые хотят получать голосовую почту только по электронной почте. Допустимые параметры: `yes` или `no`. Опция может быть установлена только для каждого почтового ящика.
`dialout`	Если определено, опция `4` из расширенного меню позволит вам набрать номер из приложения `VoicemailMain()`. Аргумент определяет, из какого контекста будет выполняться вызов. Если этот параметр не определен, для вызова не будет запрашиваться опция набора номера.
`envelope`	Включение или выключение воспроизведения конверта перед воспроизведением сообщения голосовой почты. Допустимые параметры: `yes` или `no`. По умолчанию `yes`.
`exitcontext`	Контекст для выхода из приложения `Voicemail()` при нажатии `*` или `0` . Работает так же как с опцией `operator`. Должен иметь расширение `a` в контексте с `*`. Должен иметь расширение `o` в контексте с `0`.
`forcegreeting`	Заставляет записывать приветствие для новых почтовых ящиков. Новый почтовый ящик определяется номером почтового ящика и паролем. Допустимые значения: `yes` или `no`. По умолчанию `no`.
`forcename`	Заставляет записывать имя пользователя для новых почтовых ящиков. Новый почтовый ящик определяется номером почтового ящика и паролем. Допустимые значения: `yes` или `no`. По умолчанию `no`.
`hidefromdir`	Если установлено `yes`, этот почтовый ящик будет скрыт от приложения `Directory()`. По умолчанию `no`.
`locale`	Позволяет установить языковой стандарт для почтового ящика, чтобы управлять форматированием строк даты/времени. Дополнительную информацию см. в *voicemail.sample.conf*.
`messagewrap`	Позволяет разворачивать первое и последнее сообщения; например, разрешить с последнего сообщения переходить к первому при выборе следующего или первому сообщению к последнему при переходе к предыдущему. Допустимые параметры: `yes` или `no`. По умолчанию `no`.
`minpassword`	Устанавливает минимальную длину пароля. Аргумент должен быть целым числом.
`nextaftercmd`	Переход к следующему сообщению после нажатия клавиши `7` (удаление) или `9` (сохранение). Допустимые значения: `yes` или `no`. По умолчанию `yes`.
`operator`	Позволит отправителю голосовой почты нажать `0` до, во время или после записи голосовой почты. Выйдет на расширение `o` в том же контексте или в контексте, определяемом параметром `exitcontext`. Допустимые параметры: `yes` или `no`. По умолчанию `no`.
`passwordlocation`	По умолчанию пароль для голосовой почты сохраняется в файле *voicemail.conf* и изменяется при помощи Asterisk при каждом изменении пароля. Это может быть нежелательно, особенно если вы хотите проанализировать пароль из внешней локации (или скрипта). Альтернативным вариантом для определения пароля является `spooldir`, который поместит пароль для пользователя голосовой почты в файл *secret.conf* в каталоге голосовой почты пользователя.

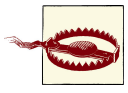
Опция	Описание
	Допустимыми параметрами являются `voicemail.conf` и `spooldir`. Опция по умолчанию `voicemail.conf`.
`review`	Когда включено, разрешит пользователю записывать сообщение голосовой почты для повторной записи своего сообщения. После нажатия клавиши # чтобы сохранить голосовую почту им будет предложено переписать или сохранить сообщение. Допустимые параметры: `yes` или `no`. По умолчанию `no`.
`saycid`	Если включено и в каталоге */var/spool/asterisk/voicemail/recordings/callerids* существует приглашение, этот файл будет воспроизводиться до сообщения, воспроизводя файл вместо того, чтобы указывать номер вызывающего абонента. Допустимые параметры: `yes` или `no`. По умолчанию `no`.
`sayduration`	Определяет, следует ли воспроизводить длительность сообщения до его воспроизведения. Допустимые параметры: `yes` или `no`. По умолчанию `yes`.
`saydurationm`	Позволяет установить минимальную продолжительность воспроизведения (в минутах). Например, если вы установите значение 2, то не будете информированы о длине сообщения для сообщений длиной менее 2 минут. Допустимые значения - целые числа. Значение по умолчанию - 2.
`searchcontexts`	Для таких приложений, как `Voicemail()`, `VoicemailMain()` и `Directory()` контекст голосовой почты является необязательным аргументом. Если контекст голосовой почты не указан, то используется поиск контекста `default`. Если этот параметр включен, будут просмотрены все контексты. Это связано с той оговоркой, что если включено, номер почтового ящика должен быть уникальным во всех контекстах, иначе будет коллизия, и система не поймет, какой почтовый ящик использовать. Допустимыми параметрами являются `yes` и `no`. По умолчанию `no`.
`sendvoicemail`	Позволяет пользователю составлять и отправлять сообщение голосовой почты из приложения `VoicemailMain()`. Доступно в качестве опции `5` в расширенном меню. Если этот параметр отключен, опция `5` в расширенном меню не будет запрашиваться. Допустимые параметры: `yes` или `no`. По умолчанию `no`.
`tempgreetwarn`	Включает уведомление пользователя, когда включено их временное приветствие. Допустимые параметры: `yes` или `no`. По умолчанию `no`.
`tz`	Устанавливает часовой пояс для пользователя голосовой почты (или глобально). См. */usr/share/timezone* для разных доступных часовых поясов. Не применимо, если `emvelop=no`.
`volgain`	Опция `volgain` позволяет вам установить усиление громкости для сообщений голосовой почты. Значение устанавливается в децибелах (дБ). Для этого нужно установить приложение *sox*.

Почтовые ящики, которые вы определяете в файле voicemail.conf, могут выглядеть следующим образом:

```
[default]
100 => 5542,Mike Loukides,mike@shifteight.org
101 => 67674,Tim OReilly,tim@shifteight.org
102 => 36217,Mary JonesSmith,mary.jones-smith@shifteight.org

; *** Нужно, чтобы всё было на одной линии
103 => 5426,Some Guy,,,dialout=fromvm|callback=fromvm|review=yes|operator=yes|
envelope=yes

[shifteight]
100 => 0107,Leif Madsen,leif@shifteight.org
101 => 0523,Jim VanMeggelen,jim@shifteight.org,,attach=no|maxmsg=100
102 => 11042,Tilghman Lesher,,,attach=no|tz=central
```



Каталог Asterisk не может справиться с концепцией фамилии, которое является чем-то другим, кроме простого слова. Это означает, что фамилии, такие как *О'Рейли*, *Джонс-Смит*

и да, даже *Ван Меггелен*, перед добавлением в голосовую почту должны быть удалены любые знаки препинания и пробелы перед добавлением в *voicemail.conf*.

Контексты в *voicemail.conf* - отличная и мощная концепция, но вы, скорее всего, обнаружите, что контекст `default` будет всем, что вам нужно при нормальном использовании. Основная причина для нескольких контекстов почтовых ящиков может заключаться в том, что ваша система имеет более одной АТС, и вам необходимо разделить почтовые ящики.

Исходный файл voicemail.conf

Мы рекомендуем следующий образец в качестве отправной точки. Вы можете обратиться к *~/asterisk-complete/asterisk/11/configs/voicemail.conf.sample* для получения подробной информации о различных настройках:

```
; Voicemail Configuration

[general]
format=wav49|wav
serveremail=voicemail@shifteight.org
attach=yes
skipms=3000
maxsilence=10
silencethreshold=128
maxlogins=3
emaildateformat=%A, %B %d, %Y at %r
pagerdateformat=%A, %B %d, %Y at %r
sendvoicemail=yes ; Разрешить пользователю составлять и отправлять
                  ; голосовую почту, находясь внутри

[zonemessages]
eastern=America/New_York|'vm-received' Q 'digits/at' IMp
central=America/Chicago|'vm-received' Q 'digits/at' IMp
central24=America/Chicago|'vm-received' q 'digits/at' H N 'hours'
military=Zulu|'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'
european=Europe/Copenhagen|'vm-received' a d b 'digits/at' HM

[shifteight.org]
100 => 1234,Leif Madsen,leif@shifteight.org
101 => 1234,Jim Van Meggelen,jim@shifteight.org
102 => 1234,Russell Bryant,russell@shifteight.org
103 => 1234,Jared Smith,jared@shifteight.org
```



Настройка сервера Linux для обработки отправки электронной почты - это задача администрирования Linux, выходящая за рамки этой книги. Вам нужно будет протестировать вашу службу голосовой почты, чтобы убедиться в том, что электронное письмо обрабатывается надлежащим образом агентом передачи почты (MTA)[3] и что спам-фильтры входящей почты не отклоняют сообщения (одна из причин, по которым это может произойти, - это сервер Asterisk использует имя хоста в теле письма, которое на самом деле не разрешается).

Стандартные клавиши голосовой почты

Здесь мы рассмотрим стандартную конфигурацию клавиатуры для Comedian Mail. Некоторые параметры могут быть включены или отключены на основе конфигурации *voicemail.conf* (например, `envelope=no`), но наш обзор на Рисунке 8-1 покажет стандартные параметры, доступные с минимальной конфигурацией.

3 Также иногда называется агентом передачи сообщений.

Интеграция диалплана

Существует два основных приложения диалплана, которые предоставляются модулем `app_voicemail.so` в Asterisk. Первый, названный просто `VoiceMail()`, делает именно то, что вы ожидаете от него, а именно - запись сообщения в почтовый ящик. Второй, `VoiceMailMain()`, позволяет вызывающему абоненту входить в почтовый ящик для извлечения сообщений.

Приложение диалплана VoiceMail()

Если вы хотите передать вызов голосовой почте, вам необходимо указать два аргумента: почтовый ящик (или почтовые ящики), на который должно быть отправлено сообщение, и любые параметры, относящиеся к нему, например, приветствие для воспроизведения или отметить сообщение как срочнное. Структура команды `VoiceMail()` следующая:

```
VoiceMail(mailbox[@context][&mailbox[@context][&...]][,options])
```

Параметры, которые вы можете передать в `VoiceMail()` для обеспечения более высокого уровня контроля, приведены в Таблице 8-5.

```
VoiceMailMain() --> Аутентификация --> Уведомление: Количество старых и
новых сообщений
```

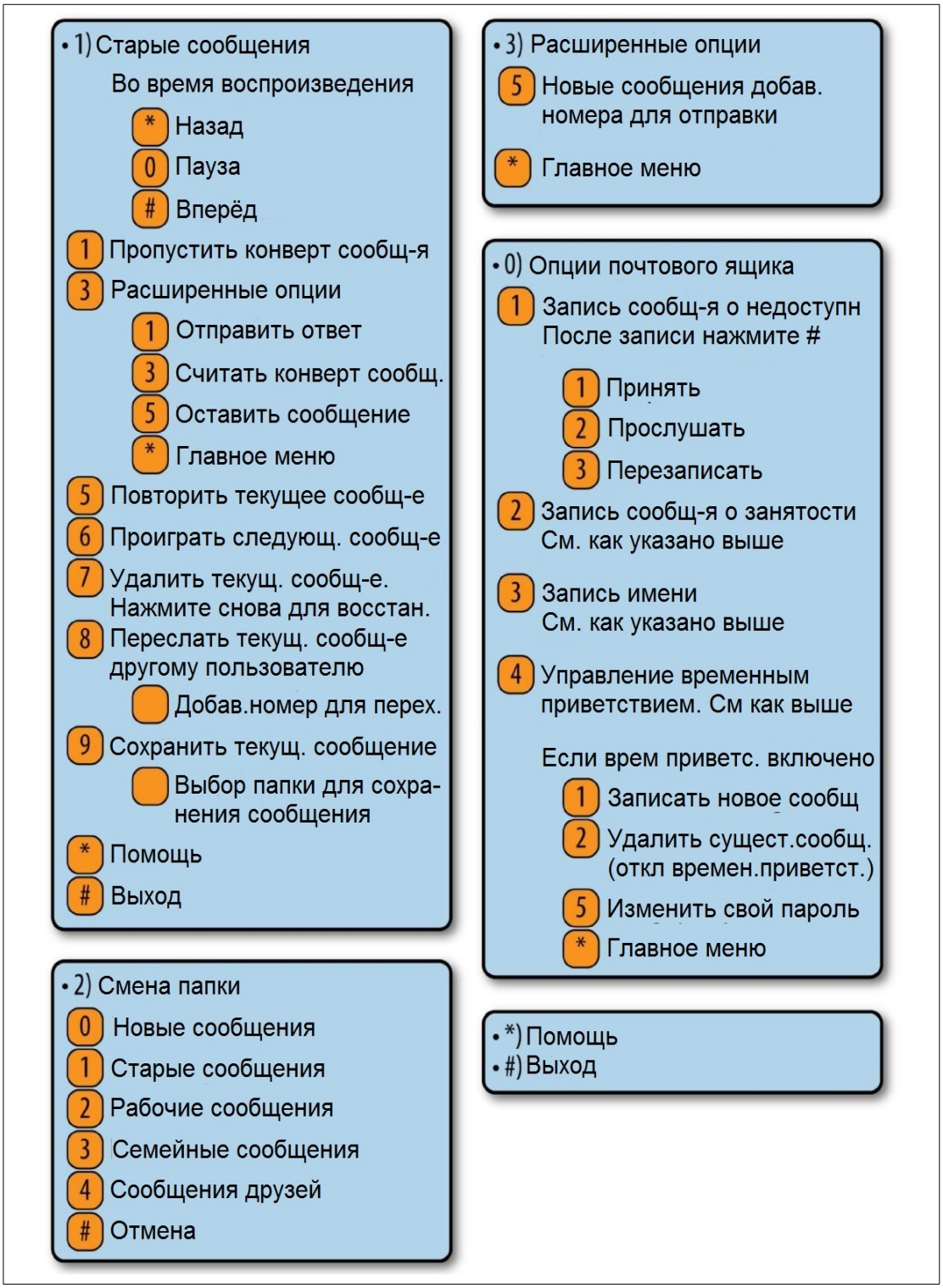


Рисунок 8-1. Конфигурация ключей для Комейдиной почты

Таблица 8-5. Дополнительные аргументы VoiceMail()

Аргумент	Назначение
`b`	Предписывает Asterisk воспроизводить приветствие "занято" для почтового ящика (если приветствие "занято" отсутствует, будет воспроизводиться приветствие "недоступно").
`d([c])`	Принимает цифры, подлежащие обработке в контексте `c`. Если контекст не указан, по умолчанию будет использоваться текущий контекст.
`g(#)`	Применяет заданную величину усиления (в децибелах) к записи. Работает только на каналах DAHDI.
`s`	Подавляет воспроизведение инструкций вызывающим абонентам после воспроизведения приветствия.

Аргумент	Назначение
u	Поручает Asterisk воспроизводить уведомление о недоступном почтовом ящике (это поведение по умолчанию).
U	Указывает, что это сообщение должно быть отмечено как срочное. Наиболее заметным эффектом этого является то, что голосовая почта хранится на сервере IMAP. В этом случае электронное письмо будет отмечено как срочное. Когда владелец почтового ящика звонит в систему голосовой почты Asterisk, ему также следует сообщить, что сообщение является срочным.
P	Указывает, что это сообщение должно быть помечено как приоритетное

Приложение `VoiceMail()` отправляет вызывающего абонента в указанный почтовый ящик, чтобы он мог оставить сообщение. Почтовый ящик должен быть указан как *`mailbox@context`*, где *`context`* - это имя контекста голосовой почты. Буквы опций `b` или `u` могут быть добавлены для запроса типа приветствия. Если используется буква `b`, вызывающий абонент услышит сообщение о *занятости* владельца почтового ящика. Если используется буква `u`, вызывающий абонент будет слышать сообщение о *недоступности* владельца почтового ящика (если такое существует).

Рассмотрим простой пример добавочного номера 101, который позволяет людям вызывать Джона:

```
exten => 101,1,NoOp()
    same => n,Dial(${JOHN})
```

Давайте добавим сообщение о недоступности, которое будет воспроизводиться вызывающим абонентам, если Джон не ответит на звонок. Помните, что второй аргумент приложения `Dial()` - это тайм-аут. Если вызов не отвечен до истечения времени ожидания, он отправляется на следующий приоритет. Давайте добавим 10-секундный тайм-аут и приоритет для отправки вызывающего абонента на голосовую почту, если Джон не ответит вовремя:

```
exten => 101,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,VoiceMail(101@default,u)
```

Теперь давайте изменим его так, что если Джон занят (по другому вызову), вызывающий будет отправлен на его голосовую почту, где он услышит сообщение о занятости Джона. Для этого мы будем использовать переменную `${DIALSTATUS}`, которая содержит одно из нескольких значений состояния (введите **`core show application Dial`** в консоли Asterisk для перечисления всех возможных значений):

```
exten => 101,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(101@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(101@default,b)
    same => n,Hangup()
```

Теперь звонящие получат ответ голосовой почты Джона (с соответствующим приветствием), если Джон занят либо недоступен. Альтернативный синтаксис заключается в использовании функции `IF()` для определения того, какое из сообщений (недоступен или занят) использовать:

```
exten => 101,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,Voicemail(101@default,${IF($["${DIALSTATUS}" = "BUSY"]?b:u)})
    same => n,Hangup()
```

Однако остается небольшая проблема в том, что Джон не имеет способа получить свои сообщения. Мы исправим это в следующем разделе.

Приложение диалплана VoiceMailMain()

Пользователи могут получать сообщения голосовой почты, изменять свои параметры и записывать приветствия голосовой почты с помощью приложения `VoiceMailMain()`. `VoiceMailMain()` принимает два аргумента: номер почтового ящика (и, возможно, контекст) для доступа и некоторые параметры. Оба аргумента являются необязательными.

Структура приложения `VoiceMailMain()` выглядит следующим образом:

```
VoiceMailMain([mailbox][@context][,options])
```

Если вы не передадите какие-либо аргументы в `VoiceMailMain()`, он будет воспроизводить приглашение, предлагая вызывающему абоненту указать номер своего почтового ящика. Параметры, которые могут быть подставлены, перечислены в Таблице 8-6.

Таблица 8-6. Необязательные аргументы VoiceMailMain()

Аргумент	Назначение
`p`	Позволяет обрабатывать параметр *mailbox* в качестве префикса для номера почтового ящика.
`g(#)`	Увеличивает коэффициент усиления на `#` децибел во время воспроизведения сообщений.
`s`	Пропускает проверку пароля.
`a(folder)`	Запуск сеанса в одной из следующих папок голосовой почты (по умолчанию `0`): • `0 - INBOX` • `1 - Old` • `2 - Work` • `3 - Family` • `4 - Friends` • `5 - Cust1` • `6 - Cust2` • `7 - Cust3` • `8 - Cust4` • `9 - Cust5`

Чтобы пользователи могли набирать номер для проверки своей голосовой почты, вы можете добавить внутренний номер к диалплану следующим образом:

```
[Services]
exten => *98,1,NoOp(Access voicemail retrieval.)
    same => n,VoiceMailMain()
```

Тогда вам просто нужно добавить include в контекст `[LocalSets]`, чтобы вы могли набрать `*98`:

```
[LocalSets]
; существующий диалплан выше
include => Services
```

Создание каталога Набор-по-имени

Последней особенностью системы голосовой почты Asterisk, которую мы должны рассмотреть, является каталог набора-по-имени. Он создается приложением `Directory()`. Это приложение использует имена, определенные для почтовых ящиков в *voicemail.conf*, чтобы предоставить вызывающему абоненту каталог для набора пользователей по имени.

`Directory()` принимает до трех аргументов: контекст голосовой почты, из которого можно читать имена; необязательный контекст диалплана, в котором набирать пользователя, и строку параметров (которая также является необязательной). По умолчанию `Directory()` ищет пользователя по фамилии, но передача параметра `f` заставляет вместо этого искать по имени. Давайте добавим два

набора по названию каталогов во входящий контекст нашего образца диалплана, чтобы вызывающие могли выполнять поиск по имени или фамилии:

```
exten => 8,1,Directory(default,incoming,f)
exten => 9,1,Directory(default,incoming)
```

Если вызывающие абоненты нажимают `8`, они получат каталог по имени. Если они набирают `9`, они получат каталог по фамилии.

Использование Jitterbuffer'a

При использовании Asterisk в качестве сервера голосовой почты[4], вы можете добавить *jitterbuffer* между голосовой почтой и вызывающим абонентом. Цель jitterbuffer'a заключается в том, чтобы помочь справиться с ситуацией, когда вызов проходит через IP-сеть, трафик может не прибывать с идеальным временем и в идеальном порядке. Если пакеты иногда прибывают с небольшой задержкой (jitter - дрожанием) или не по порядку, джиттер-буфер может исправить это, чтобы система голосовой почты получала голосовой поток вовремя и по порядку. Если джиттер-буффер обнаруживает, что пакет был потерян (или может появиться так поздно, что он больше не будет иметь значения), он может выполнять скрытие потери пакетов. То есть, он попытается создать аудиокадр, чтобы поставить на место потерянного, чтобы было сложнее услышать потерю звука.

В Asterisk поддержка jitterbuffer может быть включена на мосту между двумя каналами двумя способами. В случае голосовой почты обычно имеется только один канал, подключенный к одному из приложений голосовой почты. Старый метод (который требуется в версиях Asterisk до 10) заключается в том, чтобы включить использование jitterbuffer перед голосовой почтой, создав мост между двумя каналами с использованием канала `Local` и указав опцию `j`.

Указание опции `n` для канала `Local` дополнительно гарантирует, что канал `Local` не оптимизирован на пути вызова в Asterisk:

```
[Services]
; старый метод -- требуется только в версиях Астериска до 10
exten => *98,1,Dial(Local/vmm@Services/nj)

exten => vmm,1,VoiceMailMain()
```

Начиная с Asterisk 10, существует новая функция диалплана `JITTERBUFFER()`, которая с точки зрения пользователя выполняет те же функции. Просто установив значения в диалоговом окне, мы можем включить jitterbuffer перед доступом к приложению диалплана, например `Voicemail()`:

```
[Services]
; новый метод -- поддерживается в Астериске 10 и позднее
exten => *98,1,NoOp()
    same => n,Set(JITTERBUFFER(fixed)=default)
    same => n,VoiceMailMain()
```

Существует как фиксированный, так и адаптивный джиттер-буфер, а также несколько различных настроек. Мы использовали фиксированный jitterbuffer с настройками по умолчанию, которые следующие. См. *core show function JITTERBUFFER* для получения дополнительных параметров конфигурации:

- Длина буфера 200 мс
- Если существует разница в метках времени более 1000 мс, то джиттер-буфер будет повторно синхронизирован

4 Этот совет относится к любой ситуации, когда Asterisk является конечной точкой вызова. Другим примером может быть использование приложений `MeetMe()` или `ConfBridge()` для конференц-связи.

Внутреннее хранение

Хранение сообщений в традиционных системах голосовой почты всегда было чрезмерно сложным[5]. Asterisk, с другой стороны, не только предоставляет вам простой, логичный механизм хранения на основе файловой системы, но также предлагает несколько дополнительных параметров хранения сообщений.

Файловая система Linux

По умолчанию Asterisk хранит голосовые сообщения в папке spool, в */var/spool/asterisk/voicemail/<context>/<mailbox>*. Сообщения могут храниться в нескольких форматах (например, WAV и GSM), в зависимости от того, что вы указали в качестве формата в разделе `[general]` вашего файла *voicemail.conf*. Ваши приветствия также хранятся в этой папке.



Asterisk не будет создавать папку для любых почтовых ящиков, у которых пока нет записей (как в случае с новым почтовым ящиком), поэтому эта папка не может использоваться как надежный метод определения, какие почтовые ящики существуют в системе.

Вот пример того, что может быть в папке почтового ящика. Этот почтовый ящик не имеет новых сообщений в INBOX, имеет два сохраненных сообщения в папке Old и имеет записи (прветствия) по занятости (busy), недоступности (unavail) и имени (greet) (как показано на рисунке 8-2).

```
/var/spool/asterisk/voicemail/default/301
     ./INBOX
     ./Old
          ./Old/msg0000.WAV
          ./Old/msg0000.txt
          ./Old/msg0001.WAV
          ./Old/msg0001.txt
     ./Urgent
     ./busy.WAV
     ./unavail.WAV
     ./greet.WAV
```

Рисунок 8-2. Образец папки почтового ящика



Для каждого сообщения есть соответствующий файл *msg####.txt*, который содержит информацию о конверте для сообщения. Файл *msg####.txt* также имеет критически важное значение для индикации ожидающего сообщения (MWI), так как это файл, который Asterisk ищет в *INBOX*, чтобы определить, должен ли индикатор сообщения для пользователя быть включен или нет.

ODBC

В централизованной или распределенной системе вам может потребоваться хранить сообщения как двоичные объекты в базе данных, а не как файлы в файловой системе. Мы подробно обсудим это в «Хранение сообщений голосовой почты ODBC» в Главе 16.

IMAP

Многие люди предпочитают управлять своей голосовой почтой в рамках своей электронной почты. Это называется *унифицированным обменом сообщениями* в телекоммуникационной отрасли, и его

5 Nortel использовал для хранения своих сообщений специальный раздел в проприетарном формате, что делало невозможным извлечение сообщений из системы или отправку их по электронной почте, а также архивирование или вообще возможность что-то с ними делать.

реализация традиционно является дорогостоящей и сложной. Asterisk позволяет довольно просто интегрировать голосовую и электронную почту либо через встроенный обработчик голосовой почты, либо через связь с сервером IMAP. Мы подробно обсудим интеграцию IMAP в разделе «Интеграция IMAP голосовой почты».

Использование Asterisk в качестве автономного сервера голосовой почты

В традиционной телекоммуникационной среде сервер голосовой почты обычно представляет собой автономную единицу (представляющую либо отдельный сервер в целом, либо дополнительную карту системы). Очень немногие УАТС имели полностью интегрированную голосовую почту (в том смысле, что голосовая почта была неотъемлемой частью УАТС, а не периферийного устройства).

Asterisk вполне может выступать в качестве автономной системы голосовой почты. Двумя наиболее распространенными аргументами, которые могут потребоваться, являются:

1. Если вы создаете большую централизованную систему и имеете несколько серверов, каждый из которых предоставляет определенную функцию (прокси-сервер, медиа-шлюз, голосовую почту, конференц-связь и т.д.)
2. Если вы хотите заменить систему голосовой почты на традиционной АТС голосовой почтой Asterisk

Asterisk может выступать в любой из этих ролей.

Интеграция Asterisk в среду SIP как автономный сервер голосовой почты

Если вы хотите, чтобы Asterisk выступал в роли выделенного сервера голосовой почты (т.е. без каких-либо аппаратов, зарегистрированных на нем, и никаких других типов вызовов, проходящих через него), процесс с точки зрения диалплана довольно прост. Однако получение сообщения, ожидающего обработки, может быть немного сложнее.

Начнем с быстрой диаграммы. На Рисунке 8-3 показан слишком упрощенный пример типичной корпоративной среды SIP. У нас даже нет сервера Asterisk (кроме голосовой почты), чтобы дать вам общее представление о том, как Asterisk может выступать в качестве автономного сервера голосовой почты в среде, отличной от Asterisk.

К сожалению, Asterisk не может отправлять уведомления о сообщениях конечному оборудованию, если не знает, где оно находится. В типичной системе Asterisk, где заданная регистрация и голосовая почта обрабатываются на одном компьютере, это не является проблемой, поскольку Asterisk знает, где находятся аппараты. Но в среде, где аппараты не зарегистрированы в Asterisk, это может стать большой проблемой.

В Интернете есть несколько решений, которые рекомендуют использовать опцию `externnotify` в *voicemail.conf*, вызывая внешний скрипт, когда сообщение остается в почтовом ящике (или удаляется). Хоть мы не можем сказать, что это плохой подход, мы находим его немного глупым, и это требует от администратора понимания того, как писать внешний скрипт или программу для обработки фактического прохождения сообщения.

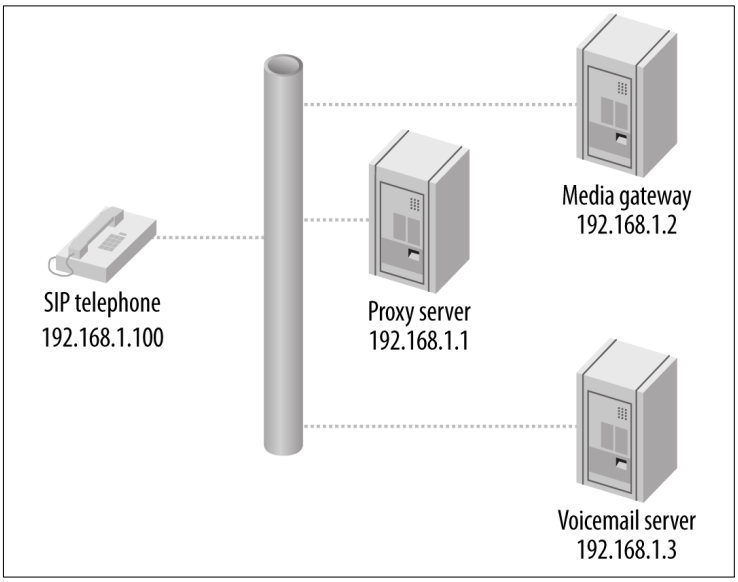


Рисунок 8-3. Упрощенная корпоративная среда SIP

Вместо этого вы можете статически определять запись для каждого почтового ящика в файле *sip.conf* сервера голосовой почты, указывая куда должны отправляться уведомления о сообщениях. Однако, вместо определения адреса каждой конечной точки, сервер голосовой почты может отправлять все сообщения в прокси-сервер, который будет обрабатывать ретрансляцию уведомлений о сообщениях соответствующим конечным точкам.

Сервер голосовой почты по-прежнему должен знать о конечных точках SIP, даже если устройства не зарегистрированы непосредственно на нем. Это можно сделать либо с помощью файла *sip.conf*, который идентифицирует каждую конечную точку SIP, либо через статическую базу данных реального времени, которая делает то же самое. Если вы используете *sip.conf* или Asterisk Realtime Architecture (ARA), для каждой конечной точки потребуется запись, подобная этой:

```
[messagewaiting](!)           ; шаблон для обработки параметров, общий
                              ; для всех почтовых ящиков
type=peer
subscribecontext=voicemailbox ; контекст диалплана на сервере голосовой
                              ; почты
context=voicemailbox                ; контекст диалплана на сервере голосовой
                                    ; почты
host=192.168.1.1                    ; ip-адрес сервера присутствия
[0000FFFF0001](messagewaiting)      ; это должно соответствовать имени
                                    ; абонента на прокси-сервере
mailbox=0000FFFF0001@DIR1           ; должен быть в формате
                                    ; почтовый_ящик@контекст_почт_ящика
defaultuser=0000FFFF0001            ; это должно соответствовать имени
                                    ; абонента на прокси-сервере
```



Обратите внимание, что динамический realtime Asterisk не будет работать с этой конфигурацией, поскольку информация пира загружается в память, только когда есть фактический вызов с участием этого пира. Поскольку уведомление о сообщении не является вызовом в отношении Asterisk, использование динамического realtime не позволит ожидать сообщения для всех пиров, не зарегистрированных в Asterisk.

Вы не захотите реализовать это, если у вас нет прототипа базовой операции решения. Хотя мы все согласны с тем, что протокол SIP является протоколом, не все согласны с правильным способом реализации протокола. В результате существует множество проблем взаимодействия, которые нуждаются в этом решении. Мы предоставили базовое введение в эту концепцию в этой книге, но детали реализации будут зависеть от других факторов, внешних по отношению к Asterisk, таких, как возможности прокси.

Тот факт, что ни одно устройство не должно регистрироваться в Asterisk, значительно снизит нагрузку на сервер Asterisk, и, в результате этот проект должен позволить серверу голосовой почты поддерживать несколько тысяч подписчиков.

Требования к диалплану

Диалоговое окно сервера голосовой почты может быть довольно простым. Должны быть удовлетворены две потребности:

1. Получать входящие звонки и направлять их в соответствующий почтовый ящик
2. Обрабатывать входящие вызовы от пользователей, желающих проверить их сообщения

Система, которая передает вызовы серверу голосовой почты, должна устанавливать некоторые заголовки SIP для передачи дополнительной информации на сервер голосовой почты. Как правило, эта информация будет включать почтовый ящик/имя пользователя, которое имеет отношение к вызову. В нашем примере мы собираемся установить заголовки `X-Voicemail-Mailbox` и `X-Voicemail-Context`, которые будут содержать информацию, которую мы хотим передать серверу голосовой почты.[6]



Если исходная система также является системой Asterisk, вы можете установить заголовки, используя приложение голосовой почты `SIPAddHeader()`, аналогично этому:

```
exten => sendtovoicemail,1,Verbose(2,Set SIP headers for voicemail)
    same => n,SIPAddHeader(X-Voicemail-Mailbox: mailbox number)
    same => n,SIPAddHeader(X-Voicemail-Context: voicemailbox)
```

Обратите внимание, что этот диалплан не переводит на сервер голосовой почты. Было бы полезно, если бы один из других серверов в вашей среде был также сервером Asterisk. Если вы используете другой тип сервера, вам нужно будет узнать, как установить пользовательские заголовки на этой платформе или выяснить, использует ли он уже определенные заголовки для такого рода вещей и, возможно, модифицировать диалплан на сервере голосовой почты для обработки этих заголовков.

Серверу голосовой почты потребуется файл *extensions.conf*, содержащий следующее:

```
[voicemailbox]
; прямые входящие вызовы в почтовый ящик
exten => Deliver,1,NoOp()
    same => n,Set(Mailbox=${SIP_HEADER(X-Voicemail-Mailbox)})
    same => n,Set(MailboxContext=${SIP_HEADER(X-Voicemail-Context)})
    same => n,VoiceMail(${Mailbox}@${MailboxContext})
    same => n,Hangup()

; подключить пользователей к своему почтовому ящику, чтобы они
; могли получать сообщения
exten => Retrieve,1,NoOp()
    same => n,Set(Mailbox=${SIP_HEADER(X-Voicemail-Mailbox)})
    same => n,Set(MailboxContext=${SIP_HEADER(X-Voicemail-Context)})
    same => n,VoiceMailMain(${Mailbox}@${MailboxContext})
    same => n,Hangup()
```

Требования к sip.conf

В файле *sip.conf* на сервере голосовой почты требуются не только записи, необходимые для всех почтовых ящиков для уведомления об ожидающем сообщении, но и для определения соединения между сервером голосовой почты и остальной средой SIP:

6 Насколько нам известно, нет никаких конкретных заголовков SIP, которые стандартизированы для такого рода вещей, поэтому вы можете именовать заголовки как пожелаете. Мы выбрали эти заголовки просто потому, что они имеют какой-то смысл. Вы можете обнаружить, что другие заголовки лучше подходят для ваших нужд.

```
[VOICEMAILTRUNK]
type=peer
defaultuser=voicemail
fromuser=voicemail
secret=s0m3th1ngs3cur3
canreinvite=no
host=<адрес прокси/сервера регистрации>
disallow=all
allow=ulaw
dtmfmode=rfc2833
context=voicemailbox
```

Другой конец соединения (возможно, ваш прокси-сервер) должен быть настроен для передачи голосовых сообщений на сервер голосовой почты.

Запуск Asterisk в качестве автономного сервера голосовой почты требует определенных знаний о кластеризации и интеграции, но вы не можете победить стоимость.

SMDI (Simplified Message Desk Interface)

Протокол Simplified Message Desk Interface (SMDI) предназначен для обмена информацией базовыми сообщениями между телефонными системами и системами голосовой почты.

Asterisk поддерживает SMDI, но, учитывая, что это старый протокол, который проходит через последовательное соединение, вероятно, будут проблемы с интеграцией. Поддержка в различных УАТС и других устройств может быть неполной. Тем не менее, это довольно простой протокол, поэтому, конечно его стоит попробовать, если вы планируете использовать Asterisk в качестве замены голосовой почты на старой УАТС.

Ниже приведено не подробное объяснение того, как настроить SMDI для Asterisk, а скорее введение в концепции, с некоторыми базовыми примерами. Если вы планируете внедрять SMDI, вам нужно будет написать сложную логику диалплана и хорошо понять, как подключать системы через последовательные соединения.[7]

SMDI включен в Asterisk с использованием двух опций в разделе `[general]` файла *voicemail.conf*:

```
smdienable=yes
smdiport=/dev/ttyS0     ; или любой другой последовательный порт, к
                        ; которому вы подключаете свой SMDI-сервис
```

Кроме того, вам понадобится файл *smdi.conf* в папке */etc/asterisk* чтобы определить детали конфигурации SMDI. Он должен выглядеть примерно так (см. файл *smdi.conf.sample* для получения дополнительной информации о доступных параметрах):

```
[interfaces]
charsize=7
paritybit=even
baudrate=1200           ; надеюсь, поддерживается более высокий битрейт
smdiport=/dev/ttyS0     ; или любой другой последовательный порт, который вы
                        ; будете использовать для обработки сообщений SMDI
                        ; на Asterisk

[mailboxes]       ; сопоставить входящие строки цифр (обычно номера
                  ; DID) с верным почтовый_ящик@контекст в
                  ; voicemail.conf
smdiport=/dev/ttyS0     ; сначала объявите, какой порт SMDI будут
                        ; использовать следующие почтовые ящики
```

7 Если у вас нет опыта в настройке и устранении неполадок последовательных подключений, вы можете обнаружить, что весь этот процесс несет намного больше проблем, чем того стоит.

```
4169671111=1234@default
4165551212=9999@default
```

В диалплане есть две функции, которые будут нужны для конфигурации SMDI. Функция `SMDI_MSG_RETRIEVE()` вытаскивает соответствующее сообщение из очереди сообщений SMDI. Вам необходимо передать функции ключ поиска (обычно DID, на который ссылается в сообщении), и он вернет идентификационный номер, на который может ссылаться функция `SMDI_MSG()`:

```
SMDI_MSG_RETRIEVE(<smdi port>,<search key>[,timeout[,options]])
```

После того, как у вас есть идентификатор сообщения SMDI, вы можете использовать функцию `SMDI_MSG()` для доступа к различным сведениям о сообщении, таким как `station`, `callerID`, и `type` (тип сообщения SMDI):

```
SMDI_MSG(<message_id>,<component>)
```

В вашем диалплане вам нужно будет обрабатывать поиск входящих сообщений SMDI, чтобы обеспечить правильную обработку вызовов. Например, если входящий вызов предназначен для доставки в почтовый ящик, тип сообщения может быть одним из `B` (для занято) или `N` (для неотвеченных вызовов). Если, с другой стороны, вызов предназначен для перехода на `VoiceMailMain()`, поскольку вызывающий абонент хочет получить свои сообщения, тип сообщения SMDI будет `D`, и это нужно будет обрабатывать.

Интеграция базы данных

Приложение голосовой почты Asterisk может быть интегрировано в базу данных. Это может быть очень полезно, особенно в кластерных и распределенных системах. Это подробно обсуждается в Главе 16.

Вывод

В то время как система голосовой почты Asterisk довольно старая с точки зрения кода Asterisk, тем не менее она представляет из себя мощное приложение, которое может (и делает) успешно конкурировать с дорогостоящими, запатентованными системами голосовой почты.

ГЛАВА 9

Интернационализация

Дэвид Дюффетт

Я много путешествовал по всему миру,
и я хорошо ладил во всех этих зарубежных странах,
потому что у меня есть теория, что это их страна,
и они получили право запускать ее, как хотят.
-Уилл Роджерс

Телефония - одна из тех областей жизни, где, будь то дома или на работе, людям не нравятся сюрпризы. Когда люди используют телефоны, что-либо вне нормы - это нежелательное ожидание, и, как человек, который, вероятно, занимается поставками телефонных систем, вы будете знать, что неудовлетворернные ожидания могут привести к невыразимым страданиям с точки зрения дополнительной работы, потери денег и другим проблемам, связанным с неудовлетворенностью клиентов.

В дополнение к тому, что пользовательский опыт соответствует ожиданиям пользователей, также необходимо, чтобы ваш Asterisk чувствовал себя «дома». Например, если исходящий вызов проходит по аналоговой линии (FXO), Asterisk будет необходимо интерпретировать тоны, которые он «слышит» на линии (занят, звонит и т.д.).

По умолчанию (и, возможно, как и следовало ожидать, поскольку он был «рожден в США»), Asterisk настроен на работу в Северной Америке. Однако, поскольку Asterisk развертывается во многих местах и (к счастью) люди со всего мира вносят свой вклад в это, вполне возможно настроить Asterisk для правильной работы практически в любом месте, где вы решите его развернуть.

Если вы читали эту книгу с самого начала от главы к главе вы уже сделали некоторые шаги во время установки и начальной настройки, которые заставили бы ваш Asterisk работать в вашей локальной сети (и оправдать ожидания ваших клиентов) ,

В нескольких главах этой книги содержится информация, которая поможет вам интернационализировать[1] или (возможно, более правильно) локализовать реализацию Asterisk. Цель этой главы - предоставить единое место, где все аспекты изменений, которые необходимо внести в вашу телефонную систему на основе Asterisk, могут быть рассмотрены, обсуждены и объяснены. Причина использования фразы «телефонная система на основе Asterisk», а не просто «Asterisk» заключается в том, что некоторые изменения необходимо будет сделать в других частях системы (IP-телефоны, ATA и т.д.), Тогда как другие изменения будут реализованы в конфигурационных файлах Asterisk и DAHDI.

1 i18n - термин, используемый для сокращения слова *интернационализация* из-за его длины. Формат *<первая_буква><число><последняя_буква>*, где *<число>* - количество букв между первой и последней буквами. Другие слова, такие как локализация (L10n), модуляция (m12n) и т. д., также нашли дом с этой схемой, которую Лейф находит немного смешной. Более подробную информацию можно найти в глоссарии W3C онлайн.

Начнем с того, что мы собрали список (в определенном порядке) вещей, которые может потребоваться изменить, чтобы оптимизировать вашу телефонную систему на базе Asterisk для определенного места за пределами Северной Америки. Можете выкрикнуть, если хотите ...

- Язык/акцент на подсказки
- Физическое соединение для интерфейсов PSTN (FXO, BRI, PRI)
- Тоны, которые слышат пользователи IP-телефонов и/или ATA
- Формат CallerID вызывающего абонента, отправленный и/или полученный аналоговыми интерфейсами
- Тоны для аналоговых интерфейсов, которые должны быть подставлены или обнаружены Asterisk
- Формат штампов времени и даты для голосовой почты
- Как указано выше, отметки времени/даты, объявленые в Asterisk
- Шаблоны в диалплане (IP-телефонов, ATA и самого Asterisk, если вы используете образец диалплана)
- Способ указания на аналоговое устройство, которое ожидает голосовую почту (MWI)
- Тоны, передаваемые абонентам Asterisk (они вступают в игру, когда пользователь «внутри» системы, например, сигналы, слышимые во время передачи вызова)

Мы рассмотрим все в этом списке, приняв стратегию работы от внешнего края системы к самому ядру (самому Asterisk). В заключение мы получим удобный контрольный список того, что вам может понадобиться изменить, и где это изменить.

Хотя принципы, описанные в этой главе, позволят вам адаптировать вашу установку Asterisk специально для вашего региона (или вашего клиента), для удобства, все наши примеры будут сосредоточены на том, как адаптировать Asterisk для одного региона: Объединенное королевство.

Внешние устройства для сервера Asterisk

Существуют серьезные различия между хорошим старомодным аналоговым телефоном и любым из большого количества IP-телефонов и нам нужно подобрать одно из действительно фундаментальных различий, чтобы пролить свет на следующее разъяснение, охватывающее настройки, которые может потребоваться изменить на устройствах, внешних по отношению к Asterisk, например, на IP-телефонах.

Вы когда-нибудь считали, что аналоговый телефон является абсолютно немым устройством (мы знаем, что базовая модель очень и очень дешевая), которое должно подключаться к интеллектуальной сети (PSTN), тогда как IP-телефон (например, SIP или IAX2) - очень интеллектуальное устройство, которое подключается к немой сети (Интернет или любая обычная IP-сеть)? Рисунки 9-1 и 9-2 иллюстрируют разницу.

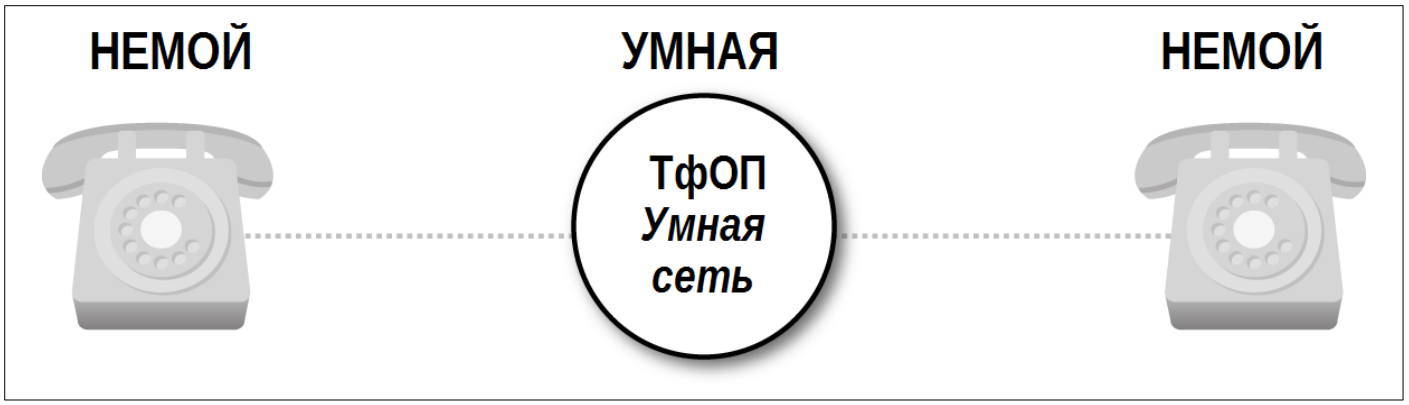


Рисунок 9-1. В старые времена: немое устройство подключается к умной сети

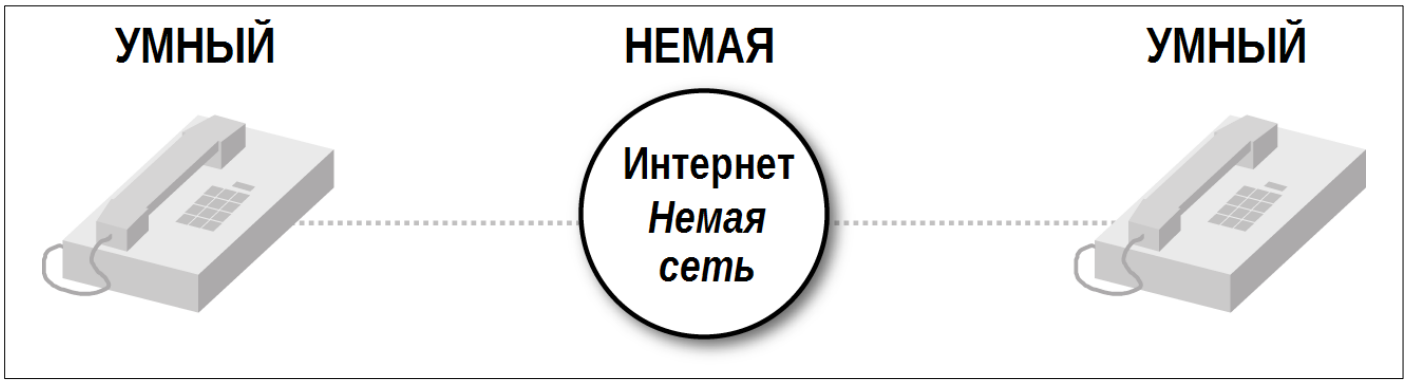


Рисунок 9-2. Ситуация сегодня: умные устройства подключаются через немую сеть

Можем ли мы взять два аналоговых телефона, подключить их напрямую друг к другу и иметь функциональность, которую мы обычно связываем с обычным телефоном? Нет, конечно нет, потому что сеть поставляет все: фактическое питание телефона, тон готовности (от местной станции или CO), информацию callerID, тон вызова (от удаленной [ближайшей к телефону назначения] станции или CO), все необходимые сигналы и т.д.

И наоборот, можем ли мы взять два IP-телефона, подключить их напрямую друг к другу и получить некоторые разумные функции? Конечно, мы могли бы, потому что весь интеллект находится внутри самих IP-телефонов - они обеспечивают слышимые тональные сигналы (тон готовности, вызова, занятости) и запускают протокол, который выполняет всю необходимую сигнализацию (обычно SIP). Фактически, вы можете попробовать это у себя: у большинства средних IP-телефонов есть встроенный коммутатор Ethernet, поэтому вы можете фактически подключить два IP-телефона напрямую друг к другу с помощью обычного (прямого) Ethernet-кабеля или просто подключить их через обычный коммутатор. Им нужно будет иметь фиксированные IP-адреса в отсутствие DHCP-сервера, и вы можете набирать IP-адрес другого телефона только с помощью клавиши * для точек в адресе.

Рисунок 9-2 указывает на то, что на IP-телефоне мы отвечаем за настройку всего того, что сеть предоставила бы в старые времена. Это можно сделать одним из (по крайней мере) двух способов. Первый - настроить тональные сигналы, предоставляемые IP-телефоном, на собственном веб-интерфейсе устройства. Это делается путем просмотра IP-адреса телефона (IP-адрес обычно можно получить с помощью опции меню на телефоне), а затем выбрав соответствующие параметры. Например, на IP-телефоне Yealink тональные сигналы задаются на странице веб-интерфейса *Телефон* на вкладке *Тоны* (где вы найдете список различных типов тонов, которые можно изменить - в случае Yealink, это Dial, Ring Back, Busy, Congestion, Call Waiting, Dial Recall, Record, Info, Stutter, Message и Auto Answer).

Другой способ, которым эта конфигурация может быть применена, - это автоматическое конфигурирование (автопровижинг) телефона с этими настройками. Полное объяснение механизма автопровижинга выходит за рамки этой книги, но вы можете обычно настраивать тоны в соответствующих атрибутах соответствующих элементов в файле XML.

Хотя мы меняем настройки на IP-телефонах, есть две другие вещи, которые может потребоваться изменить, чтобы телефоны выглядели правильно и правильно функционировали как часть системы.

Большинство телефонов отображают время в режиме ожидания, и многие люди считают особенно раздражающим, когда их телефоны показывают неправильное время, поэтому нам нужно убедиться, что отображается правильное местное время. Должно быть достаточно легко найти соответствующую страницу веб-интерфейса (или атрибутов XML), чтобы указать сервер времени. Вы также обнаружите, что есть настройки для летнего времени и других соответствующих вещей.

Последнее, что нужно изменить, - это потенциальный тупик, касающийся телефонного звонка - диалплан. Мы говорим не о диалплане, который находится в */etc/asterisk/extensions.conf*, а об диалплане телефона. Не все понимают, что IP-телефоны также имеют диалплан, хотя эти диалпланы больше связаны с тем, какие строки набора разрешены, чем с тем, что делать при данном наборе.

Главное правило заключается в том, что если трубка положена, встроенный диалплан обходится, но если вы возьмете трубку, диалплан начнет исполняться, и может случиться так, что диалплан не позволит вам использовать строку набора номера для вызова. Хотя эта проблема может проявиться с отказом телефона передавать определенные типы номеров через Asterisk, это может также повлиять на любые коды функций, которые вы планируете использовать. Это можно легко устранить с помощью гугления номера модели телефона вместе с «UK dialplan» (или конкретный регион, который вам нужен), или вы можете перейти на соответствующую страницу в веб-интерфейсе и вручную настроить диалплан или выбрать соответствующую страну в выпающем списке (в зависимости от типа телефона, с которым вы работаете).

Предыдущее обсуждение конфигурации IP-телефона также применимо к любым аналоговым телефонным адаптерам (ATA), которые вы планируете использовать, в частности, для тех, которые поддерживают интерфейс FXS. Кроме того, Вам может потребоваться указать некоторые электрические характеристики интерфейса телефонии, такие как линейное напряжение и импеданс, вместе с форматом CallerID, который будет работать с локальными телефонами. Все, что отличается, - это то, как вы получите IP-адрес для входа в веб-интерфейс. Обычно это делается путем набора определенного кода на подключенном аналоговом телефоне, после чего, IP-адрес считывается вызывающему абоненту.

Конечно, ATA может также иметь интерфейс FXO, который также должен быть сконфигурирован для правильного взаимодействия с аналоговой линией, предоставляемой в вашем регионе. Типы параметров, которые необходимо изменить, аналогичны интерфейсу FXS.

Что делать, если вы подключаете аналоговый телефон или линию к карте Digium? Мы рассмотрим это дальше.

Подключение к ТфОП, DAHDI, карты Digium и аналоговые телефоны

Прежде чем мы перейдем к конфигурации DAHDI и Asterisk, нам необходимо физически подключиться к ТфОП. К сожалению, мировых стандартов для этих соединений нет; на самом деле часто встречаются изменения в одной стране в зависимости от региона.

Интерфейсы первичной скорости (PRI) в наши дни обычно оканчиваются соединением RJ45, хотя импеданс соединений может меняться. В некоторых странах (особенно в Южной Америке) все еще можно найти PRI, которые заканчиваются двумя BNC-разъемами: один для передачи и один для приема.

Вообще говоря, PRI, завершенный в RJ45, будет соединением ISDN, и если вы обнаружите, что соединение выполнено с помощью пары разъемов BNC (коаксиальный разъем нажать-и-повернуть), указывает на вероятность того, что вы имеете дело с CAS-протоколом (например, R2).

На Рисунке 9-3 показан требуемый адаптер, если ваш телефон снабжен разъемами BNC (для плат Digium требуется соединение RJ45). Он называется *balun*, поскольку преобразует сбалансированное соединение (RJ45) в несимметричное (BNC) в дополнение к изменению импеданса соединения.



Интерфейсы базовой скорости (BRI) распространены в континентальной Европе и почти всегда поставляются через соединение RJ45.

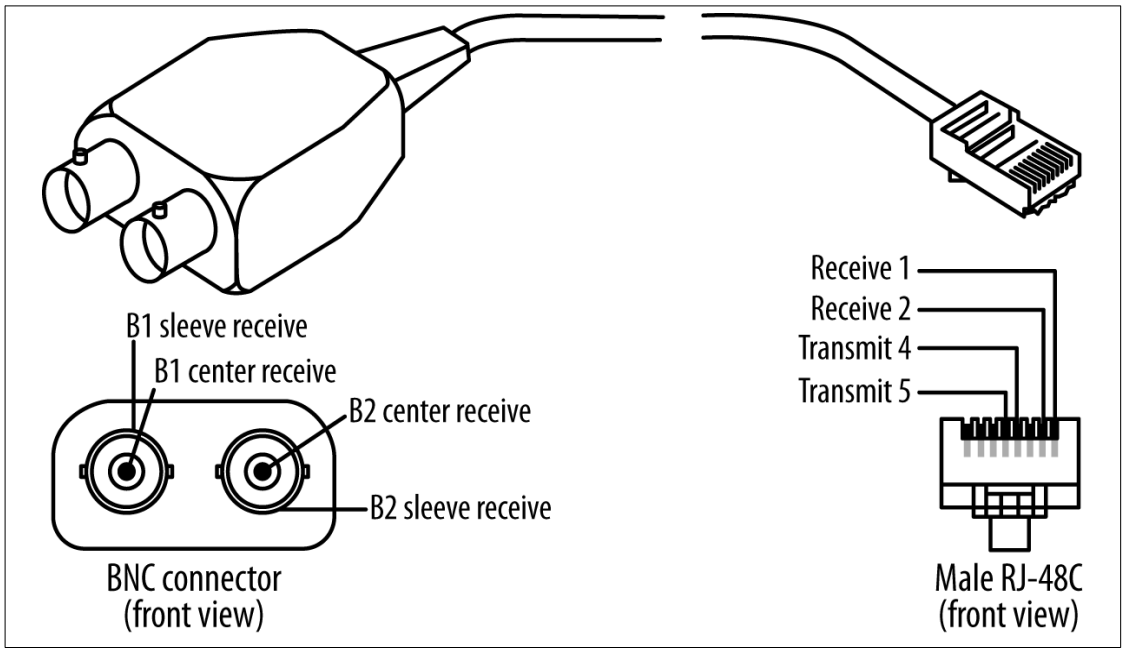


Рисунок 9-3. Balun

Аналоговые соединения значительно варьируются от места к месту - вы узнаете, какой разъем используется в вашей местности. Важно помнить, что аналоговая линия состоит всего из двух проводов, и они должны подключаться к средним двум выводам штекера RJ11, который входит в карту Digium, а другой конец является локальным. На Рисунке 9-4 показан штекер, используемый в Великобритании, где два провода подключены к контактам 2 и 5.



Рисунок 9-4. Штепсельная вилка BT, используемая для аналоговых соединений PSTN в Великобритании (обратите внимание, что имеются только контакты 2-5)

Digium Asterisk Hardware Device Interface (интерфейс Asterisk аппаратного устройства Digium), или DAHDI, фактически охватывает несколько вещей. Он содержит драйверы ядра для карт адаптеров телефонии, которые работают в рамках DAHDI, а также утилиты автоматической настройки и инструменты тестирования. Эти части содержатся в двух отдельных пакетах (*dahdi-linux* и *dahdi-tools*), но мы также можем использовать один полный пакет, называемый *dahdi-linux-complete*. Все три пакета доступны на сайте Digium. Установка DAHDI была рассмотрена в Главе 3.

В Главе 7 описано использование аналоговых и цифровых соединений PSTN и мы не будем повторять эти детали здесь. Если вы используете цифровые PSTN-соединения, ваша задача - выяснить, какое соединение вы получаете от телекоммуникационной компании. Как правило, если вы запросили PRI, это будет T1 в Северной Америке, J1 в Японии или E1 в значительной части остального мира.

После того, как вы установили тип подключения PRI, предоставленный вами телефонной компанией, вам потребуется дополнительная информация, необходимая для правильной настройки DAHDI и Asterisk (например, является ли соединение ISDN или CAS-протоколом). Опять же, вы найдете их в Главе 7.

Драйверы DAHDI

Соединения, в которых должна выполняться некоторая реальная локализация - это соединения аналоговых интерфейсов. Для того, чтобы настроить телефонную систему на основе Asterisk для улучшенной работы в данной местности, вам сначала нужно будет сконфигурировать некоторые низкоуровневые аспекты того, как карта Digium взаимодействует с подключенным устройством или линией. Это делается с помощью драйверов ядра DAHDI в файле под названием */etc/dahdi/system.conf*.

В следующих строках (взятых из примера конфигурации, которую вы получаете со свежей установкой DAHDI) вы найдете как настройки `loadzone`, так и настройки `defaultzone`. Настройка `loadzone` позволяет выбрать, какой комплект(ы) тональных сигналов будет воспроизводить карта (для подачи на аналоговые телефоны) и распознавать (на подключенных аналоговых телефонных линиях):

```
# Данные тональных зон
# ^^^^^^^^^^^^^^
# Наконец, вы можете предварительно загрузить некоторые тональные зоны, чтобы
# они не могли быть перезаписаны другими пользователями (если вы разрешаете
# сторонним пользователям открывать /dev/dahdi/* какие-нибудь интерфейсы).
# Также это означает, что их не нужно загружать во время выполнения.
# Формат - «loadzone=<zone>», где zone является двухбуквенным кодом страны.
#
# Вы также можете указать зону по умолчанию с «defaultzone=<zone>», где zone
# является двухбуквенным кодом страны.
#
#
# Обновленный список зон можно найти в файле zonedata.c

#
loadzone = us
#loadzone = us-old
#loadzone=gr
#loadzone=it
#loadzone=fr
#loadzone=de
#loadzone=uk
#loadzone=fi
#loadzone=jp
#loadzone=sp
#loadzone=no
#loadzone=hu
#loadzone=lt
#loadzone=pl
defaultzone=us
#
```



Файл */etc/dahdi/system.conf* использует символ хеша (#), чтобы указать комментарий вместо точки с запятой (;) как файлы в */etc/asterisk*.

Хотя можно загрузить несколько различных наборов тонов (вы можете увидеть все наборы тонов подробно в *zonedata.c*) и переключаться между ними, в большинстве практических ситуаций вам понадобится только:

```
loadzone=uk       # для загрузки набора тонов
defaultzone=uk    # по умолчанию DAHDI использует этот набор
```

... или то, что вам нужно для вашего региона.

Если вы выполняете *dahdi_genconf* для автоматической (или должно быть автомагической?) настроки своих адаптеров DAHDI, вы заметите, что во вновь созданном */etc/dahdi/system.conf* будет использоваться по умолчанию как для `loadzone`, так и для `defaultzone` значение `us`. Несмотря на предупреждения о том, чтобы не редактировать файлы вручную, вполне можно изменить эти настройки на то, что вам нужно.

Если вам интересно, как мы определяем, есть ли в почтовом ящике, связанном с каналом, подключенным к аналовогому телефону, какие-либо голосовые сообщения, это делается с заикающимся тоном набора. Формат этого заикающегося гудка определяется с помощью комбинации `loadzone/defaultzone`, которую вы использовали.

В отличие от аналоговых телефонов, у которых есть индикатор ожидания сообщения (например, светодиод или лампочка, которые мигают, чтобы указывать на новую голосовую почту), вы можете автоматически переключаться между ними и прослушивать заикающийся тон набора. Вы можете быть свидетелем этого, наблюдая за командной строкой Asterisk, чтобы увидеть, насколько активен канал DAHDI (если вам нечего делать!).

Это на уровне DAHDI. Мы выбрали протокол(ы) для соединений PRI или BRI, тип сигнализации для аналоговых каналов (все рассмотрены в Главе 7) и тоны для аналоговых соединений, которые только что обсуждались.

Связь между Linux, DAHDI и Asterisk (и, следовательно, */etc/dahdi/system.conf* и */etc/asterisk/chan_dahdi.conf*) показана на Рисунке 9-5.



После того, как вы завершили настройку на уровне DAHDI (в */etc/dahdi/system.conf*), вам необходимо выполнить *dahdi_cfg -vvv*, чтобы DAHDI перечитал конфигурацию. Это также хорошее время для использования *dahdi_tool*, чтобы проверить все ли в порядке на уровне Linux.
Таким образом, если после настройки Asterisk для работы с адаптерами DAHDI если что-то не работает должным образом, вы можете быть уверены, что проблема ограничена *chan_dahdi.conf* (или *#include dahdi-channels.conf*, если вы используете эту часть вывода *dahdi_genconf*).

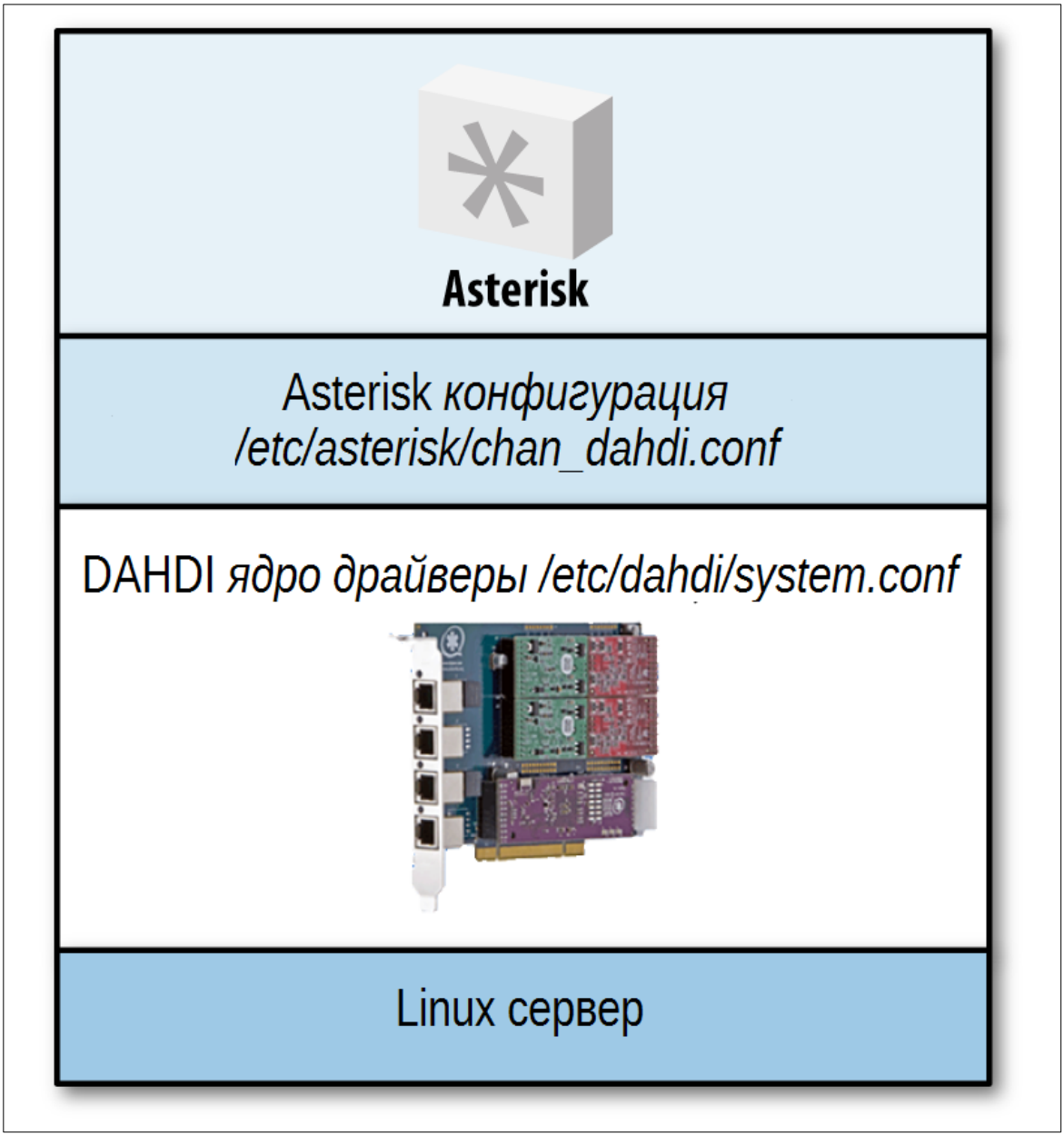


Рисунок 9-5. Связь между Linux, DAHDI, и Asterisk

Asterisk

Теперь, когда всё настроено на уровне Linux, нам нужно настроить Asterisk, чтобы использовать только что подключенные каналы на уровне Linux и настроить способ, которым Asterisk интерпретирует и генерирует информацию, которая поступает или выходит за пределы этих каналов. Эта работа выполняется в */etc/asterisk/chan_dahdi.conf*.

В этом файле мы не только расскажем Asterisk о том, какие каналы у нас есть (эти настройки будут соответствовать тому, что мы уже сделали в DAHDI), но и настроим ряд вещей, которые гарантируют, что Asterisk хорошо подходит для своего нового дома.

Caller ID

Ключевым компонентом этого изменения является CallerID. Хотя методы доставки CallerID являются довольно стандартными в мире BRI и PRI, они широко варьируются в аналоговом мире; таким образом, если вы подключили американский аналоговый телефон к телефонной сети Великобритании, он действительно работал бы как телефон, но информация CallerID не отображалась бы. Это связано с тем, что он передается по-разному в разных местах по всему миру, а американский телефон будет искать сигнализацию CallerID в формате США, в то время как

телефонная сеть Великобритании будет поставлять его (если он включен - это нестандартно в Великобритании, вам нужно заплатить за CallerID!) в британском формате.

Мало того, что формат отличается, метод указания телефона (или Asterisk) для поиска CallerID может отличаться и от места к месту. Это важно, так как мы не хотим, чтобы Asterisk тратил время на поиск информации об CallerID, если он не отображается на линии.

Опять же, Asterisk по умолчанию использует североамериканский формат CallerID (нет записей в */etc/asterisk/chan_dahdi.conf*, это просто по умолчанию), и для его изменения нам нужно будет сделать несколько записей, описывающих технические детали системы CallerID. В случае Великобритании доставка информации CallerID сигнализируется изменением полярности на телефонной линии (другими словами, стороны A и B пары телефонных проводов временно переключаются), а фактическая информация CallerID предоставляется в формате, известном как V.23 (частотная манипуляция или FSK). Таким образом, записи в *chan_dahdi.conf* для получения CallerID в Великобритании на любых интерфейсах FXO будет выглядеть так:

```
cidstart=polarity ; доставка caller ID будет путем
                  ; смены полярности сигнализации
cidsignalling=v23 ; доставка информации called ID
                  ; будет в формате V23
```

Конечно, Вам также может потребоваться отправить CallerID используя ту же самую локальную информацию о сигнализации, на любые аналоговые телефоны, подключенные к FXS-интерфейсам, и может потребоваться еще одна запись, так как в некоторых местах информация CallerID отправляется после определенного количества звонков. Если это так, вы можете использовать эту запись:

```
sendcalleridafter=2
```

Прежде чем вы сможете сделать эти записи, вам нужно будет установить информацию о вашей местной системе CallerID (кто-то из вашей местной телефонной компании или Google может быть вашим в этом, но есть также хорошая информация в образце */etc/asterisk/chan_dahdi.conf*).

Язык и/или акцент подсказок

Как вы знаете, подсказки (или записи), которые будет использовать Asterisk, хранятся в */var/lib/asterisk/sounds*. В старых версиях Asterisk все звуки были в этом фактическом расположении, но в наши дни вы найдете ряд подкаталогов, которые позволяют использовать разные языки или акценты. Имена этих подкаталогов произвольны; вы можете называть их как пожелаете.

Обратите внимание, что имена файлов в этих каталогах должны быть такими, какие ожидает Asterisk, например, в */var/lib/asterisk/sound/en*, файл *hello.gsm* будет содержать слово «Hello» (скажет прекрасная Эллисон), тогда как *hello.gsm* в */var/lib/asterisk/sounds/es* (для испанского в этом случае) будет содержать слово «Hola» (голосом испанской эквиваленты прекрасной Эллисон[2]).

По умолчанию используется каталог: /var/lib/asterisk/sounds/en, так как же вы его измените?

Есть два пути. Один из них заключается в том, чтобы установить язык в файле конфигурации канала, который звонит, используя языковую директиву. Например, строка:

```
language=en_UK
```

помещенная в *chan_dahdi.conf*, *sip.conf* и т.д. (для применения вообще или только для данного канала или профиля) будет указывать Asterisk на использование звуковых файлов, найденных в */var/lib/asterisk/sounds/en_UK* (которые могут содержать британские акценты) для всех вызовов, которые поступают по этим каналам.

2 Кто, по сути, та же Эллисон, которая делает английские подсказки; Июнь Уоллак делает французские подсказки. Мужской австралийских акцент сделан Камероном Туми. Все таланты озвучивания доступны для записи дополнительных подсказок. Дополнительную информацию см. на странице Digium IVR.

Другой способ - изменить язык во время телефонного звонка через диалплан. Это (наряду со многими атрибутами индивидуального вызова) можно установить с помощью функции диалплана `CHANNEL()`. См. Главу 10 для полного изучения функций диалплана.

Следующий пример позволит вызывающему абоненту выбрать один из трех языков для продолжения вызова:

```
; Предоставляет выбор (1) Французский, (2) Испанский, или (3) Немецкий
exten => s,1,Background(choose-language)
    same => n,WaitExten(5)

exten => 1,1,Set(CHANNEL(language)=fr)
exten => 2,1,Set(CHANNEL(language)=es)
exten => 3,1,Set(CHANNEL(language)=de)
; следующий приоритет для для добавочных номеров 1, 2, или 3 будет
; рассмотрен здесь
exten => _[123],n,Goto(menu,s,1)
```

Если вызывающий абонент нажал `1`, будут воспроизводиться звуки из */var/lib/asterisk/sounds/fr*; если он нажал `2`, звуки появятся из */var/lib/asterisk/sounds/es* и т.д.

Как уже упоминалось, имена этих каталогов произвольны и необязательно должны содержать только два символа - главное, чтобы они соответствовали названию поддиректории, которую создали в директиве `language` в конфигурации канала, или когда вы устанавливаете аргумент `CHANNEL(language)` в диалплане.

Штампы время/дата и произношение

Asterisk использует системное время Linux с хост-сервера, как и следовало ожидать, но у нас могут быть пользователи системы, которые находятся в разных часовых поясах или даже в разных странах. Голосовая почта - это то место, где резина попадает на дорогу, так как здесь пользователи вступают в контакт с информацией об отметке времени/даты.

Рассмотрим сценарий, при котором некоторые пользователи системы находятся в США, а другие - в Великобритании.

Помимо разницы во времени, еще одна вещь, которую следует учитывать, - это то, какой формат даты и времени люди используют в разных местах - в США даты обычно указываются как месяц, день, год и время, указанные в 12-часовом формате (например, 2:54 PM).

В британских датах же указываются день, месяц, год и время, которые часто указываются в формате 24-часов (14:54) - хотя некоторые люди в Великобритании предпочитают 12-часовой формат, поэтому мы рассмотрим его тоже.

Поскольку все эти вещи связаны с голосовой почтой, вы могли бы догадаться, что мы сконфигурируем их в файле */etc/asterisk/voicemail.conf* - в частности в разделе `[zonemessages]`.

Вот часть `[zonemessages]` образца файла *voicemail.conf*, который поставляется с Asterisk, с UK24 (для британских людей, которые любят 24-часовой формат времени) и UK12 (для британцев, которые предпочитают 12-часовой формат времени) добавлены зоны:

```
[zonemessages]
; Пользователи могут находиться в разных часовых поясах или могут иметь
; разные сообщения для своего ввода сообщения, когда они входят в
; систему голосовой почты. Установите сообщение и часовой пояс, который
; каждый пользователь слышит здесь. Установите используя одну из этих зон
; с помощью tz=attribute в поле параметров почтового ящика. Конечно,
; замена языка по-прежнему применяется здесь, поэтому у вас может быть
; несколько деревьев каталогов, которые имеют альтернативный выбор языка.
```

```
;
; Смотрите /usr/share/zoneinfo/ для имён timezone.
; Посмотрите страницу руководства для strftime для быстрого руководства
; о том, как выполняется замена переменных по приведенным ниже значениям.
;
; Поддерживаемые значения:
; 'filename' имя звукового файла (требуются одинарные кавычки вокруг
; имени файла)
; ${VAR} переменная замены
; A или a День недели (Saturday, Sunday, ...)
; B или b или h Название месяца (January, February, ...)
; d или e число месяца (first, second, ... thirty-first)
; Y Год
; I или l Час, 12 часовой формат
; H Hour, 24 часовой формат (одна цифра часов, предшествующиая "oh")
; k Hour, 24 часовой формат (одна цифра часов, НЕ предшествующая "oh")
; M Минуты, с 00 произносится как "o'clock"
; N Минуты, с 00 произносится как "hundred"(сто) (US военное время)
; P или p AM или PM (до полудня и после)
; Q "today"(сегодня), "yesterday"(вчера) или ABdY
; (*примечание: нестандартное значение strftime)
; q " (для today), "yesterday", weekday, или ABdY
; (*примечание: нестандартное значение strftime)
; R 24 часовое время, включая минуты
;
eastern=America/New_York|'vm-received' Q 'digits/at' IMp
central=America/Chicago|'vm-received' Q 'digits/at' IMp
central24=America/Chicago|'vm-received' q 'digits/at' H N 'hours'
military=Zulu|'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'
european=Europe/Copenhagen|'vm-received' a d b 'digits/at' HM

UK24=Europe/London|'vm-received' q 'digits/at' H N 'hours'
UK12=Europe/London|'vm-received' Q 'digits/at' IMp
```

Эти зоны не только определяют время, но также определяют, как упорядочиваются и считываются время и даты.

Создав эти зоны, мы можем перейти к разделу контекста голосовой почты в *voicemail.conf*, чтобы связать соответствующие почтовые ящики с правильными зонами:

```
[default]
4001 => 1234,Russell Bryant,rb@shifteight.org,,|tz=central
4002 => 4444,David Duffett,dd@shifteight.org,,|tz=UK24
4003 => 4450,Mary Poppins,mp@shifteight.org,,|tz=UK12|attach=yes
```

Как вы можете видеть, когда мы объявляем почтовый ящик, мы также (необязательно) связываем его с определенной зоной. Полную информацию о голосовой почте можно найти в Главе 8.

Последнее, что нужно локализовать в нашей конфигурации Asterisk - это тональные сигналы, воспроизводимые Asterisk вызывающим абонентам, когда они находятся внутри системы (например, тона, слышымые звонящим во время трансфера).

Как было указано ранее в этой главе, начальные тоны, которые люди слышат при звонке в систему, поступают от IP-телефона или от DAHDI для аналоговых каналов.

Эти тоны установлены в */etc/asterisk/indications.conf*. Вот часть файла образца, где вы можете видеть данный регион, указанный в директиве страны. Нам просто нужно изменить код страны:

```
;
; indications.conf
```

```
; Конфигурационный файл для определения местоположения
;
; Замечание:
; При добавлении стран в этот файл, пожалуйста, держите их в алфавитном
; порядке в соответствии с кодами стран в 2 символа!
;
; Категория [general] - это определенные глобальные переменные.
; Все остальные категории интерпретируются как указание местоположения
;
[general]
country=uk  ; по умолчанию US, поэтому мы изменили на UK
```

Ваш диалплан должен будет отражать схему нумерации для вашего региона. Если вы еще не знаете схему для своего региона, ваш местный регулятор телекоммуникаций, как правило, сможет предоставить подробную информацию об этом. Кроме того, пример диалплана в файле */etc/asterisk/extensions.conf*, конечно же, заполнен североамериканскими номерами и шаблонами.

Вывод — Простая шпаргалка

Как вы теперь видите, есть несколько вещей, которые нужно изменить чтобы полностью локализовать вашу телефонную систему на базе Asterisk и не все они находятся в конфигурации Asterisk или даже DAHDI - некоторые вещи необходимо изменить на подключенных IP-телефонах или ATA.

Прежде чем покинуть эту главу, взгляните на Таблицу 9-1: Шпаргалка чтобы узнать что изменить и где это изменить.

Таблица 9-1. Шпаргалка интернационализации

Что менять	Где менять
Тоны прогресса вызова	• IP-телефоны - на самом телефоне • ATA - на самом ATA • Аналоговые телефоны—DAHDI (*/etc/dahdi/system.conf*)
Тип PRI/BRI и протокол	DAHDI—*/etc/dahdi/system.conf* и */etc/asterisk/chan_dahdi.conf*
Физические соединения с ТфОП	• Balun, если требуется для PRI • Подключите аналоговую пару к средним 2 контактам RJ11, подключающимся к карте Digium
Caller ID на аналоговых линиях	Asterisk—*/etc/asterisk/chan_dahdi.conf*
Голосовые подсказки и/или акцент	• Канал—*/etc/asterisk/sip.conf*, */etc/asterisk/iax.conf*, */etc/asterisk/chan_dahdi.conf*, etc. • Диалплан—функция `CHANNEL(language)`
Голосовая почта штамп времени/даты и произношение	Asterisk—*/etc/asterisk/voicemail.conf*
Тоны, представляемые Asterisk	Asterisk—*/etc/asterisk/voicemail.conf*

Пусть после всех развертываний Asterisk чувствует себя как дома…

Погружение в диалплан

Для списка всех способов, по которым технология не улучшила качество жизни, пожалуйста, нажмите три.

—Элис Кан

Хорошоу. У вас есть основы диалпланов, но вы знаете, что многое еще впереди. Если вы еще не поняли Главу 6 - вернитесь назад и прочитайте еще раз. Мы собираемся заняться более продвинутыми темами.

Выражения и манипуляции с переменными

По мере того, как мы начинаем погружаться в более глубокие аспекты диалплана, настало время познакомить вас с несколькими инструментами, которые значительно повысят мощь, которую вы можете использовать в своем диалплане. Эти конструкции добавляют невероятный интеллект к вашему диалплану, позволяя ему принимать решения на основе разных критериев, которые вы определяете. Наденьте свой мыслительный колпачок, и давайте начнем.



В этой главе мы используем лучшие практики, которые были разработаны на протяжении многих лет в создании диалплана. Основная из них - вы заметите, что все первые приоритеты начинаются с приложения `NoOp()`, что означает просто «Нет операции»; ничего функционального не произойдет. Другая заключается в том, что все следующие строки начинаются с `same => n`, что является ярлыком, который гласит: «Использовать то же расширение, которое было только что определено ранее». Кроме того, отступ составляет четыре пробела.

Базовые выражения

Выражения представляют собой комбинации переменных, операторов и значений, которые вы объединяете вместе для получения результата. Выражение может проверять значения, изменять строки или выполнять математические вычисления. Допустим, у нас есть переменная с именем `COUNT`. На простом русском языке два выражения, использующие эту переменную, могут быть «`COUNT` плюс 1» и «`COUNT` разделить на 2.». Каждое из этих выражений имеет конкретный результат или значение в зависимости от значения данной переменной.

В Asterisk выражения всегда начинаются со знака доллара и открытой квадратной скобки и заканчиваются закрывающей квадратной скобкой, как показано здесь:

```
$[expression]
```

Таким образом, мы могли бы написать два наших примера:

```
$[${COUNT} + 1]
$[${COUNT} / 2]
```

Когда Asterisk встречает выражение в диалплане, он заменяет всё выражение полученным значением. Важно отметить, что это происходит *после* подстановки переменной. Чтобы продемонстрировать, давайте рассмотрим следующий код:[1]

```
exten => 321,1,NoOp()
    same => n,Answer()
    same => n,Set(COUNT=3)
    same => n,Set(NEWCOUNT=$[${COUNT} + 1])
    same => n,SayNumber(${NEWCOUNT})
```

Во втором приоритете мы присваиваем значение `3` переменной с именем `COUNT`.

В третьем приоритете задействовано только одно приложение `Set()` - но на самом деле происходят три вещи:

1. Asterisk заменяет `${COUNT}` числом `3` в выражении. Выражение эффективно становится следующим:

   ```
   same => n,Set(NEWCOUNT=$[3 + 1])
   ```

2. Asterisk оценивает выражение, добавляя от `1` до `3` и заменяя его вычисленным значением `4`:

   ```
   same => n,Set(NEWCOUNT=4)
   ```

3. Приложение `Set()` присваивает значение `4` переменной `NEWCOUNT`

Третий приоритет просто вызывает приложение `SayNumber()`, которое говорит текущее значение переменной `${NEWCOUNT}` (устанавливается в значение `4` в приоритете два).

Попробуйте в своем диалплане.

Операторы

Когда вы создаете диалплан Asterisk, вы действительно пишете код на специализированном языке сценариев. Это означает, что диалплан Asterisk, как любой язык программирования, распознает символы, называемые операторами, которые позволяют вам манипулировать переменными. Давайте посмотрим на типы операторов, которые доступны в Asterisk:

Булевы (логические) операторы

Эти операторы оценивают «истину» утверждения. В вычислительных терминах это в основном относится к тому, является ли оператор чем-то или нет (отличным от нуля или нулевым, истинным или ложным, включенным или выключенным и т.д.). Булевы операторы:

`expr1 | expr2`

Этот оператор (называемый оператором «или» или «пайп»(труба)) возвращает оценку *`expr1`*, если она истинна (не пустая строка, не нуль). В противном случае он возвращает оценку *`expr2`*.

`expr1 & expr2`

Этот оператор (называемый «и») возвращает оценку *`expr1`*, если оба выражения истинны (т. е. ни одно выражение не оценивает пустую строку или ноль). В противном случае он возвращает ноль.

`expr1 {=, >, >=, <, <=, !=} expr2`

1 Помните, что когда вы *ссылаетесь* на переменную, вы можете вызывать ее по ее имени, но когда вы ссылаетесь на *значение* переменной, вы должны использовать знак доллара и скобки вокруг имени переменной.

Эти операторы возвращают результаты целочисленного сравнения, если оба аргумента являются целыми числами; в противном случае они возвращают результаты сравнения строк. Результатом каждого сравнения является `1`, если указанное отношение истинно, или `0`, если отношение ложно. (Если вы выполняете сравнения строк, они будут выполняться таким образом, чтобы это соответствовало текущим локальным настройкам вашей операционной системы.)

Математические операторы

Хотите выполнить расчет? Вам понадобится один из них:

`expr1 {+, -} expr2`

Эти операторы возвращают результаты сложения или вычитания целочисленных аргументов.

`expr1 {*, /, %} expr2`

Они возвращают соответственно результаты умножения, целочисленного деления или остатка целочисленных аргументов.

Операторы регулярных выражений

Вы также можете использовать оператор регулярных выражений в Asterisk:



Некоторая дополнительная информация об особенностях операторов регулярных выражений в Asterisk содержится на веб-сайте Walter Doekes.

`expr1 : expr2`

Этот оператор сопоставляет *`expr1`* с *`expr2`*, где *`expr2`* должно быть регулярным выражением.[2] Регулярное выражение привязывается к началу строки с неявным ^ .[3]

Если шаблон не содержит подвыражения, возвращается количество совпадающих символов. Это будет `0`, если совпадение не выполнено. Если шаблон содержит подэлемент `-- \(...\) --` возвращается строка, соответствующая `\1`. Если совпадение не выполняется, возвращается пустая строка.

`expr1 =~ expr2`

Этот оператор работает так же, как оператор `:` кроме того, что он не привязан к началу.

В версии Asterisk 1.0 синтаксический анализатор был довольно прост, поэтому потребовалось, чтобы вы поместили хотя бы один пробел между оператором и любыми другими значениями. Следовательно, следующее, возможно, не сработало, как ожидалось:

```
exten => 123,1,Set(TEST=$[2+1])
```

Оно присвоило переменной `TEST` строку `2+1` вместо значения `3`. Чтобы исправить это, мы помещаем пробелы вокруг оператора, например:

```
exten => 234,1,Set(TEST=$[2 + 1])
```

Это больше не требуется в текущих версиях Asterisk, поскольку парсер выражений стал более упрощенным в этих сценариях. Однако, для удобства чтения, мы по-прежнему рекомендуем включать пробелы вокруг ваших операторов.

Чтобы присоединить текст в начало или конец переменной, просто поместите их вместе, например:

```
exten => 234,1,Set(NEWTEST=blah${TEST})
```

2 Для получения дополнительной информации о регулярных выражениях, возьмите копию окончательной ссылки, Jeffrey E. F. Friedl's Mastering Regular Expressions (O'Reilly, 2006) или посетите http://www.regular-expressions.info.

3 Если вы не знаете, что ^ имеет отношение к регулярным выражениям, вы просто должны прочитать «Освоение регулярных выражений». Это изменит вашу жизнь!

Функции диалплана

Функции диалплана позволяют добавлять больше мощи вашим выражениям; вы можете рассматривать их как интеллектуальные переменные. Функции диалплана позволяют вычислять длины строк, даты и время, контрольные суммы MD5 и т. д. все из выражения диалплана.



Вы увидите использование `Playback(silence/1)` во всех примерах этой главы. Мы делаем это, так как оно ответит на эту строку, если на нее еще не ответили, и воспроизведёт некоторую тишину на линии. Это позволяет другим приложениям, таким как `SayNumber()` воспроизводить аудио без пробелов.

Синтаксис

Функции диалплана имеют следующий базовый синтаксис:

```
FUNCTION_NAME(argument)
```

Вы ссылаетесь на имя функции так же, как на имя переменной, но вы ссылаетесь на значение функции с добавлением знака доллара, открывающейся фигурной скобкой и закрывающейся фигурной скобкой:

```
${FUNCTION_NAME(argument)}
```

Функции также могут инкапсулировать другие функции, например:

```
${FUNCTION_NAME(${FUNCTION_NAME(argument)})}
^              ^ ^             ^        ^ ^^^
1              2 3             4        4 321
```

Как вы, наверное, уже выяснили, вы должны быть очень осторожны, чтобы убедиться, что у вас есть все круглые скобки и фигурные. В предыдущем примере мы обозначили открывающие круглые скобки и фигурные скобки цифрами и их соответствующими закрывающими аналогами с одинаковыми номерами.

Примеры функций диалплана

Функции часто используются вместе с приложением `Set()` для получения или установки значения переменной. В качестве простого примера рассмотрим функцию `LEN()`. Эта функция вычисляет длину строки аргумента. Давайте вычислим длину строки переменной и вернем ее вызывающей стороне:

```
exten => 123,1,NoOp()
    same => n,Set(TEST=example)
    same => n,Playback(silence/1)
    same => n,SayNumber(${LEN(${TEST})})
```

В этом примере сначала будет оцениваться `$TEST` со значением "example". Строка "example" затем передается функции `LEN()`, которая будет вычислять длину строки, `7`. Наконец, `7` передается в качестве аргумента в приложение `SayNumber()`.

Давайте рассмотрим еще один простой пример. Если бы мы хотели установить один из различных тайм-аутов канала, мы могли бы использовать функцию `TIMEOUT()`. Функция `TIMEOUT()` принимает один из трех аргументов: `absolute`, `digit` и `response`. Чтобы установить тайм-аут с помощью функции `TIMEOUT()`, мы могли бы использовать приложение `Set()`, например:

```
exten => s,1,Set(TIMEOUT(digit)=30)
```

Обратите внимание на отсутствие `${ }`, окружающего функцию. Так же, как если бы мы присваивали значение переменной, мы присваиваем значение функции без использования инкапсуляции `${ }`.

Полный список доступных функций можно найти, набрав *core show functions* в интерфейсе командной строки Asterisk.

Условное ветвление

Теперь, когда вы немного узнали о выражениях и функциях, пришло время их использовать. Используя выражения и функции, вы можете добавить еще более сложную логику вашему диалплану. Чтобы позволить вашему диалплану принимать решения, вы будете использовать условное ветвление. Давайте посмотрим поближе.

Приложение GotoIf()

Ключом к условному ветвлению является приложение `GotoIf()`. `GotoIf()` оценивает выражение и отправляет вызывающего абонента в конкретный пункт назначения основываясь на оценке выражения, выраженном в `true` или `false`.

`GotoIf()` использует специальный синтаксис, который часто называют *условным синтаксисом*:

```
GotoIf(expression?destination1:destination2)
```

Если выражение принимает значение `true`, вызывающий объект отправляется в *`destination1`*. Если выражение принимает значение `false`, вызывающий объект отправляется во второе назначение. Итак, что истинно и что ложно? Пустая строка и число `0` оцениваются как false. Все остальное оценивается как истинное.

Каждый из пунктов назначения может быть одним из следующих:

- Метка приоритета в том же расширении, как `weasels`
- Расширение и метка приоритета в том же контексте, например, `123,weasels`
- Контекст, внутренний номер и метка приоритета, такая как `incoming,123,weasels`

Любой из пунктов назначения может быть опущен, но не оба. Если пропущенный пункт назначения должен соблюдаться, Asterisk просто переходит к следующему приоритету в текущем внутреннем номере.

Давайте используем `GotoIf()` в примере:

```
exten => 345,1,NoOp()
    same => n,Set(TEST=1)
    same => n,GotoIf($[${TEST} = 1]?weasels:iguanas)
    same => n(weasels),Playback(weasels-eaten-phonesys)
    same => n,Hangup()
    same => n(iguanas),Playback(office-iguanas)
    same => n,Hangup()
```



Вы заметите, что мы использовали приложение `Hangup()` после каждого использования `Playback()`. Это делается для того, что когда мы переходим на метку `weasels`, вызов останавливается до того, как выполнение переходит к звуковому файлу *office-iguanas*. Становится все более распространенным видеть расширения, разбитые на несколько компонентов (защищенных друг от друга командой `Hangup()`), каждая из которых совершает отдельную последовательность шагов, выполняемых после `GotoIf()`.

Предоставление только ложного условного пути

Если бы мы захотели, мы могли бы выполнить предыдущий пример следующим образом:

```
exten => 345,1,NoOp()
    same => n,Answer()
    same => n,Set(TEST=1)
    same => n,GotoIf($[${TEST} = 1]?:iguanas)   ; у нас больше нет ярлыка
                                                ; weasels, но это всё равно
                                                ; работает
    same => n,Playback(weasels-eaten-phonesys)
    same => n,Hangup()
    same => n(iguanas),Playback(office-iguanas)
    same => n,Hangup()
```

Ничего нет между `?` и `:` поэтому, если оператор оценивает значение как `true`, выполнение будет продолжено на следующем шаге. Поскольку это то, чего мы хотим, ярлык не нужен.

Мы действительно не рекомендуем это делать, потому что это трудно читать, но вы увидите диалпланы, подобные этому, поэтому хорошо знать, что этот синтаксис абсолютно корректный.

Как правило, когда у вас есть этот тип разбивки, в котором вы хотите избежать перехода Asterisk на следующий приоритет после того, как вы выполнили этот переход, скорее всего, лучше перейти на отдельные расширения вместо ярлыков приоритета. Во всяком случае, это делает его более понятным при чтении. Мы могли бы переписать предыдущий диалплан следующим образом:

```
exten => 345,1,NoOp()
    same => n,Answer()
    same => n,Set(TEST=1)
    same => n,GotoIf($[${TEST} = 1]?weasels,1:iguanas,1)    ; теперь мы идем к
                                                            ; extension,priority

exten => weasels,1,NoOp()
    same => n,Playback(weasels-eaten-phonesys)  ; это НЕ метка
                                                ; это другое расширение
    same => n,Hangup()

exten => iguanas,1,NoOp()
    same => n,Playback(office-iguanas)
    same => n,Hangup()
```

Изменив значение, присвоенное `TEST` во второй строке, вы сможете заставить сервер Asterisk воспроизводить другое приветствие.

Давайте посмотрим на другой пример условного разветвления. На этот раз мы будем использовать `Goto()` и `GotoIf()` для отсчета с 10, а затем повесим трубку:

```
exten => 123,1,NoOp()
    same => n,Answer()
    same => n,Set(COUNT=10)
    same => n(start),GotoIf($[${COUNT} > 0]?:goodbye)
    same => n,SayNumber(${COUNT})
    same => n,Set(COUNT=$[${COUNT} - 1])
    same => n,Goto(start)
    same => n(goodbye),Hangup()
```

Давайте проанализируем этот пример. Во втором приоритете мы устанавливаем переменную `COUNT` равным 10. Затем мы проверяем, будет ли `COUNT` больше `0`. Если это так, переходим к следующему приоритету. (Не забывайте, что если мы опускаем пункт назначения в приложении `GotoIf()`, управление переходит к следующему приоритету.) Оттуда мы проговариваем число, вычитаем `1` из

COUNT и возвращаемся к началу приоритета. Если значение COUNT меньше или равно 0, управление переходит на приоритет goodbye, и кладется трубка.

Кавычки и префиксы переменных в условных ветвлениях

Сейчас самое подходящее время уделить минутку, чтобы посмотреть на какие-то небрежные вещи с условными ветлениями. В Asterisk недопустимо иметь нулевое значение по обе стороны оператора сравнения. Давайте рассмотрим примеры, которые приведут к ошибке:

```
$[    = 0 ]
$[ foo = ]
$[ > 0 ]
$[ 1 + ]
```

Любой из наших примеров может вызвать предупреждение:

```
WARNING[28400][C-000000eb]: ast_expr2.fl:470 ast_yyerror: ast_yyerror():
syntax error: syntax error, unexpected '=', expecting $end; Input:
 = 0
 ^
```

Маловероятно (если у вас нет опечатки), что вы целенаправленно реализуете что-то вроде наших примеров. Тем не менее, когда вы выполняете математику или сравнение с неустановленной переменной канала, это фактически то, что вы делаете.

Примеры, которые мы использовали чтобы показать вам, как работает условное ветвление, не являются недопустимыми. Поскольку мы сначала инициализировали переменную и можем ясно видеть, что переменная канала, которую мы используем в нашем сравнении, была установлена, мы в безопасности. Но что, если ты не всегда так уверен?

В Asterisk строки не обязательно должны быть заключены в двойные или одинарные кавычки, как во многих языках программирования. На самом деле, если вы используете двойную или одинарную кавычку, это будет буквенной конструкцией в строке. Если мы посмотрим на следующие две строки Set() ...

```
exten => n,Set(TEST_1=foo)
exten => n,Set(TEST_2='foo')
exten => n,NoOp($[${TEST_1} = ${TEST_2}])
```

... тогда значение, возвращаемое нашим сравнением в NoOp() не будет равно 1 (значения совпадают или *true*), возвращаемое значение будет 0 (значения не совпадают или *false*).

Мы можем использовать это в наших интересах при выполнении сравнений путем заключения переменных канала в одинарные или двойные кавычки. Делая это, мы убеждаемся, что даже когда переменная канала не может быть установлена, наше сравнение действительно.

В следующем примере мы получим ошибку:

```
exten => 1000,1,NoOp()
    same => n,GotoIf($[${TEST} = invalid]?error_handling)
    same => n(error_handling),NoOp()
```

Однако мы можем обойти это, обернув то, что мы сравниваем в кавычки. Тот же пример, но сделан действительным:

```
exten => 1000,1,NoOp()
    same => n,GotoIf($["${TEST}" = "invalid"]?error_handling)
    same => n(error_handling),NoOp()
```

Даже если ${TEST} не был установлен, мы делаем сравнение:

```
$["" = "invalid"]
```

Мы можем сделать такой же пример сети безопасности при сравнении, которую выполняют численные сравнения, добавив префикс для наших чисел в виде нуля. Поскольку `01` совпадает с `1` при сравнении чисел, наш потенциально недействительный пример:

```
exten => 1001,1,NoOp()
    same => n,GotoIf($[${TEST} < 1]?error_handling)
    same => n(error_handling),NoOp()
```

Можно сделать более безопасным:

```
exten => 1001,1,NoOp()
    same => n,GotoIf($[0${TEST} < 1]?error_handling)
    same => n(error_handling),NoOp()
```

Если вы привыкли распознавать эти ситуации и использовать методы обертывания и префикса, которые мы изложили, вы напишете гораздо более безопасные диалпланы.

Классический пример условного ветвления ласково известен как логика анти-подруги. Если номер CallerID входящего вызова соответствует номеру телефона бывшей подруги получателя, Asterisk дает другое сообщение, чем обычно, для любого другого вызывающего абонента. В то время как этот пример несколько простой и примитивный, он хорош изучения условного ветвления в диалплане Asterisk.

В этом примере используется функция `CALLERID`, которая позволяет нам получать информацию CallerID во входящем вызове. Предположим ради этого примера, что номер телефона жертвы - 888-555-1212:

```
exten => 123,1,NoOp()
    same => n,GotoIf($[${CALLERID(num)} = 8885551212]?reject:allow)
    same => n(allow),Dial(DAHDI/4)
    same => n,Hangup()
    same => n(reject),Playback(abandon-all-hope)
    same => n,Hangup()
```

В приоритете `1` мы вызываем приложение `GotoIf()`. Оно сообщает Asterisk, о переходе к метке приоритета `reject`, если номер CallerID соответствует `8885551212`, а в противном случае - присвоить метку приоритета `allow` (мы могли бы просто опустить имя метки, в результате чего `GotoIf()` провалится). Если номер CallerID совпадает, управление вызовом переходит к метке приоритета `reject`, которое воспроизводит недоброжелательное сообщение нежелательному абоненту. В противном случае вызов пытается набрать получателя на канале `DAHDI/4`.

Временное условное ветвление с GotoIfTime()

Другой способ использования условного разветвления в вашем диалплане - это приложение `GotoIfTime()`. В то время как `GotoIf()` оценивает выражение, чтобы решить, что делать, `GotoIfTime()` смотрит на текущее системное время и использует его, чтобы решить, следовать или нет другой ветке в диалплане.

Наиболее очевидное использование этого приложения - дать вашим абонентам другое приветствие до и после обычных рабочих часов.

Синтаксис приложения `GotoIfTime()` выглядит следующим образом:

```
GotoIfTime(время,дни_недели,числа_месяца,месяцы?label)
```

Короче говоря, `GotoIfTime()` отправляет вызов указанной метке (label), если текущая дата и время соответствуют критериям, указанным во *времени*, *днях_недели*, *днях_месяца* и *месяцах*. Давайте рассмотрим каждый аргумент более подробно:

время

Это список одного или нескольких диапазонов времени в 24-часовом формате. В качестве примера, с 9:00 Д.П. до 5:00 П.П. будет указано как `09:00-17:00`. День начинается в 0:00 и заканчивается в 23:59.



Стоит отметить, что время будет правильно оборачиваться. Поэтому, если вы хотите указать время закрытия своего офиса, вы можете написать `18:00-9:00` в параметре *время* и он будет работать как ожидалось. Обратите внимание, что этот метод работает также и для других компонентов `GotoIfTime()`. Например, вы можете написать `sat-sun`, чтобы указать выходные дни.

дни_недели

Это список одного или нескольких дней недели. Дни должны быть указаны как `mon`, `tue`, `wed`, `thu`, `fri`, `sat` и/или `sun`. С понедельника по пятницу будет выражаться как `mon-fri`. Вторник и четверг будут выражены как `tue&thu`.



Обратите внимание, что вы можете указать комбинацию диапазонов и отдельных дней, например: `sun-mon&wed&fri-sat` или, проще говоря: `wed&fri-mon`.

числа_месяца

Это список чисел месяца. Дни обозначаются цифрами от `1` до `31`. Седьмое-двенадцатое будет выражаться как `7-12`, а 15 и 30 числа месяца будут записаны как `15&30`.

месяцы

Это список одного или нескольких месяцев в году. Месяцы должны быть записаны как `jan-apr` для диапазона и разделены амперсандами, когда вы хотите включить непоследовательные месяцы, например `jan&mar&jun`. Вы также можете комбинировать их так: `jan-apr&jun&oct-dec`.

Если вы хотите сопоставить все возможные значения для любого из этих аргументов, просто поместите аргумент * для него.

Аргумент метки может быть любым из следующих:

- Метка приоритета в пределах одного расширения, например, `time_has_passed`
- Расширение и приоритет в одном контексте, например `123,time_has_passed`
- Контекст, расширение и приоритет, такие как `incoming,123,time_has_passed`

Теперь, когда мы рассмотрели синтаксис, давайте рассмотрим несколько примеров. Следующий пример будет соответствовать *9:00 до 17:59 вечера, с понедельника по пятницу, в любой день месяца, в любой месяц в году:*

```
exten => s,1,NoOp()
    same => n,GotoIfTime(09:00-17:59,mon-fri,*,*?open,s,1)
```

Если вызывающий звонит в течение этих часов, вызов будет отправлен на первый приоритет расширения `s` в контексте с именем `open`. Если вызов выполняется за пределами указанного времени, он будет отправлен на следующий приоритет текущего расширения. Это позволяет вам легко разветвляться несколько раз, как показано в следующем примере (обратите внимание, что вы всегда должны ставить ваши наиболее конкретные совпадения времени перед наименее конкретными):

```
; Если это любой час дня, любой день недели,
; в течение четвертого дня месяца, в июле, мы закрыты
exten => s,1,NoOp()
    same => n,GotoIfTime(*,*,4,jul?closed,s,1)
```

```
; В рабочее время отправлять вызовы в контекст open
    same => n,GotoIfTime(09:00-17:59,mon-fri,*,*?open,s,1)
    same => n,GotoIfTime(09:00-11:59,sat,*,*?open,s,1)
; В противном случае мы закрыты
    same => n,Goto(closed,s,1)
```



Если вы столкнетесь с ситуацией, когда задаете вопрос: «Но я указал 17:58, и сейчас 17:59. Почему он все еще делает то же самое?», Следует отметить, что детализация приложения `GotoIfTime()` - это двухминутный период. Таким образом, если вы укажете 18:00 в качестве конечного времени периода, система будет продолжать выполнять тот же путь до 18:01:59.

Макрос

Макрос - очень полезная конструкция, разработанная, чтобы избежать повторения в диалплане. Они также помогают вносить изменения в диалплан.



Хотя `Macro()` кажется универсальной подпрограммой диалплана, у нее есть проблема переполнения стека, что означает, что вы не должны пытаться вложить вызовы `Macro()` более пяти уровней. Если вы планируете использовать макросы в макросах (и вызывать сложные функции внутри них), вы можете столкнуться с проблемами стабильности. Вы узнаете, что у вас есть проблема только с одним тестовым вызовом, поэтому, если ваш диалплан тестовый, всё хорошо - продолжаем. Мы также рекомендуем вам взглянуть на приложения `GoSub()` и `Return()` (см. «GoSub»), так как многие макрофункции могут быть реализованы без фактического использования `Macro()`.
Начиная с Asterisk 11 приложение `Macro()` устарело в пользу приложения `GoSub()`. Однако знание `Macro()` полезно, так как почти любая существующая система, которую вы поддерживаете или изменяете, скорее всего, содержит хотя бы одно использование `Macro()`.

Чтобы проиллюстрировать этот момент, давайте снова рассмотрим наш образец диалплана. Если вы помните изменения, внесенные нами для голосовой почты, мы закончили с внутр.номером Джона:

```
exten => 101,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(101@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(101@default,b)
    same => n,Hangup()
```



В нашем примере `GotoIf()` мы заключаем переменную канала `${DIALSTATUS}` в двойные кавычки. Для чего мы это сделали, см. «Кавычки и префиксы переменных в условных ветвлениях».

Теперь представьте, что у вас есть сто пользователей в вашей системе Asterisk, настройка внутренних номеров потребует много копирования и вставки. Затем представьте, что вам нужно внести изменения в способ работы ваших внутренних номеров. Это потребует много редактирования, и вы почти наверняка будете иметь ошибки.

Вместо этого вы можете определить макрос, содержащий список шагов, которые необходимо предпринять, а затем все внутренние номера телефонов относящиеся к этому макросу. Все, что вам нужно изменить - это макрос, и все в диалплане, что ссылается на этот макрос, также изменится.



Если вы знакомы с компьютерным программированием, вы поймете, что макросы похожи на подпрограммы во многих современных языках программирования. Если вы не знакомы с компьютерным программированием, не волнуйтесь - мы проведем вас через создание макроса.

Лучший способ оценить макросы - увидеть их в действии, поэтому давайте двигаться дальше.

Определение макроса

Давайте возьмем логику диалплана, которую мы использовали для настройки голосовой почты для Джона и превратим ее в макрос. Затем мы будем использовать макрос, чтобы дать Джону и Джейн (и остальным их сотрудникам) ту же функциональность.

Определения макросов очень похожи на контексты. (Фактически, вы можете утверждать, что это действительно небольшие, ограниченные контексты.) Вы определяете макрос, помещая команду `macro-` и имя своего макроса в квадратные скобки, например:

```
[macro-voicemail]
```

Имена макросов должны начинаться с `macro-`. Это отличает их от обычных контекстов. Команды внутри макроса построены почти идентично всем остальным в диалплане; единственным ограничивающим фактором является то, что макросы используют только расширение `s`. Давайте добавим нашу логику голосовой почты в макрос, изменив расширение на `s` по мере продвижения:

```
[macro-voicemail]
exten => s,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(101@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(101@default,b)
    same => n,Hangup()
```

Это начало, но оно не идеально, поскольку это все еще характерно для Джона и его номера почтового ящика. Чтобы создать макрос, который будет работать не только для Джона, но и для всех его сотрудников, мы воспользуемся еще одним свойством макросов: аргументы. Но сначала посмотрим, как мы вызываем макросы в нашем диалплане.

Вызов макроса из диалплана

Чтобы использовать макрос в нашем диалплане, мы используем приложение `Macro()`. Это приложение вызывает указанный макрос и передает ему любые аргументы. Например, чтобы вызвать наш макрос голосовой почты из диалплана, мы можем сделать следующее:

```
exten => 101,1,Macro(voicemail)
```

Приложение `Macro()` также определяет несколько специальных переменных для нашего использования. Они включают:

`${MACRO_CONTEXT}`

Исходный контекст, в котором был вызван макрос.

`${MACRO_EXTEN}`

Исходное расширение, в котором был вызван макрос.

`${MACRO_PRIORITY}`

Исходный приоритет, в котором был вызван макрос.

`${ARG` *n*`}`

n-й аргумент передаваемый макросу. Например, первым аргументом будет `${ARG1}`, вторым `${ARG2}` и т. д.

Как мы объясняли ранее, способ, которым мы изначально определяли наш макрос, был жестко запрограммирован для Джона, а не был общим. Давайте изменим наш макрос, чтобы использовать `${MACRO_EXTEN}` вместо `101` для номера почтового ящика. Таким образом, если мы вызываем макрос из внутреннего номера 101, сообщения голосовой почты будут отправляться в почтовый ящик 101; если мы вызываем макрос из номера 102, сообщения отправляются в почтовый ящик 102; и так далее:

```
[macro-voicemail]
exten => s,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(${MACRO_EXTEN}@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(${MACRO_EXTEN}@default,b)
    same => n,Hangup()
```

Использование аргументов в макросе

Теперь мы приближаемся к тому, чтобы макрос был таким, какой мы хотим, но нам остается изменить все: нам нужно пройти в канал для набора, так как он в настоящее время все еще жестко запрограммирован для `${JOHN}` (помните, что мы определил переменную `JOHN` как канал для вызова, когда мы хотим связаться с Джоном). Передим его в канал как аргумент, и тогда наш первый макрос будет завершен:

```
[macro-voicemail]
exten => s,1,NoOp()
    same => n,Dial(${ARG1},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(${MACRO_EXTEN}@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(${MACRO_EXTEN}@default,b)
    same => n,Hangup()
```

Теперь, когда наш макрос сделан, мы можем использовать его в нашем диалплане. Вот как мы можем позвонить в наш макрос, чтобы предоставить голосовую почту Джону, Джейн и Джеку:[4]

```
exten => 101,1,Macro(voicemail,${JOHN})
exten => 102,1,Macro(voicemail,${JANE})
exten => 103,1,Macro(voicemail,${JACK})
```

С 50 или более пользователями этот диалплан по-прежнему будет выглядеть аккуратно и организованно; мы просто будем иметь одну строку для каждого пользователя, ссылаясь на макрос, который может быть более сложным. Мы могли бы даже иметь несколько разных макросов для различных типов пользователей, таких как `executives` (руководители), `courtesy_phones` (справочная служба), `call_center_agents` (агенты колл-центра), `analog_sets` (аналоговые комплекты), `sales_department` (отдел продаж) и т.д.

Более продвинутая версия макроса может выглядеть примерно так:

```
[macro-voicemail]
exten => s,1,NoOp()
    same => n,Dial(${ARG1},20)
    same => n,Goto(s-${DIALSTATUS},1)
```

4 В начале главы мы сказали, что используем приложение `NoOp()` в качестве первого приоритета, но в некоторых случаях имеет смысл нарушить эту традицию. В основном случае, когда расширение будет содержать только одну строку. Если после `Macro()` был бы добавлен дополнительный функционал, мы бы изменили первый приоритет, чтобы начать с `NoOp()`, как обычно.

```
exten => s-NOANSWER,1,VoiceMail(${MACRO_EXTEN}@default,u)
    same => n,Goto(incoming,s,1)

exten => s-BUSY,1,VoiceMail(${MACRO_EXTEN}@default,b)
    same => n,Goto(incoming,s,1)

exten => _s-.,1,NoOp()
    same => n,Goto(s-NOANSWER,1)
```



Так как теперь мы знаем, как использовать функции диалплана, вот еще один способ контроля того, какая голосовая подсказка (недоступности или занятости) воспроизводится вызывающему. В следующем примере мы будем использовать функцию диалплана `IF()`:

```
[macro-voicemail]
exten => s,1,NoOp()
    same => n,Dial(${ARG1},20)
    same => n,VoiceMail(${MACRO_EXTEN}@default,${IF($[${DIALSTATUS} =
BUSY]?b:u)})
```

Этот макрос зависит от хорошего побочного эффекта приложения `Dial()`: когда вы используете приложение `Dial()`, оно устанавливает переменную `DIALSTATUS` чтобы указать, был ли вызов успешным или нет. В этом случае мы обрабатываем `NOANSWER` и `BUSY` и обрабатываем все остальные результаты как `NOANSWER`.

GoSub

Приложение диалплана `GoSub()` аналогично приложению `Macro()`, поскольку цель состоит в том, чтобы разрешить вам вызывать блок функциональности диалплана, передавать информацию этому блоку и возвращать его (необязательно с возвращаемым значением), `GoSub()` работает иначе, чем `Macro()`, хотя в нем нет требований к пространству стека, поэтому он эффективно монтируется. По сути, `GoSub()` действует как `Goto()` с памятью о том, откуда он появился.

В этом разделе мы перейдем к тому, что мы узнали в «Макросах». При необходимости вы можете просмотреть этот раздел: он объясняет, почему мы можем использовать подпрограмму и цель, которую мы пытаемся выполнить.

Определение подпрограмм

В отличие от `Macro()`, при использовании `GoSub()` в диалплане особых требований к именам не существует. Фактически, вы можете использовать `GoSub()` в том же контексте и расширении, если хотите. В большинстве случаев, однако, `GoSub()` используется аналогично `Macro()`, поэтому определение нового контекста является общим. При создании контекста нам нравится добавлять имя с `sub`, поэтому мы знаем, что контекст обычно вызывается из приложения `GoSub()` (конечно, нет необходимости, чтобы вы это делали, но это похоже на разумное соглашение).

Вот простой пример того, как мы можем определить подпрограмму в Asterisk:

```
[SubVoicemail]
```

Давайте возьмем наш пример из «Макросов» и преобразуем его в подпрограмму. Вот как он определяется для использования с `Macro()`:

```
[macro-voicemail]
exten => s,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(101@default,u)
    same => n,Hangup()
```

```
    same => n(busy),VoiceMail(101@default,b)
    same => n,Hangup()
```

Если мы собираемся преобразовать это для использования в подпрограмму, то может выглядеть так:

```
[subVoicemail]
exten => start,1,NoOp()
    same => n,Dial(${JOHN},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(101@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(101@default,b)
    same => n,Hangup()
```

Не так много изменений, не так ли? Все, что мы изменили в этом примере - это имя контекста, от `[macro-voicemail]` до `[subVoicemail]` и расширения от `s` до `start` (поскольку нет требования, чтобы расширение было вызвано чем-то конкретным, в отличие от `Macro()`, который ожидает что расширение будет `s`).

Конечно, как и в примере из раздела «Макросы», мы не передали никаких аргументов подпрограмме, поэтому всякий раз, когда мы вызываем `[subVoicemail]`, всегда будет вызываться `${JOHN}`, и будет использоваться поле голосовой почты 101. В следующих разделах мы будем копать немного глубже. Сначала рассмотрим, как мы будем называть подпрограмму, а затем узнаем, как передавать аргументы.

Вызов подпрограмм из диалплана

Подпрограммы вызываются из диалплана с помощью приложения `GoSub()`. Аргументы для `GoSub()` несколько отличаются от аргументов для `Macro()`, поскольку `GoSub()` не имеет требований к именам для контекста или расширения (или приоритета), которые используются. Кроме того, никакие специальные переменные канала не задаются при вызове подпрограммы, отличной от переданных аргументов, которые сохраняются в `${ARG`*n*`}` (где первый аргумент равен `${ARG1}`, второй аргумент - `${ARG2}` и так далее).

Теперь, когда мы обновили наш макрос голосовой почты, который вызывается как подпрограмма, давайте посмотрим, как мы его вызываем, используя `GoSub()`:

```
exten => 101,1,GoSub(subVoicemail,start,1())
```



Вы заметите, что мы разместили набор открывающих и закрывающих круглых скобок в нашем приложении `GoSub()`. Это заполнители для любых аргументов, которые мы можем передать подпрограмме и, хотя они не обязательно будут существовать, это стиль программирования, который мы предпочитаем использовать.

Затем давайте посмотрим, как мы можем передавать аргументы нашей подпрограмме, чтобы сделать ее более общей.

Использование аргументов в подпрограмме

Возможность использования аргументов является одной из основных особенностей использования `Macro()` или `GoSub()`, поскольку позволяет абстрагироваться от кода, который в противном случае был бы дублирован в вашем диалплане. Без необходимости дублировать код, мы можем лучше управлять им, и можем легко добавить функциональность большому числу пользователей, изменив одно местоположение. Вам рекомендуется помещать код в эту форму всякий раз, когда он дублируется.

Прежде чем мы начнем использовать нашу подпрограмму, нам нужно обновить ее для принятия аргументов, чтобы она была общей для использования несколькими пользователями:

```
[subVoicemail]
exten => start,1,NoOp()
    same => n,Dial(${ARG1},10)
    same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
    same => n(unavail),VoiceMail(${ARG2}@default,u)
    same => n,Hangup()
    same => n(busy),VoiceMail(${ARG2}@default,b)
    same => n,Hangup()
```

Напомним, что ранее мы жестко закодировали переменную канала `${JOHN}` в качестве местоположения для набора номера и почтовый ящик `101` в качестве ящика голосовой почты для использования, если `${JOHN}` недоступен. В этом коде мы заменили `${JOHN}` и `101` на `${ARG1}` и `${ARG2}` соответственно. В более сложных подпрограммах мы можем даже присваивать переменным `${ARG1}` и `${ARG2}` нечто вроде `${DESTINATION}` и `${VMBOX}`, чтобы было ясно, что представляют собой `${ARG1}` и `${ARG2}`.

Теперь, когда мы обновили нашу подпрограмму, мы можем использовать ее для нескольких внутренних номеров:

```
[LocalSets]
exten => 101,1,GoSub(subVoicemail,start,1(${JOHN},${EXTEN}))
exten => 102,1,GoSub(subVoicemail,start,1(${JANE},${EXTEN}))
exten => 103,1,GoSub(subVoicemail,start,1(${JACK},${EXTEN}))
```

Опять же, наш диалплан хорош и опрятен. Мы могли бы даже изменить нашу подпрограмму всего на три строки:

```
[subVoicemail]
exten => start,1,NoOp()
    same => n,Dial(${ARG1},10)
    same => n,VoiceMail(${ARG2}@default,${IF($[${DIALSTATUS} = BUSY]?b:u)})
    same => n,Hangup()
```

Одно из отличий между `GoSub()` и `Macro()` заключается в том, что если мы оставим нашу подпрограмму подобным образом, мы никогда не вернемся в исходный диалплан. В этом конкретном примере это не проблема, так как после того, как голосовая почта оставлена, мы ожидаем, что вызывающий абонент повесит трубку в любом случае. В ситуациях, когда мы хотим сделать что-то ещё после выполнения подпрограммы, нам нужно реализовать приложение `Return()`.

Возврат из подпрограммы

В отличие от `Macro()` приложение `GoSub()` не возвращается автоматически после его выполнения. Чтобы вернуться туда, откуда мы пришли, нам нужно использовать приложение `Return()`. Теперь, когда мы знаем, как вызвать подпрограмму и передать аргументы, можем посмотреть пример, где нам может понадобиться вернуться из подпрограммы.

Используя наш предыдущий пример, мы можем разбить часть набора и часть голосовой почты на отдельные подпрограммы:

```
[subDialer]
exten => start,1,NoOp()
    same => n,Dial(${ARG1},${ARG2})
    same => n,Return()

[subVoicemail]
exten => start,1,NoOp()
```

```
    same => n,VoiceMail(${ARG1}@${ARG2},${ARG3})
    same => n,Hangup()
```

Созданный здесь контекст `[subDialer]` принимает два аргумента: `${ARG1}`, который содержит назначение для набора; и `${ARG2}` содержащий время звонка, определенное в секундах. Мы завершаем контекст `[subDialer]` с помощью приложения `Return()`, которое вернет к приоритету, следующему за вызовом `GoSub()` (следующая строка диалплана).

Контекст `[subVoicemail]` содержит приложение `VoiceMail()`, которое использует три аргумента, переданных ему: `${ARG1}` содержит номер почтового ящика, `${ARG2}` содержит контекст голосовой почты, а `${ARG3}` содержит значение, указывающее, тип сообщения голосовой почты (недоступно или занято) для воспроизведения вызывающему абоненту.

Вызов этих подпрограмм может выглядеть так:

```
exten => 101,1,NoOp()
    same => n,GoSub(subDialer,start,1(${JOHN},30))
    same => n,GoSub(subVoicemail,start,1(${EXTEN},default,u))
```

Здесь мы использовали подпрограмму `subDialer`, которая пытается вызвать `${JOHN}`, набрав его в течение 30 секунд. Если приложение `Dial()` вернется (например, если линия занята или не было ответа в течение 30 секунд), мы делаем `Return()` из подпрограммы и выполняем следующую строку нашего диалплана, которая вызывает подпрограмму `subVoicemail`. Оттуда мы передаем внутренний номер, который был набран (например, `101`) в качестве номера почтового ящика, и передаем значения `default` для контекста голосовой почты и букву `u` для воспроизведения сообщения о недоступности.

Наш пример жестко запрограммирован для воспроизведения сообщения голосовой почты о недоступности, но мы можем изменить приложение `Return()`, чтобы вернуть `${DIALSTATUS}`, чтобы мы могли играть сообщение «занято», если его значение `BUSY`. Для этого мы будем использовать переменную канала `${GOSUB_RETVAL}`, которая устанавливается каждый раз, когда мы передаем значение в приложение `Return()`:

```
[subDialer]
exten => start,1,NoOp()
    same => n,Dial(${ARG1},${ARG2})
    same => n,Return(${DIALSTATUS})
[subVoicemail]
exten => start,1,NoOp()
    same => n,VoiceMail(${ARG1}@${ARG2},${ARG3})
    same => n,Hangup()
```

В этой версии мы сделали только одно изменение: `Return()` на `Return(${DIALSTATUS})`.

Теперь мы можем изменить расширение `101`, чтобы использовать переменную канала `${GOSUB_RETVAL}`, которая будет установлена через `Return()`:

```
exten => 101,1,NoOp()
    same => n,GoSub(subDialer,start,1(${JOHN},30))
    same => n,Set(VoicemailMessage=${IF($[${GOSUB_RETVAL} = BUSY]?b:u)})
    same => n,GoSub(subVoicemail,start,1(${EXTEN},default,${VoicemailMessage}))
```

В нашем диалплане теперь есть новая строка, которая устанавливает переменную канала `${VoicemailMessage}` в значение `u` или `b`, используя функцию диалплана `IF()` и значение `${GOSUB_RETVAL}`. Затем мы передаем значение `${VoicemailMessage}` в качестве третьего аргумента в нашу подпрограмму `subVoicemail`.

Прежде чем двигаться дальше, вы можете вернуться назад и просмотреть «Макросы» и «GoSub». Мы дали вам массу возможностей чтобы переварить здесь, но эти концепции сэкономят вам много работы и времени как только вы начинаете строить свой диалплан.

Локальные (Local) каналы

Локальные каналы (Local) - это метод выполнения других областей диалплана из приложения `Dial()` (в отличие от отправки канала вызова). Они могут показаться немного странной концепцией когда вы впервые начнете их использовать, но поверьте нам, когда вы узнаете что они являются славной и чрезвычайно полезной функцией, вы почти наверняка захотите использовать, когда начнете писать расширенные диалпланы. В качестве примера можно привести лучший пример использования локальных каналов. Предположим, что у нас есть ситуация, когда нам нужно позвонить нескольким людям, но нужно предоставить задержки разной длины, прежде чем набирать каждого из них. Использование локальных каналов - единственное решение проблемы.

С помощью приложения `Dial()` вы можете звонить по нескольким конечным точкам, но все три канала будут звонить одновременно и в один и тот же промежуток времени. Набор нескольких каналов одновременно выполняется следующим образом:

```
[LocalSets]
exten => 107,1,NoOp()
    same => n,Verbose(2,Dialing multiple locations simultaneously)
    same => n,Dial(SIP/0000FFFF0001&DAHDI/g0/14165551212&SIP/MyITSP/
12565551212,30)
    same => n,Hangup()
```

В этом примере набирают трёх адресатов в течение 30 секунд. Если ни один из них не отвечает на вызов в течение 30 секунд, диалплан продолжит на следующей строке и звонок завершится.

Однако предположим, что мы хотим ввести некоторые задержки и прекратить звонки в разное время. Использование локальных каналов дает нам независимый контроль над каждым каналом, который мы хотим набрать, поэтому мы можем вводить задержки и управлять периодом времени, для которого каждый канал звонит независимо. Мы покажем вам, как это делается в диалплане, как внутри таблицы, которая визуально отображает задержки, так и все вместе в боксе как это было сделано для других частей диалплана. Мы будем строить диалплан в соответствии с началом и остановками времени, описанными на Рисунке 10-1.

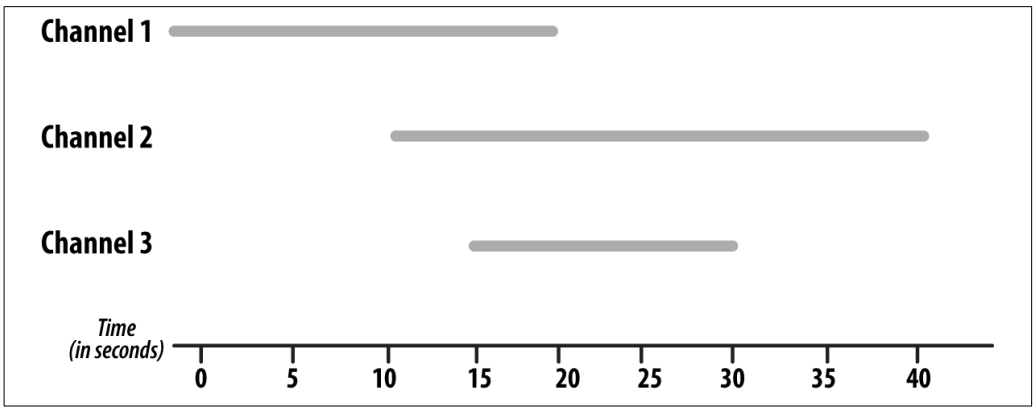


Рисунок 10-1. Набор с задержкой по времени в локальных каналах

Сначала нам нужно вызвать три локальных канала, которые будут выполнять разные части диалплана. Мы делаем это с помощью приложения `Dial()`, например:

```
[LocalSets]
exten => 107,1,Verbose(2,Dialing multiple locations with time delay)

; *** Это всё должно быть на одной линии
    same => n,Dial(Local/channel_1@TimeDelay&Local/channel_2@TimeDelay
&Local/channel_3@TimeDelay,40)
    same => n,Hangup()
```

Теперь наше приложение `Dial()` набирает три локальных канала. Целями назначения будут внутренние номера `channel_1`, `channel_2` и `channel_3`, расположенные в контексте диалплана

TimeDelay. Помните, что локальные каналы - это способ выполнения диалплана из приложения Dial(). Наш главный тайм-аут для всех каналов составляет 40 секунд, что означает, что любой локальный канал, который не имеет более короткого тайм-аута, будет завершаться, если он не отвечает на вызов в течение этого периода времени.

Как и было обещано, Таблица 10-1 иллюстрирует конфигурации задержки.

Table 10-1. Задержка набора с использованием локальных каналов

Период времени (в секундах)	channel_1	channel_2	channel_3
0	Dial(SIP/0000FFFF0001,20)	Wait(10)	Wait(15)
5			
10		Dial(DAHDI/g0/14165551212)	
15			Dial(SIP/MyITSP/12565551212,15)
20	Hangup()		
25			
30			Hangup()
35			
40			

В этой таблице мы видим, что channel_1 сразу начал набирать местоположение SIP/0000FFFF0001 и ждал в течении в 20 секунд. Через 20 секунд этот локальный канал повесил трубку. Наш channel_2 ждал 10 секунд до набора конечной точки DAHDI/g0/14165551212. Максимальное время, связанное с этим Dial() не было достигнуто, поэтому его временной период закончится, когда истечет время ожидания 40 секунд (которое мы установили при изначальном именовании локальных каналов) истечёт. Наконец, channel_3 ждал за 15 секунд до набора номера, затем набрал SIP/MyITSP/12565551212 и ждал 15 секунд перед тем, как повесить трубку.

Если мы соединим все это, мы получим следующий диалплан:

```
[LocalSets]
exten => 107,1,Verbose(2,Dialing multiple locations with time delay)

; *** Это все должно быть на одной линии
    same => n,Dial(Local/channel_1@TimeDelay&Local/channel_2@TimeDelay
&Local/channel_3@TimeDelay,40)
    same => n,Hangup()

[TimeDelay]
exten => channel_1,1,Verbose(2,Dialing the first channel)
    same => n,Dial(SIP/0000FFFF0001,20)
    same => n,Hangup()

exten => channel_2,1,Verbose(2,Dialing the second channel with a delay)
    same => n,Wait(10)
    same => n,Dial(DAHDI/g0/14165551212)

exten => channel_3,1,Verbose(2,Dialing the third channel with a delay)
    same => n,Wait(15)
    same => n,Dial(SIP/MyITSP/12565551212,15)
    same => n,Hangup()
```

Вы увидите локальные каналы используемые в этой книге для различных целей. Локальные каналы позволяют выполнять логику диалплана из приложений, которые обычно ожидают прямого подключения к каналу. Например, вы можете назначить локальный канал в качестве участника очереди и запускать всякую причудливую логику диалплана всякий раз, когда очередь пытается доставить вызов агенту. Об этом мы поговорим в разделе «Использование локальных каналов» в Главе 13.

Дополнительные сценарии и информация о локальных каналах и флаги модификатора (`/n`, `/j`, `/m`, `/b`) доступны в вики Asterisk. Если вы будете регулярно использовать локальные каналы, это очень важный документ для прочтения.

Использование базы данных Asterisk (AstDB)

Уже весело? А будет еще лучше!

Asterisk предоставляет мощный механизм для хранения значений, называемый *базой данных Asterisk* (AstDB). AstDB предоставляет простой способ хранения данных для использования в диалплане.



Для тех из вас, кто пользуется реляционными базами данных, такими как PostgreSQL или MySQL, база данных Asterisk не является традиционной базой данных; это база данных с поддержкой SQLite с использованием пар ключ/значение. Существует несколько способов хранения данных из Asterisk в реляционной базе данных. Более подробно о реляционных базах данных вы найдете в Главе 16.
Первоначально (и на протяжении многих лет) AstDB использовала базу данных Berkeley (которая применительно к диалплану, функциям и командам CLI работает так же, как и новая база данных SQLite). Другими словами, это изменение должно быть прозрачным для вас.

База данных Asterisk хранит свои данные в группах, называемых *семействами* (*families*), со значениями, идентифицируемыми *ключами* (*keys*). Внутри семьи ключ может использоваться только один раз. Например, если у нас было семейство, называемое `test`, мы могли бы сохранить только одно значение с помощью ключа, называемого `count`. Каждое сохраненное значение должно быть связано с семейством.

Хранение данных в AstDB

Чтобы сохранить новое значение в базе данных Asterisk, мы используем приложение `Set()`[5], но вместо того, чтобы использовать его для установки переменной канала, мы используем его для установки переменной AstDB. Например, чтобы присвоить ключу `count` в семействе `test` значение `1`, мы должны написать следующее:

```
exten => 456,1,NoOp()
    same => n,Set(DB(test/count)=1)
```

Если в семействе `test` уже существует ключ с именем `count`, его значение будет перезаписано новым. Вы также можете сохранить значения из командной строки Asterisk, выполнив команду *database put <family> <key> <value>*. В нашем примере вы бы набрали *database put test count 1*.

Извлечение данных из AstDB

Чтобы получить значение из базы данных Asterisk и назначить его переменной, мы снова используем приложение `Set()`. Получим значение `count` (опять же, из семейства тестов), назначте его переменной `COUNT`, а затем произнесите значение вызывающему:

```
exten => 456,1,NoOp()
```

5 Предыдущие версии Asterisk имели приложения, называемые `DBput()` и `DBget()`, которые использовались для установки и получения значений из AstDB. Если вы используете старую версию Asterisk, вы должны использовать эти приложения.

```
    same => n,Set(DB(test/count)=1)
    same => n,Set(COUNT=${DB(test/count)})
    same => n,Answer()
    same => n,SayNumber(${COUNT})
```

Вы также можете проверить значение заданного ключа из командной строки Asterisk, запустив команду *database get <family> <key>*. Чтобы просмотреть все содержимое AstDB, используйте команду *database show*.

Удаление данных из AstDB

Существует два способа удаления данных из базы данных Asterisk. Чтобы удалить ключ, вы можете использовать приложение `DB_DELETE()`. Он принимает путь к ключу в качестве своих аргументов, например:

```
; удаляет ключ и возвращает его значение за один шаг
exten => 457,1,Verbose(0, The value was ${DB_DELETE(test/count)})
```

Вы также можете удалить целое семейство ключей с помощью приложения `DBdeltree()`. Приложение `DBdeltree()` принимает один аргумент: имя семейства ключей для удаления. Чтобы удалить все семейство `test`, выполните следующие действия:

```
exten => 457,1,DBdeltree(test)
```

Чтобы удалить ключи и семейства ключей из AstDB через интерфейс командной строки, используйте команды *database del <key>* и *database deltree <family>*, соответственно.

Использование AstDB в диалплане

Существует множество способов использования базы данных Asterisk в диалплане. Чтобы представить AstDB, мы рассмотрим два простых примера. Первый пример - простой подсчет, показывающий, что база данных Asterisk является постоянной (что означает, что она выживает при перезагрузке системы). Во втором примере мы будем использовать функцию `BLACKLIST()` для оценки того, находится ли номер в черном списке и должен быть заблокирован.

Чтобы начать пример подсчета, давайте сначала выберем число (значение ключа `count`) из базы данных и назначим его переменной с именем `COUNT`. Если ключ не существует, `DB()` вернет `NULL` (нет значения). Поэтому мы можем использовать функцию `ISNULL()`, чтобы проверить, было ли возвращено значение. Если нет, мы инициализируем AstDB с помощью приложения `Set()`, где установим значение в базе данных равным `1`. Следующий приоритет вернет нам `1`. Это произойдет в первый раз, когда мы набираем этот внутренний номер:

```
exten => 678,1,NoOp()
    same => n,Set(COUNT=${DB(test/count)})
    same => n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
    same => n,Set(DB(test/count)=1)
    same => n,Goto(1)
    same => n(continue),NoOp()
```

Затем мы скажем текущее значение `COUNT`, а затем увеличим его:

```
exten => 678,1,NoOp()
    same => n,Set(COUNT=${DB(test/count)})
    same => n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
    same => n,Set(DB(test/count)=1)
    same => n,Goto(1)
    same => n(continue),NoOp()
    same => n,Playback(silence/1)
```

```
    same => n,SayNumber(${COUNT})
    same => n,Set(COUNT=$[${COUNT} + 1])
```

Теперь, когда мы увеличили значение `COUNT`, вернем новое значение в базу данных. Помните, что сохранение значения для существующего ключа перезаписывает предыдущее значение:

```
exten => 678,1,NoOp()
    same => n,Set(COUNT=${DB(test/count)})
    same => n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
    same => n,Set(DB(test/count)=1)
    same => n,Goto(1)
    same => n(continue),NoOp()
    same => n,Playback(silence/1)
    same => n,SayNumber(${COUNT})
    same => n,Set(COUNT=$[${COUNT} + 1])
    same => n,Set(DB(test/count)=${COUNT})
```

Наконец, мы вернемся к первому приоритету. Таким образом, приложение будет продолжать считать:

```
exten => 678,1,NoOp()
    same => n,Set(COUNT=${DB(test/count)})
    same => n,GotoIf($[${ISNULL(${COUNT})}]?:continue)
    same => n,Set(DB(test/count)=1)
    same => n,Goto(1)
    same => n(continue),NoOp()
    same => n,Playback(silence/1)
    same => n,SayNumber(${COUNT})
    same => n,Set(COUNT=$[${COUNT} + 1]
    same => n,Set(DB(test/count)=${COUNT})
    same => n,Goto(1)
```

Попробуйте этот пример. Послушайте, чтобы он подсчитал какое-то время, а затем повесьте трубку. Когда вы снова набеёте этот добавочный номер, он должен продолжать отсчет с того места, где был остановлен. Значение, хранящееся в базе данных, будет постоянным даже при перезапуске Asterisk.

В следующем примере мы создадим логику диалплана вокруг функции `BLACKLIST()`, которая проверяет, существует ли текущий callerID номера в черном списке. (Черный список - это просто семейство, называемое `blacklist` в AstDB.) Если `BLACKLIST()` находит номер в черном списке, она возвращает значение `1`; в противном случае вернет `0`. Мы можем использовать эти значения в сочетании с `GotoIf()`, чтобы контролировать, будет ли вызов выполнять приложение `Dial()`:

```
exten => 124,1,NoOp()
    same => n,GotoIf($[${BLACKLIST()}]?blocked,1)
    same => n,Dial(${JOHN})

exten => blocked,1,NoOp()
    same => n,Playback(silence/1)
    same => n,Playback(privacy-you-are-blacklisted)
    same => n,Playback(vm-goodbye)
    same => n,Hangup()
```

Чтобы добавить номер в черный список, запустите *database put blacklist <номер> 1* из интерфейса командной строки Asterisk.

Создание приложения горячего стола с AstDB

Благодаря встроенной базе данных Asterisk вы можете создавать всевозможные приложения без необходимости взаимодействия с чем-либо внешним. В приведенном ниже примере мы объединили

все знания, которые обсудили в этой главе, в одном наборе внутренних номеров, которые вы можете включить в свой контекст `LocalSets`.



Вы также можете найти версию с реляционной базой данных с использованием SQL и `func_odbc` в разделе «Веселимся с func_odbc: горячий стол» в Главе 16.

Hot-desking (горячий рабочий стол) - довольно распространенная функция, которая вызывает повышенный интерес по мере развертывания систем Asterisk из-за присущей гибкости диалплана. Старые традиционные системы УАТС применяют добавочный номер к линии в системе или к самому устройству. С Asterisk у нас есть возможность применять логику диалплана и информацию, хранящуюся в локальной базе данных (или внешней), чтобы определить, где вызывать внутренний номер. Мы могли бы легко разработать систему, в которой добавочный номер ничего не делает, кроме звонка на сотовый телефон или комбинацию устройств (например, в пейджинговой системе или группе торговых агентов).

В диалплане, предоставленном для этого примера горячего стола, мы разрешили людям регистрироваться на любом устройстве, набрав `71`*`XX`*, где `1`*`XX`* является внутренним номером в диапазоне от 100 до 199. Чтобы зарегистрировать внутренний номер вне устройства, пользователь просто набирает `7000` с устройства. Несмотря на то, что диалплан и логика стали более сложными, диалплан также учитывает другие внутренние номера, которые уже подключились к устройству, с которого кто-то хочет войти в систему, и автоматически регистрирует их в первую очередь. Кроме того, если ранее мы вошли с другого устройства и не выходили из системы до изменения местоположения, диалплан будет регистрировать внутренний номер с того устройства, прежде чем записывать его в новом месте.



Имейте в виду, что мы не добавили никакой логики для аутентификации абонентов. Кроме того, мы не добавили никаких подсказок, уведомляющих абонентов о том, что существующий внутренний номер был зарегистрирован до их выхода из системы, поскольку мы хотели сохранить основную логику приложения горячий стол, чтобы у вас была база для работы.

Чтобы понять логику диалплана, которую мы предоставили, полезно увидеть маршрут потока вызовов. Мы показали его на Рисунке 10-2.

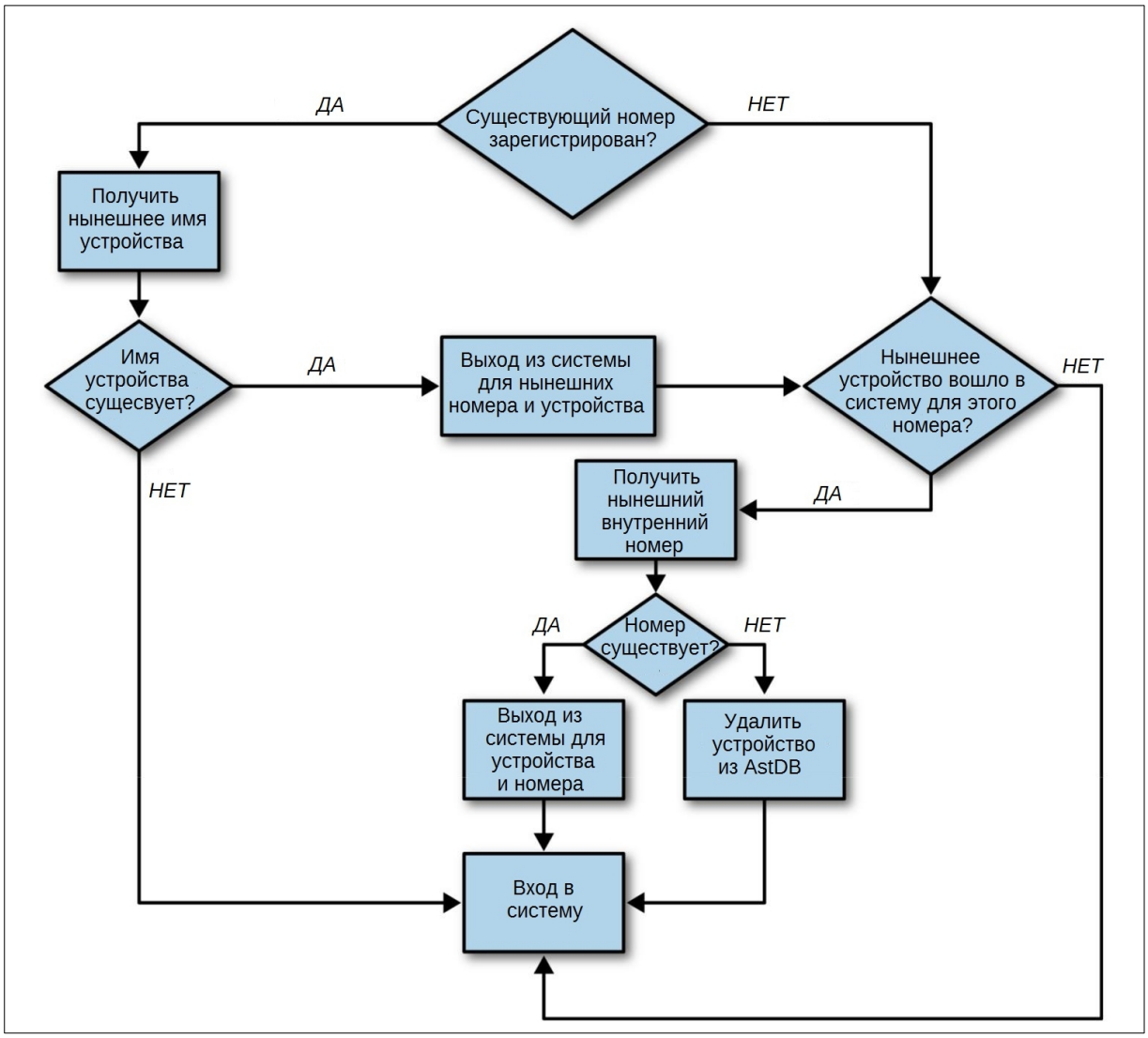
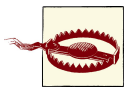


Рисунок 10-2. Прохождение вызова для приложения hot-desking



Возможно, если два человека попытаются войти в один и тот же внутренний номер одновременно или если кто-то еще входит в устройство, которое ранее использовалось внутренним номером для hot-desking (и пыталось войти в систему одновременно с другим человеком), что база данных может выйти из синхронизации. Здесь не было блокировки, чтобы логика была максимально чистой и простой. Если существует большая вероятность того, что люди меняются местами и часто входят и выходят друг за друга, возможно, вы захотите изучить возможность добавления блокировки диалплана, которая может быть выполнена с использованием функций диалплана `LOCK()` и `UNLOCK()`.

```
[HotDesking]
; Контроль диапазона внутренних номеров для использования совпадения шаблонов

; Вход с 71XX выйдет из существующего внутреннего номера в этом месте
; и запишет это устройство с новым номером.
; Выход с 7000 для любого устройства.
;

exten => 7000,1,Verbose(2,Attempting logoff from device ${CHANNEL(peername)})
    same => n,Set(PeerName=${CHANNEL(peername)})
    same => n,Set(CurrentExtension=${DB(HotDesk/${PeerName})})
    same => n,GoSubIf($[${EXISTS(${CurrentExtension})}]?
subDeviceLogoff,1(${PeerName},${CurrentExtension}):loggedoff)
    same => n,GotoIf($[${GOSUB_RETVAL} = 0]?loggedoff)
    same => n,Playback(an-error-has-occurred)
    same => n,Hangup()
```

```
    same => n(loggedoff),Playback(silence/1&agent-loggedoff)
    same => n,Hangup()

exten => _71XX,1,Verbose(2,Attempting to login device ${CHANNEL(peername)}
to extension ${EXTEN:1})
    same => n,Set(NewPeerName=${CHANNEL(peername)})
    same => n,Set(NewExtension=${EXTEN:1})

; Проверьте зарегистрировался ли внутр.номер с этого устройства (NewPeerName)
; -- Если уже существует внутр.номер (ExistingExtension)
;    -- получить имя существующего устройства
;       -- Если устройство не существует
-- (login)  поскольку мы перезапишем существующий внутр. номер для этого
устройства
        -- Если существует имя устройства
           -- logoff ExistingExtension + ExistingDevice
              -- Переход к check_device ----------------------------------+
; -- Если внутр.номер не существует                                       |
;    -- Проверьте, зарегистрировано ли устройство для этого внутр. номера |
;       (NewExtension) <-------------------------------------------------+
;       -- Если устройство присутствует
;          -- Получить существующий внутр.номер
;             -- Если внутр.номер присутствует
;                -- Выйти из устройства и внутр.номера
;                   -- Авторизоваться
;                -- Если внутр номер не присутствует
;                   -- Удалить устройство из AstDB
;                      -- Авторизоваться
;      -- Если для NewExtension нет устройства
;         -- Авторизоваться

; Тесты:
; * Login 100 to 0000FFFF0001
; * Login 101 to 0000FFFF0001 (Результат: Только 101 залогинился)
; * Login 101 to 0000FFFF0002 (Результат: Только 101 залогинился в новом
                                                          ; местоположении)
; * Login 100 to 0000FFFF0001 (Результат: Оба 100 и 101 залогинились)
; * Login 100 to 0000FFFF0002 (Результат: Только 100 залогинился на
                                  ; 0000FFFF0002 -- смена местоположения)
; * Login 100 to 0000FFFF0001 (Результат: Только 100 залогинился)

    same => n,Set(ExistingExtension=${DB(HotDesk/${NewPeerName})})
    same => n,GotoIf($[${EXISTS(${ExistingExtension})}]?get_existing_device)
    same => n(check_device),NoOp()
    same => n,Set(ExistingDevice=${DB(HotDesk/${NewExtension})})
    same => n,GotoIf($[${EXISTS(${ExistingDevice})}]?get_existing_extension)
    same => n,NoOp(Nothing to logout)[6]
    same => n,Goto(login)

    same => n(get_existing_device),NoOp()
    same => n,Set(ExistingDevice=${DB(HotDesk/${ExistingExtension})})
    same => n,GotoIf($[${ISNULL(${ExistingDevice})}]?login)
    same => n,GoSub(subDeviceLogoff,1(${ExistingDevice},${ExistingExtension}))
    same => n,GotoIf($[${GOSUB_RETVAL} = 0]?check_device)
    same => n,Playback(silence/1&an-error-has-occurred)
    same => n,Hangup()
```

6 Не удалось выйти из системы (прим. переводчика)

```
    same => n(get_existing_extension),NoOp()
    same => n,Set(ExistingExtension=${DB(HotDesk/${ExistingDevice})})
    same => n,GoSubIf($[${EXISTS(${ExistingExtension})}]?
subDeviceLogoff,1(${ExistingDevice},${ExistingExtension}):remove_device)
    same => n,GotoIf($[${GOSUB_RETVAL} = 0]?loggedoff)
    same => n,Playback(silence/1&an-error-has-occurred)
    same => n,Hangup()

    same => n(remove_device),NoOp()
    same => n,Set(Result=${DB_DELETE(HotDesk/${ExistingDevice})})
    same => n,Goto(loggedoff)
    same => n(loggedoff),Verbose(2,Existing device and extensions have
been logged off prior to login)[7]
    same => n(login),Verbose(2,Now logging in extension ${NewExtension}
to device ${NewPeerName})[8]
    same => n,GoSub(subDeviceLogin,1(${NewPeerName},${NewExtension}))
    same => n,GotoIf($[${GOSUB_RETVAL} = 0]?login_ok)
    same => n,Playback(silence/1&an-error-has-occurred)
    same => n,Hangup()
    same => n(login_ok),Playback(silence/1&agent-loginok)
    same => n,Hangup()

exten => subDeviceLogoff,1,NoOp()
    same => n,Set(LOCAL(PeerName)=${ARG1})
    same => n,Set(LOCAL(Extension)=${ARG2})
    same => n,ExecIf($[${ISNULL(${LOCAL(PeerName)})} |
${ISNULL(${LOCAL(Extension)})}]?Return(-1))
    same => n,Set(PeerNameResult=${DB_DELETE(HotDesk/${LOCAL(PeerName)})})
    same => n,Set(ExtensionResult=${DB_DELETE(HotDesk/${LOCAL(Extension)})})
    same => n,Return(0)

exten => subDeviceLogin,1,NoOp()
    same => n,Set(LOCAL(PeerName)=${ARG1})
    same => n,Set(LOCAL(Extension)=${ARG2})
    same => n,ExecIf($[${ISNULL(${LOCAL(PeerName)})} | ${ISNULL($
{LOCAL(Extension)})}]?Return(-1))
    same => n,Set(DB(HotDesk/${LOCAL(PeerName)})=${LOCAL(Extension)})
    same => n,Set(DB(HotDesk/${LOCAL(Extension)})=${LOCAL(PeerName)})
    same => n,Set(ReturnResult=${IF($[${DB_EXISTS(HotDesk/${LOCAL(PeerName)})}
& ${DB_EXISTS(HotDesk/${LOCAL(Extension)})}]?0:-1)})
    same => n,Return(${ReturnResult})
```

Полезные функции Asterisk

Теперь, когда мы рассмотрели некоторые из основ, давайте рассмотрим несколько полезных функций, которые были включены в Asterisk.

Zapateller()

`Zapateller()` - это простое приложение Asterisk, которое воспроизводит специальный информационный тон в начале вызова, что приводит к тому, что автодиалеры (обычно используемые телемаркетингами) считают, что линия была отключена. Они не только повесят трубку, но их системы будут отмечать ваш номер как неработающий, что может помочь вам избежать всех видов телемаркетинга. Чтобы использовать эту функцию в своем диалплане, просто вызовите приложение `Zapateller()`.

7 Существующие устройства и внутренние номера были отключены до входа в систему (прим. переводчика)

8 Теперь залогинился внутр.номер `${NewExtension}` на устройстве `${NewPeerName}` (прим. переводчика)

Мы также будем использовать необязательный параметр `nocallerid`, чтобы тон воспроизводился только тогда, когда на входящем вызове нет CallerID абонента. Например, вы можете использовать `Zapateller()` в расширении `s` вашего контекста `[incoming]`, например:

```
[incoming]
exten => s,1,NoOp()
    same => n,Zapateller(nocallerid)
    same => n,Playback(enter-ext-of-person)
```

Парковка вызова

Еще одна удобная функция называется *парковка вызова* (*call parking*). Парковка вызова позволяет поместить вызов на удержание на «парковочное место», чтобы его можно было забрать с другого внутреннего номера. Параметры для парковки вызова (например, расширения для использования, количество мест и т.д.) управляются в файле конфигурации *features.conf*. Раздел `[general]` файла *features.conf* содержит четыре параметра, связанных с парковкой вызовов:

`parkext`

Это внутренний номер для парковки. Передайте вызов этому внутр.номеру, и система сообщит вам, в каком состоянии парковки находится звонок. По умолчанию внутр.номер парковки - `700`.

`parkpos`

Этот параметр определяет количество парковочных мест. Например, установка `701-720` создает 20 парковочных позиций с номерами 701-720.

`context`

Это название контекста парковки. Чтобы иметь возможность парковки вызовов, вы должны включить этот контекст.

`parkingtime`

Если установлено, этот параметр определяет как долго (в секундах) может оставаться звонок на парковке. Если вызов не будет принят в течение указанного времени, номер, который припарковал вызов, будет вызван.

Также обратите внимание, что, поскольку пользователь должен иметь возможность передавать вызовы на внутр.номер парковки, вы должны убедиться, что используете параметры `t` и/или `T` для приложения `Dial()`.

Итак, давайте создадим простой диалплан, чтобы показать парковку вызовов:

```
[incoming]
include => parkedcalls

exten => 103,1,Dial(SIP/Bob,,tT)
exten => 104,1,Dial(SIP/Charlie,,tT)
```

Чтобы показать, как работает парковка вызова, предположим что Алиса звонит в систему и набирает номер 103, чтобы связаться с Бобом. Через некоторое время Боб передает вызов на добавочный номер 700, который сообщает ему, что звонок от Алисы был припаркован в позицию 701. Затем Боб набирает Чарли на добавочном номере 104 и сообщает ему, что Алиса находится на добавочном номере 701. Затем Чарли набирает номер 701 и начинает разговаривать с Алисой. Это простой и эффективный способ передачи абонентов между пользователями.



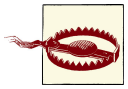
Парковка вызова часто используется в сочетании с пейджинговыми интерфейсами. Регистратор может принять вызов для машинистов в задней части магазина, где они имеют один телефон. Звонок припаркован, и регистратор затем отправляется обратно в магазин, говорит, кому нужен звонок, и какое место парковки им нужно набрать, чтобы получить вызов.

Она также может быть полезна, когда вы совершаете вызов и вам нужно сменить место в здании. Вы можете припарковать вызов, а затем получить его в другом месте в офисе, не возвращаясь в свое первоначальное местоположение, чтобы поговорить с человеком.

Конференц-связь с MeetMe()

И последнее, но не менее важное: давайте рассмотрим настройку аудиоконференц.моста с помощью приложения `MeetMe()`.[9] Это приложение позволяет нескольким абонентам общаться вместе, как если бы они находились в одном и том же физическом местоположении. Некоторые из основных функций:

- Возможность создания конференций, защищенных паролем
- Администрирование конференции (закрытие конференции, блокировка конференции или запуск участников)
- Возможность отключения всех, кроме одного участника (полезная для корпоративных объявлений, трансляций и т. д.)
- Статическое или динамическое создание конференции



Вы должны иметь `res_timing_dahdi`, выбранный в *menuselect*, чтобы иметь возможность компилировать `app_meetme`. Это означает, что вы уже установили DAHDI, даже без аппаратуры телефонии, поскольку `MeetMe()` использует компоненты DAHDI для выполнения микширования звука. Если вы не можете использовать `res_timing_dahdi` в своей системе, вы должны посмотреть на `ConfBridge()`, который может использовать разные источники синхронизации. См. «Конференц-связь с Conf-Bridge()».

Давайте пройдем через базовый конференц-зал. Параметры конфигурации для конференц-системы MeetMe находятся в файле *meetme.conf*. Внутри файла конфигурации вы определяете конференц-залы и необязательные числовые пароли. (Если здесь определен пароль, его должны будут вводить все участники конференции, использующие эту комнату.) Для нашего примера, давайте создадим конференц-зал с номером `600`. Сначала мы создадим конференц-зал в *meetme.conf*. Мы назовем его `600`, и не будем назначать пароль для начала:

```
[rooms]
conf => 600
```

Теперь, когда файл конфигурации готов, нам нужно перезапустить Asterisk, чтобы он мог перечитать файл *meetme.conf*. Затем мы добавим поддержку конференц-зала к нашему диалплану с помощью приложения `MeetMe()`. `MeetMe()` принимает три аргумента: имя конференц-зала (как определено в файле *meetme.conf*), набор параметров и пароль, которые пользователь должен ввести, чтобы присоединиться к этой конференции. Давайте создадим простую конференцию, используя комнату `600`, параметр `i` (который объявляет, когда люди входят и выходят из конференции) и пароль `54321`:

```
exten => 600,1,MeetMe(600,i,54321)
```

Вот и все! Когда вызывающие абоненты вводят внутренний номер `600`, им будет предложено ввести пароль. Если они правильно вводят `54321`, то будут добавлены в конференцию. Вы можете запустить *core show application MeetMe* из CLI Asterisk для списка всех параметров, поддерживаемых приложением `MeetMe()`.

Другим полезным приложением является `MeetMeCount()`. Как следует из названия, это приложение подсчитывает количество пользователей в конкретном конференц-зале. Оно принимает до двух аргументов: конференц-зал, в котором можно подсчитать количество участников и, возможно, имя переменной, чтобы назначить счет. Если имя переменной не передается как второй аргумент, счетчик считывается вызывающему:

9 В мире проприетарных УАТС этот тип функциональности очень дорог. Либо вы должны платить большие деньги за услугу набора номера, либо вам нужно добавить дорогостоящий мост для конференц-связи в свою собственную АТС.

```
exten => 601,1,NoOp()
    same => n,Playback(conf-thereare)
    same => n,MeetMeCount(600)
    same => n,Playback(conf-peopleinconf)
```

Если вы передадите переменную в качестве второго аргумента в `MeetMeCount()`, подсчет присваивается переменной, и воспроизведение счетчика пропускается. Вы можете использовать это, чтобы ограничить количество участников, например:

```
; ограничить конференц-зал 10 участниками
exten => 600,1,NoOp()
    same => n,MeetMeCount(600,CONFCOUNT)
    same => n,GotoIf($[${CONFCOUNT} <= 10]?meetme:conf_full,1)
    same => n(meetme),MeetMe(600,i,54321)

exten => conf_full,1,Playback(conf-full)
```

Разве Asterisk не забава?

Конференц-связь с ConfBridge()

Приложение `ConfBridge()` - это новая горячая точка. Это, по сути, замена приложения `MeetMe()` для Asterisk 10 и более поздних версий.



Asterisk 1.8 также содержит `ConfBridge()`, но его набор функций значительно сокращен от того, что доступно в Asterisk 10 и более поздних версиях.

`ConfBridge()` был введен с использованием новых модулей моста, которые позволяют использовать альтернативные источники синхронизации для микширования, а не только `res_timing_dahdi` как MeetMe(). Кроме того, были добавлены несколько новых функций в `ConfBridge()`, которые недоступны для `MeetMe()`, например:

- Звук высокой четкости, который может быть смешан с частотой дискретизации от 8 кГц до 96 кГц
- Возможности видео, включая добавление динамически переключаемых видеопотоков на основе громкоговорителя
- Динамически управляемая система меню для администраторов конференций и пользователей
- Дополнительные параметры, доступные в файле конфигурации *confbridge.conf*

Мы собираемся начать с базовой конфигурации, чтобы настроить ваш мост для конференции. `ConfBridge()` настроен через файл *confbridge.conf*, который содержит множество опций, включая расширенные функции для пользователей и мостов. Мы начнем с очень простой конфигурации, использующей настройки по умолчанию. Во-первых, давайте создадим разделы `default_user` и `default_bridge` в *confbridge.conf*:

```
$ cat >> confbridge.conf
[general]

[default_user]
type=user

[default_bridge]
type=bridge
```
Ctrl+D

После создания файла *confbridge.conf* нам нужно загрузить модуль *app_confbridge.so*. Это можно сделать на консоли Asterisk:

```
$ asterisk -rx "module load app_confbridge.so"
```

С загруженным модулем мы можем построить простой диалплан для доступа к нашему конференц-мосту:

```
[ConferenceRooms]
exten => 602,1,NoOp()
    same => n,ConfBridge(${EXTEN})

[LocalSets]
include => ConferenceRooms
```



Мы могли бы также определить профили пользователя и моста в приложении `ConfBridge()`. По умолчанию приложение диалплана `ConfBridge()` будет использовать профили `default_user` и `default_bridge`, поэтому мы не определили их в нашем примере, но эквивалент будет выглядеть так:

```
same => n,ConfBridge(${EXTEN},default_bridge,default_user)
```

Конечно, теперь нам нужно перезагрузить наш диалплан:

```
$ asterisk -r
*CLI> dialplan reload
```

Если вы сейчас наберете добавочный номер 602 со своего телефона, вы войдёте в конферен-мост:

```
  == Using SIP RTP CoS mark 5
    -- Executing [602@LocalSets:1] NoOp("SIP/0000FFFF0001-00000001", "") in
new stack
    -- Executing [602@LocalSets:2] ConfBridge("SIP/0000FFFF0001-00000001",
"602") in new stack
    -- <SIP/0000FFFF0001-00000001> Playing 'conf-onlyperson.gsm' (language
'en')
    -- <SIP/0000FFFF0001-00000001> Playing 'confbridge-join.gsm' (language
'en')
```

Это только верхушка айсберга. У нас есть базовая конфигурация, но есть гораздо больше функций для настройки. Перейдите к разделу «Расширенные возможности конференц-связи» в Главе 11.

Вывод

В этой главе мы рассмотрели несколько других приложений в диалплане Asterisk, и, надеюсь, мы предоставили вам еще несколько инструментов, которые вы можете использовать для дальнейшего изучения создания собственных диалпланов. Как и в других главах, мы приглашаем вас вернуться и перечитать любые разделы, требующие разъяснений.

ГЛАВА 11

Парковка, пейджинг и конференц-связь

Я не верю в ангелов, нет. Но у меня есть парковочный ангел. Он на моей приборной панели, и ты его заводишь. Крылья машут, что должно дать вам парковочное место. Это работает до сих пор.

-Billy Connolly B

В этой главе будут рассмотрены два важных аспекта системы УАТС: парковочные вызовы, которые могут быть приняты из местоположения, отличного от того, куда они изначально поступили, и пейджинг, который позволяет объявить для кого вызов и как его можно получить.

В Asterisk эти две функциональности являются эксклюзивными друг для друга и могут использоваться независимо друг от друга. Некоторые предприятия, которые имеют большие склады или сотрудников, которые много перемещаются по офису и необязательно сидят за столом весь день, используют функции пейджинга и парковки своих систем для прямого вызова по всему офису. В этой главе мы покажем вам, как использовать парковку и пейджинг в традиционной настройке, а также несколько более современных решений этих часто используемых функций.

features.conf

Asterisk также предоставляет несколько функций, общих для большинства современных АТС, многие из которых имеют необязательные параметры. В файле *features.conf* вы можете настроить или определить различные параметры функций в Asterisk.

Базовые особенности DTMF

Многие из параметров *features.conf* применяются только при запуске вызовов, которые были запущены в диалплане приложениями `Dial()` или `Queue()` с одним или несколькими из вариантов `K`, `k`, `H`, `h`, `T`, `t`, `W`, `w`, `X` или `x`. Доступные таким образом функции основаны на DTMF (это означает, что они не могут быть доступны через обмен SIP-сообщениями, а только через тональные сигналы в звуковом канале, вызванные пользователями, набирающими требуемые цифры на их панели набора).[1]

Передачу по SIP-каналам (например, с SIP-телефона) можно обрабатывать с использованием возможностей самого телефона что никоим образом не будет затронуто в файле *features.conf*.

1 Да, мы понимаем, что сообщение SIP INFO на самом деле является SIP-сообщением, а не технической частью аудиоканала, но дело в том, что вы не можете использовать кнопку «transfer» или «park» на вашем SIP-телефоне для доступа к этим функциям во время вызова. Вам придется отправить DTMF.

Раздел [general]

В разделе `[general]` файла *features.conf* вы можете определить параметры, которые точно настраивают поведение парковки и переносят функции в Asterisk. Эти параметры перечислены в Таблице 11-1.

Таблица 11-1. features.conf раздел [general]

Опция	Значение/ Пример	Примечание
`parkext`	`700`	Устанавливает внутренний номер по умолчанию, используемый для парковки вызовов.
`parkpos`	`701-720`	Устанавливает диапазон внутренних номеров, используемых в качестве парковки. Припаркованные вызовы могут быть получены путем набора номеров в этом диапазоне.
`context`	`parkedcalls`	Устанавливает контекст по умолчанию, в котором создаются номера для парковки и номера парковочных мест.
`parkinghints`	`no`	Включает/отключает автоматическое создание хинтов диалплана для номеров парковки чтобы телефоны могли подписаться на состояние номеров парковки и парковочных мест. По умолчанию - `no`.
`parkingtime`	`45`	Указывает количество секунд, в течение которых вызов будет ждать на парковке до истечения времени ожидания.
`comebacktoorigin`	`yes`	Настраивает обработку таймаутов припаркованных вызовов. Для получения дополнительной информации о поведении этого параметра см. врезку под названием «Обработка таймаутов припаркованных вызовов с опцией comebacktoorigin».
`courtesytone`	`beep`	Указывает звуковой файл, который будет воспроизводиться у припаркованного абонента, когда вызов будет забран с парковки.
`parkedplay`	`caller`	Указывает, на какой стороне вызова будет воспроизводиться `courtesytone`, когда припаркованный вызов будет забран. Допустимые параметры включают `callee`, `caller`, `both` или `no`. По умолчанию - `no`.
`parkedcalltransfers`	`caller`	Контролирует, какая сторона вызова имеет возможность выполнить передачу DTMF в вызове, который возникает при наборе припаркованного вызова. Допустимые параметры включают `callee`, `caller`, `both` или `no`. По умолчанию - `no`.
`parkedcallreparking`	`caller`	Контролирует, какая сторона вызова имеет возможность выполнить парковку, основанную на DTMF в вызове, который возникает при наборе припаркованного вызова.[a] Допустимые параметры включают `callee`, `caller`, `both` или `no`. По умолчанию - `no`.
`parkedcallhangup`	`caller`	Контролирует, какая сторона вызова имеет возможность выполнить завершение разговора на основе DTMF в вызове, который возникает при наборе припаркованного вызова. Допустимые параметры включают `callee`, `caller`, `both` или `no`. По умолчанию - `no`.
`parkedcallrecording`	`caller`	Контролирует, какая сторона вызова имеет возможность инициировать запись в одно касание на основе DTMF в вызове, который возникает в результате захвата припаркованного вызова. Допустимые параметры включают `callee`, `caller`, `both` или `no`. По умолчанию - `no`.
`parkeddynamic`	`yes`	Включает динамическое создание парковочных мест в диалплане. Должны быть заданы переменные канала `PARKINGDYNAMIC`, `PARKINGDYNCONTEXT` и `PARKINGDYNPOS`.
`adsipark`	`yes`	Передает информацию ADSI о припаркованном вызове обратно в исходный набор.
`findslot`	`next`	Настраивает поведение выбора места для парковки. Дополнительную информацию см. в разделе «Парковочные места».
`perkedmusicclass`	`default`	Указывает класс MOH, который будет использоваться для воспроизведения припаркованным абонентам. Класс музыки, установленный в диалплане с использованием функции канала диалплана `CHANNEL(musicclass)`, переопределит этот параметр.
`transferdigittimeout`	`3`	Устанавливает количество секунд для ожидания каждой цифры от

		вызывающего абонента, выполняющего передачу.
`xfersound`	`beep`	Указывает звук, который будет воспроизводиться, чтобы указать, что перевод завершен.
`xferfailsound`	`beeperr`	Указывает звук, который будет воспроизводиться, чтобы указать, что перевод не завершился.
`pickupexten`	`*8`	Настраивает расширение, используемое для парковки вызова.
`pickupsound`	`beep`	Указывает звук, который будет воспроизводиться при успешной попытке захвата вызова. По умолчанию звук не воспроизводится.
`pickupfailsound`	`beeperr`	Указывает звук, который будет воспроизводиться при неудачной попытке захвата вызова. По умолчанию звук не воспроизводится.
`featuredigittimeout`	`1000`	Устанавливает количество миллисекунд ожидания между цифрами, нажатыми во время вызова, при сопоставлении с DTMF-активируемыми функциями вызова.
`atxfernoanswertimeout`	`15`	Настраивает количество секунд ожидания ответа цели передачи, прежде чем считать, что время ожидания истекло.
`atxferdropcall`	`no`	Настраивает поведение обработки подтверждаемого трансфера, когда передатчик кладет трубку до завершения трансфера и он не выполняется. По умолчанию этот параметр имеет значение `no`, и будет инициирован вызов, чтобы попытаться осуществить трансфер к абоненту, который инициировал его. Если установлено значение `yes`, вызов будет сброшен после недачного трансфера.
`atxferloopdelay`	`10`	Устанавливает количество секунд ожидания между повторами обратного вызова, если `atxferdropcall` установлен на `no`.
`atxfercallbackretries`	`2`	Устанавливает количество попыток обратного вызова, если `atxferdropcall` установлен на `no`. По умолчанию установлены 2 попытки обратного вызова.

[a] Прочтите это снова. Это имеет смысл.

Обработка таймаутов припаркованных вызовов с опцией comebacktoorigin

Этот параметр настраивает поведение парковки вызова, когда время ожидания припаркованных вызовов (см. параметры `parkingtime`). У `comebacktoorigin` может быть одно из двух значений:

`yes` *(по умолчанию)*

Когда превышен тайм-аут припаркованного вызова Asterisk попытается отправить вызов обратно пиру, который припарковал этот вызов. Если канал больше не доступен для Asterisk, вызывающий абонент будет отключен.

`no`

Этот параметр будет использоваться, если вы хотите выполнить пользовательскую функциональность набора номера на припаркованных вызовах, которые превысили тайм-ауты. Абонент будет отправлен в определенную область диалплана, где логика может корректно обрабатывать оставшуюся часть вызова (это может включать просто возврат вызова к другому внутр.номеру, или выполнение какого-либо поиска).

Вам также может потребоваться учитывать вызовы, в которых исходный канал не может обрабатывать возвращенный припаркованный вызов. Если, например, вызов был припаркован каналом, который является транком для другой системы, не было достаточной информации для отправки обратного вызова нужному человеку в той системе. Действия, следующие за тайм-аутом, были бы более сложными, чем `comebacktoorigin=yes`, который мог бы обрабатывать изящно.

Припаркованные вызовы, чей тайм-аут с `comebacktoorigin=no` всегда будут отправлены в контекст `parkedcallstimeout`.



Диалплан (и контексты) подробно обсуждались в Главе 6.

Расширение, на которое они будут отправлены, будет создано из имени канала, который припарковал вызов. Например, если узел с SIP-именем `0004F2040808` припарковал этот вызов, расширение будет `SIP_0004F2040808`.

Если это расширение не существует, вызов будет отправлен на расширение `s` в контексте `parkedcallstimeout`. Наконец, если расширение `s` в `parkedcallstimeout` не существует, вызов будет отправлен на расширение `s` контекста `default`.

Кроме того, для любых вызовов, где `comebacktoorigin=no`, будет существовать расширение `SIP_0004F2040808`, созданное в контексте `park-dial`. Это расширение будет настроено для выполнения `Dial()` на `SIP/0004F2040808`.[2]

Раздел [featuremap]

В этом разделе вы можете определить последовательности DTMF запускающие различные функции на каналах, которые были соединены через опции в приложениях `Dial()` или `Queue()`. Параметры приведены в Таблице 11-2.

Таблица 11-2. feature.conf раздел [featuremap]

Опция	Значение/ Пример	Примечание	Флаги Dial()/Queues()
`blindxfer`	`#1`	Вызывает слепой (неконтролируемый) трансфер	`T, t`
`disconnect`	`*0`	Завершает вызов	`H, h`
`automon`	`*1`	Запускает запись текущего вызова с использованием приложения `Monitor()` (нажатие этой последовательности клавиш второй раз останавливает запись)	`W, w`
`atxfer`	`*2`	Выполняет автоматический трансфер	`T, t`
`parkcall`	`#72`	Парковка вызовов	`K, k`
`automixmon`	`*3`	Запускает запись текущего вызова с использованием приложения `MixMonitor()` (нажатие этой последовательности клавиш снова останавливает запись)	`X, x`



По умолчанию `blindxfer` и `disconnect` коды: `#` и `*` соответственно. Обычно их изменяют по умолчанию, поскольку они будут мешать другим вещам, которые вы, возможно, захотите сделать (например, если вы используете параметр `Tt` в вашей команде `Dial()` каждый раз когда вы нажимаете клавишу `#` вы инициируете передачу).

Раздел [applicationmap]

Этот раздел *features.conf* позволяет сопоставлять коды DTMF с приложениями диалплана. Вызывающий абонент будет переведен на удержание, пока приложение не завершит выполнение.

Синтаксис для определения карты приложения выглядит следующим образом (она должна располагаться в одной строке, разрывы строк не допускаются):[3]

```
<FeatureName> => <DTMF_sequence>,<ActivateOn>[/<ActivatedBy>]
,<Application>([<AppArguments>])[,MOH_Class]
```

2 Мы надеемся вы поймете, что фактическое расширение будет связано с именем канала, которое припарковало вызов, и не будет `SIP_0004F2040808` (если только Leif не продаст Вам телефон Polycom из своей лаборатории).

3 В синтаксисе имеется определенная гибкость (вы можете посмотреть пример файла для подробностей), но наш пример использует стиль, который мы рекомендуем, так как он наиболее соответствует типичному синтаксису диалплана.

Вы делаете следующее:

1. Дайте карте имя чтобы его можно было включить в диалплане за счет использования переменной канала `DYNAMIC_FEATURES`.
2. Определите последовательность DTMF, которая активирует эту функцию (мы рекомендуем использовать для этого как минимум две цифры).
3. Определите на каком канале будет активирована функция, и (необязательно) участника, которому разрешено активировать функцию (по умолчанию это позволяет обоим каналам использовать/активировать эту функцию).
4. Дайте имя приложению, которое эта карта вызывает, и его аргументы.
5. Предоставьте дополнительный класс музыки на удержании (MOH) для назначения этой функции (которую будет слышать противоположный канал при выполнении приложения). Если вы не определите какой-либо класс MOH, вызывающий абонент будет слышать только тишину.

Вот пример карты приложения, которое будет запускать скрипт AGI:

```
agi_test => *6,self/callee,AGI(agi-test.agi),default
```



Поскольку приложения, созданные из карты приложения, запускаются за пределами ядра УАТС, вы не можете использовать приложения, запускающие диалплан (такие как `Goto()`, `Macro()`, `Background()` и т.д.). Если вы хотите использовать карту приложения на запуск внешних процессов (в том числе выполнение кода диалплана) Вам нужно будет запускать внешнее приложение с помощью `AGI()` вызова или приложения `System()`. Дело в том, что если вы хотите, выполнить что-то сложное с помощью карты приложения, то нужно будет очень тщательно протестировать, так как не все будет работать так, как вы могли бы ожидать.

Чтобы использовать карту приложения, вы должны объявить её в диалплане, установив переменную `DYNAMIC_FEATURES` где-то перед командой `Dial()`, которая соединяет каналы. Используйте модификатор в виде двойного подчеркивания для имени переменной, чтобы гарантировать, что карта приложения доступна для обоих каналов в течение всего вызова. Например:

```
exten => 101,1,NoOp()
    same => n,Set(__DYNAMIC_FEATURES=agi_test)
    same => n,Dial(SIP/0000FFFF0002)
```



Если вы хотите разрешить доступ к нескольким картам приложений при вызове, вам нужно будет использовать символ `#` в качестве разделителя между несколькими именами карт:

```
Set(__DYNAMIC_FEATURES=agi_test#my_other_map)
```

Причина, по которой символ `#` был выбран вместо простой запятой заключается в том, что старые версии приложения `Set()` интерпретировали запятую иначе, чем более поздние, а синтаксис карт приложений никогда не обновлялся.

Не забудьте перезагрузить модуль функций после внесения изменений в файл features.conf:

```
*CLI> features reload
```

Вы можете проверить, что ваши изменения произошли с помощью команды *features show*. Убедитесь, что вы проверили свою карту приложений, прежде чем передавать ее своим пользователям!

Наследование переменных канала

Переменные канала всегда связаны с исходным каналом, который их установил, и более недоступны после трансфера канала.

Чтобы позволить канальным переменным следить за каналом, когда он передается по системе, необходимо использовать наследование переменной канала. Существует два модификатора, которые могут позволить переменной канала следовать за каналом: одинарное подчеркивание и двойное подчеркивание.

Одинарное подчеркивание (_) заставляет канальную переменную наследоваться каналом для одного трансфера, после чего она больше недоступна для дополнительных трансферов. Если вы используете двойное подчеркивание (__) переменная канала будет наследоваться на протяжении всего срока жизни этого канала.

Настройка переменных канала для наследования требует, чтобы вы использовали один или два символа подчеркивания в качестве префикса имени канала. Затем на переменные канала ссылаются точно так же, как и в обычном режиме (например, не пытайтесь считывать значения переменных канала с символами подчеркивания в имени переменной).

Вот пример установки переменной канала для единичного наследования:

```
exten => example,1,Set(_MyVariable=thisValue)
```

Вот пример установки переменной канала для бесконечного наследования:

```
exten => example,1,Set(__MyVariable=thisValue)
```

Чтобы прочитать значение переменной канала, не используйте подчеркивания:

```
exten => example,1,Verbose(1,Value of MyVariable is: ${MyVariable})
```

И если вы считаете, что эта врезка была захватывающей, посмотрите ещё одну!

Динамическое создание Feature-Map из диалплана.

Начиная с Asterisk 11, вы можете создавать сопоставление функций из диалплана напрямую, что делает определение функции и ее отображение DTMF динамическим по каждому каналу. Это делается с помощью функций диалплана `FEATURE()` и `FEATUREMAP()`. Допустимые значения для `FEATUREMAP()` включают следующие значения, которые устанавливают или извлекают используемую последовательность DTMF для запуска функциональности:

atxfer

Трансфер вызова с уведомлением

blindxfer

Слепой трансфер

automon

Авто `Monitor()` (Запись вызова)

disconnect

Разъединение вызова

parkcall

Парковка вызова

automixmon

Авто MixMonitor() (запись вызова)

С помощью `FEATUREMAP()` функция может быть использована для получения текущей последовательности DTMF для этой функциональности:

```
exten => 199,1,Verbose(2,Current DTMF для atxfer: ${FEATUREMAP (atxfer)})
```

Или вы можете использовать последовательность DTMF для особой функции на текущем канале:

```
exten => 199,1,NoOp()
    same => n,Set(FEATUREMAP(atxfer)=*9)
```

Если вы хотите установить тайм-аут парковки для канала, то можете сделать это с помощью функции `FEATURE()`. Она содержит один аргумент - время парковки, которое является значением в секундах до того, как припаркованный вызов возвращается к абоненту (или назначению, в зависимости от того как вы настроили парковку):

```
exten => 199,1,NoOp()
    same => n,Set(FEATURE(parkingtime)=60)
```

Группировка сопоставлений приложений

Если у вас есть множество функций, которые необходимо активировать для определенного контекста или внутреннего номера, вы можете группировать несколько элементов вместе в группе сопоставлений приложений, так что одно назначение переменной `DYNAMIC_FEATURES` присваивает все назначенные функции этому сопоставлению.

Группировка сопоставлений приложений добавляется в конец файла *features.conf*. Каждой группе присваивается имя, а затем перечисляются соответствующие функции:

```
[shifteight]
unpauseMonitor => *1    ; присвоение пользовательского ключа
pauseMonitor => *2      ; присвоение пользовательского ключа
agi_test =>             ; не присвоено пользовательского ключа
```



Если вы хотите указать пользовательское сопоставление ключей для объекта в группировке сопоставлений приложений, просто выполните `=>` с соответствующим сопоставлением ключей. Если вы не укажете сопоставление ключей, будет использовано сопоставление ключей по умолчанию для этой функции (как указано в разделе `[featuremap]`). Независимо от того, хотите ли вы назначить сопоставление настраиваемых ключей или нет требуется оператор `=>`.

В диалплане вы должны назначить эту группу приложений с помощью приложения `Set()`:

```
Set(__DYNAMIC_FEATURES=shifteight)  ; используйте двойное подчеркивание,
                                    ; если вы хотите, чтобы оба плеча вызова
                                    ; имели назначенную переменную.
```

Парковочные места

Парковочные места (лоты) позволяют проводить вызов в системе без привязки к определенному номеру. Затем вызов может быть получен любым, кто знает код парковки для этого вызова. Эта функция часто используется в сочетании с системой служебного оповещения (система PA или Tannoy для наших читателей в Великобритании). По этой причине её часто называют парк-энд-пейдж[4]; однако следует отметить, что парковка и пейджинг фактически раздельны.

Для парковки вызова в Asterisk вам необходимо передать вызывающего абонента в код функции, назначенный для парковки, который назначается в файле *features.conf* директивой `parkext`. По умолчанию это `700`:

```
parkext => 700    ; Какой номер набрать для парковки (все парковочные места)
```

4 park-and-page - парковка и пэйджинг (прим. переводчика)

Вам нужно подождать, чтобы завершился трансфер пока не получите номер места для поиска парковки из системы, иначе у вас не будет способа получить звонок. По умолчанию поисковые слоты, назначенные с помощью директивы `parkpos` в *features.conf*, пронумерованы как `701-720`:

```
parkpos =>  701-720     ; Какие номера использовать для парковки вызовов
                        ; (парковочные места по умолчанию)
```

После того, как вызов припаркован, любой из пользователей системы может получить его, набрав номер поискового слота (`parkpos`), назначенный этому вызову. Затем вызов будет подключен к каналу, набирающему код поиска.

Существует два способа назначения слотов для поиска. Это делается с помощью директивы `findslot` в файле *features.conf*. Метод по умолчанию (`findslot => first`) всегда использует слот с наименьшим номером, если он доступен, и, при необходимости присваивает коды с более высоким номером. Второй метод (`findslot => next`) будет перемещаться по кодам извлечения при каждой последующей парковке, возвращаясь к первому коду извлечения после использования последнего. Какой метод вы выберете, будет зависеть от того, насколько заняты ваши парковочные места. Если вы используете парковку редко, по умолчанию `findslot` в `first` будет лучше (люди будут использовать их припаркованные вызовы всегда находящиеся в одном и том же слоте). Если вы используете парковку часто (например, в автосалоне), гораздо лучше для каждой последующей парковки назначать следующий слот, так как вы часто будете иметь более одного одновременно припаркованного вызова. Ваши пользователи привыкнут внимательно слушать фактический номер парковки (вместо того, чтобы просто всегда набирать `701`) и это минимизирует вероятность того, что люди случайно получат неправильный вызов в занятой системе.

Если вы используете парковку, вам, вероятно, также понадобится способ объявлять о припаркованных вызовах, чтобы заинтересованные стороны знали, как их получить. Хотя можно просто бегать по коридору и кричать “Вася, там тебе звонят на `701`!” более профессиональный метод заключается в использовании системы пейджинга (более формально известной как система публичных адресов), которую мы обсудим в следующем разделе.

Пейджинг потолочный и «под носом» (aka система оповещения).

Во многих системах УАТС желательно иметь возможность пользователю отправлять свой голос с телефона в систему оповещения. Обычно это включает в себя набор кода функции или номера, который делает соединение с каким-либо ресурсом системы оповещения, а затем делает объявление через трубку телефона, которое транслируется на все устройства, связанные с этим пейджинговым ресурсом. Часто это будет внешняя пейджинговая система, состоящая из усилителя, подключенного к потолочным громкоговорителям; однако пейджинг через громкоговорители офисных телефонов также популярен (в основном по соображениям стоимости). Если у вас есть бюджет (или существующая потолочная пейджинговая система), потолочный пейджинг обычно лучше, но настольный пейджинг (aka пейджинг «под носом») может хорошо работать во многих местах. Что пожалуй является наиболее распространенным, состоит в том, чтобы иметь сочетание настольного и потолочного пейджинга где, например, настольный пейджинг может использоваться для офисов, а потолочный будет использоваться для склада, прихожей и общественных мест (столовая, приемная , и т.д.).

В Asterisk для пейджинга используется приложение `Page()`. Это приложение просто берет список каналов в качестве аргумента, одновременно вызывает все перечисленные каналы, и, как они отвечают, помещает каждый из них в конференц-зал. Имея это в виду, становится очевидным, что одно требование для работы поискового вызова заключается в том, что каждый канал назначения должен иметь возможность автоматически отвечать на входящее соединение и выводить звук на какой-то динамик (другими словами, `Page()` не будет работать, если все телефоны просто звонят).

Таким образом, в то время как приложение `Page()` само по себе является безболезненным и простым в использовании, получение всех каналов назначения для правильной обработки входящего пейджинга немного сложнее. Мы вернемся к этому в ближайшее время.

Приложение `Page()` принимает три аргумента, определяя группу каналов, которые должны быть подключены к пейджингу, параметры и таймаут:

```
exten => *724,1,Page(${ChannelsToPage},i,120)
```

Параметры (обозначенные в Таблице 11-3) дают вам некоторую гибкость в отношении того, как работает `Page()`, но большая часть конфигурации будет связана с тем, как целевые устройства обрабатывают входящее соединение. Мы рассмотрим различные способы настройки устройств для приема пейджинга в следующем разделе.

Таблица 11-3. Параметры Page()

Опция	Описание	Разъяснение
`d`	Включает полнодуплексный звук	Иногда называется «пейджинг для разговора», использование этого параметра означает, что оборудование, которое получает пейджинг, имеет возможность передавать звук обратно на тот же канал, откуда оно получает звук. Как правило, вы не захотите использовать его, если у вас нет конкретной потребности в нем.
`i`	Игнорирует попытки переадресации вызова	Обычно вы хотите, чтобы эта опция была включена.
`q`	Не воспроизводит звуковой сигнал для вызывающего абонента (тихий режим)	Обычно вы не будете использовать это, но если у вас есть внешний усилитель, который обеспечивает свой собственный тон, вы можете установить эту опцию.
`r`	Записывает пейджинг в файл	Если вы собираетесь использовать один и тот же пейджинг несколько раз в будущем, вы можете записать его, а затем использовать позже, вызывая с помощью `Originate()` или используя параметр `A(x)` для `Page()`.
`s`	Набирает канал только в том случае, если состояние устройства `NOT_INUSE`	Этот вариант, вероятно, полезен (и надежен) для SIP-каналов, но даже в этом случае может не работать, если одной линии разрешено одновременно выполнять несколько вызовов. Поэтому во всех случаях не полагайтесь на эту опцию.
`A(x)`	Воспроизводит объявление `x` для всех участников	Вы можете использовать ранее записанный файл для воспроизведения по пейджинговой системе. Если вы объедините это с `Originate()` и `Record()`, то можете реализовать систему отложенного пейджинга.
`n`	Не воспроизводит объявление вызывающему абоненту (подразумевает `A(x)`)	По умолчанию система будет воспроизводить исходящий звук как для вызывающего абонента, так и для вызываемого абонента. Если этот параметр включен, то звук вызова не будет воспроизводиться вызывающему абоненту (персональный пейджинг).

Из-за того, как работает команда `Page()`, она очень ресурсоемка. Мы не можем этого достаточно подчеркнуть. Внимательно читайте дальше, и мы расскажем, как обеспечить, чтобы пейджинг не вызывал проблем с производительностью в рабочей среде (что почти наверняка произойдет, если пейджинг не будет правильно спроектирован).

Места для отправки пейджинга

Как мы уже говорили ранее, `Page()` сама по себе очень проста. Хитрость заключается в том, как собрать всех вместе. Пейджинги могут быть отправлены на различные типы каналов, и все они требуют различной конфигурации.

Внешний пейджинг

Если в здании установлена система оповещения, обычно подключается телефонная система к внешнему усилителю и вызов отправляется на нее по каналу. Один из способов сделать это - подключить звуковую карту вашего сервера к усилителю и отправлять вызовы на канал с именем `Console/DSP`, но это предполагает, что звуковые драйверы на вашем сервере работают правильно, а уровни звука нормализованы правильно на этом канале. Другой способ (потенциально простой, и возможно более надежный) для обработки внешнего поискового вызова заключается в использовании FXS устройства некоторого типа (например, ATA), который подключается к интерфейсу поискового вызова, такому как Bogen UTI1,[5] который затем подключается к пейджинговому усилителю.[6]

В вашем диалплане пейджинг на внешний усилитель будет выглядеть как простой `Dial()` для устройства, подключенного к пейджинговому оборудованию. Например, если у вас был ATA, настроенный в *sip.conf* как `[PagingATA]` и вы подключили ATA к Bogen UTI1, вы выполните пейджинг, набрав:

```
exten => *724,1,Verbose (2, Пейджинг на внешний усилитель)  ; обратите внимание
                                                            ; '*' в наборе является частью того
                                                            ; что вы на самом деле набираете
    same => n, Set(PageDevice=SIP/PagingATA)
    same => n, Page(${PageDevice},i,120)
```

Обратите внимание, что для этого вам придется регистрировать ATA как SIP устройство в *sip.conf*, и в этом случае мы назвали устройство `[PagingATA]`. Вы можете назвать это устройство как угодно (например, мы часто используем MAC-адрес как имя SIP-устройства), но для чего-либо, что не является телефоном пользователя, может быть полезно использовать имя, которое делает его выделяющимся из других устройств.

Если в вашей системе была FXS-карта, и вы подключили UTI1 к ней, вы бы делали `Dial()` на канал для этого порта FXS:

```
same => n,Dial(DAHDI/25)
```

UTI1 отвечает на вызов и открывает канал к пейджинговой системе; вы затем делаете свое объявление и вешаете трубку.

Настольный пейджинг

Настольный пейджинг стал популярным в кнопочных телефонных системах, где спикеры офисных телефонов используются в качестве системы оповещения бедняков. Большинство SIP-телефонов имеют возможность автоответа на вызов по громкой связи, который выполняет то, что в основном требуется от каждого телефона. В дополнение к этому, однако, необходимо передать аудио более чем на один аппарат одновременно. Asterisk использует встроенный механизм конференц-связи для обработки деталей под капотом. Вы используете приложение `Page()`, чтобы это произошло.

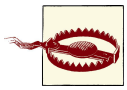
Подобно `Dial()`, приложение `Page()` может обрабатывать несколько каналов. Так как вы, как правило будете хотеть, чтобы `Page()` сигнализировал нескольким аппаратам сразу (возможно даже всем телефонам в вашей системе) вы можете получить длинные строки с устройствами, которые выглядят примерно так:

```
Page(SIP/SET1&SIP/SET2&SIP/SET3&SIP/SET4&SIP/SET5&SIP/SET6&SIP/SET7&...
```

> При превышении определенного размера система Asterisk не сможет создать несколько наборов пейджинга. Например, в офисе с 200 телефонами использование SIP для пейджинга

5 Bogen UTI1 полезен, поскольку он может обрабатывать всевозможные входящие и исходящие соединения, что почти гарантирует, что вы сможете безболезненно подключить свою телефонную систему к любому внешнему пейджинговому оборудованию, независимо от того, насколько оно старое или малоизвестное.

6 В этой книге мы предполагаем, что внешнее пейджинговое оборудование уже установлено и работает со старой телефонной системой.



каждого телефона было бы невозможно; трафик и нагрузка на процессор сервера Asterisk были бы слишком велики. В подобных случаях вы должны смотреть или на многоадресный пейджинг или на внешний.

Возможно, самая сложная часть пейджинга на основе SIP - заключается в том, что вам обычно приходится указывать каждому аппарату, что он должен отвечать автоматически, но у разных производителей SIP-телефонов для этой цели используются разные SIP-сообщения. Таким образом, в зависимости от используемой модели телефона команды, необходимые для выполнения пейджингового вызова на основе SIP, будут отличаться. Вот несколько примеров:

- Для Aastra:

```
exten => *724,1,Verbose(2,Paging to Aastra sets)
    same => n,SIPAddHeader(Alert-Info: info=alert-autoanswer)
    same => n,Set(PageDevice=SIP/00085D000000)
    same => n,Page(${PageDevice},i)
```

- Для Polycom:

```
exten => *724,1,Verbose(2,Paging to Polycom sets)
    same => n,SIPAddHeader(Alert-Info: Ring Answer)
    same => n,Set(PageDevice=SIP/0004F2000000)
    same => n,Page(${PageDevice},i)
```

- Для Snom:

```
exten => *724,1,Verbose(2,Paging to Snom sets)
    same => n,Set(VXML_URL=intercom=true)

; заменить 'domain.com' на домен вашей системы
    same => n,SIPAddHeader(Call-Info: sip:domain.com\;answer-after=0)
    same => n,Set(PageDevice=SIP/000413000000)
    same => n,Page(${PageDevice},i)
```

- Для Cisco SPA (бывшие телефоны Linksys, но не серии 79XX):

```
exten => *724,1,Verbose(2,Paging для Cisco SPA, но не Cisco 79XX)
    same => n,SIPAddHeader(Call-Info:\;answer-after=0)      ; Cisco SPA phones
    same => n,Set(PageDevice=SIP/0004F2000000)
    same => n,Page(${PageDevice},i)
```

Предполагаем, что вы поняли, что произойдет, если у вас смесь телефонов в вашей рабочей среде? Как вы будете контролировать, какие заголовки отправлять на телефоны?[7]

Как бы то ни было, это не красиво.

К счастью, многие из этих аппаратов поддерживают многоадресную (multicast) IP-рассылку, которая является гораздо лучшим способом отправки пейджинга на несколько телефонов (подробности читайте далее). Тем не менее, если у вас есть лишь горстка телефонов в вашей системе и все они принадлежат одному и тому же производителю, то пейджинговая система на основе SIP может быть самым простым методом, поэтому мы не хотим запугивать вас.

Многоадресный пейджинг через канал MulticastRTP

Если вы серьезно относитесь к пейджинговому вызову через телефоны в вашей системе и у вас больше, чем горстка телефонов, вам нужно будет посмотреть, как использовать многоадресную IP-

7 Подсказка: канал Local будет вашим другом в этом случае.

рассылку. Понятие multicast IP существует уже давно,[8] но широко не используется. Тем не менее, оно идеально подходит для пейджинга в одном месте.

Asterisk имеет канал (`chan_multicast_rtp`), предназначенный для создания многоадресной рассылки RTP. Затем этот поток подписывается различными телефонами, и результат заключается в том, что всякий раз когда появляется медиапоток в многоадресной рассылке, телефоны передают этот медиапоток на свои динамики.

Поскольку `MulticastRTP` является драйвером канала, у него нет приложения, но вместо этого он будет работать в любом месте диалплана, где вы могли бы иначе использовать канал. В нашем случае мы будем использовать приложение `Page()` для инициирования многоадресной рассылки.

Чтобы использовать многоадресный канал, вы просто отправляете ему вызов так же, как и любому другому каналу. Синтаксис для канала выглядит следующим образом:

```
MulticastRTP/<тип>/<ip address:port>[/<linksys address:port>]
```

Тип может быть `basic` или `linksys`. Основной синтаксис канала `MulticastRTP` выглядит следующим образом:

```
exten => *723,1,Page(MulticastRTP/basic/239.0.0.1:1234)
```

Не все устройства поддерживают IP multicast, но мы протестировали его на Snom,[9] Linksys/Cisco, Polycom (прошивка 4.x или позже), и Aastra, и он работает замечательно.

Многоадресный пейджинг на Cisco SPA-телефонах

Функция многоадресной рассылки на телефонах Cisco SPA немного странна, но после ее настройки она работает нормально. Хитрость заключается в том, что адрес, который вы помещаете в телефон, не является адресом многоадресной передачи, на который ссылается пейджинг, а скорее своего рода сигнальным каналом.

Мы обнаружили, что вы можете сделать этот адрес таким же, как адрес многоадресной рассылки, но просто используйте другой номер порта.

Диалплан выглядит так:

```
exten => *724,1,Page(MulticastRTP/linksys/239.0.0.1:1234/239.0.0.1:6061)
```

В SPA-телефоне вам необходимо войти в интерфейс администрирования и перейти к вкладке *SIP*. В самой нижней части страницы вы найдете раздел *Linksys Key System Parameters* (Параметры системы клавиш Linksys). Вам необходимо установить следующие параметры:

- `Linksys Key System: Yes`
- `Multicast Address: 239.0.0.1:6061`

Обратите внимание, что адрес многоадресной рассылки, который вы назначаете для телефона, является вторым в определении канала (в нашем примере тот, который использует порт 6061).

Обратите внимание, что вы можете написать команду `Page()` в этом формате в среде, где есть сочетание телефонов SPA (fka Linksys, теперь Cisco) и других типов телефонов. Другие телефоны будут использовать первый адрес и будут работать так же, как если бы вы использовали `basic` вместо `linksys`.

8 У него даже есть свое собственное зарезервированное IP-адресное пространство класса D от 224.0.0.0 до 239.255.255.255 (но считайте IP multicast, прежде чем вы просто захватите один из них и назначите его). Части этого адресного пространства являются частными, части-общедоступными, а части предназначены для целей, отличных от тех, для которых их можно использовать. Дополнительные сведения о многоадресной рассылки см. на этой странице Википедии.

9 Очень громко, и нет возможности настроить усиление.

VoIP-пейджинговые адаптеры

В последнее время на рынке появились различные пейджинговые колонки на основе VoIP. Эти устройства адресуются в диалплане точно так же, как SIP ATA, подключенный к UTI1, но они могут быть установлены так же, как и потолочные громкоговорители. Поскольку они автоматически отвечают, нет необходимости передавать им какую-либо дополнительную информацию, как вам необходимо в случае SIP-телефона.

Для небольших установок (где требуется не более полудюжины динамиков), эти устройства могут быть экономически эффективными. Однако для чего-либо большего (или установки в сложной среде, такой как склад или автостоянка), вы получите более высокую производительность с гораздо меньшей стоимостью с помощью традиционной аналоговой пейджинговой системы, подключенной к телефонной системе аналоговым (FXS) интерфейсом.

Мы не знаем, поддерживают ли эти устройства многоадресную рассылку. Имейте это в виду, если вы планируете использовать их в большом количестве.

Комбинации пейджинга

Во многих организациях может возникнуть необходимость как в настольном, так и во внешнем пейджинге. Например, производственное предприятие может захотеть использовать оповещение на основе телефонов для офисной зоны, а внешний пейджинг для завода и склада. С точки зрения Asterisk это довольно просто. Когда вы вызываете приложение `Page()`, вы просто указываете различные ресурсы, на которые хотите направить пейджинг, разделенные символом & и все они будут включены в конференцию приложением `Page()`.

Соединим все вместе

На этом этапе у вас должен быть список различных типов каналов, на которых вы хотите разместить пейджинг. Поскольку `Page()` почти всегда будет сигнализировать более одного канала, мы рекомендуем установить глобальную переменную, которая определяет список каналов, а затем вызвать приложение `Page()` этой строкой:

```
[global]
MULTICAST=MulticastRTP/linksys/239.0.0.1:1234
;MULTICAST=MulticastRTP/linksys/239.0.0.1:1234/239.0.0.1:6061     ; если у вас
                                                                  ;SPA телефоны
                                                              ; (Linksys / Cisco)
BOGEN=SIP/ATAforPaging  ; Это предполагает ATA в файле sip.conf с именем;
                        ; [ATAforPaging]
;BOGEN=DAHDI/25         ; Мы могли бы это сделать, полагая, что у нас есть
                        ; аналоговый FXS на 25 канале DAHDI
PAGELIST=${MULTICAST}&${BOGEN}      ; Все эти имена переменных произвольны.
                                    ; Asterisk все равно, как вы называете эти
                                    ; строки

[page_context]    ; Вам не нужен контекст пейджинга, если вы используете номер
                  ; телефона, пейджинг можно набирать вашими телефонами
exten => *724,1,Page(${PAGELIST},i,120)
```

В этом примере предлагается несколько возможных конфигураций, в зависимости от аппаратного обеспечения. Хотя необязательно иметь переменную `PAGELIST`, мы обнаружили что это будет способствовать упрощению управления несколькими ресурсами пейджингового вызова, особенно во время процесса настройки и тестирования.

Мы создали контекст для пейджингового вызова в этом примере. Чтобы это сработало, вам нужно либо сделать `include` этого контекста в контексты диалплана, в которые входят ваши телефоны или код `Goto()` в этих контекстах, чтобы добавить пользователя в этот контекст и номер (то есть

`Goto(page_context,*724,1)`). В качестве альтернативы вы можете жестко задавать внутренний номер для приложения `Page()` каждом контексте, как номер набора сервисов.

Зонирование пейджинга

Зонирование пейджинга популярно в таких местах, как автосалоны, где отдел запчастей, отдел продаж и, возможно, отдел подержанных автомобилей требуют пейджинга, но им не нужно слышать пейджинги друг друга.

В зонировании пейджинга человек, отправляющий пейджинг, должен выбрать, в какую зону он хочет его отправить. Контроллер зонирования, такой как Bogen PCM2000, обычно используется для сигнализации различных зон: приложение `Page()` сигнализирует контроллеру зоны, отвечает контроллеру зоны, а затем отправляется дополнительная цифра, чтобы выбрать, в какую зону должен быть отправлен пейджинг. Большинство контроллеров зонирования позволяют использовать пейджинг для всех зон, в дополнение к объединению зон (например, пейджинг для отделов продаж новых и подержанных автомобилей).

Вы также можете иметь отдельные внутренние номера в диалплане, чтобы разделить ATA (или группы телефонов), но это может оказаться более сложным и дорогостоящим, чем просто покупка пейджингового контроллера, который предназначен для обработки этого. Зонирование пейджинга не требует какой-либо существенно другой технологии, но для этого требуется немного больше размышлений и планирования относительно диалплана и аппаратного обеспечения.

Расширенные возможности конференц-связи

Приложение `ConfBridge()` представляет собой расширенное приложение для конференц-связи в Asterisk, которое позволяет осуществлять мелкомасштабный контроль участников конференции, помимо аудио- и видеоконференций высокой четкости. Текущая реализация `ConfBrifge()` была выпущена вместе с Asterisk 10.[10] Начиная с Asterisk 10 приложение `ConfBridge()` является фактической заменой `MeetMe()`, которая все еще работает хорошо, но не содержит расширенных функций `ConfBridge()`. Ранее мы представили базовую рабочую настройку для `ConfBridge()` (см. «Конференц-связь с ConfBridge()» в Главе 10) и предложили Вам начать работу, если вы еще не настроили свой первый конференц-мост.

В файле *confbridge.conf* мы настраиваем как `users` (пользователей) так и типы `bridge` (мостов). Тип `users` позволяет нам создавать профили, которые мы можем назначить пользователям, прежде чем они войдут в конференцию, чтобы контролировать свои права доступа и назначать разные функции. Например может быть профиль участника и профиль администратора, где администратор может отключать участников конференции. Другой пример - на конференцию может не передаваться аудиопоток, пока не войдет лидер конференции. Помимо профиля `users` мы также имеем профили `bridge`, которые используются для определения атрибутов самого моста конференции. Некоторые параметры включают определение максимального количества участников или запись конференции (и место сохранения записанных файлов).

В разделе «Конференц-связь с ConfBridge()» в Главе 10 мы настроили наши типы пользователей и мостов по умолчанию с минимальным набором данных. Давайте рассмотрим доступные варианты для каждого из типов.

Раздел [general]

Раздел `[general]` зарезервирован для будущего использования и в настоящее время не содержит каких-либо глобальных параметров.

10 `ConfBridge()` существует с Asterisk 1.8, но это была простая реализация, которая не содержала аудио- или видеоконференции высокой четкости.

Параметры профилей пользователей

Параметры в Таблице 11-4 для профилей пользователей, которые определяют опции, доступные участникам конференции.

Таблица 11-4. Параметры профиля пользователя в confbridge.conf

Опция	Описание
admin	Определяет, помечен ли пользователь как администратор конференции. Пользователям, отмеченным как администраторы, могут быть предоставлены различные параметры, доступные только администраторам в пользовательском меню. Меню определяется в *confbridge.conf* и выбирается при вызове приложения ConfBridge(). Доступными параметрами являются yes или no. По умолчанию - no.
marked	Задает, должен ли пользователь в этом профиле быть отмечен (промаркирован) или нет. Используется для запуска конференции при ожидании маркированного пользователя. См. wait_marked и end_marked. Доступные параметры: yes или no. По умолчанию - no.
startmuted	Устанавливает для пользователей в этом профиле значение “звук отключен” при первоначальном присоединении к конференции. Доступные параметры: yes или no. По умолчанию - no.
music_on_hold_when_empty	Определяет, следует ли воспроизводить MOH, когда существует только один участник конференции или когда конференция ожидает маркированного пользователя. Доступные параметры: yes или no. По умолчанию - no.
music_on_hold_class	Устанавливает, какой класс MOH должен использоваться. Значение по умолчанию default.
quiet	Если этот параметр включен, он ограничивает звуки, воспроизводимые в конференции, например звуки присоединения и объявления пользователей. Доступные параметры: yes или no. По умолчанию - no.
announce_user_count	Если включено, количество пользователей в конференции объявляется присоединяющемуся участнику до входа в конференцию. Доступные параметры: yes или no. По умолчанию - no.
announce_user_count_all	Используется для объявления количества участников всем участникам конференции. Если задано число, то объявление воспроизводится только тогда, когда число участников превышает заданное число. Доступные параметры: yes, no или целое число. По умолчанию - no.
announce_only_user	Если включено, подсказка будет воспроизводиться когда первый участник конференции присоединяется, сообщая, что он является единственным участником конференции. Доступные параметры: yes или no. По умолчанию - yes.
wait_marked	Если этот параметр включен, участник конференции должен дождаться присоединения маркированного пользователя. Доступные параметры: yes или no. По умолчанию - no.
end_marked	Определяет, удаляются ли оставшиеся пользователи после того, как последний маркированный пользователь покинет конференцию. Доступные параметры: yes или no. По умолчанию - no.
dsp_drop_silence	Когда включено, Asterisk отбросит то, что он определяет как тишину в конференции, вызывая резкое нарастание фонового шума в конференции. Рекомендуется для больших конференций, где фоновый шум может стать проблемой. Доступные параметры: yes или no. По умолчанию - no.
dsp_talking_threshold	Значение в миллисекундах—продолжительность звука остающаяся выше исходного

	значения, установленного DSP. Значение не должно быть изменено, пока вы не знакомы с внутренним устройством того, как это может повлиять на вашу конференцию. Смотри файл *confbridge.conf.sample* в каталоге *contribs* исходников Asterisk для получения дополнительной информации.
`dsp_silence_threshold`	Похож на `dsp_talking_threshold`, но ищет тишину. Значение в миллисекундах. Не рекомендуется настраивать.
`talk_detection_events`	Если включено, уведомление о том, когда оратор начинает и заканчивает говорить, передается как событие по AMI. Доступные параметры: `yes` или `no`. По умолчанию - `no`.
`denoise`	Опция `denoise` полезна, если вы используете кодек *speex*, а диктор имеет повышенный уровень фонового шума. При включении этой опции будет предпринята попытка удалить фоновый шум до того, как звук будет микширован в конференцию, сохраняя желаемый звук речи. Этот параметр не следует путать с параметром `dsp_drop_silence`. Кроме того, эта опция имеет небольшое влияние на производительность. Доступные параметры: `yes` или `no`. По умолчанию - `no`.
`jitterbuffer`	Если включено, то jitterbuffer будет включен на аудиоканале пользователя до микширования. Это желательно в том случае, если может помочь сгладить звук, воспроизводимый в конференции за счет небольшой задержки. Эта опция использует адаптивный режим функции `JITTERBUFFER()`. Если требуется точная настройка jitterbuffer, отключите эту опцию и используйте функцию `JITTERBUFFER()` перед вызовом приложения `ConfBridge()`. Доступные параметры: `yes` или `no`. По умолчанию - `no`.
`pin`	Если установлен, то человеку, при входе в конференцию будет предложено ввести PIN-код. Допустимое значение - любое целое число.
`announce_join_leave`	Если эта функция включена, пользователю, входящему в конференцию, будет предложено записать свое имя до присоединения к конференции. Затем будет воспроизведено имя, объявляющее человека, присоединяющегося и покидающего конференцию. Доступные параметры: `yes` или `no`. По умолчанию - `no`.
`dtmf_passthrough`	Если этот параметр включен, DTMF будет передаваться через конференцию. Это полезно, когда мост конференции может быть связан с оконечной точкой, от которой вы хотите получить DTMF; иначе, он поглощается Asterisk. Доступные параметры: `yes` или `no`. По умолчанию - `no`.
`announcement`	Если этот параметр задан, при присоединении пользователей к конференции будет воспроизводиться объявление с именем. Значение должно быть путем к файлу объявления.

Параметры для профилей соединения

Параметры, перечисленные в Таблице 11-5 для профилей соединений и определения параметров для самой конференции.

Таблица 11-5. Опции профиля соединения в confbridge.conf

Опция	Описание
`max_members`	Определяет максимальное число участников конференции для одного моста. При достижении этого лимита конференция будет заблокирована до тех пор, пока лишний участник не покинет конференцию. Единственным исключением является то, что администраторы всегда могут присоединиться к конференции, независимо от количества участников. Значение должно быть целым числом. По умолчанию количество участников не ограничено.
`record_conference`	Если эта опция включена, запись конференции начнется, когда первый участник присоединится к конференции, и прекратится, когда последний участник покинет конференцию. Имя файла для записи имеет формат *confbridge-<имя моста*

	конференции-<*время начала*>.*wav*. Файл по умолчанию будет записываться линейно с частотой 8 кГц. Запись будет сохранена в каталоге монитора, настроенном в *asterisk.conf*. Доступные параметры: `yes` или `no`.
`record_file`	Если параметр `record_conference` включена, можно указать имя файла для записанной конференции. Однако, несколько конференций могут использовать один и тот же профиль моста, не рекомендуется задавать этот параметр в самом файле *confbridge.conf*. Вместо этого используйте функцию `CONFBRIDGE()` для динамической установки имени файла в диалплане перед входом в конференцию.
`internal_sample_rate`	Эта опция установит внутреннюю частоту дискретизации конференции, при которой будет происходить микширование. По умолчанию частота дискретизации выбирается автоматически, однако можно указать значение от 8 000 до 192 000. Если задать частоту дискретизации, которую не поддерживает Asterisk, будет использоваться ближайшая, поддерживаемая Asterisk. Доступные значения: `auto` или значение от `8000` до `192000`. Значение по умолчанию - `auto`.
`mixing_interval`	Этот параметр задает внутренний интервал микширования моста конференции в миллисекундах. Установка более высокого интервала микширования может уменьшить количество нагрузки, используемой большими конференциями за счет более слабосвязанной конференции (например, задержки). Допустимые значения: `10`, `20`, `40`, и `80`. Значение по умолчанию - `20`.
`video_mode`	Параметр `video_mode` используется для управления распределением видео среди участников конференции, которые могут создавать и/или просматривать видеопотоки.[a] Участники, которые хотят просматривать и быть источником видео, должны совместно использовать один и тот же видеокодек, например H.264 (в *sip.conf* используйте `allow=h264` в дополнение к аудиокодекам). Кроме того, рекомендуется отключить jitterbuffer, так как jitterbuffer работает только на аудиочасти конференции, и, таким образом, может привести к сбою синхронизации аудио и видео. Доступные режимы можно увидеть в Таблице 11-6.

[a] Микширование различных видеокодеков невозможно. Кроме того, видео, отображаемое в конференции, транслирует только одного участника за раз (т. е. Вы не можете сделать видео “Brady Bunch”).

Таблица 11-6. Доступные видеорежимы

Опция	Описание
`none`	По умолчанию источники видео не выбраны. Можно все еще включить источник видеосигнала для конференции через DTMF или AMI.
`follow_talker`	Источником видео будет тот, кто говорит на конференции (самый громкий) и у которого есть источник видео. Источник видео будет видеть последний выбранный Источник видео, а не свой собственный.
`last_marked`	Источником видео будет последний маркированный пользователь, присоединившийся к конференции. Если к конференции присоединилось несколько маркированных участников конференции и ушел последний маркированный пользователь, то источником станет участник, присоединившийся непосредственно перед последним (и так далее).
`first_marked`	Как и в `last_marked`, первый маркированный участник конференции с источником видеосигнала будет источником видеосигнала для конференции. Если этот участник уйдет, то следующий маркированный участник с видеосигналом станет источником видеосигнала для конференции.

Пользовательские звуки конференц-связи

Также в разделе профиля моста вы можете определить пользовательские звуки, которые будут использоваться для этого профиля. Параметры, перечисленные в Таблице 11-7 показывают доступные настройки звука.

Таблица 11-7. Доступные параметры звука для профилей связи

Опция	Описание

sound_join	Звук, воспроизводимый, когда участник присоединяется к конференции
sound_leave	Звук воспроизводится, когда участник покидает конференцию
sound_has_joined	Звук, воспроизводимый при объявлении имени присоединяющегося участника
sound_has_left	Звук, воспроизводимый при объявлении имени покидающего участника
sound_kicked	Звук воспроизводимый участнику, когда он был удален из конференции
sound_muted	Звук, воспроизводимый участнику, когда у него был отключен звук
sound_unmuted	Звук, воспроизводимый участнику, когда у него был возвращен звук
sound_only_person	Звук, воспроизводимый участнику, когда он является единственным участником конференции
sound_only_one	Звук, воспроизводимый присоединяющемуся участнику, если в конференции есть только один другой участник
sound_there_are	Звук, воспроизводимый при объявлении количества участников конференции
sound_other_in_party	Смежный звук используется с sound_there_are; звуковые файлы объединяются следующим образом: sound_there_are *number_of_participants* sound_other_in_party
sound_place_into_conference	Звук, воспроизводимый участнику при помещении в конференцию после ожидания присоединения маркированного пользователя
sound_wait_for_leader	Звук, воспроизводимый участникам, уведомляющий их об ожидании присоединения к конференции маркированного пользователя
sound_leader_has_left	Звук воспроизводится, когда последний макрированный пользователь покинул конференцию
sound_get_pin	Звук, воспроизводимый участнику при запросе пин-кода для конференции
sound_invalid_pin	Звук, воспроизводимый участнику при вводе неверного PIN-кода
sound_locked	Звук, воспроизводимый участнику при попытке присоединиться к заблокированной конференции
sound_locked_now	Звук, воспроизводимый администратору после блокировки конференции
sound_unlocked_now	Звук, воспроизводимый администратору после разблокировки конференции
sound_error_menu	Звук воспроизводится при вводе недопустимого параметра меню

Параметры меню ConfBridge

Приложение диалплана ConfBridge() также имеет возможность настройки меню для участников конференции. Параметры доступны через DTMF и могут быть указаны в качестве аргумента для приложения ConfBridge(), такого как профили пользователей и профили мостов. В Таблице 11-8 мы рассмотрим доступные параметры для пользовательских меню, как определено в *confbridge.conf*.



Можно объединить несколько параметров вместе с использованием одной последовательности DTMF. Выполнение нескольких действий выполняется путем разграничения каждого действия запятой (,).

Таблица 11-8. Параметры меню ConfBridge()

Опция	Описание
playback(<*audio_filename*>[[&<audio_filename>]])	Опция playback может использоваться для воспроизведения аудио участнику, вводящему строку DTMF. Аудио не может быть прервано этой опцией. Похожее по стилю на приложение Playback().
playback_and_continue(<*audio_filename*>[[&<audio_filename>]])	Подобно playback, за исключением того, что оно будет прослушивать DTMF во время воспроизведения звука. Это полезно в ситуациях, когда вы создаете аудио-меню и хотите разрешить DTMF во время воспроизведения. Аналогично стилю приложения Background().
toggle_mute	Включает отображение состояния включения и выключения звука для участника. В то время как mute включен, звук участника не будет воспроизводиться в конференц-мосте, но он все равно сможет слушать.
no_op	Опция no_op означает No Operation (Не операция). Её цель - просто зарезервировать последовательности DTMF в меню.

`decrease_listening_volume`	Уменьшить громкость прослушивания участника.
`increase_listening_volume`	Увеличить громкость прослушивания участника.
`reset_listening_volume`	Сбросить громкость прослушивания на значение по умолчанию.
`decrease_talking_volume`	Уменьшить громкость разговора участника.
`increase_talking_volume`	Увеличить громкость разговора участника.
`reset_talking_volume`	Сбросить громкость разговора участника на значение по умолчанию.
`dialplan_exec(context,extension,priority_label)`	Использование опции `dialplan_exec` позволяет участнику покинуть конференцию, выполнить диалплан и в конце диалплана вернуться в конференцию.
`leave_conference`	Позволяет участнику покинуть конференцию с помощью последовательности DTMF. Выполнение диалплана будет продолжено после приложения `ConfBridge()`.
`admin_kick_last`	Позволяет администратору выгнать последнего присоединившегося участника из конференции. Этот параметр доступен только администраторам, поэтому его можно безопасно включить в общее меню для пользователей и администраторов.
`admin_toggle_conference_lock`	Позволяет администратору переключаться между включением и выключением конференции. Может использоваться только администраторами, даже если включено в меню пользователя.
`set_as_single_video_src`	Позволяет участнику установить себя в качестве единственного источника видеосигнала для конференции. Это позволяет прикрепить видео к одному участнику, независимо от того, в каком режиме установлен `video_mode`.
`release_as_single_video_src`	Освобождает участника от роли единственного источника видеосигнала для моста. После освобождения участника мост конференции возвратится к `video_mode`, который был установлен для конференции.
`admin_toggle_mute_participants`	Позволяет администратору переключаться между отключением и включением звука всех участников конференции, не являющихся администраторами. Администраторы по-прежнему смогут выступать на конференции. Если этот параметр включен, все участники, включая администраторов, будут уведомлены о том, что звук конференции отключен.
`participant_count`	При использовании, скажет участнику сколько всего участников в конференции.

Включение PIN-кода

В «Конференц-связь с ConfBridge()» в Главе 10 мы создали базовую конфигурацию, которая позволила нам присоединиться к конференц-связи на основе `ConfBridge()`. Теперь мы собираемся расширить её и добавить PIN-код в профиль пользователя. Если вы еще не настроили базовую конференцию, то вернитесь и сделайте это сейчас, прежде чем продолжить.

Сначала мы начнем с нашей базовой конфигурации *confbridge.conf* и просто добавим опцию `pin` для профиля `default_user`:

```
[general]

[default_user]
type = user
pin = 1234
```

```
[default_bridge]
type = bridge
```

После этого мы просто запускаем *module reload app_confbridge.so* из консоли и попытаемся позвонить нашему мосту конференции на номер 602. При этом мы увидим вывод консоли, похожий на следующий:

```
*CLI> module reload app_confbridge.so
    -- Reloading module 'app_confbridge.so' (Conference Bridge Application)
  == Using SIP RTP CoS mark 5
    -- Executing [602@LocalSets:1] NoOp("SIP/0000FFFF0001-00000003", "") in new
stack
    -- Executing [602@LocalSets:2] ConfBridge("SIP/0000FFFF0001-00000003",
"602") in new stack
    -- <SIP/0000FFFF0001-00000003> Playing 'conf-getpin.gsm' (language 'en')
    -- <SIP/0000FFFF0001-00000003> Playing 'conf-onlyperson.gsm' (language 'en')
    -- <SIP/0000FFFF0001-00000003> Playing 'confbridge-join.gsm' (language 'en')
    -- <Bridge/0x7fdddc004378-input> Playing 'confbridge-join.gsm' (language
'en')
```

Довольно просто, правда? А что, если мы хотим сделать ПИН-код динамичным? Возможно, мы хотим найти ПИН-код в базе данных, которая может быть изменена внешним приложением. Мы можем сделать это просто с помощью динамических профилей в нашем диалплане. Сначала мы начнем со статически определенного PIN-кода, но установим его из диалплана.

Мы начнем с этого простого диалплана:

```
[ConferenceRooms]
exten => 602,1,NoOp()
    same => n,ConfBridge(${EXTEN})
```

Затем мы используем функцию `CONFBRIDGE()` для динамического задания параметров профиля до присоединерия к конференции. Мы будем использовать `CONFBRIDGE()`, чтобы установить другой PIN-код, который мы определили в профиле `default_user`:

```
[ConferenceRooms]
exten => 602,1,NoOp()
    same => n,Set(CONFBRIDGE(user,pin)=4321)
    same => n,ConfBridge(${EXTEN})
```

После внесения изменений просто выполните *dialplan reload* и наберите добавочный номер 602. Вам все равно будет предложено ввести ПИН-код, но на этот раз это будет 4321 вместо 1234.

Рискуя раздражать вас разговорами о PIN-кодах конференций, давайте рассмотрим еще один пример. На этот раз мы переопределим ПИН-код по умолчанию, указанный в AstDB. Мы также могли бы вставить ПИН-код из внешней базы данных, такой как MySQL, но AstDB является приятной и простой. (См. Главу 16 для получения информации о `func_odbc`, которую вы могли бы использовать для вставки ПИН-кода из реляционной базы данных.)

Сначала давайте поместим ПИН-код в нашу AstDB из консоли Asterisk:

```
*CLI> database put confbridge 602/pin 1200
Updated database successfully
*CLI> database show confbridge
/confbridge/602/pin                               : 1200
```

С нашим PIN-кодом 1200, установленным для конференц-моста 602, мы можем изменить наш диалплан, чтобы найти ПИН-код, который мы должны использовать для `ConfBridge()`:

```
[ConferenceRooms]
exten => 602,1,NoOp()
```

```
; поиск пин-кода для конференции
    same => n,Set(CONF_PIN=${DB(confbridge/${EXTEN}/pin)})

; проверка на пустое возвращаемое значение
    same => n,GotoIf($[${ISNULL(${CONF_PIN})}]?join)

; установка пин-кода
    same => n,Set(CONFBRIDGE(user,pin)=${CONF_PIN})

; вход в конференцию
    same => n(join),ConfBridge(${EXTEN})
```

После выполнения *dialplan reload* наш консольный вывод будет выглядеть так при наборе внутреннего номера 602:

```
    -- Executing [602@LocalSets:1] NoOp("SIP/0000FFFF0001-00000006", "")
in new stack
    -- Executing [602@LocalSets:2] Set("SIP/0000FFFF0001-00000006",
"CONF_PIN=1200") in new stack
    -- Executing [602@LocalSets:3] GotoIf("SIP/0000FFFF0001-00000006",
"0?join") in new stack
    -- Executing [602@LocalSets:4] Set("SIP/0000FFFF0001-00000006",
"CONFBRIDGE(user,pin)=1200") in new stack
    -- Executing [602@LocalSets:5] ConfBridge("SIP/0000FFFF0001-00000006",
"602") in new stack
```

Вы заметите, что мы добавили ярлык приоритета, называемый `join`, помимо вызова `ConfBridge()`. Мы сделали это, чтобы у нас было место для перехода на нулевое значение[11], возвращаемое из поиска AstDB. Он просто дает нам один дополнительный уровень контроля, так что, если PIN-код не был возвращен, мы перейдем к ПИН-по умолчанию, установленному в *confbridge.conf*.

Ожидание присоединения маркированного пользователя

В некоторых случаях полезно создать конференцию, в которой все участники могут присоединиться к конференции до начала и ждать в зале ожидания до того, как присоединится лидер конференции. Мы можем включить эту функцию с использованием пары опций и маркированного пользователя. Маркированный пользователь будет лидером конференции и, когда он присоединится, то заставит всех участников переместиться в конференц-зал, где они смогут разговаривать друг с другом в обычном режиме.



Мы будем использовать функцию `CONFBRIDGE()` для настройки нашего обычного и маркированного участника, но Вам можно просто изменить файл *confbridge.conf*, если вы предпочитаете этот метод.

Во-первых, давайте настроим наш диалплан, чтобы люди, присоединившиеся к конференции, должны были ждать маркированного пользователя и удаляться из конференции когда последний маркированный пользователь покидает конференцию. (Несколько маркированных пользователей могут присоединиться к конференции, поэтому, если, скажем, три менеджера присоединились к конференции, и одному нужно уйти раньше, конференция продолжится до тех пор, пока не уйдет последний менеджер.) Например:

```
[ConferenceRooms]
; обычный участник
exten => 602,1,NoOp()
    same => n,Set(CONFBRIDGE(user,wait_marked)=yes)
```

11 Даже если бы мы сделали что-то вроде `Set(CONFBRIDGE(user,pin)=)`, статически определенный ПИН в *confbridge.conf* будет иметь приоритет, что означает, что невозможно переопределить ПИН-код ни на что. Мы считаем это функцией безопасности.

```
    same => n,Set(CONFBRIDGE(user,end_marked)=yes)
    same => n,Goto(conference,1)

; маркированный участник (с пин)
exten => 603,1,NoOp()
    same => n,Set(CONFBRIDGE(user,marked)=yes)
    same => n,Set(CONFBRIDGE(user,pin)=1200)
    same => n,Goto(conference,1)

; тот же мост для всех
exten => conference,1,NoOp()
    same => n,ConfBridge(primary)
```

Давайте немного пройдемя по нашему диалплану. Во-первых, у нас есть два внутренних номера: 602 и 603. Номера 602 не имеет PIN-кода и настроит людей, присоединяющихся к конференции, дожидаться, когда присоединится первый маркированный пользователь и удалит их из конференции, когда последний маркированный пользователь уйдет из конференция. После установки параметров `wait_marked` и `end_marked` на `yes` - `Goto()` переводит выполнение диалплана туда, где запущено приложение `ConfBridge()`. Номер 603 используется для установки маркированного пользователя перед присоединением к конференции. Мы также требуем, чтобы они вводили ПИН-код, поскольку мы не хотим, чтобы любой входил в качестве маркированного пользователя.

Этого достаточно. После того, как вы изменили свой диалплан, просто выполните *dialplan reload*, и все это будет соусом.

Использование меню ConfBridge()

Меню `ConfBridge()` - это метод выполнения различных функций на конференциях, вводя последовательности DTMF с вашего телефона и вызывая действие, оставаясь в конференции. Использование меню на основе DTMF в `ConfBridge()` осуществляется путем определения его в файле *confbridge.conf*. После того, как вы включили свое меню, вы можете передать имя своего меню в приложение `ConfBridge()`, которое сделает его активным для этого участника. Различные параметры меню перечислены и описаны в «Параметры меню ConfBridge». Сначала давайте посмотрим, как определить новое меню.

Наше первое меню: управление громкостью

Давайте создадим простое меню в файле *confbridge.conf*. Просто поместите его в нижней части файла ниже раздела `[default_bridge]`:



Не забывайте, что `type=menu` является первой строкой под заголовком, иначе ваше меню не будет загружаться в Asterisk.

```
[volume_ctrl_menu]
type=menu
*5=toggle_mute
1=increase_listening_volume
4=decrease_listening_volume
7=reset_listening_volume
3=increase_talking_volume
6=decrease_talking_volume
9=reset_talking_volume
```

Меню, которое мы создали, позволяет участнику переключать свой статус отключения звука, увеличивать и уменьшать громкость разговоров и прослушивания, а также сбрасывать их. Чтобы включить это меню, мы просто передаем `volume_ctrl_menu` в качестве аргумента приложению `ConfBridge()`. Включение меню выполняется путем перезагрузки модуля *app_confbridge.so*:

```
*CLI> module reload app_confbridge.so
    -- Reloading module 'app_confbridge.so' (Conference Bridge Application)
  == Parsing '/etc/asterisk/confbridge.conf': Found
```

Теперь давайте изменим наш диалплан, чтобы любой, кто присоединится к нашей конференции, мог управлять громкостью на конференции. Мы будем использовать наш пример из раздела "Ожидание присоединения маркированного пользователя" в качестве основы для этой конфигурации:

```
 ; same bridge for everyone
exten => conference,1,NoOp()
same => n,ConfBridge(primary,,,volume_ctrl_menu)
```

Единственное изменение, которое мы должны были сделать, это добавить `,,,volume_ctrl_menu` после слова `primary` в строке, которая загружает приложение `ConfBridge()`. Нам нужны три ведущие запятые, так как меню является четвертым аргументом. (второй аргумент - профиль моста; третий - профиль пользователя, который мы оставили пустым, так как мы используем профили по умолчанию.)

После сохранения изменений перезагрузите диалплан:

```
*CLI> dialplan reload
```

Войдите в мост конференций. После входа в систему, попробуйте набрать `*5`. Вы должны услышать сообщение, в котором говорится: “Ваш микрофон выключен." Если вы снова наберете `*5`, вы услышите "Ваш микрофон включён.”

Расширенное меню: использование dialplan_exec

Допустим, мы подумали о некоторой функциональности, которую хотим выполнить с моста конференции, но просматривая доступные опции меню `ConfBridge()` ни один из них не удовлетворяет нашему требованию. К счастью, у нас есть опция `dialplan_exec`, которая позволяет нам выполнять любую функциональность, которую мы хотим, используя программирование диалплана.

Пример, который мы собираемся использовать - это временный выход из конференции, предоставление меню для ввода номера телефона, который вы хотите присоединить к конференции и запрос на возвращение на конференцию вместе с новым вызванным участником. Давайте посмотрим, как мы можем сделать это с помощью опции `dialplan_exec`:

```
[ConferenceRooms]
; обычный участник
exten => 602,1,NoOp()
    same => n,Goto(conference,1)

exten => conference,1,NoOp()
    same => n,Set(thisBridge=primary)
    same => n,ConfBridge(${thisBridge},,,volume_ctrl_menu)

exten => conference_joiner,1,NoOp()
    same => n,Read(numberToDial,vm-enter-num-to-call)
    same => n,Originate(SIP/my_itsp/${numberToDial},exten,ConferenceRooms,602,1)
```

В первой части нашего диалплана мы определяем номер 602, который просто делает `Goto()` в расширение конференции. Мы удалили конфигурацию функции `CONFBRIDGE()` так как она здесь не требуется. Расширение конференции затем присоединяется к мосту конференции и переходит в наше `volume_ctrl_menu`, который будет содержать последовательность DTMF для выполнения следующей части диалплана. Расширение `conference_joiner` где происходит волшебство. Первая строка выполняет `Read()`, которое просит участника ввести номер, которому он хочет позвонить.



В этом случае мы установили минимальный функционал, необходимый для того, чтобы заставить систему делать то, что мы хотим. В производственной системе вы, вероятно, захотите ввести некоторые дополнительные элементы управления и проверки, чтобы убедиться, что набраны действительные номера и т.д.

В следующей строке используется приложение `Originate()` для вызова участника. Первый аргумент - конечная точка, которую мы вызываем; мы используем пира `SIP/my_itsp` вместе с номером, запрошенным в приложении `Read()` для создания строки запроса. Второй аргумент - это слово `exten`, означающее, что мы хотим подключить вызываемого пользователя к расширению в диалплане. Следующие три аргумента являются местоположением в диалплане: контекст `ConferenceRooms`, расширение `602` и приоритет `1`.

После внесения изменений в наш диалплан, нам нужен способ выполнить расширение `conference_joiner` один раз внутри конференц-моста. Мы делаем это с модификацией файла `confbridge.conf`. Мы изменим существующий параметр `volume_ctrl_menu` меню `ConfBridge()`, чтобы добавить параметр DTMF `dialplan_exec`:

```
[volume_ctrl_menu]
type=menu
*5=toggle_mute
1=increase_listening_volume
4=decrease_listening_volume
7=reset_listening_volume
3=increase_talking_volume
6=decrease_talking_volume
9=reset_talking_volume
0=dialplan_exec(ConferenceRooms,conference_joiner,1)
```

После внесения изменений необходимо перезагрузить наш диалплан и приложение конференц-моста:

```
*CLI> dialplan reload
*CLI> module reload app_confbridge.so
```

И теперь мы можем проверить нашу функциональность, позвонив в конференц-мост, а затем нажав цифру `0`. Нам будет предложено ввести номер, по которому мы хотим позвонить. После ввода номера, Asterisk вызовет его и попытается присоединиться к ним к мосту конференции после ответа.

Включение видеоконференций

Одной из основных особенностей нового приложения `ConfBridge()` является использование видеоконференций. Вероятно, лучшая часть этого заключается в том, что это действительно довольно просто реализовать. Есть несколько вещей, которые вы должны знать о возможных проблемах в реализации видеоконференций:

- Все участники видео должны использовать один и тот же видеокодек; перекодирование видео в Asterisk недоступно.
- В Asterisk нет мультиплексирования видео; одновременно участнику может быть показан только один источник видео.

Помимо этих двух предостережений, видеоконференция в Asterisk работает довольно хорошо. Давайте начнем с включения нескольких опций в *sip.conf*, так как мы сосредоточимся на использовании SIP-клиентов для нашей конфигурации.

В *sip.conf* есть два варианта, которые мы должны включить, оба из них мы можем включить в разделе `[general]` (если мы просто хотим включить его вообще для всех наших пиров) или в конфигурации пира. Мы будем контролировать, какие пиры получают параметры, изменив существующий шаблон office-phone:

```
[office-phone](!)
type=friend
context=LocalSets
host=dynamic
nat=force_rport,comedia
secret=welcome
dtmfmode=auto
videosupport=yes
disallow=all
allow=g722
allow=ulaw
allow=alaw
allow=h264
```

Мы добавили `videosupport=yes` и `allow=h264` в наш шаблон `office-phone` в *sip.conf*. С нашими изменениями там, давайте перезагрузим модуль `chan_sip` пока мы думаем об этом:

```
*CLI> module reload chan_sip.so
```

Мы можем проверить наши изменения, запустив *sip show peer <peer_identifier>* в консоли Asterisk:

```
*CLI> sip show peer 0000FFFF0001
...
  User=Phone      : No
  Video Support   : Yes
  Text Support    : No
...
  Def. Username   : 0000FFFF0001
  SIP Options     : (none)
  Codecs          : (ulaw|alaw|g722|h264)
  Codec Order     : (g722:20,ulaw:20,alaw:20,h264:0)
...
```

Следующий шаг - включить поддержку видео в файле *confbridge.conf*. Включение `video_mode` в профиле моста позволит нам установить тип видеорежима, включенного в конференц-мосте. Существует четыре типа, которые мы описываем в разделе “Параметры для профилей мостов”:

- `none`
- `follow_talker`
- `last_marked`
- `first_marked`

Наиболее популярным вариантом является `follow_talker`, который заставляет источник видео переключаться между участниками моста конференции на основе того, кто говорит. Мы включим его в нашем профиле моста в *confbridge.conf*:

```
[default_bridge]
type=bridge
video_mode=follow_talker
```

При любых изменениях в *confbridge.conf*, нам необходимо перезагрузить *app_confbridge.so*:

```
*CLI> module reload app_confbridge.so
```

Вот и все. Мы можем теперь войти в конференц-мост с клиентом SIP, который поддерживает видео H.264. Клиент, который был хорошо протестирован с Asterisk - это Jitsi, он является кросс-платформенным для Windows, OSX и Linux.

Вывод

В этой главе мы рассмотрели файл *features.conf*, который содержит функциональные возможности для включения передачи на основе DTMF, включение записи вызовов во время вызова и настройки парковки для одной или нескольких компаний. Мы также рассмотрели различные способы объявления звонков и информации людям в офисе, используя несколько методов пейджинга, включая традиционные системы пейджинга и многоадресный пейджинг телефонных аппаратов на столах сотрудников. После этого мы углубились в приложение `ConfBridge()`, которое отличается исключительной гибкостью в настройке и богатством доступных функций. Это исследование различных методов реализации традиционных функций парковки, пейджинга и конференц-связи современным способом, как мы надеемся, показало вам гибкость, которую может предложить Asterisk.

Маршрутизация интернет-вызовов

There ain't no such thing as a free lunch (TANSTAAFL).
Не существует такой вещи, как бесплатный обед
-Robert Heinlein.

Одна из достопримечательностей VoIP - это концепция избегания использования PSTN в целом и маршрутизация всех вызовов непосредственно между конечными точками, использующими Интернет, практически без затрат. Хотя технология для этого уже давно существует, реальность такова, что большинство телефонных вызовов по-прежнему стоят денег, даже те, которые маршрутизируются через VoIP-услуги.

С технологической точки зрения все еще существует множество систем, которые не могут обрабатывать маршрутизацию VoIP-вызовов с использованием чего-либо, кроме клавиатуры на телефоне.

С культурной точки зрения мы все еще привыкли звонить друг другу, используя цифровую строку (ака номер телефона). С VoIP, идея иметь возможность звонить кому-то с помощью *имя@домен* (так же, как мы делаем с электронной почтой) имеет смысл, но есть несколько вещей, которые нужно рассмотреть, прежде чем мы сможем туда добраться.

Так что все держит?

freenum.org

Первые несколько разделов этой главы могут полностью отвлечь вас от всей идеи, поэтому мы хотим начать с того, что *freenum.org* предлагает временное решение для всего беспорядка, который настолько изящен, что мы не можем видеть причин, по которым все в сообществе VoIP не смогут его охватить.[1] Основная идея *freenum.org* в том, что люди привыкли использовать цифровую клавиатуру на своем телефоне и потому, что в Интернете есть неиспользуемый RFC, описывающий то, что позволяет индивидуальные планы нумерации, почему бы не связать эти вещи вместе и предоставить DNS-совместимый план нумерации, который легко настраивается и работает с существующими технологиями? Это может показаться немного сложным, но это довольно просто.

1 Серьезно, получите доступ к *freenum.org* и получите свой ISN сегодня. Это просто и бесплатно, и скоро все крутые дети будут иметь его.

DNS и SIP URI

Система доменных имен (DNS) предназначена для того, чтобы люди могли локализовать ресурсы в Интернете. Хотя в конечном счете все соединения между конечными точками обрабатываются с помощью численных IP-адресов, это может быть очень полезно чтобы связать имя (например, *www.google.com*)с тем, что на самом деле может быть несколькими IP-адресами.

В случае VoIP использование доменного имени может выглядеть примерно `100@192.168.1.1` (*`внутренний_номер@сервер`*) и сделать его доступным как `leif@shifteight.org` (который выглядит гораздо более сексуально на визитной карточке).

SIP URI

SIP обычно выглядит как `sip:`*`endpoint@domain.tld`*. В зависимости от вашего SIP-клиента вы можете набирать SIP URI каке *`endpoint@domain.tld`* или даже точно так же, как *`endpoint`* (если у вас есть прокси-сервер и конечная точка, которую вы вызываете, является частью вашего домена).

Для телефона SIP, который часто имеет только цифровую клавиатуру, может быть проблематично набирать SIP URI по имени[2], поэтому стало привычным использовать числовой набор для доступа к внешним ресурсам. Мы также используем «телефонные звонки» с использованием «телефонных номеров». Однако сам протокол SIP понимает только *`resource@address`*, поэтому все, что вы набираете, должно в конечном итоге быть преобразовано в этот формат, прежде чем SIP сможет что-либо с этим сделать. Обычно единственная причина, по которой вы можете набрать что-либо по «номеру телефона» с вашего телефона SIP - это то, что вы зарегистрированы на ресурсе, который понимает, как преобразовать числовые строки, которые вы набираете, в SIP URI.

В Asterisk часть URI *`resource`* (часть перед @) должна соответствовать расширению в диалплане.[3] Часть *`address`* будет адресом (или именем хоста) самого сервера Asterisk. Итак, URI `sip:100@shifteight.org` закончится на расширении `100`, где-то в диалплане сервера, который предоставляет услугу SIP для `shifteight.org`.

То, что набирается (`100`), никоим образом не может относиться к фактическому идентификатору подключенной конечной точки. Например, у нас может быть пользователь по имени Leif, чей телефон может быть устройством, которое регистрируется по его MAC-адресу и поэтому может быть чем-то вроде `0000FFFF0001@192.168.1.99`[4]. Значительная часть диалплана Asterisk заключается в упрощении адресации для пользователей и обработке сложностей различных протоколов, поддерживаемых Asterisk.

SRV записи

Service Record (SRV) представляет собой несколько новый тип DNS записи, которая предоставляет информацию о доступных услугах. Определенный в RFC 2782, он часто используется более новыми протоколами (одним из них является SIP). Если вы хотите поддерживать поиск SIP в своем домене, для правильного ответа вам потребуется соответствующая запись SRV.

Когда SIP-соединение выполняет поиск по `leif@shifteight.org`, для целей SIP запись SRV может реагировать на то, что запрошенная услуга (SIP) фактически найдена на сервере `pbx.shifteight.org` (или, возможно, даже на полностью другом домене, таком как `pbx.tothemoon.net`).

Интернет хостинг-провайдеры обычно предлагают веб-интерфейс для настройки DNS-записей, но многие из них не обеспечивают хороший интерфейс для SRV записей (при условии что они

2 Вы знаете, где находится символ `@` на вашей телефонной цифровой клавиатуре?

3 Имейте в виду, что расширение в Asterisk может быть любой буквенно-цифровой строкой, такой как `leif` или `100`.

4 Вы можете фактически набрать этот URI непосредственно с телефона, минуя сервер Asterisk, но вы можете увидеть как набрав `100` будет гораздо более простым чем пытаясь понять, как ввести `0004f2a1b2c3@192.168.1.99` в вашем телефоне, используя только цифровую клавиатуру (кстати, это может быть сделано).

предлагают что-либо вообще). Обычно вы можете легко настроить записи A и MX, но записи SRV могут быть более сложными. Если ваш хост не поддерживает записи SRV, вам нужно будет переместить свой DNS-хостинг другому провайдеру, если вы хотите иметь возможность поддерживать поисковые запросы SIP SRV для вашего домена.

Большинство DNS-серверов используют BIND (Berkeley Internet Name Daemon). Запись BIND для записи SRV для SIP будет выглядеть примерно так:

```
_sip._udp.shifteight.org. 86400 IN SRV 0 0 5060 pbx.shifteight.org.
```

Форма записи подробно описана в Таблице 12-1.

Таблица 12-1. Компоненты записи SIP SRV

Имя	Описание	Пример
Service	Символьное имя службы	`_sip.`
Proto	Транспортный протокол	`_udp.`
Name	Доменное имя для этой записи[a]	`shifteight.org.`
TTL	Время жизни (в секундах)	`86400`
Class	Поля класса DNS (всегда IN)	`IN`
Priority	Целевой приоритет хоста	`0`
Weight	Относительный вес этой записи	`0`
Port	TCP/UDP номер порта	`5060`
Target	Имя хоста машины, предоставляющей эту услугу	`pbx.shifteight.org`

[a] Обратите внимание на конечную точку.

Когда вы настраиваете запись SRV, вы можете протестировать ее со следующей командой Linux:

```
# dig SRV _sip._udp.shifteight.org
```

Результат будет содержать несколько компонентов, но интересующий вас раздел:

```
        ;; ANSWER SECTION:
_sip._udp.shifteight.org. 14210 IN SRV 0 0 5060 pbx.shifteight.org.
```

Это означает, что ваш DNS-сервер правильно отвечает на запрос SRV для SIP в вашего домена, отвечая именем хоста вашей АТС (в данном случае, `pbx.shifteight.org`).

Любые запросы SIP в ваш домен будут переданы на ваш сервер Asterisk, который будет отвечать за обработку входящих SIP-соединений.[5]

Если ваш диалплан не понимает часть имени/ресурса/конечной точки SIP URI, вызовы не будут выполняться. Это означает, что если вы хотите иметь возможность предлагать ресурсы в системе Asterisk по имени, вам понадобятся соответствующие записи диалплана.

Приём вызовов в вашу систему

Когда SIP URI входит в вашу систему Asterisk, часть ресурса URI будет поступать в диалплан как `${EXTEN}`. Так, например, `leif@shifteight.org` прибудет в диалплан в качестве `leif` внутри переменной канала `${EXTEN}` в любом контексте, который вы используете для обработки

5 Это может быть просто прокси-сервер или любой другой сервер, способный обрабатывать входящие SIP-соединения.

неаутентифицированных SIP-вызовов (если вы строите свой диалплан, используя примеры в этой книге, это будет контекстом диалплана `unauthenticated`).

Изменение sip.conf

Как только вы знакомы с последствиями безопасности для разрешенных SIP-подключений без аутентификации, вам необходимо убедиться, что ваш файл *sip.conf* позволяет им. Хотя Asterisk позволяет их по умолчанию в предыдущих главах этой книги мы поручили отключить неаутентифицированные вызовы SIP. Логика для этого проста: если она вам не нужна, не включайте ее.

Поскольку мы теперь заинтересованы в разрешении звонков из Интернета, нам нужно будет разрешить неавторизованные вызовы SIP. Мы делаем это, установив общую переменную в файле */etc/asterisk/sip.conf*, следующим образом:

```
[general]
context = unauthenticated      ; контекст по умолчанию для входящих вызовов
allowguest = yes               ; включить неаутентифицированные вызовы.
```

После внесения этого изменения не забудьте перезагрузить SIP, используя эту команду из командной строки:

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

или из Asterisk CLI:

```
* CLI> sip reload
```

Вы можете проверить, что изменения применились, используя команду Asterisk CLI *sip show settings*. То, что вы хотите увидеть, это `Allow unknown access: Yes` под разделом `Global Settings` и `Context: unauthenticated` под заголовком `Default Settings`.

Стандартный диалплан

Чтобы обработать входящее имя, в вашем диалплане должно быть расширение, соответствующее этому имени.

Запись диалплана в системе `pbx.shifteight.org` может выглядеть так:

```
[unauthenticated]
exten => leif,1,Goto(PublicExtensions,100,1)

exten => jim,1,Goto(PublicExtensions,101,1)

exten => tilghman,1,Goto(PublicExtensions,102,1)

exten => russell,1,Goto(PublicExtensions,103,1)
```

Это, безусловно, самый простой способ реализации набора имен, но его также сложно поддерживать, особенно в системах с сотнями пользователей.

Чтобы более эффективно реализовать обработку имен, вы можете добавить что-то вроде следующего в файл *extensions.conf*. Обратите внимание, что некоторые строки были обернуты пробелами в этом примере из-за ограничений пространства. Эти строки должны отображаться в одной строке в диалплане. Все строки должны начинаться с `exten =>`, `same =>` или индикатора комментария (`;`).

```
[unauthenticated]
exten => _[A-Za-z0-9].,1,Verbose(2,НЕСАНКЦИОНИРОВАННЫЙ ЗАПРОС К ${EXTEN} ОТ $
{CALLERID(all)})
    same => n,Set(FilteredExtension=${FILTER(A-Za-z0-9,${EXTEN})})
```

```
    same => n,Set(CheckPublicExtensionResult=$
{DIALPLAN_EXISTS(PublicExtensions,
${FilteredExtension},1)})
    same => n,GotoIf($["${CheckPublicExtensionResult}" = "0"]?
CheckEmailLookup)
    same => n,Goto(PublicExtensions,${FilteredExtension},1)

; Это наш обработчик, когда кто-то набирает SIP URI с именем
    same => n(CheckEmailLookup),GoSub(subEmailToExtensionLookup,start,1
(${TOLOWER(${FilteredExtension})}))
    same => n,GotoIf($["${GOSUB_RETVAL}" = "NoResult"]?i,1:PublicExtensions,
${GOSUB_RETVAL},1)
    same => n,Goto(i,1)

; Это обрабатывает недействительные номера/имена
exten => i,1,Verbose(2,Входящий вызов от ${CALEERID(all)} в контексте $
{CONTEXT} не найдено никакого результата)
    same => n,Playback(silence/1&invalid)
    same => n,Hangup()

; Это явные совпадения расширений (полезно для небольших систем)
exten => leif,1,Goto(PublicExtensions,100,1)

exten => jim,1,Goto(PublicExtensions,101,1)

exten => tilghman,1,Goto(PublicExtensions,102,1)

exten => russell,1,Goto(PublicExtensions,103,1)
```

Когда вызов входит в диалплан, он может совпадать в одном из двух мест: он может соответствовать нашему совпадению шаблонов в сверху, или он может соответствовать явным именованным расширениям ближе к нижней части нашего примера (например, `leif`, `jim`, `tilghman` или `russell`).

Если вызов явно не совпадает с нашими именованными расширениями, будет использоваться совпадение шаблонов. Наш шаблон совпадающий с `_[A-Za-z0-9].` соответствует любой строке начинающейся с буквенно-цифрового символа, за которым следует ещё один или несколько других символов.

Входящую строку нужно сделать безопасной, поэтому мы используемфункцию `FILTER()` для удаления иных символов и присваиваем результат переменной канала `FilteredExtension`.

Функция `DIALPLAN_EXISTS()` будет использоваться, чтобы увидеть, соответствует ли запрос чему-либо в контексте `PublicExtensions`. Эта функция вернет либо `0` (если совпадение не найдено), либо `1` (если найдено совпадение) и присвоит результат переменной канала `CheckPublicExtensionResult`.

Следующая строка - `GotoIf()`, которая проверяет состояние переменной `CheckPublicExtensionResult`. Если результат был равен `0`, диалплан будет продолжаться на метке приоритета `CheckEmailLookup`. Если результат был отличным от `0` (в этом случае результат может быть `1`), будет выполнена следующая строка диалплана. Эта строка выполнит `Goto()` и продолжит выполнение в контексте `PublicExtensions` (предположительно, чтобы набрать конечную точку назначения).

Предполагая, что наша переменная `CheckPublicExtensionResult` была `0`, наш диалплан будет продолжаться на метке приоритета `CheckEmailLookup`, где мы используем подпрограмму `subEmailToExtensionLookup` через `GoSub()`.[6] Мы передаем значение, содержащееся в переменной канала `FilteredExtension` в подпрограмму, но вы заметите, что мы завернули в функцию

6 Мы объясняем использование `subEmailToExtensionLookup` в следующем разделе.

диалплана TOLOWER(), которая ожидает, что ваши адреса электронной почты будут сохранены в нижнем регистре, а не в смешанном).

По возвращении из подпрограммы subEmailToExtensionLookup мы проверяем переменную канала GOSUB_RETVAL (которая автоматически устанавливается при возврате из подпрограммы). Результат будет одним из двух: добавочный номер, который соответствует имени, переданному подпрограмме, или строке NoResult. Наш диалплан проверяет ${GOSUB_RETVAL}, и если она содержит NoResult, вызывающий передается в расширение i (недействителен), где мы информируем вызывающего абонента о том, что набранный номер недействителен. Если все будет хорошо, вызов продолжит выполнение в контексте PublicExtensions.

Разбор файлов

Этот небольшой трюк позволит вам использовать файл *voicemail.conf* для поиска действительных имен пользователей напротив их адреса электронной почты. Это может привести к тому, что будет дублирование, и для этого требуется, чтобы поле электронной почты в *voicemail.conf* было заполнено и содержало имя пользователя (перед символом @), которое вы будете поддерживать в своем диалплане, но его легко закодировать в диалплане, и если ничто другое не даст вам иных идей о том, как вы могли бы предоставить более автоматизированный способ связывания имен с внутренними номерами для набора SIP URI. Обратите внимание, что этот метод не позволит вам исключить некоторых людей из набора номера. Это все или ничего.

Мы написали это как подпрограмму, которая вызывается чтото вроде этого:

```
; где ‘name’ является именем пользователя, содержащееся в адресе email
GoSub(subEmailToExtensionLookup,start,1(name))
```

Подпрограмма выглядит следующим образом:

```
[subEmailToExtensionLookup]
exten => start,1,Verbose(2,проверка для пользователя в voicemail.conf)
    same => n,Set(LOCAL(FilteredExtension)=${FILTER(a-z0-9,${ARG1})})
    same => n,Set(LOCAL(Result)=${SHELL(grep"${LOCAL(FilteredExtension)}@"
/etc/asterisk/voicemail.conf)})
    same => n,GotoIf($[${ISNULL(${LOCAL(Result)})}]?No_Result,1)
    same => n,Set(LOCAL(ExtensionToDial)=${CUT(${LOCAL(Result)},=,1)})
    same => n,Set(LOCAL(ExtensionToDial)=${FILTER(0-9,${LOCAL
(ExtensionToDial)})})
same => n,Return(${LOCAL(ExtensionToDial)})

exten => no_Result,1,Verbose(2,Пользователь ${ARG1} не найден в
voicemail.conf)
    same => n,Return(NoResult)
```



Хотя это может быть полезно для обозначения ${LOCAL(Result)} как локальной переменной, как только имя этой переменной было установлено как LOCAL() с помощью приложения Set() больше не нужно обращаться к нему через LOCAL(). Вы можете дополнительно задать имя переменной через Set() во второй раз (без префикса LOCAL()), и оно все еще ссылается на конструкцию LOCAL(), и она все равно исчезнет, когда возвращается из подпрограммы.

Давайте рассмотрим этот код, потому что есть некоторые полезные действия, которые вы можете применить к другим целям.

Сначала создается канальная переменная с именем FilteredExtension. Эта переменная локальна для подпрограммы:

```
Set(LOCAL(FilteredExtension)=${FILTER(a-z0-9,${ARG1})})
```

Функция FILTER() просматривает весь ${ARG1} и удаляет любые нецифробуквенные символы. Это прежде всего по соображениям безопасности. Мы передаем эту строку в оболочку, поэтому важно, чтобы она содержала только символы, которые мы ожидаем.

Следующий шаг - это место, где проходит холодок:

```
Set(LOCAL(Result)=${SHELL(grep "${LOCAL(FilteredExtension)}@"
/etc/asterisk/voicemail.conf)})
```

Оболочка вызывается для запуска приложения shell grep, которая будет искать в файле *voicemail.conf*, вернет все строки, содержащие имя@ и присвоит результат переменной ${Result}:

```
GotoIf($[${ISNULL(${LOCAL(Result)})}]?no_result,1)
```

Если строки не содержат искомую строку, мы вернем из подпрограммы значение NoResult (которое будет найдено в переменной канала ${GOSUB_RETVAL}). Раздел диалплана, который называется подпрограммой, должен будет обрабатывать это условие.



В качестве альтернативы вместо использования no_result,1 для вашего местоположения вы можете использовать same с произвольно высоким приоритетом, а также меткой приоритета, чтобы сделать эти вещи немного чище. Если вы это сделаете, то пункт назначения будет просто no_result (вместо no_result,1), и ваш пункт назначения будет выглядеть так:

```
same => 10000(no_result),Verbose(2,No user ${ARG1} found in
voicemail.conf)
same => n,Return(NoResult)
```

Мы создали расширение с именем no_result для этой цели:

```
exten => no_result,1,Verbose(2,Пользователь ${ARG1} не найден в voicemail.conf)
    same => n,Return(NoResult)
```

Если ${Result} не является нулевым, следующие шаги очистят ${Result}, чтобы извлечь добавочный номер[7] пользователя с именем, переданным в ${ARG1}:

```
Set(LOCAL(ExtensionToDial)=${CUT(${LOCAL(Result)},=,1)})
```

Функция CUT() будет использовать символ = в качестве разделителя полей и присвоит значение из первого поля, найденного в ${Result}, новой переменной ExtensionToDial. Оттуда нам просто нужно обрезать любые конечные пробелы, фильтруя все нечетные символы:

```
Set(LOCAL(ExtensionToDial)=${FILTER(0-9,${LOCAL(ExtensionToDial)})})
```

Теперь мы можем вернуть имя добавочного номера, которое мы получили:

```
Return(${LOCAL(ExtensionToDial)})
```

Этот пример был тем, что мы взломали для иллюстрации некоторые методы, которые вы можете использовать для легкости сопоставления имен с внутренними номерами для набора SIP URI. Это отнюдь не лучший способ сделать это, но его довольно просто реализовать, и во многих случаях это может быть тем, что вам нужно.

Поиск в базе данных

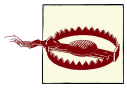
Использование базы данных на сегодняшний день является лучшим способом обработки информации о пользователях в более крупных и сложных системах. Мы обсудим интеграцию Asterisk с базами данных более подробно в Главе 16, но здесь полезно ввести понятие.

7 На самом деле то, что мы извлекаем, это номер ящика голосовой почты; однако это число, как правило, будет таким же, как внутренний внутренний добавочный номер пользователя. Если это не одно и то же, этот конкретный метод не будет выполнять поиск по имени расширения, и нужно будет найти другой способ.

База данных идеально подходит для обработки поиска имен, поскольку это упрощает обслуживание пользовательских данных (и интеграцию с внешними системами, такими как веб-интерфейсы). Однако для проектирования и реализации требуется немного больше усилий.

Пример, который мы будем использовать в этой главе, будет работать, но для производственной среды он, вероятно, слишком упрощен. Наша цель - просто дать вам достаточно информации, чтобы понять концепцию; более тесная интеграция является частью того, что описано в Главе 16.

Во-первых, нам понадобится таблица для обработки нашего сопоставления между именами. Это может быть отдельная таблица из главной пользовательской таблицы или она может быть обработана в главной пользовательской таблице, при условии, что она содержит поле, которое будет содержать точные строки, которые пользователи будут публиковать в качестве своих SIP URI (например, некоторые компании имеют правила относительно того, как выглядят адреса электронной почты, поэтому у Leif может быть URI, например, `lmadsen@shifteight.org` или `leif.madsen@shifteight.org`).



Если вы серьезно относитесь к реализации этого примера в производственной системе, убедитесь, что вы знакомы с материалом Главы 16, так как в ней рассматриваются некоторые ключевые понятия, которые мы здесь опускаем.

Наша таблица `NameMapping` выглядит так, как показано в Таблице 12-2.

Таблица 12-2. NameMapping

Имя	Номер	Контекст
leif	100	publicExtensions
leif.madsen	100	publicExtensions
lmadsen	100	publicExtensions
jim	101	publicExtensions
reception	0	Services[a]
voicemail	*98	Services

[a] Убедитесь что этот контекст существует в вашей системе.

Мы считаем, что использование отдельной таблицы, которая обрабатывает только сопоставление между именами и расширениями/контекстами, является наиболее полезным решением, поскольку эта таблица может использоваться для обработки не только пользователей с помощью телефонных аппаратов. Вам предлагается придумать другие способы решения этой проблемы, которые могут быть более подходящими для вашей среды.

В диалплане мы будем ссылаться на эту таблицу с помощью функции Asterisk `func_odbc`:

```
[subLookupNameInNameMappingTable]
exten => start,1,Verbose(2,Поиск ${ARG1})
; где ‘name’ - это имя пользователя, найденное в адресе электронной почты:
    same => n,Set(ARRAY(CalleeExtension,CalleeContext)=${ODBC_NAME_LOOKUP
(${ARG1})})
    same => n,GotoIf($[${ISNULL(${CalleeExtension})}]?No_result,1)
    same => n,GotoIf($[${ISNULL(${CalleeContext})}?No_result,1)
    same => n,Return()  ; Вам нужно будет обрабатывать новые
; переменные CalleeExtension и CalleeContext в
; коде, который вызвал эту подпрограмму
exten => no_result,1,Verbose(2,Имя не было найдено в базе данных.)
    same => n,Return(NoResult)
```

Файл */etc/asterisk/func_odbc.conf* потребует следующую запись:

```
[NAME_LOOKUP](DB)
readsql=SELECT Extension,Context FROM NameMapping WHERE Name='${SQL_ESC(${
ARG1})}'
```



Имейте в виду, что нет причин не ссылаться на несколько хранилищ данных для поиска имен. Например, у вас может быть таблица, такая как описанная здесь, но также будет вторичный поиск, как скажем, база данных LDAP, чтобы попытаться искать имена там. Это может усложниться при настройке и обслуживании, но, если настроено правильно, это также может означать, что ваша система Asterisk может быть тесно интегрирована с другими системами вашего предприятия.

Подробности о том, как обрабатывать все это в вашем диалплане, выходят за рамки этой книги. Достаточно сказать, что в вашем диалплане вам все равно придется обрабатывать значения, которые создает или назначает ваша подпрограмма.

Набор SIP URI из Asterisk

Asterisk может набирать SIP URI так же легко, как и любой другой пункт назначения, но конечная точка (а именно, ваш телефон) в конечном итоге будет нести бремя составления адреса, и в этом заключается трудность.

Большинство SIP-телефонов позволят вам составить SIP URI с помощью клавиатуры набора. Сначала это звучит как отличная идея, но поскольку на телефоне нет клавиш пишущей машинки, чтобы набрать что-то вроде `jim.vanmeggelen@shifteight.org`, то, что вам нужно было бы фактически ввести в телефон, было бы чем-то вроде строки:

```
5-444-6-*-888-2-66(пауза)-6-33-4(пауза)-4-33-555-33-66-#-7777-44(пауза)
-444-333-8-33-
444-(пауза)-4(пауза)-44-8-*-666-777-4
```

Чтобы поддержать это в своем диалплане, вам понадобится нечто подобное:[8]

```
exten => _[0-9a-zA-Z].,1,Verbose()
    same => n,Set(FilteredExtension=${FILTER(0-9a-zA-Z@-_.,${EXTEN})})
    same => n,Dial(SIP/${FilteredExtension})
```

Это просто, весело, и все работает! ...?

Реальность такова, что до тех пор, пока все телефоны не будут поддерживать сложные и гибкие адресные книги, а также клавиатуру в стиле QWERTY (возможно, через сенсорный экран), набор SIP URI не будет взлетать.

Если у вас есть SIP URI который вы хотите набирать на регулярной основе (например, во время написания этой книги было много звонков между Джимом и Лейфом), вы можете добавить что-то вроде этого к вашему диалплану:

```
exten => 5343,1,Dial(SIP/leif.madsen@shifteight.org).
```

В этом диалплане вы можете набрать `5343` (*LEIF*) на своем телефоне, и диалплан Asterisk переведёт Вас на соответствующий SIP URI. Это не практично для большого количества URI, но для некоторых это может быть полезным ярлыком.

Но продолжайте читать, потому что есть очень полезные компоненты DNS, которые упрощают процесс набора непосредственно между системами без использования ТфОП.

8 Технически, символы `! # $% & '* + / =? ^ `{ | } ~` также действительны как `local-part` часть адреса электронной почты; однако они необычны, и мы решили не допускать их в нашем примере диалплана.

ENUM и E.164

Хотя протокол SIP действительно не мыслит с точки зрения телефонных номеров, реальность такова, что телефонные номера не уйдут в ближайшее время, и если вы хотите правильно интегрировать систему VoIP с таким количеством телефонных сетей, вам понадобится обработать ТфОП в некотором роде.

ENUM сопоставляет телефонные номера в Системе доменных имен (DNS). Теоретически ENUM - отличная идея. Почему бы не отключить ТфОП полностью и просто направлять телефонные звонки непосредственно между конечными точками, используя тот же план нумерации? Мы не уверены, что эта идея когда-либо станет тем, что, возможно, станет новым сообществом телекоммуникаций. Причина? Никто не может сказать, кому принадлежат номера телефонов.

E.164 и ITU (МСЭ)

ITU - Международный союз электросвязи (МСЭ) является учреждением Организации Объединенных Наций, которое на самом деле старше, чем сама ООН. Он был основан в 1865 году как Международный телеграфный союз. МСЭ-Т сектор, известный на протяжении многих десятилетий как CCITT (международный консультационный комитет по телефонии и телеграфии), является органом стандартизации отвечает за все протоколы используемые ТфОП, а также многие, которые используются в VoIP. До появления VoIP, работа сектора ITU-T представляла небольшой интерес для среднего человека, и членство, как правило, ограничивалось отраслями и учреждениями, которые заинтересованы в телекоммуникационных стандартах.

Стандарты МСЭ, как правило, соответствуют формату буквенно-числовых номеров. Стандарты МСЭ-Т, о которых вы, возможно, слышали, включают H.323, H.264, G.711, G.729 и т.д.

E.164 является стандартом МСЭ-Т, который определяет международный план нумерации для ТфОП.

Если вы когда-либо пользовались телефоном, вы использовали адресацию E.164.

Каждой стране в мире присваивается код страны[9], а контроль за обращением в этих странах осуществляется местными властями.

Номера E.164 ограничены 15 цифрами (исключая префикс).

В Asterisk нет ничего особенного в том, что нужно обрабатывать адресацию E.164, кроме того, чтобы ваш диалплан соответствовал потребностям любых ТфОП-совместимых каналов, которые у вас могут быть.

Например, если вы работаете в стране NANP, Вам, вероятно, придется иметь следующие шаблоны:

```
_NXXNXXXXXX
_1NXXNXXXXXX
_011X.
_N11
```

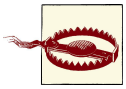
В Великобритании вам может понадобиться нечто большее:

```
_0[123789]XXXXXXXXX
_0[123789]XXXXXXXX
```

И в Австралии ваш диалплан может иметь следующие шаблоны:

```
_NXXXXXXX
_0XXXXXXXXX
```

9 За исключением 24 стран и территорий в страновом коде 1, которые являются частью Североамериканского центра нумерации (NANPA).



Пожалуйста, не копируйте и не вставляйте эти совпадения шаблонов в свой диалплан. Особенности региональных диалпланов сложны и постоянно меняются. Одним из важных вопросов, которые необходимо тщательно рассмотреть, является номер для конкретного региона экстренного вызова, как описано в разделе «Экстренный набор» в Главе 7. Вы не хотите получать этот материал неправильно.

Североамериканский центр нумерации

На большей части Северной Америки используется Нормандский план нумерации (NANP). Всем странам в NANP присваивается код страны 1. Канада и США являются наиболее известными из этих стран, но NANP фактически включает около 24 различных стран и территорий (в основном в Карибском бассейне).

ENUM

Чтобы разрешить отображение номеров E.164 в пространство имен DNS, должен был быть разработан способ представления телефонных номеров в качестве имен DNS.

Эта концепция определена в RFC3761, с благосклонным названием «Приложение E.164 для унифицированных идентификаторов ресурсов (URI) для системы динамической децентрализации (DDDS) приложения (ENUM)». ENUM, как сообщается, означает Electronic NUmber Mapping (электронное отображение номера).

Согласно RFC, преобразование телефонного номера в ENUM-совместимый адрес требует следующего алгоритма:

```
1. Удалить все символы, за исключением цифр.
Например, первое хорошо известное правило создало ключ «+442079460148». Этот
шаг просто удалит «+», оставив «442079460148».
2. Поместите точки («.») Между каждой цифрой. Пример:
4.4.2.0.7.9.4.6.0.1.4.8
3. Обратный порядок цифр. Пример:
8.4.1.0.6.4.9.7.0.2.4.4
4. Дополнить строку ".e164.arpa" до конца. Пример:
8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa
```

Ясно как день?

ENUM не взлетел. Причины, по-видимому, носят преимущественно политический характер. Проблема связана с тем, что нет ни одной организации, которая контролирует нумерацию на ТфОП так, как это делает IANA для Интернета. Поскольку ни одна организация не имеет четкого мандата по управлению номерами E.164 во всем мире, задача сохранения точной и авторитетной базы данных для ENUM оказалась неуловимой.

Некоторые страны Европы хорошо зарекомендовали себя в предоставлении надежных баз данных ENUM, но в коде страны 1 (NANP), который содержит несколько стран и, следовательно, несколько регулирующих органов, ситуация стала нелогичной. Это неудивительно, так как провайдеры, которые контролируют адресацию E.164, не могут с полным основанием ожидать энтузиазма по поводу того, что вы можете обойти их сети. Организации, ответственные за внедрение ENUM в Северной Америке, как правило, работают над созданием сети ТфОП в Интернете, что может сэкономить им деньги, но не вам или мне.

Это совсем не то, что нужно. Зачем мне направлять VoIP-звонки из моей системы в сеть, которая хочет взимать плату за эту привилегию? SIP предназначен для маршрутизации вызовов между конечными точками и не имеет реального использования для концепции оператора.

Преимущество всего этого заключается в том, что при выполнении ENUM-поиска возвращается действительный SIP-URI.

Asterisk и ENUM

Asterisk может выполнять поиск по базам данных ENUM, используя либо `ENUMLOOKUP()` или комбинацию функций диалплана `ENUMQUERY()` и `ENUMRESULT()`. `ENUMLOOKUP()` возвращает только одно значение из поиска и полезно, когда вы знаете, что, вероятно, будет только одно возвращаемое значение (например, SIP URI, который вы хотите, чтобы набирала система), или если вы просто хотите получить количество доступных записей.

Статус ENUM во всем мире

В странах NANP (и многих других), официальная зона *e164.arpa* формально не была внедрена, и, следовательно нет никакого официального места для поиска ENUM для NANP.

Список статусов реализаций ENUM различных стран можно найти по адресу *http://enumdata.org/*. Для тех стран, которым посчастливилось иметь ENUM в производстве вы можете выполнять поиск ENUM непосредственно в своих *e164.arpa* зонах.

Для стран без *e164.arpa* зон существует несколько альтернативных мест для поиска, наиболее популярным в настоящее время является *http://www.e164.org*. Обратите внимание, что у этих организаций нет официального мандата на сохранение зон, которые они представляют. Это проекты на основе сообщества, проекты с наилучшими проектами, а данные, содержащиеся в них, часто будут устаревшими.

Поиск ENUM в диалплане может выглядеть так:

```
exten => _X.,1,Set(CurrentExten=${FILTER(0-9,${EXTEN})})
    same => n,Set(LookupResult=${ENUMLOOKUP(${CurrentExten},sip,,,e164.arpa)})
    same => n,GotoIf($[${EXISTS(${LookupResult})}]?HaveLocation,1)
    same => n,Set(LookupResult=${ENUMLOOKUP(${CurrentExten},sip,,,e164.org)})
    same => n,GotoIf($[${ISNULL(${LookupResult})}]?
NormalCall,1:HaveLocation,1)

exten = > HaveLocation,1,Verbose(2,Возвращен ручной набор по SIP URI)
    exten => ...

exten => NormalCall,1,Verbose(2,Ручной набор по стандартному маршруту ТфОП)
    exten => ...
```

Код диалплана, на который мы только что посмотрели, будет набирать номер и передавать его функции `ENUMLOOKUP()`. Он запрашивает тип метода `sip` (мы хотим, чтобы SIP URI был возвращен), и поиск сначала выполнялся из списков в DNS, найденных в зоне *e164.arpa*, а затем из записей, найденных по адресу *http://www.e164.org*.

За пределами стран, которые его внедрили, слабое понимание об ENUM. Таким образом, многие запросы ENUM не вернут никаких результатов. Ожидается, что это не изменится в ближайшем будущем, и ENUM останется диковинкой до тех пор, пока оно не будет широко распространено.

ISN, ITAD и freenum.org

Наконец, мы перейдем к самой крутой части этой главы.

Самым большим недостатком ENUM является то, что он использует систему нумерации, которая не контролируется никакими интернет-провайдерами.[10] Проект freenum.org решает эту проблему, используя схему нумерации, управляемую IANA. Это означает, что формальная глобально действительная, негеографическая система нумерации для VoIP может быть немедленно и легко реализована без потасовки в бюрократии и политике, которые обременяют систему нумерации E.164.

Цель проекта freenum.org

Джон Тодд, управляющий проектом отмечает что:

> Freenum.org - это служба DNS, которая использует методы сопоставления типа ENUM, позволяющие преобразовать многие службы в строку, удобную для клавиатуры. Самый очевидный и широко используемый метод для этого - это подключение пользователей VoIP бесплатно, создавая легко запоминаемую строку набора, которая сопоставляется с SIP URI в фоновом режиме. Однако все, что может появиться в записи NAPTR (электронная почта, мессенджер, веб-адреса), можно сопоставить с адресом *freenum.org* в стиле ISN. Цель проекта - предоставить бесплатные числовые указатели на миллиарды телефонов, которые поддерживают только символы 0-9, * и # и позволяют этим устройствам общаться через VoIP или другие протоколы следующего поколения. Проект распространяется на более чем 30 DNS-серверах по всему миру.

Получили ISN?

Сердцем концепции *freenum.org* является ITAD Абонентский номер (ISN). ISN представляет собой числовую строку, состоящую из добавочного номера в вашей системе, разделителя символов звездочки (*)[11] и номера, уникального для вашей организации, называемого номером административного домена IP-телефонии (ITAD). Преимущество ISN заключается в том, что его можно набирать с любого телефона. ISN будет выглядеть примерно так:

```
0*1273
```

который будет представлять собой *внутренний номер ноль в ITAD 1273*[12] и будет разрешен для `sip:0@shifteight.org`.

Вы управляете своими добавочными номерами (все что слева от *). Ваш ITAD назначается IANA (той же организацией, которая управляет IP-адресами и MAC-адресами).[13]

После того, как ваш ITAD назначен, вы сможете публиковать ISN на своем веб-сайте или на визитных карточках или где вы обычно публиковали телефонные номера.



В качестве консультантов мы недавно начали присваивать номера ITAD всем организациям, для которых мы выполняем развертывание. Многие компании не поймут для чего они могли бы использовать номера ISN, но это становится приятной особенностью, когда их списки контактов содержат свой собственный номер ISN для службы или они начинают взаимодействовать с другими организациями, имеющими ITAD. Лейф даже поставил его в офис жены, чтобы он мог бесплатно позвонить ей на обед.

10 Более того, возможно, что номера E.164 контролируются слишком многими организациями, каждая из которых подчиняется различным правилам и имеет цели, которые не всегда совместимы с концепцией глобального бесплатного VoIP-вызова.

11 Этот символ не имеет ничего общего с программным обеспечением, которое является предметом этой книги; он просто ссылается на *, который находится на клавиатуре набора каждого телефона. Интересно, что могло бы быть, если бы вместо Asterisk Марк Спенсер решил назвать его создание Octothorpe.

12 ITAD 1273 присвоен *shifteight.org*.

13 Если вы думаете о строке 0*1273 с точки зрения форматирования адреса электронной почты, то 0 - ваше имя, символ звездочки как символ символа адресов электронной почты @, а номер ITAD - как числовое представление имени домена вашей компании.

Любая система, способная набирать ISN, позволит своим пользователям звонить вам, набрав ваш ISN. Вызовы будут направляться непосредственно между двумя системами, используя URI SIP, который возвращает freenum.org.

Абонентские номера ITAD(ISNs)

ISN не заменяет SIP URI, а дополняет его, разрешая набор номеров VoIP, используя только символы, найденные на стандартной телефонной клавиатуре. Чтобы перевести ISN в действительный URI, система DNS будет запрашивать ISN по домену в *freenum.org*. Любой поиск DNS по вашему ISN вернет URI, который определяет как ваша система ожидает получать звонки на этот ISN.[14]

Управление интернет-нумерацией

Орган, уполномоченный на присвоение номеров в Интернете (IANA), является органом, ответственным за управление любой системой нумерации, которая существует в результате RFC, для которой требуется численная база данных. Наиболее известная ответственность IANA - это делегирование IP-адресов пяти региональным интернет-регистраторам, которые контролируют все публичные IP-адреса на планете.[15] Эти организации несут ответственность за присвоение IP-адресов в своих регионах.

Существует много других схем нумерации, которые были созданы в результате RFC. Другие управляемые IANA номера включают в себя MAC-адреса, конкретно, организационно-уникальный идентификатор (OUI) в пространстве адресации MAC.

Несколько лет назад был создан протокол TRIP (Telephony Routing over IP). Хотя этот протокол никогда не взлетит и вряд ли увидит какой-либо рост в будущем, он предложил нам одну невероятно полезную вещь: ITAD. Поскольку ITAD являются частью RFC, IANA уполномочено поддерживать базу данных ITAD. Именно это делает freenum.org возможным.

Административные домены IP-телефонии (ITAD)

Freenum.org использует ответственность IANA за ведение базы данных ITAD и позволяет нам создавать простые, основанные на стандартах, глобально релевантные и ориентированные на сообщества планы нумерации для VoIP.[16] Вы можете найти список присвоенных в настоящее время номеров ITAD на веб-сайте IANA.

Вы можете получить свой собственный номер ITAD, отправив форму, расположенную по адресу *http://www.iana.org/cgi-bin/assignments.pl*.

Эта форма должна быть заполнена, как показано на Рисунке 12-1.

14 Хотя *freenum.org* может обрабатывать ITAD, которые разрешают не-SIP URI, обработка нескольких протоколов выходит за рамки этой книги. На данный момент мы рекомендуем вам ограничить ваш ISN обработкой SIP URI.

15 AfriNIC, APNIC, ARIN, LACNIC и RIPE NCC.

16 Обратите внимание, что *freenum.org* консультировался с представителями IANA в отношении использования ITAD с протоколами, отличными от TRIP.

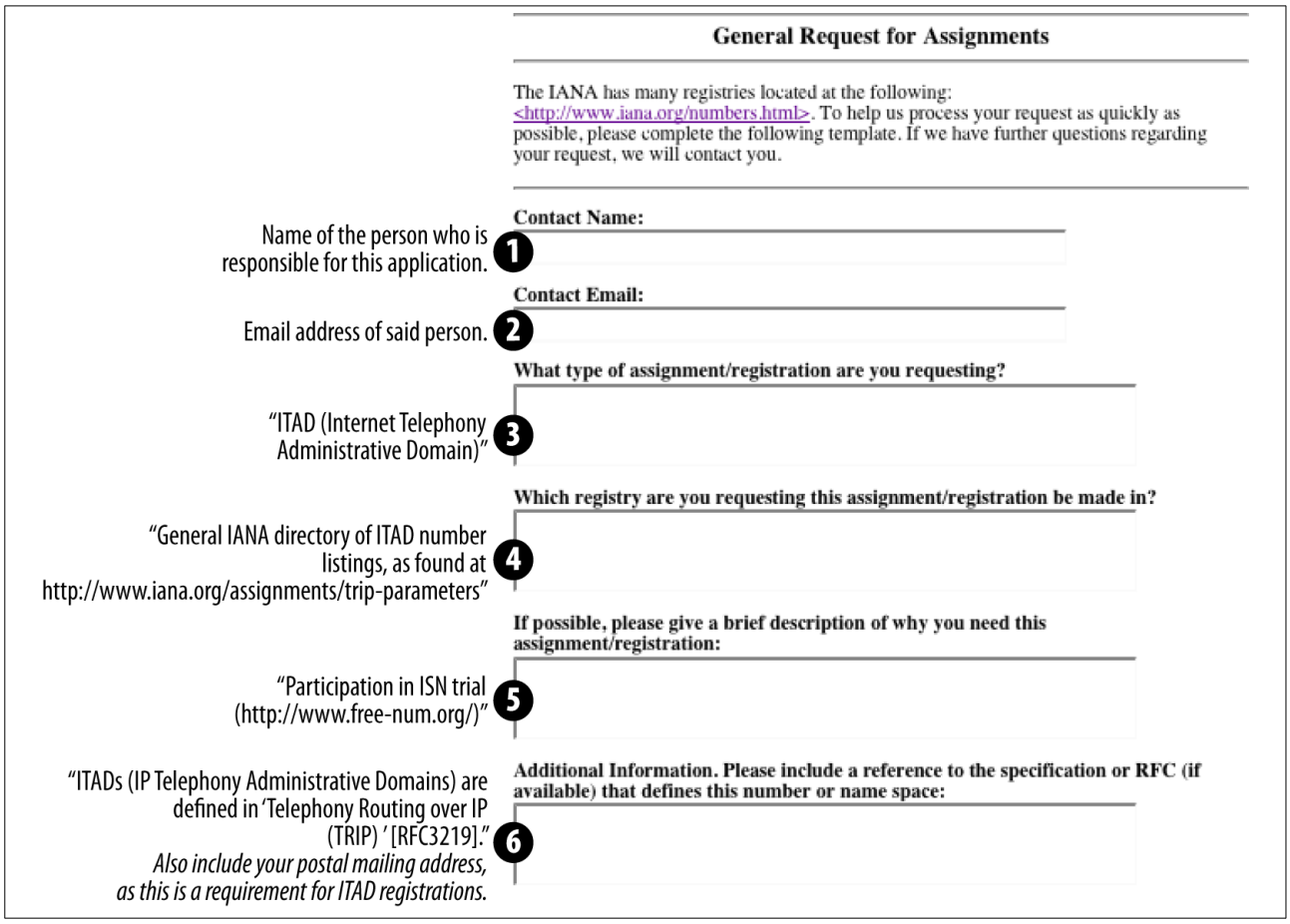


Рисунок 12-1. Форма запроса для присвоения

В форме требуется следующая информация:

1. Имя лица, ответственного за это заявление.
2. Адрес электронной почты указанного лица.
3. ITAD (Административный домен IP-телефонии).
4. Общий каталог IANA для списков номеров ITAD, см. *http://www.iana.org/ assignments/trip-parameters*.
5. Участие в исследовании ISN (*http://www.freenum.org*).
6. ITAD (Административный домен IP-телефонии) определены в разделе «Маршрутизация телефонии через IP (TRIP)» [RFC 3219].

   Кроме того, обязательно укажите свой почтовый почтовый адрес, так как это требуется при регистрации ITAD.

Ваша заявка будет рассмотрена Real Human Being ™, и в течении нескольких дней вам будет назначен ITAD IANA. Через несколько дней вы также получите информацию для своей учетной записи freenum.org (в настоящее время существует простой процесс проверки, чтобы боты и спамеры не злоупотребляли системой). Затем вам нужно зайти на сайт freenum.org и определить параметры для вашего ITAD.

Создание записи DNS для своего ITAD

В правом верхнем углу сайта freenum.org вы увидите ссылку «Sign in Here». Ваше имя пользователя - это адрес электронной почты, который вы зарегистрировали в IANA, и ваш пароль будет отправлен вам по электронной почте системой *freenum.org*.[17]

17 Это может занять несколько дней, поэтому, если вы получили свой ITAD от IANA, но не получили пароль от *freenum.org*, подождите некоторое время.

Вам будет представлен список ваших назначенных ITAD. Чтобы ваш новый ITAD работал, вам необходимо обеспечить, чтобы записи DNS были обновлены.



Существует два метода обработки DNS для вашего ITAD. Первым (и самым простым) является запись указателя имени (NAPTR), вставленная в зону *freenum.org*. Другой способ - создать зону для вашего ITAD и передать эту зону *freenum.org* на ваши серверы имен. Мы обсудим здесь первый метод, но если вы знакомы с администрированием NAPTR/ENUM для DNS-сервера, вы можете использовать второй метод.

Люди freenum.org создали автоматический инструмент самообслуживания Freenum (FASST), чтобы упростить запись DNS для вас. Основные поля уже заполнены. Единственное, что вам нужно изменить, находится в разделе формы «Настройка DNS»: укажите имя хоста вашей ATC и сохраните изменения. Средство FASST использует регулярное выражение для преобразования поиска ISN в SIP URI.

Для того, чтобы указать имя вашего компьютера, вам необходимо изменить примеры регулярных выражений, представленных FASST, изменяя образец имени хоста `sip.yourdomain.com` на имя хоста вашей ATC. Так, например, в нашем случае мы хотели бы изменить

```
:!^\\+*([^\\*]*)!sip:\\1@sip.yourdomain.com!
```

на:

```
:!^\\+*([^\\*]*)!sip:\\1@pbx.shifteight.org!
```

Другие поля в записи DNS не должны быть изменены, если вы не знаете, что делаете. Остальные поля в форме являются необязательными и могут быть заполнены по вашему усмотрению.

Использование делегированных зон для географически распределенных офисов

Джон Тодд отмечает:

> для тех сайтов, которые имеют чрезвычайно сложные конфигурации или географически распределенные офисы с различными серверами SIP, обрабатывающие разные префиксы (например: 12xxx отправляется на сервер Asterisk во Франции, 13xxx на сервер Asterisk в Германии и т. д.), тогда есть более сложные методы, когда вы запускаете свою собственную делегированную зону из домаена freenum.org, но это выходит за рамки этой книги, но ее можно узнать на сайте *freenum.org*.

Тестирование ITAD

Как часто бывает с изменениями DNS, это может занять несколько дней для ваших изменений в системе. Чтобы проверить, вы можете использовать Google для «онлайн-инструмента поиска», чтобы найти инструмент поиска в Интернете, или использовать инструмент *dig* под Linux:

```
$ dig NAPTR 4.3.2.1.1273.freenum.org
```

Как только ваша запись будет обновлена в системе, результат будет содержать следующее:

```
;; ANSWER SECTION:
4.3.2.1.1273.freenum.org. 86400 IN NAPTR 100 10 "u" "E2U+sip"
"!^\\+*([^\\*]*)!sip:\\1@shifteight.org!" .
```

Если секция ответа не содержит регулярное выражение, содержащее имя домена, записи не обновлены и вы должны подождать ещё несколько часов (или даже оставить его на один день).

Использование ISN в вашей системе Asterisk

Итак, теперь, когда у вас есть собственный ITAD (вы зарегистрировались, верно?), Вы захотите сделать его доступным для других, а также настроить свой диалплан, чтобы вы могли набирать другие ITAD.

В разделе `[globals]` вашего диалплана (*/etc/asterisk/extensions.conf*) добавьте глобальную переменную, содержащую ваш ITAD:

```
[globals]
ITAD = 1273 ; замените «1273» на свой собственный номер ITAD.
```

Чтобы разрешить вызов в ITAD из вашей системы, вам потребуется что-то вроде следующего кода диалплана:[18]

```
[OutgoingISN]
exten => _X*X!,1,GoSub(subFreenum,start,1(${EXTEN}))
exten => _XX*X!,1,GoSub(subFreenum,start,1(${EXTEN}))
exten => _XXX*X!,1,GoSub(subFreenum,start,1(${EXTEN}))
exten => _XXXX*X!,1,GoSub(subFreenum,start,1(${EXTEN}))
exten => _XXXXX*X!,1,GoSub(subFreenum,start,1(${EXTEN}))
; вам может понадобиться добавить еще несколько строк для обработки ...

[subFreenum]
exten => start,1,Verbose(2,Выполнение поиска ISN)
    same => n,Set(ISN=${FILTER(0-9*,${ARG1})})
    same => n,Set(Result=${ENUMLOOKUP(${ISN},sip,s,,freenum.org)})
    same => n,GotoIf($[${EXISTS(${Result})}]?call,1:no_result,1)

exten => call,1,Verbose(2,Размещение вызова ISN - ${ISN} - через ${Result})
    same => n,Dial(SIP/${Result})
    same => n,Return()

exten => no_result,1,Verbose(2,Поиск ISN: - ${ISN} - не вернул результат)
    same => n,Playback(silence/1&invalid)
    same => n,Return()
```

Мы добавили два новых контекста к нашему диалплану: `OutgoingISN` и `subFreenum`. Контекст `OutgoingISN` контролирует кто может набирать номера ISN из вашего диалплана. Если вы следовали нашим примерам в этой книге, у Вас должен быть контекст с именем `LocalSets`, который является контекстом, с которого все ваши телефоны входят в диалплан. Включение `OutgoingISN` в `LocalSets` позволяет набирать номера ISN:

```
[LocalSets]
include =>  OutgoingISN ; включить контекст, который позволяет набор
                        ; номера ISN
include => external           ; использовать подпрограмму чтобы определить
                              ; что вы можете набрать
```



Мы поместили `OutgoingISN` выше `external`, потому что Asterisk будет выполнять согласование расширения в порядке следования включений (include), а поскольку `external` имеет более общий поиск шаблону, чем наш контекст `OutgoingISN`, нам нужно убедиться, что `OutgoingISN` отображается первым.

Магия для набора номеров ISN обрабатывается в контексте `subFreenum`. Наш контекст `OutgoingISN` будет передавать запрошенный номер (например, `1234*256`) в подмаршрут

18 Если люди публикуют полные DID пользователей, а не их внутренние внутренние номера, совпадения шаблонов должны поддерживать до 15 цифр.

subFreenum. После первого вызова Verbose() в первой строке подмаршрут будет отфильтровывать запрос на номера и символ звездочки (*), чтобы сделать расширение безопасным. Затем результат будет присвоен переменной канала ISN:

```
exten => start,n,Set(ISN=${FILTER(0-9*,${ARG1})})
```

Затем подпрограмма выполнит поиск ISN через DNS-систему с использованием функции диалплана ENUMLOOKUP(). Параметры, переданные функции ENUMLOOKUP(), включают в себя:

- Номер ISN для поиска.
- Тип метода для поиска и возврата (SIP).
- Параметр s , который сообщает Asterisk о выполнении поиска в стиле ISN вместо стандартного поиска ENUM.
- Суффикс зоны для выполнения поиска (мы будем использовать freenum.org, но по умолчанию это e164.arpa).

Наш код для выполнения поиска выглядит следующим образом:

```
exten => start,n,Set(Result=${ENUMLOOKUP(${ISN},sip,s,,freenum.org)})
```

После поиска и сохранения результата в переменной канала ${Result} наш подмаршрут проверит, получили ли мы результат или нет:

```
exten => start,n,GotoIf($[${EXISTS(${Result})}]?call,1:no_result,1)
```

Если результат не получен, вызов будет обрабатываться в расширении no_result. Если результат получен из нашего поиска, выполнение будет продолжено в расширении call, где вызов будет совершен с использованием результата, сохраненного в переменной канала ${Result}.

Получение звонков на ваш ITAD

Прием звонков на ваш ITAD намного проще. Если ваша система поддерживает входящие SIP URI, ISN уже будут работать для вас.[19] Мы показали конфигурацию, необходимую для приема вызовов в вашу систему в разделе «Приём вызовов в вашу систему».

Безопасность и идентификация.

Печально, что в Интернете есть несколько эгоистичных, жадных криминальных типов, которые не упустят попытки воспользоваться преимуществами людей ради собственной выгоды. В телекоммуникации это поведение представляет собой несколько рисков для вас.

В этом разделе мы сосредоточим внимание на проблемах безопасности, связанных с частями вашей системы, которые вы намерены сделать общедоступными через Интернет. Хотя было бы просто отказаться от каких-либо внешних подключений, реальность такова, что если вы хотите, чтобы люди могли звонить вам бесплатно из Интернета (например, если вы намереваетесь публиковать SIP URI вашей компании на своей веб-странице), вам нужно будет определить безопасное место в вашей системе, куда будут поступать эти вызовы. Защита ваших входящих VoIP-соединений концептуально аналогична реализации DMZ в традиционных сетях.[20]

В Asterisk определенным контекстам в вашем диалплане нельзя доверять. Это означает, что вам нужно будет тщательно изучить, какие ресурсы доступны для каналов, которые входят в систему через эти контексты, и обеспечить доступность только определенных сервисов и функций.

19 Если вы правильно настроили свои ITAD и ISN, преобразование из строки набора ISN в SIP URI произойдет до того, как звонок поступит на ваш порог.

20 DMZ - это любая часть вашей сети, которую вы публикуете в Интернете (например, ваш веб-сайт) и, следовательно, не можете полностью доверять. Организации нередко размещают АТС в пределах DMZ.

Мошенничество с оплатой

Мошенничество с оплатой является, безусловно, самым большим риском для вашей телефонной системы с точки зрения потенциальной потери стоимости. Не случайно мошенники в течение нескольких дней набирают десятки тысяч долларов на чужих телефонных звонках.

Мошенничество с оплатой не является новой вещью, существовавшей до VoIP; тем не менее, благоприятный характер VoIP позволяет мошенникам использовать преимущества незащищенных систем. Большинство провайдеров не будет брать на себя ответственность за эти расходы, и, таким образом, если ваша система будет скомпрометирована вы можете в очень большом счете за телефон. В то время как провайдеры становятся все лучше и лучше, предупреждая своих клиентов о подозрительной деятельности, это не освобождает вас от ответственности за то, чтобы ваша система была укреплена против этой очень реальной и очень опасной угрозой.

В вашей системе Asterisk жизненно важно, чтобы вы знали, какие ресурсы в вашей системе находятся во внешнем мире, и убедились, что эти ресурсы защищены.

Самая распространенная форма мошенничества с оплатой в эти дни достигается путем атаки брутфорсом. В этом случае воры имеют скрипт, который связывается с вашей системой и пытается зарегистрироваться как действительный пользователь. Если они могут зарегистрироваться в качестве телефона в вашей системе, начнется поток вызовов, и вы будете получите большой счет. Если вы используете простые добавочные номера и легко угадываемые пароли, и ваша система принимает регистрацию вне вашего брандмауэра, вы наверняка будете жертвой мошенничества с платной подпиской.

Атаки брутфорсом также могут вызвать проблемы с производительностью в вашей системе, так как один из этих сценариев может нагрузить ваш маршрутизатор и АТС огромным количеством попыток регистрации.

Следующая тактика оказалась успешной в минимизации риска мошенничества с платной почтой:

1. Не используйте легкоподбираемые пароли. Пароли должны содержать не менее восьми символов и содержать комбинацию цифр, букв и символов. `8a$j03H%` хороший пароль,[21] `1234` - нет.
2. Не используйте внутренние номера для регистрации SIP в *sip.conf*. Вместо `[1000]` используйте что-то наподобие MAC-адреса (что-то вроде `[0004f2123456]` будет намного сложнее для сценария брутфорса).
3. Используйте сценарий анализа, например *fail2ban*, чтобы настроить ваш внутренний брандмауэр, для блокировки IP-адреса, которые показывают грубое поведение, например массивные потоки пакетов.



Демон f*ail2ban* становится популярным способом автоматического реагирования на угрозы безопасности. Мы обсудим это далее в Главе 26.

Спам по интернет-телефонии (SPIT)

VoIP-спам еще не принят, но будьте уверены, он это сделает. Спамеры во всем мире млеют от счастья на пути к возможности свободно нападать на всех и каждого с помощью телефонной системы с поддержкой Интернета.

Подобно электронной почте, VoIP влечет за собой определенный уровень доверия, поскольку он предполагает, что каждый телефонный звонок является законным. К сожалению, как и в случае спама по электронной почте, для всех нас требуется только несколько плохих яблок.

21 Фактически, поскольку он опубликован в этой книге, это уже не хороший пароль, но вы поняли идею.

Сейчас многие организации и люди работают над тем, как решать проблему SPIT, прежде чем она станет проблемой. Некоторые разрабатываемые концепции включают сертификаты и «белые списки». В качестве окончательного решения не было найдено ни одного метода.

Хотя было бы просто заблокировать наши системы от мира, факт состоит в том, что интернет-телефония - это то, что каждый бизнес будет поддерживать в недалеком будущем. SPIT все чаще становится проблемой, поскольку все больше и больше сомнительных персонажей решают, что это новый путь к богатству.

Решение проблемы SPIT будет продолжаться: битва между нами и The Bad Guys ™.

Распределенные атаки на отказ в обслуживании

Атаки на отказ в обслуживании SIP уже происходят в Интернете. Облако EC2 от Amazon стало популярным местом для создания этих атак, а другие облачные или скомпрометированные системы станут популярными и для этих видов деятельности. Фактические атаки - это не строго отказ от атак службы (в том смысле, что они не намеренно пытаются задушить вашу систему); скорее, они являются атакующими кампаниями, которые обычно пытаются использовать брутфорс для обнаружения любых дыр в любых системах, которые они смогут найти. По мере того, как количество этих атак увеличивается, эффект на сеть будет аналогичен влиянию спама электронной почты.

Ранее упомянутый демон *fail2ban* может быть полезен для минимизации последствий этих атак. См. Главу 26 для получения подробностей.

Фишинг

Когда система VoIP была скомпрометирована, одним из популярных применений системы с нарушенным доступом является передача мошеннических кампаний с использованием идентичности скомпрометированной системы. Преступники, занимающиеся так называемыми фишинговыми экспедициями, будут совершать случайные звонки в списки номеров, пытаясь получить информацию о кредитной карте или другую конфиденциальную информацию, представляясь от имени вашей организации.

Безопасность - это непрерывный процесс.

В отличие от предыдущих выпусков, в этой книге мы попытались привести примеры и передовые методы, которые учитывают безопасность на всех этапах. Где бы вы ни работали, вы должны думать о безопасности. В то время как реализация хорошей безопасности требует больше усилий по проектированию, разработке и тестированию, это сэкономит вам время и деньги в долгосрочной перспективе.

Большинство дыр в безопасности происходят в результате чего-то, что было поспешно реализовано и не было заблокировано позже. «Я сейчас просто построю это, и почищу его позже» - это слова, которые вы никогда не хотите говорить (или слышать).

Вывод

Одна из фантазий VoIP заключалась в том, что она собиралась сделать бесплатные звонки. Спустя более десятилетия мы все еще платим за наши телефонные звонки. Технология существует в течение небольшого времени, но удобство использования там не было.

Не стоит регистрировать ваш ITAD и настраивать систему для обработки ISN. Если бы каждая развернутая система Asterisk имела ITAD, и люди начали публиковать свои ISN на веб-сайтах, картах и визитных карточках, вес сообщества Asterisk привел бы к принятию индустрии.

Необходимо учитывать соображения безопасности для VoIP, но мы ожидаем, что выгоды перевесят риски.

Наша коллективная мечта бесплатного интернет-вызова может быть ближе, чем мы думаем.

Очереди автоматического распределения вызовов (ACD)

Англичанин, даже если он один,
формирует упорядоченную очередь из одного.
-George Mikes

Автоматическое распределение вызовов (Automatic Call Distribution - ACD, а по-русски АРВ) или организация очередности вызовов предоставляет УАТС возможность помещать в очередь входящие вызовы от группы пользователей: она объединяет несколько вызовов в шаблон удержания, назначает каждому вызову ранг и определяет порядок в котором этот вызов должен быть доставлен доступному агенту (как правило, с начала списка). Когда агент становится доступным, вызывающий абонент с наивысшим рангом в очереди доставляется этому агенту, а все остальные перемещаются вверх по рангу.

Если вы когда-либо звонили в организацию и слышали «Все наши операторы заняты», вы испытали АРВ. Преимуществом АРВ для вызывающих абонентов является то, что им не нужно повторять набор номера, чтобы попытаться связаться с кем-то, а преимущества для организаций заключаются в том, что они могут лучше обслуживать своих клиентов и временно обрабатывать ситуации, когда абонентов больше, чем агентов.[1]



Существует два типа центров обработки вызовов: входящий и исходящий. АРВ относится к технологии, которая обрабатывает входящие вызовы, тогда как термин Dialer (или Predictive Dialer) относится к технологии, которая обрабатывает исходящие вызовы. В этой книге мы будем в основном сосредотачиваться на входящем вызове.

Все мы были разочарованы плохо спроектированными и управляемыми очередями: длительная музыка на удержание с радио, которое не настроено, ошеломляющее время ожидания и бессмысленные сообщения, которые каждые 20 секунд повторяют, насколько важен ваш звонок, несмотря на тот факт, что вы ждали 30 минут и слышали сообщение столько раз, что можете процитировать его по памяти. С точки зрения обслуживания клиентов проектирование очереди может быть одним из наиболее важных аспектов вашей телефонной системы. Как и в случае с автосекретарем, следует иметь в виду, прежде всего, то, что *ваши абоненты не заинтересованы в удержании в очереди*. Они позвонили, потому что *они хотят поговорить с тобой*. Все ваши

1 Это-распространенное заблуждение, что очередь может позволить вам обрабатывать больше вызовов. Это не совсем верно: ваши абоненты все равно захотят поговорить с живым человеком, и они будут готовы ждать так долго. Другими словами, если у вас не хватает персонала, ваша очередь может оказаться не более чем препятствием для ваших абонентов. Это то же самое, будь вы на телефоне или на кассе WalMart. Никто не любит стоять в очереди. Идеальная очередь невидима для абонентов, так как на их звонки отвечают немедленно, без необходимости ожидания.

проектные решения должны держать этот важный факт в центре внимания: люди хотят поговорить с другими людьми, а не с вашей телефонной системой.[2]

Цель этой главы - научить вас создавать и проектировать очереди, которые доставят абонентов к намеченным местам назначения как можно быстрее и безболезненнее.



В этой главе мы можем переключаться между использованием терминов *участники очереди*[3] и *агенты*. Поскольку мы не будем тратить много времени на модуль Asterisk с именем `chan_agent` (используя `AgentLogin()`), мы должны четко указать, что в этой книге, когда мы используем термин «агент», мы имеем в виду конечную точку - человека, а не технологию канала в Asterisk с именем `chan_agent`. Читайте дальше, и тогда узнаете больше.

Создание простой очереди ACD

Для начала мы создадим простую очередь ACD. Она будет принимать абонентов и пытаться доставить их участнику очереди.



В Asterisk термин участник относится к каналу (как правило, SIP-пир), назначенному очереди, которому можно набрать, например `SIP/0000FFFF0001`. Агент технически относится к каналу Agent, также используемому для набора конечных точек. К сожалению, канал Agent является устаревшей технологией в Asterisk, поскольку он ограничен гибкостью и может вызвать неожиданные проблемы, которые трудно диагностировать и решить. Мы не будем использовать `chan_agent`, поэтому имейте в виду, что мы обычно будем использовать термин *участник* для ссылки на телефонное устройство и *агент* для обозначения человека, который обрабатывает вызов. Поскольку один из них обычно не действует без другого, любой термин может относиться к обоим.

Мы создадим очереди в файле *queues.conf* и вручную добавим участников очереди через консоль Asterisk. В разделе "Участники очереди" мы рассмотрим, как создать диалплан, который позволяет нам динамически добавлять и удалять участников очереди (а также приостанавливать и возобновлять их).

Первым шагом является создание пустого файла *agent.conf* в каталоге конфигурации */etc/asterisk*. Мы не будем использовать или редактировать этот файл, однако модуль `app_queue` ожидает его найти и не будет загружаться, если он не существует:

```
$ cd /etc/asterisk
$ touch agents.conf
```

Затем вам нужно создать *queues.conf*, в котором определена конфигурация для реальных очередей:

```
$ touch queues.conf
```

Заполните ее следующей конфигурацией, которая создаст две очереди с именами `[sales]` и `[support]`. Вы можете назвать их как захотите, но мы будем использовать в книге эти имена и далее, поэтому, если вы используете другие имена очередей, отличные от тех что мы здесь рекомендовали, обратите внимание на свой выбор для использования в будущем:

```
[general]
autofill=yes            ; распределять всех ожидающих абонентов доступным
                        ; участникам
shared_lastcall=yes     ; соблюдать время отдыха для участников,
                        ; зарегистрированных в более чем одной очереди

[StandardQueue](!)            ; шаблон для предоставления общих функций
musicclass=default            ; воспроизводить  музыку [default]
```

2 Существует несколько книг, в которых обсуждаются показатели центра обработки вызовов и доступные стратегии очередей, такие как James C. Abbott's *The Executive Guide to Call Center Metrics* (Robert Houston Smith).

3 В оригинале *queue members*, что можно перевести как *члены очереди*, но я выбрал вариант *участники* (прим. переводчика).

```
strategy=rrmemory       ; использование стратегии Round Robin Memory
joinempty=no            ; не присоединяться к очереди, если нет доступных
                        ; участников
leavewhenempty=yes      ; покинуть очередь, когда нет доступных
                        ; участников
ringinuse=no            ; не звонить участникам, когда они уже заняты
                        ; (предотвращает множественные вызовы агенту)

[sales](StandardQueue)  ; создать очередь sales с использованием параметров
; шаблона StandardQueue

[support](StandardQueue)      ; создать очередь support с использованием
                              ; параметров шаблона StandardQueue
```

Раздел `[general]` определяет поведение по умолчанию и глобальные параметры. Мы указали только два варианта в разделе `[general]`, так как для для наших нужд на данный момент достаточно встроенных значений по умолчанию.

Первый параметр - `autofill`, который указывает очереди распределять всех ожидающих абонентов всем доступным участникам немедленно. Предыдущие версии Asterisk распределяли только одного вызывающего абонента за раз - это означало, что, хотя Asterisk сигнализировал агенту, все остальные вызовы удерживались (даже если были доступны другие агенты) до тех пор, пока первый вызывающий абонент не был подключен к агенту (что, очевидно, приводило к узким местам в старых версиях Asterisk, где использовались большие очереди). Если у вас нет особой необходимости обратной совместимости, *этот параметр всегда должен быть установлен в значение* `yes`.

Второй вариант в секции `[general]` *queues.conf* является `shared_lastcall`. Когда мы включаем `shared_lastcall`, последний вызов агента, который зарегистрирован в нескольких очередях, будет вызовом, который подсчитывается для времени отдыха[4], чтобы избежать отправки вызова агенту из другой очереди в это время. Если для этой опции установлено значение `no`, время отдыха будет применяться только к очереди, из которой пришел последний вызов, что означает - агент, который завершает вызов из очереди `support`, все равно может получить вызов из очереди `sales`. Эта опция также должна всегда быть установлена в `yes` (по умолчанию).

Следующий раздел `[StandardQueue](!)` - это шаблон, который мы будем применять к нашим очередям `sales` и `support` (мы объявили его шаблоном, добавив `(!)`). Мы определили `musicclass` как `default` в режиме ожидания, как это было настроено в файле *musiconhold.conf*. `strategy` мы будем использовать `rrmemory`, что означает Round-Robin with Memory (циклический перебор с памятью). Стратегия `rrmemory` работает путем ротации через агентов в очереди в последовательном порядке, отслеживая какой агент получил последний вызов, и представляя следующий вызов следующему агенту. Когда она добирается до последнего агента, то возвращается наверх (когда агенты входят в систему, они добавляются в конец списка). Мы установили `joinempty` в `no`, поскольку это вообще дурной тон помещать вызывающих абонентов в очередь, где нет агентов, готовых принять их звонки.



Вы можете установить в `yes` для облегчения тестирования, но мы не рекомендуем запускать в продакшен, если очередь не используется для какой-либо функции, которая не предназначена для передачи вызывающих абонентов агентам. Никто не хочет ждать в очереди, которая никуда не движется.

Опция `leavewhenempty` используется для управления тем, должны ли вызывающие абоненты выпадать из приложения `Queue()` в диалплан если нет доступных участников для принятия их вызовов. Мы установили значение `yes`, так как абоненты не должны стоять в очереди без зарегистрированных агентов.

4 Время отдыха (или время подведения итогов) используется для агентов, которым может потребоваться выполнить какую-либо регистрацию в журнал или другую функцию после завершения вызова. Это дает им льготный период в несколько секунд, чтобы выполнить эту задачу, прежде чем принимать другой вызов.



С точки зрения бизнеса вы должны сообщать своим агентам чтобы они очищали все вызовы из очереди перед выходом из системы в течение дня. Если вы обнаружите, что в конце дня в очереди много вызовов, то можете рассмотреть вопрос о продлении чьей-то смены чтобы справиться с ними. В противном случае они просто добавят вам стресса, когда вернутся на следующий день в плохом настроении.
Вы можете использовать `GotoIfTime()` ближе к концу дня, чтобы перенаправить абонентов на голосовую почту или другое подходящее место в вашем диалплане, в то время как ваши агенты очищают все оставшиеся вызовы в очереди.

Наконец, мы настроили `ringinuse` на `no`, что говорит Asterisk не звонить участникам когда их устройства уже звонят. Цель установки `ringinuse` в `no` состоит в том, чтобы избежать нескольких вызовов одного и того же участника из разных очередей.



Следует отметить, что, `joinempty` и `leavewhenempty` ищут либо отсутствие участников в очереди, либо их недоступность. Агенты, которые являются `Ringing` или `InUse` не считаются недоступными, поэтому не будут блокировать звонящих от присоединения к очереди или выкидывать их при `joinempty=no` и/или `leavewhenempty=yes`.

После того, как вы закончили настройку в вашем файле *queues.conf*, вы должны сохранить его и перезагрузить модуль `app_queue.so` из своего Asterisk CLI:

```
$ asterisk -r
* CLI> module reload app_queue.so
    -- Reloading module 'app_queue.so' (True Call Queuing)
```

Затем убедитесь что ваши очереди были загружены в память (не забудьте обеспечить наличие пустого файла agents.conf):

```
localhost*CLI> queue show
support     has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   No Members
   No Callers

sales       has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   No Members
   No Callers
```

Вывод *queue show* предоставляет различные фрагменты информации, включая те детали, которые приведены в таблице 13-1.

Таблица 13-1. Вывод команды CLI queue show

Поле	Описание
`W:`	Вес очереди
`C:`	Количество вызовов, направленных в очередь
`A:`	Количество вызовов, отвеченных участниками
`SL:`	Уровень обслуживания

Теперь, когда вы создали очереди, вам нужно настроить ваш диалплан, чтобы разрешить вызовам входить в очередь.

Добавьте следующую логику в файл диалплана *extensions.conf*:

```
[Queues]
exten => 7001,1,Verbose(2,${CALLERID(all)} вход в очередь support)
   same => n,Queue(support)
   same => n,Hangup()

exten => 7002,1,Verbose(2,${CALLERID(all)} вход в очередь sales)
   same => n,Queue(sales)
   same => n,Hangup()

[LocalSets]
include = > Queues            ; разрешить телефонам вызов очередей
```

Мы включили `Queues` в контекст `LocalSets` чтобы наши телефоны могли вызывать очереди, которые мы настроили. В Главе 15 мы определим пункты меню, которые ведут в эти очереди. Сохраните изменения в файле *extensions.conf* и перезагрузите диалплан с помощью команды CLI *dialplan reload*.

Если вы наберете номер `7001` или `7002` на этом этапе, то получите следующий результат:

```
 -- Executing [7001@LocalSets:1] Verbose("SIP/0000FFFF0003-00000001",
    "2,"Leif Madsen" <100> entering the support queue") in new stack
== "Leif Madsen" <1--> entering the support queue
 -- Executing [7001@LocalSets:2] Queue("SIP/0000FFFF0003-00000001",
    "support") in new stack
    [2011-02-14 08:59:39] WARNING[13981]: app_queue.c:5738 queue_exec:
    Unable to join queue 'support'
 -- Executing [7001@LocalSets:3]
    Hangup("SIP/0000FFFF0003-00000001", "") in new stack
== Spawn extension (LocalSets, 7001, 3) exited non-zero on
   'SIP/0000FFFF0003-00000001
```

Вы не присоединитесь к очереди на этом этапе, так как в очереди нет агентов для ответа на вызовы. Поскольку у нас есть `joinempty=no` и `leavewhenempty=yes`, настроенные в *queues.conf*, абоненты не будут помещены в очередь. (Это было бы хорошей возможностью для экспериментов с вариантами `joinempty` и `leavewhenempty` в *queues.conf*, чтобы лучше понять их влияние на очереди.)

В следующем разделе мы продемонстрируем, как добавлять участников в очередь (а также другие взаимодействия участников с очередью, такие как пауза/возобновление).

Участники очереди

Очереди не очень полезны, если кто-то не отвечает на входящие в них вызовы, поэтому нам нужен метод, позволяющий агентам регистрироваться в очереди для ответа на вызовы. Существуют различные способы для этого, поэтому мы покажем вам как добавлять участников в очередь как вручную (как администратор либо через CLI, либо жестко в файле *queues.conf*), так и динамически (в качестве агента через расширение определенное в диалплане). Мы начнем с метода CLI Asterisk, который позволит вам легко добавлять участников в очередь для тестирования с минимальными изменениями диалплана. Далее мы покажем как можно определить элементы в *queues.conf* (которые будут добавлять определенных участников когда `app_queue` перезагружается). Наконец, мы покажем вам, как добавить логику диалплану, которая позволит агентам входить в очереди и выходить из них, а также приостанавливать и возобновлять себя в очередях, в которые они вошли.

Управление участниками очереди с помощью CLI

Мы можем добавить участников очереди в любую доступную очередь через команду Asterisk CLI *queue add*. Формат команды *queue add* (все в одной строке):

```
* CLI> queue add member <channel> to <queue> [[[penalty <penalty>] as
```

```
<membername>] state_interface <interface>]
```

Элемент *<channel>* является каналом, который мы хотим добавить в очередь, например `SIP/0000FFFF0003`, а имя *<queue>* будет чем-то вроде `support` или `sales` - любое имя очереди, которое существует в */etc/asterisk/queues.conf*. Пока мы будем игнорировать *<penalty>*, но обсудим это в "Расширенных очередях" (`penalty` используется для управления рангом участника в очереди, что может быть важным для агентов, которые вошли в несколько очередей или имеют разные навыки). Мы можем определить *<membername>* чтобы предоставить подробные сведения механизму логирования очередей.

Опция `state_interface` - это то, что нам нужно рассмотреть более внимательно. Поскольку это так важно для всех аспектов очередей и их участников в Asterisk мы написали небольшой раздел об этом, так что прочитайте "Введение в состояния устройств". Как только вы это настроите, вернитесь сюда и продолжайте. Не волнуйтесь, мы подождем.

Теперь, когда вы добавили `callcounter=yes` в *sip.conf* (мы будем использовать SIP-каналы во всех остальных наших примерах), давайте посмотрим как добавить участников в наши очереди из Asterisk CLI.

Добавление участника очереди `support` может быть выполнено командой *queue add member*:

```
* CLI> queue add member SIP/0000FFFF0001 to support
Added interface 'SIP/0000FFFF0001' to queue 'support'
```

Запрос очереди проверит, добавлен ли наш новый участник:

```
CLI> queue show support
support has 0 calls (max unlimited) in ‘rrmemory’ strategy
(0s holdtime, 0s talktime), W: 0, C: 0, A: 0, SL: 0.0% внутри 0s
   Members:
      SIP/0000FFFF0001 (dynamic) (Not in Use) has taken no calls yet
   No callers
```

Чтобы удалить участника очереди, вы должны использовать команду *queue remove member*:

```
* CLI> queue remove member SIP/0000FFFF0001 from support
Removed interface 'SIP/0000FFFF0001' from queue 'support'
```

Конечно, вы можете ещё раз использовать команду *queue show* чтобы убедиться, что ваш участник удален из очереди.

Мы также можем приостанавливать и возобновлять участников в очереди из консоли Asterisk командами *queue pause member* и *queue unpause member*. Они принимают аналогичный формат с предыдущими командами, которые мы использовали:

```
*CLI> queue pause member SIP/0000FFFF0001 queue support reason DoingCallbacks
paused interface 'SIP/0000FFFF0001' in queue 'support' for reason
'DoingCallBacks'

*CLI> queue show support
support     has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   Members:
      SIP/0000FFFF0001 (dynamic) (paused) (Not in use) has taken no calls yet
No Callers
```

Добавив причину приостановки участника очереди, например `lunchtime` (время обеда) можно гарантировать, что журналы очереди будут содержать некоторые дополнительные сведения, которые могут быть полезны. Вот как вернуть участника:

```
*CLI> queue unpause member SIP/0000FFFF0001 queue support reason off-break
unpaused interface 'SIP/0000FFFF0001' in queue 'support' for reason 'off-
break'

*CLI> queue show support
support     has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   Members:
      SIP/0000FFFF0001 (dynamic) (Not in use) has taken no calls yet
   No Callers
```

В рабочей среде CLI обычно не является лучшим способом управления состоянием агентов в очередях. Вместо этого есть приложения диалплана, которые позволяют агентам сообщать очереди об их доступности.

Определение участников очереди в файле queues.conf

Если вы определяете участника очереди в файле *queues.conf*, то этот участник всегда будет регистрироваться в очереди, когда она перезагружается. Как правило, вы будете использовать этот метод только в том случае, если хотите чтобы участник всегда регистрировался (что обычно не очень хорошо если ваши участники являются людьми, поскольку люди склонны вставать и двигаться).

В каждом определении очереди вы просто определяете элементы таким образом:

```
[general]
; все общие настройки, которые мы обсуждали до

[sales](StandardQueue)
member => SIP/0000FFFF0005    ; или любой другой канал

[service](StandardQueue)
member => SIP/0000FFFF0006
```

В обычной очереди (в которой у вас есть группа людей, отвечающих на вызовы), вы обнаружите, что определение участников в файле *queues.conf* не будет служить хорошей идеей. Агенты должны иметь возможность входа и выхода (и не должны автоматически регистрироваться каждый раз, когда очередь перезагружается). Мы не рекомендуем определять участников в файле *queues.conf*, если у них нет другой цели (например, банк устройств, которые отвечают на вызовы, где вы хотите использовать очередь для балансировки нагрузки в пул устройств или группу вызовов, где все телефоны отвечают на все вызовы постоянно).

Управления участниками очереди с помощью логики диалплана

В колл-центре, в котором работают живые агенты, наиболее часто агенты сами входят и выходят из системы в начале и конце их смены (или когда они отправляются на обед, в ванную комнату, или иным образом недоступны для очереди).

Чтобы включить это мы будем использовать следующие приложения диалплана:

- `AddQueueMember()`
- `RemoveQueueMember()`

При входе в очередь агенту может потребоваться перевести себя в состояние, когда он временно недоступен для приема вызовов. Следующие приложения позволят это:

- `PauseQueueMember()`
- `UnpauseQueueMember()`

Может быть проще подумать об этих приложениях следующим образом: приложения для добавления и удаления используются для входа и выхода из системы, а пара пауза/возобновление используется для коротких периодов недоступности. Разница просто в том, что пауза/возобновление задает участника как недоступного/доступного фактически не удаляя его из очереди. Это полезно для целей отчетности (если участник приостановлен диспетчер очереди может видеть что он зарегистрирован в очереди, но просто недоступен для вызова в данный момент). Если вы не знаете какой из них использовать мы рекомендуем, чтобы агенты использовали добавление/удаление всякий раз, когда они физически не находятся на своем телефоне и паузу/возобновление, когда они находятся за столом, но временно недоступны.

Использование Pause и Unpause

Использование паузы и возобновления является вопросом предпочтения. В некоторых средах эти параметры могут использоваться для всех видов деятельности в течение дня, которые делают агента недоступным (например, во время обеденного перерыва и при выполнении работы, не связанной с очередью). Однако в большинстве колл-центров, если агент не находится рядом с телефоном и готов принять вызов в тот момент, он не должен быть зарегистрирован вообще, даже если он отлучается только на несколько минут от своего стола (например, для перерыва в туалет).

Некоторым супервизорам нравится использовать настройки добавления/удаления и паузы/возобновления в качестве своего рода часов учёта чтобы они могли отслеживать когда их сотрудники прибудут на работу и уйдут в конце дня и сколько времени они проводят за своими столами и на перерывах. Мы не считаем это хорошей практикой, поскольку целью этих приложений является информирование очереди о доступности агента, а не возможностью отслеживания действий сотрудников.

Важная вещь, которую следует здесь отметить, относится к параметру `joinempty` в *queues.conf*, который обсуждался ранее. Если агент приостановлен - он считается вошедшим в очередь. Скажем, ближе к концу дня один агент поставил себя на паузу несколько часов назад, чтобы работать над проектом. Все остальные агенты вышли из системы и ушли домой. Поступает вызов. В очереди указывается, что агент в очереди и поэтому будет стоять в очереди на вызов, хотя реальность такова, что в настоящее время нет людей, которые обслуживают эту очередь. Этот вызывающий абонент может оставаться в очереди без персонала неограниченно долго.

Короче говоря, агенты, которые не сидят за своими столами и не планируют быть доступными для звонков в течение следующих нескольких минут, должны выйти из системы. Пауза/возобновление должны использоваться только для кратковременных недоступностей (или вообще). Если вы хотите использовать свою телефонную систему в качестве часов учёта, есть много отличных способов сделать это с помощью Asterisk, но мы не рекомендуем использовать для этого приложения участника очереди.

Давайте построим простую логику диалплана, которая позволит нашим агентам указывать свою доступность для очереди. Мы собираемся использовать функцию диалплана `CUT()` для извлечения имени нашего канала из нашего вызова в систему, чтобы очередь узнала какой канал входит в очередь.

Мы создали этот диалплан, чтобы показать простой процесс входа и выхода из очереди, а также изменение приостановленного состояния участника в очереди. Мы делаем это только для одной очереди, которую предварительно определили в файле *queues.conf*. Статус переменных канала `AddQueueMember()`, `RemoveQueueMember()`, `PauseQueueMember()` и `UnpauseQueueMember()` приложений может быть использован приложением `Playback()` для объявления участникам очереди после того, как они выполнили определенные функции, чтобы дать им знать об успешности входа/выхода или паузы/возобновленния):

```
[QueueMemberFunctions]
```

```
exten => *54,1,Verbose(2,Logging In Queue Member)
   same => n,Set(MemberChannel=${CHANNEL(channeltype)}/${CHANNEL(peername)})
   same => n,AddQueueMember(support,${MemberChannel})
   same => n,Verbose(1,${AQMSTATUS}) ; ADDED, MEMBERALREADY, NOSUCHQUEUE
   same => n,Playback(agent-loginok)
   same => n,Hangup()

exten => *56,1,Verbose(2,Logging Out Queue Member)
   same => n,Set(MemberChannel=${CHANNEL(channeltype)}/${CHANNEL(peername)})
   same => n,RemoveQueueMember(support,${MemberChannel})
   same => n,Verbose(1,${RQMSTATUS}; REMOVED, NOTINQUEUE, NOSUCHQUEUE
   same => n,Playback(agent-loggedoff)
   same => n,Hangup()

exten => *72,1,Verbose(2,Pause Queue Member)
   same => n,Set(MemberChannel=${CHANNEL(channeltype)}/${CHANNEL(peername)})
   same => n,PauseQueueMember(support,${MemberChannel})
   same => n,Verbose(1,${PQMSTATUS}); PAUSED, NOTFOUND
   same => n,Playback(dictate/paused)
   same => n,Hangup()

exten => *87,1,Verbose(2,Unpause Queue Member)
   same => n,Set(MemberChannel=${CHANNEL(channeltype)}/${CHANNEL(peername)})
   same => n,UnpauseQueueMember(support,${MemberChannel})
   same => n,Verbose(1,${UPQMSTATUS}); UNPAUSED, NOTFOUND
   same => n,Playback(agent-loginok)
   same => n,Hangup()
```

Автоматическая регистрация в нескольких очередях

Обычно агент является участником более чем одной очереди. Вместо того, чтобы иметь отдельный номер для входа в каждую очередь (или требовать информацию от агентов о том, в какие очереди они хотят войти), этот код использует базу данных Asterisk (astdb) для хранения информации об участии в очереди для каждого агента, и затем перебирает каждую очередь, в которую входят агенты, регистрируя их по очереди.

Чтобы этот код работал, запись должна быть добавлена в AstDB через Asterisk CLI. Например, следующее будет хранить участника `0000FFFF0001` в обеих очередях - как в `support`, так и в `sales`:[5]

```
*CLI> database put queue_agent 0000FFFF0001/available_queues support^sales
```

Вам нужно будет сделать это один раз для каждого агента, независимо от количества очередей, в которые они входят.

Если вы затем запросите базу данных Asterisk, то получите результат, похожий на следующий:

```
pbx*CLI> database show queue_agent
/queue_agent/0000FFFF0001/available_queues          : support^sales
```

Следующий код диалплана - пример того, как разрешить этому участнику очереди автоматически добавляться как в очередь `support`, так и в `sales`. Мы определили подпрограмму, которая используется для настройки трех канальных переменных (`MemberChannel`, `MemberChanType`, `AvailableQueues`). Эти переменные канала затем используются для входа (`*54`), выхода из системы (`*56`), паузы (`*72`) и возобновления (`*87`). Каждое из расширений использует подпрограмму

5 Мы будем использовать символ ^ как разделитель. Вероятно, вы могли бы использовать другой символ, только если он не является парсером Asterisk, который будет выглядеть как обычный разделитель (и, таким образом, запутать вас). Поэтому избегайте запятых, точек с запятой и т.д.

subSetupAvailableQueues для установки этих переменных канала и для проверки того, что AstDB содержит список одной или нескольких очередей для устройства, как участника очереди:

```
[subSetupAvailableQueues]
;
; Эта подпрограмма используется различными подпрограммами login/logout/
; pausing/unpausing в контексте [ACD]. Цель подпрограммы-упростить
; централизацию поиска информации.
;
exten => start,1,Verbose(2,Checking for available queues)

; Получение текущего имени канала пира (0000FFFF0001)
   same => n,Set(MemberChannel=${CHANNEL(peername)})

; Получение текущей технологии канала (SIP, IAX, etc)
   same => n,Set(MemberChanType=${CHANNEL(channeltype)})

; Получение списка очередей, доступных этому агенту
   same => n,Set(AvailableQueues=${DB(queue_agent/${MemberChannel}/
   available_queues)})
; *** Это все должно быть в одной строке

; если нет очередей, назначенных для этого агента, мы будем обрабатывать его
; в расширении no_queues_available

   same => n,GotoIf($[${ISNULL(${AvailableQueues})}]?no_queues_available,1)
   same => n,Return()

exten => no_queues_available,1,Verbose(2,No queues available for agent
   ${MemberChannel})
; *** Это все должно быть в одной строке

; воспроизведение сообщения о том, что канал еще не назначен
   same => n,Playback(silence/1&channel&not-yet-assigned)
   same => n,Hangup()

[ACD]
;
; Используется для входа агентов во все настроенные очереди согласно AstDB
;
;
; Вход в несколько очередей через систему AstDB
exten => *54,1,Verbose(2,Logging into multiple queues per the database values)

; получить доступные очереди для этого канала
   same => n,GoSub(subSetupAvailableQueues,start,1())
   same => n,Set(QueueCounter=1)    ; установка переменной счетчика

; Используя CUT(), получить первую очередь, возвращенную из AstDB
; Обратите внимание, что мы использовали '^' в качестве разделителя
   same => n,Set(WorkingQueue=${CUT(AvailableQueues,^,${QueueCounter})})

; Пока переменная канала WorkingQueue содержит значение - выполнять цикл
   same => n,While($[${EXISTS(${WorkingQueue})}])

; AddQueueMember(queuename[,interface[,penalty[,options[,membername
; [,stateinterface]]]]])
; Добавляем канал в очередь, установив интерфейс для вызова и интерфейс для
; мониторинга состояния устройства
```

```
;
; *** Это все должно быть в одной строке
   same => n,AddQueueMember(${WorkingQueue},${MemberChanType}/
${MemberChannel},,,${MemberChanType}/${MemberChannel})


   same => n,Set(QueueCounter=$[${QueueCounter} + 1]) ; увеличиваем наш
                                                      ; счетчик

; получаем следующую доступную очередь; если это ноль - наш цикл закончится
   same => n,Set(WorkingQueue=${CUT(AvailableQueues,^,${QueueCounter})})
   same => n,EndWhile()

; сообщаем агенту что он успешно вошел в систему
   same => n,Playback(silence/1&agent-loginok)
   same => n,Hangup()

exten => no_queues_available,1,Verbose(2,No queues available for
${MemberChannel})
   same => n,Playback(silence/1&channel&not-yet-assigned)
   same => n,Hangup()

; ------------------------

; Используется для выхода агентов из всех настроенных очередей согласно AstDB
exten => *56,1,Verbose(2,Logging out of multiple queues)
; Поскольку мы повторно использовали некоторый код, то поместили дубликат кода
; в подпрограмму
   same => n,GoSub(subSetupAvailableQueues,start,1())
   same => n,Set(QueueCounter=1)
   same => n,Set(WorkingQueue=${CUT(AvailableQueues,^,${QueueCounter})})
   same => n,While($[${EXISTS(${WorkingQueue})}])
   same => n,RemoveQueueMember(${WorkingQueue},${MemberChanType}/$
{MemberChannel})
   same => n,Set(QueueCounter=$[${QueueCounter} + 1])
   same => n,Set(WorkingQueue=${CUT(AvailableQueues,^,${QueueCounter})})
   same => n,EndWhile()
   same => n,Playback(silence/1&agent-loggedoff)
   same => n,Hangup()

; -------------------------

; Используется для приостановки агентов во всех доступных очередях
exten => *72,1,Verbose(2,Pausing member in all queues)
   same => n,GoSub(subSetupAvailableQueues,start,1())

; если мы не определяем очередь, участник останавливается во всех очередях
   same => n,PauseQueueMember(,${MemberChanType}/${MemberChannel})
   same => n,GotoIf($[${PQMSTATUS} = PAUSED]?agent_paused,1:agent_not_found,1)

exten => agent_paused,1,Verbose(2,Agent paused successfully)
   same => n,Playback(silence/1&unavailable)
   same => n,Hangup()

; ------------------------

; Используется для  возобновления агентов во всех доступных очередях
exten => *87,1,Verbose(2,UnPausing member in all queues)
   same => n,GoSub(subSetupAvailableQueues,start,1())
```

```
; если мы не определяем очередь, участник возобновляется во всех очередях
   same => n,UnPauseQueueMember(,${MemberChanType}/${MemberChannel})
   same => n,GotoIf($[${UPQMSTATUS} = UNPAUSED]?
agent_unpaused,1:agent_not_found,1)

exten => agent_unpaused,1,Verbose(2,Agent paused successfully)
   same => n,Playback(silence/1&available)
   same => n,Hangup()

; ------------------------

; Используется как в режиме паузы, так и возобновления
exten => agent_not_found,1,Verbose(2,Agent was not found)
   same => n,Playback(silence/1&cannot-complete-as-dialed)
```

Вы сможете усовершенствовать эти процедуры входа и выхода, учитывая, что переменные канала `AQMSTATUS` и `RQMSTATUS` устанавливаются каждый раз когда используются `AddQueueMember()` и `RemoveQueueMember()`. Например, можно установить флаг, который позволит участнику очереди узнать, что он не был добавлен в очередь, установив флаг, или даже добавив запись или систему преобразования текста в речь для воспроизведения названия конкретной очереди, которая создает проблему. Или, если вы отслеживаете это через AMI, у вас может быть всплывающее окно, или используйте `JabberSend()` чтобы сообщить участнику очереди через мгновенный обмен сообщениями, или...(разве Asterisk не веселье?).

Введение в состояния устройств

Состояния устройств в Asterisk используются для информирования различными сообщениями о том, используется ли ваше устройство в настоящее время или нет. Это особенно важно для очередей, так как мы не хотим отправлять абонентов агенту, который уже находится на телефоне.[6] Состояниями устройства управляет модуль канала и в Asterisk только `chan_sip` имеет соответствующую обработку. Когда очередь запрашивает состояние устройства, она сначала запрашивает драйвер канала (т.е. `chan_sip`). Если канал не может предоставить состояние устройства напрямую (как в случае с `chan_iax2`), он просит ядро Asterisk определить его, что и делается путем поиска по каналам, которые в настоящее время выполняются.

К сожалению, простое обращение к ядру с просьбой выполнить поиск по активным каналам не является точным, поэтому получение состояния устройства из каналов, отличных от `chan_sip` менее надежно при работе с очередями. Мы рассмотрим некоторые методы управления вызовами других типов каналов в “Расширенных очередях", но пока сосредоточимся на SIP-каналах, которые не имеют сложных требований к состояниям устройств. Дополнительные сведения о состояниях устройств см. в Главе 14.

Для правильного определения состояния устройства в Asterisk необходимо включить счетчик вызовов в *sip.conf*. Включив счетчик вызовов, мы сообщаем Asterisk отслеживать активные вызовы для устройства, чтобы эта информация могла быть возвращена в модуль канала и состояние было точно отражено в наших очередях. Во-первых, давайте посмотрим, что происходит с вашей очередью без опции счетчика вызовов:

```
*CLI> queue show support
support     has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   Members:
      SIP/0000FFFF0001 (dynamic) (Not in use) has taken no calls yet
   No Callers
```

6 В некоторых случаях может быть полезно разрешить очереди предоставлять несколько вызовов участнику, но в целом человек может обрабатывать только один вызов за раз.

Теперь предположим, что в нашем диалплане есть номер 555, который вызывает MusicOnHold(). Если мы набираем этот номер не включив счетчик вызовов, запрос очереди support (в которой SIP/0000FFFF0001 является участником) в Asterisk CLI покажет что-то подобное следующему:

```
-- Executing [555@LocalSets:1] MusicOnHold("SIP/0000FFFF0001-00000000",
   "") in new stack
-- Started music on hold, class 'default', on SIP/0000FFFF0001-00000000

*CLI> queue show support
support     has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   Members:
      SIP/0000FFFF0001 (dynamic) (Not in use) has taken no calls yet
   No Callers
```

Обратите внимание, что даже если наш телефон помечен как In Use (используемый), потому что он находится на вызове, он не отображается таковым когда мы смотрим на состояние очереди. Это является проблемой, так как очередь будет рассматривать это устройство как доступное, даже если оно уже находится на вызове.

Чтобы исправить эту проблему, нам нужно добавить callcounter=yes в раздел [general] нашего файла *sip.conf*. Мы также можем специально настроить его для любого шаблона или пира (так как это параметр конфигурации пира); однако это действительно то, что вы хотите установить для всех пиров, которые могут когда-либо быть частью очереди, поэтому лучше всего поместить эту опцию в раздел [general] или как часть каждого шаблона.

Отредактируйте свой *sip.conf* следующим образом:

```
[general]
context=unauthenticated ; контекст по умолчанию для входящих вызовов
allowguest=no           ; отключить неаутентифицированные звонки
srvlookup=yes           ; включить поиск DNS SRV записей для исходящих звонков
udpbindaddr=0.0.0.0     ; прослушивать UDP запросы на всех интерфейсах
tcpenable=no            ; отключить поддержку TCP
callcounter=yes         ; включить статусы устройств для SIP устройств
```

Затем перезагрузите модуль chan_sip и выполните тот же тест снова:

```
*CLI> sip reload
Reloading SIP
 == Parsing '/etc/asterisk/sip.conf':     == Found
```

Теперь устройство должно отображаться при выполнении вызова с него:

```
 == Parsing '/etc/asterisk/sip.conf':     == Found
 == Using SIP RTP CoS mark 5
   -- Executing [555@LocalSets:1] MusicOnHold("SIP/0000FFFF0001-00000001",
      "") in new stack
   -- Started music on hold, class 'default', on SIP/0000FFFF0001-00000001

*CLI> queue show support
support     has 0 calls (max unlimited) in 'rrmemory' strategy
(0s holdtime, 0s talktime), W:0, C:0, A:0, SL:0.0% within 0s
   Members:
      SIP/0000FFFF0001 (dynamic) (In use) has taken no calls yet
   No Callers
```

Короче говоря, Queue() должно знать состояние устройства для надлежащего управления распределением вызовов. Опция callcounter в *sip.conf* является важным компонентом правильно функционирующей очереди.

Файл queues.conf

Мы уже упоминали про файл *queues.conf*, который уже создан, но в нём есть много вариантов и мы решили, что было бы правильно и уместно рассмотреть некоторые из них.

Таблица 13-2 содержит опции, доступные в разделе `[general]` *queues.conf*.

Таблица 13-2. Доступные опции для раздела [general] queues.conf

Опция	Возможные значения	Описание
`persistentmembers`	`yes, no`	Установите значение `yes` для хранения динамически добавляемых элементов очереди в базе данных Asterisk, чтобы они автоматически добавлялись при перезапуске Asterisk.
`autofill`	`yes, no`	Если `autofill` отключено, приложение очереди будет пытаться доставлять вызовы агентам последовательно. Это означает, что одновременно предпринимается попытка передать агентам только один вызов. Дополнительные абоненты не беспокоят агентов, пока абонент не подключен к агенту. При включенном `autofill` абоненты распределяются между доступными агентами одновременно. *Если вы хотите, чтобы это всегда было включено* - `yes`.
`monitor-type`	`MixMonitor`, *<unspecified>*	Если этот параметр задан как `MixMonitor`, Asterisk будет использовать приложение `MixMonitor()` для записи вызовов в очереди. Если вы не укажете значение или не закомментируете параметр, приложение `Monitor()` будет использоваться для записи вызовов.
`updatecdr`	`yes, no`	Установите значение `yes`, чтобы при ответе агента на вызов очереди Asterisk заполнял поле `dstchannel` CDR именем участника очереди. Значение определяется при входе агента в систему с помощью приложения `AddQueueMember()`. Эта опция используется для имитации поведения каналов `chan_agent` в CDR.
`shared_lastcall`	`yes, no`	Это значение важно, если участники вошли в несколько очередей. Оно гарантирует, что ссылка на их последний вызов совместно используется во всех очередях, что гарантирует что все очереди будут уважать время отдыха других очередей

Таблица 13-3 описывает опции, доступные для настройки, определенные для каждой очереди.

Таблица 13-3. Доступные параметры для определенных очередей в queues.conf

Опция	Возможные значения	Описание
`musicclass`	Класс музыки как определено в *musiconhold.conf*	Задает класс музыки, используемый определенной очередью. Это значение также можно переопределить для каждого вызова с помощью переменной `CHANNEL(musicclass)`.
`announce`	Имя файла объявления	Используется для воспроизведения объявления агенту, который отвечает на вызов, как правило, чтобы сообщить ему, из какой очереди поступает абонент. Полезно, когда агент находится в нескольких очередях, особенно когда вызовы из очереди настроены на автоответ. Имейте в виду, что время, необходимое для воспроизведения этой записи, добавляется к времени ожидания вызывающего абонента (так как он не может ее услышать). Также учтите, что ваши агенты будут слышать эту запись несколько раз в день и не получат пользу от длинного сообщения. Если вы используете эту опцию, делайте записи короткими; одно или два слова максимум (и не более секунды в длину).

Опция	Возможные значения	Описание
strategy	ringall, leastrecent, fewestcalls, random, rrmemory, linear, wrandom	ringall Звонят все доступные участники (по умолчанию). Эта стратегия распределения на самом деле не считается APB. В традиционных терминах телефонии она известна как группа вызова. leastrecent Звонит интерфейс, который недавно получил звонок. В очереди, где есть много вызовов приблизительно той же продолжительности, это может сработать. Не сработает, если агент был на вызове в течение часа, и все его коллеги получили свой последний звонок 30 минут назад, потому что агент, который только что закончил 60-минутный вызов, получит следующий. fewestcalls Звонит интерфейс, который завершил наименьшее количество вызовов в этой очереди. Это может быть несправедливо, если вызовы не всегда одинаковой продолжительности. Агент мог обработать три звонка по 15 минут каждый, а его коллега-четыре 5-секундных; агент, который обработал три звонка, получит следующий. random Звонит случайный интерфейс. Это действительно может работать очень хорошо и очень справедливо с точки зрения равномерного распределения вызовов между агентами. rrmemory Прозвон участников в круговой манере, помня, где он остановился последним для следующего абонента. Это также может быть очень справедливым, но не настолько как random. linear Прозвон участников в указанном порядке, всегда начиная с начала списка. Это работает если у вас есть команда, в которой есть агенты, которые должны обрабатывать большинство вызовов, и другие агенты, которые должны получать вызовы, только если основные агенты заняты. wrandom Звонок случайному участнику, но используется пенальти участников в качестве веса. Стоит рассмотреть в большой очереди со сложным взвешиванием среди агентов.
servicelevel	значение в секундах	Эта настройка не так полезна, как должно быть. Идея в том, чтобы определить максимально допустимое время ожидания до ответа. Затем обратите внимание, сколько вызовов отвечают в пределах этого порога, и идут к вашему уровню обслуживания. Так, например, если ваш уровень обслуживания составляет 60 секунд, и 4 из 5 вызовов отвечают за 60 секунд или меньше, ваш уровень обслуживания составляет 80%. Причина, по которой эта метрика не так полезна в Asterisk, заключается в том, что она не разбита каким-либо полезным способом (например, по часам или дням недели). Вместо этого, она дает вам порог для всех звонков с момента последней перезагрузки

Опция	Возможные значения	Описание
		app_queue. Это хорошая идея, но не совсем практичная.
context	контекст диалплана	Позволяет абоненту выйти из очереди, нажав одну цифру DTMF. Если контекст задан и вызывающий абонент вводит цифру, то эта цифра попытается быть сопоставленной в указанном контексте, и выполнение диалплана продолжится там. Обратите внимание, что при этом абонент также теряет свое место в очереди.
penaltymemberslimit	Значение 0 или больше	Используется для игнорирования штрафных значений, если число участников в очереди меньше указанного значения.
timeout	Значение в секундах	Указывает количество секунд для вызова устройства участника. См. также timeoutpriority.
retry	Значение в секундах	Указывает количество секунд ожидания перед попыткой вызова следующего участника очереди, если значение timeout исчерпано при попытке позвонить участнику очереди.
timeoutpriority	app,conf	Используется для управления приоритетом двух возможных вариантов timeout, заданных для очереди. Приложение Queue() имеет значение тайм-аута, которое можно указать для управления абсолютным временем нахождения вызывающего абонента в очереди. Значение timeout в *queues.conf* управляет временем (наряду с повторной попыткой) для вызова участника. Иногда эти значения конфликтуют, поэтому вы можете контролировать, какое значение имеет приоритет. По умолчанию это app, так как оно работает в предыдущих версиях.
weight	Значение 0 или выше	Определяет вес очереди. Очередь с более высоким определенным весом будет иметь приоритет, если участники связаны с несколькими очередями. Имейте в виду, что если у вас очень загруженная очередь с большим весом, абоненты в очереди с меньшим весом могут никогда не получить ответа (или им придется долго ждать).
wrapuptime	Значение в секундах	Количество секунд, в течение которого участник остается недоступным в очереди после завершения вызова. Это время позволяет агенту завершить любую обработку после вызова, которая может потребоваться, прежде чем перейти к следующим вызовам.
autofill	yes,no	То же, что определено в разделе [general]. Это значение может быть определено для каждой очереди.
autopause	yas,no,all	Включает/отключает автоматическую приостановку участников, не отвечающих на вызов. Значение all приводит к приостановке этого участника во всех очередях. Этот параметр может быть сложным в реальной среде, потому что если агент не знает, что он был приостановлен, вы можете оказаться с агентами, ожидающими звонков, не зная, что они были приостановлены. Никогда не используйте это, если у вас нет способа указать участникам, что они были приостановлены, или у вас есть супервизор, который наблюдает за состоянием очереди в режиме реального времени. Также проблема в том, что

Опция	Возможные значения	Описание
		агенты должны быть обучены тому, что ответ на вызов из очереди не является необязательным; очередь ожидает, что они ответят на каждый вызов, который она им представляет.
`maxlen`	Значение 0 или выше	Указывает максимальное число абонентов, которым разрешено ожидать в очереди. Нулевое значение означает что в очереди разрешено неограниченное число абонентов.
`announce-frequency`	Значение в секундах	Определяет, как часто мы должны объявлять позицию вызывающего абонента и/или предполагаемое время ожидания в очереди. Установите это значение в ноль, чтобы отключить. В небольшом колл-центре маловероятно, что система сможет сделать точные оценки и, таким образом, абоненты с большей вероятностью сочтут эту информацию разочаровывающей.
`min-announce-frequency`	Значение в секундах	Определяет минимальное время, которое должно пройти, прежде чем мы снова объявляем позицию абонента в очереди. Это используется, когда положение вызывающего абонента может часто меняться, чтобы абонент не слышал несколько обновлений в течение короткого периода времени.
`periodic-announce-frequency`	Значение в секундах	Указывает как часто мы должны делать периодические объявления вызывающему абоненту. Имейте в виду, что воспроизведение сообщения абонентам на регулярной основе будет расстраивать их, поэтому подумайте о том, чтобы а) это сообщение было коротким и б) не воспроизводилось слишком часто. Приятная музыка будет более удачным вариантом, чем бесконечно повторяющиеся извинения или реклама.
`random-periodic-announce`	`yes,no`	Если задано значение `yes`, будут воспроизводиться определенные периодические объявления в случайном порядке. См. `periodic-announce`.
`relative-periodic-announce`	`yes,no`	Если задано значение `yes`, таймер `periodic-announce-frequency` будет запускаться с момента окончания воспроизводимого файла, а не с самого начала. По умолчанию `no`.
`announce-holdtime`	`yes,no,once`	Определяет, должно ли расчетное время ожидания воспроизводиться вместе с периодическими объявлениями. Можно установить значение `yes`, `no` или `once` (один раз).
`announce-position`	`yes,no,limit,more`	Определяет, должна ли быть объявлена позиция вызывающего абонента в очереди. Если `no`, положение не будет объявлено. Если задано значение `yes`, позиция вызывающего абонента всегда будет объявлена. Если установлено значение `limit`, звонящий услышит свою позицию в очереди, только если он находится в пределах, определенных `announce-position-limit`. Если значение больше, звонящий услышит его положение, если он не входит в число, определяемое `announce-position-limit`. Если в вашей системе есть какая-либо логика, которая может повысить абонентов по рангу (т. е. высокоприоритетные вызовы перемещаются в начало очереди), лучше не использовать эту опцию. Редко что расстраивает звонящего больше, чем уведомление о том, что его

Опция	Возможные значения	Описание
		переместили в конец очереди.
announce-position-limit	Число 0 или выше	Используется, если вы определили announce-position как limit или more.
announce-round-seconds	Значение в секундах	Если это значение не равно нулю, объявляется число секунд и округляется до заданного значения.
queue-thankyou	Имя файла подсказки для воспроизведения	Если значение не определено, воспроизводится значение по умолчанию (”Благодарим за терпение"). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
queue-youarenext	Имя файла подсказки для воспроизведения	Если значение не определено, воспроизводится значение по умолчанию (”Ваш звонок является первым в очереди и будет отвечен первым доступным оператором"). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
queue-thereare	Имя файла подсказки для воспроизведения	Если не определено, воспроизводит значение по умолчанию (”Ваша позиция в очереди"). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
queue-callswaiting	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию (”Ожидайте, пожалуйста, ответа оператора"). Если задано пустое значение, приглашение не будет воспроизводиться вообще
queue-holdtime	Имя файла подсказки для воспроизведения	Если значение не определено, воспроизводится значение по умолчанию (“Прогнозируемое время ожидания составляет”). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
queue-minutes	Имя файла подсказки для воспроизведения	Если значение не определено, воспроизводится значение по умолчанию ("минут"). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
queue-seconds	Имя файла подсказки для воспроизведения	Если значение не определено, воспроизводится значение по умолчанию ("секунд"). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
queue-reporthold	Имя файла подсказки для воспроизведения	Если значение не определено, воспроизводится значение по умолчанию ("Время ожидания"). Если задано пустое значение, приглашение не будет воспроизводиться вообще.
periodic-announce	Набор периодических объявлений для воспроизведения, разделенные запятыми	Подсказки воспроизводятся в том порядке, в котором они определены. По умолчанию используется queue-periodic-announce (“Все наши операторы заняты, пожалуйста, оставайтесь на линии и Ваш звонок будет обслужен первым доступным оператором").
monitor-format	*gsm, wav, wav49, <любой действительный формат>*	Задает формат файла, используемый при записи. Если monitor-format закомментирован, вызовы не будут записываться.
monitor-type	MixMonitor	То же, что и monitor-type, определенный в разделе [general], но для каждой очереди. Если вам нужно старое поведение монитора, просто удалите или отметьте эту опцию полностью
joinempty	paused, penalty, inuse, ringing, unavailable,	Управляет добавлением абонента в очередь при отсутствии доступных участников. Параметры, разделенные запятыми, можно включить, чтобы

Опция	Возможные значения	Описание
	invalid, unknown, wrapup	определить, как этот параметр определяет доступность элементов. Определения значений: paused Участники считаются недоступными, если они приостановлены. penalty Участники считаются недоступными, если их штрафы меньше, чем QUEUE_MAX_PENALTY. inuse Участники считаются недоступными, если status их устройства In Use. ringing Участники считаются недоступными, если состояние их устройства Ringing. unavailable Применяется в основном к каналам агента; если агент не вошел в систему, но является участником очереди, канал считается недоступным. invalid Участники считаются недоступными, если их состояние устройства Invalid. Как правило, это условие ошибки. unknown Участники считаются недоступными, если состояние их устройств неизвестно. wrapup Участники считаются недоступными, если они находятся в данный момент во времени отдыха после завершения вызова.
leavewhenempty	paused, penalty, inuse, ringing, unavailable, invalid, unknown, wrapup	Используется для контроля того, будут ли абоненты выброшены из очереди, когда участники больше не могут принимать вызовы. Смотрите joinempty для получения дополнительной информации о присваиваемых значениях.
eventwhencalled	yes, no, vars	Если задано значение yes, следующие события диспетчера будут отправлены в интерфейс управления Asterisk (AMI): • AgentCalled • AgentDump • AgentConnect • AgentComplete Если установлено значение vars, все переменные канала, связанные с агентом, также будут отправлены в AMI.
eventmemberstatus	yes,no	Если задано значение yes, событие QueueMemberStatus будет отправлено в AMI. Обратите внимание, что это может генерировать много событий менеджера.
reportholdtime	yes,no	Позволяет сообщать о времени удержания вызывающего абонента участнику очереди до установления моста. Имейте в виду, что воспроизведение этой информации для агента потребует времени, которое для вызывающего абонента будет представлять еще большее время удержания.
ringinuse	yes,no	Используется для предотвращения отправки вызовов участникам, состояние которых In Use. Напомним

Опция	Возможные значения	Описание
		из нашего обсуждения в предыдущем разделе, что только драйвер канала SIP в настоящее время может точно сообщить об этом состоянии.
`memberdelay`	Значение в секундах	Используется если требуется задержка перед соединением вызывающего абонента и участника очереди.
`timeoutrestart`	`yes,no`	Если задано значение `yes`, сбрасывает время ожидания ответа агента при получении от канала состояния `BUSY` или `CONGESTION`. Это может быть полезно, если агенту разрешено отклонить или отменить вызов.
`defaultrule`	Правило определенное в *queuerules.conf*	Привязывает правило очереди, как определено в *queuerules.conf* к этой очереди; используется для динамического изменения минимальных и максимальных штрафов, которые затем используются для выбора доступного агента. См. "Динамическое изменение штрафов (queuerules.conf)".
`member`	`Устройство`	Используется для определения статических участников в очереди. Для определения статического элемента необходимо указать его *`Technology/Device_ID`* (напр., `Agent/1234`, `SIP/0000FFFF0001`, `DAHDI/g0/14165551212`).

Значения, приведенные в Таблице 13-4 связаны с соединением вызова и участника очереди. Эти переменные могут быть полезны для целей регистрации; однако их нельзя использовать для принятия решений о маршрутизации, поскольку они устанавливаются только в тот момент, когда вызов соединен (т.е. агент отвечает), и, к сожалению, не во время вызова агента (агент звонит).

Таблица 13-4. Параметры мостового соединения

Опция	Возможные значения	Описание

`setinterfacevar`	`yes`	Если задано значение `yes`, следующие переменные канала будут установлены непосредственно перед соединением абонента с участником очереди:
		`MEMBERINTERFACE`
		Интерфейс участника, например `Agent/1234`
		`MEMBERNAME`
		Имя участника
		`MEMBERCALLS`
		Количество вызовов, выполненных интерфейсом
		`MEMBERLASTCALL`
		Последний раз, когда участник принимал вызов
		`MEMBERPENALTY`
		Штрафное значение участника
		`MEMBERDYNAMIC`
		Указывает, был ли элемент динамически добавлен в очередь
		`MEMBERREALTIME`
		Указывает, включен ли участник в режиме Realtime или нет

`setqueueentryvar`	`yes,no`	Если установлено в `yes`, следующие переменные канала будут установлены непосредственно перед соединением:
		`QEHOLDTIME`
		Время удержания вызывающего абонента в очереди
		`QEORIGINALPOS`
		Позиция, с которой абонент первоначально вошел в очередь

`setqueuevar`	`yes,no`	Если установлено в `yes`, следующие переменные канала будут

Опция	Возможные значения	Описание
		установлены непосредственно перед соединением вызова: QUEUENAME Имя очереди QUEUEMAX Максимальное число вызовов, разрешенных в этой очереди QUEUESTRATEGY Метод стратегии, определенный для очереди QUEUECALLS Количество вызовов, находящихся в очереди QUEUEHOLDTIME Текущее среднее время удержания абонентов в очереди QUEUECOMPLETED Количество завершенных вызовов в этой очереди QUEUEABANDONED Количество прерванных звонков QUEUESRVLEVEL Уровень обслуживания очереди QUEUESRVLEVELPERF Производительность очереди на уровне обслуживания
membermacro	Имя макроса, определенного в диалплане	Определяет макрос, который будет выполняться непосредственно перед соединением вызывающего объекта и участника очереди. Этот макрос можно использовать с предыдущими параметрами (которые задают дополнительные переменные канала), что позволяет оценить состояние очереди в системе в момент доставки вызова агенту. Если это вас интересует, убедитесь, что вместо этого вы используете локальный канал Asterisk (определите участников очереди как локальные каналы). Затем вы сможете реализовать решения о маршрутизации, которые выполняются непосредственно перед тем, как вызов будет предоставлен агенту, что может дать вам гораздо больше контроля над очередями чем app_queue может предоставить самостоятельно.

Файл agents.conf

Если вы просматривали примеры в каталоге *~/src/asterisk-complete/11/configs* вы, возможно, заметили файл *agents.conf*. Он может показаться заманчивым и у него есть свои места, но в целом лучший способ реализовать очереди - это использование каналов SIP. Для этого есть две причины. Во-первых, SIP-каналы являются единственным типом, который предоставляет истинную информацию о состоянии устройств. Другая причина заключается в том, что каналы агентов всегда подключаются при входе в систему, и если вы используете удаленных агентов, требования к пропускной способности для всех этих каналов могут быть больше, чем вы ожидаете. Однако в загруженных колл-центрах может быть желательно заставить агентов отвечать на вызовы немедленно, а не заставляя их нажимать кнопку ответа на телефоне и агентские каналы могут быть полезны для этого в некоторых случаях (мы все же предпочли бы использовать SIP каналы с автоответом, установленным на аппарате).

Файл *agents.conf* используется для определения агентов очередей использованием агентских каналов. Этот канал по своей природе похож на другие типы каналов в Asterisk (Local, SIP, IAX2 и т.д.), но это скорее псевдоканал в том смысле, что он используется для подключения вызывающих абонентов к агентам, которые вошли в систему с использованием других типов транспортных каналов. Например, предположим, что мы используем наш SIP-телефон для входа в Asterisk с помощью приложения диалплана `AgentLogin()`. Как только мы вошли в систему канал остается открытым все время, пока он доступен (в системе), а затем ему передаются вызовы по агентскому каналу на телефон.

Давайте посмотрим на различные параметры, доступные нам в файле *agent.conf*, чтобы лучше понять, что он нам предоставляет. Таблица 13-5 показывает единственную опцию, доступную в разделе `[general]` *agent.conf*. Таблица 13-6 покажет доступные опции под заголовком `[agents]`.

Таблица 13-5. Параметры доступные под заголовком [general] в agents.conf

Опция	Возможные значения	Описание
multiplelogin	yes,no	Если задано значение yes, одна линия на устройстве может входить в систему как несколько агентов. По умолчанию yes.

Таблица 13-6. Параметры, доступные под заголовком [agents] в agents.conf

Опция	Возможные значения	Описание
maxloginretries	Целочисленное значение	Указывает максимальное число попыток, которое агент должен выполнить для входа в систему, прежде чем система сочтет попытку неудачной и завершит вызов. По умолчанию 3. Это не блокирует учетную запись, поэтому пользователь может немедленно повторить попытку так часто, как он хочет.
autologoff	Значение в секундах	Указывает количество секунд, в течение которых устройство агента должно звонить, прежде чем агент автоматически выйдет из системы.
autologoffunavail	yes,no	Если задано значение yes, агент автоматически выходит из системы, если вызываемое устройство возвращает состояние CHANUNAVAIL.
ackcall	yes,no	Если задано значение yes, агент должен ввести одну цифру DTMF чтобы принять вызов. Должно использоваться совместно с acceptdtmf. По умолчанию no.
acceptdtmf	Одиночный символ DTMF	Используется в сочетании с ackcall, этот параметр определяет DTMF символ, который будет использоваться для принятия вызова. По умолчанию #.
endcall	yes,no	Если установлено yes, позволяет агенту завершить вызов одной цифрой DTMF. Должно использоваться совместно с enddtmf. По умолчанию yes.
enddtmf	Одиночный символ DTMF	Используется с endcall, этот параметр определяет символ DTMF, который будет использоваться для завершения вызова. По умолчанию *.
wrapuptime	Значение в миллисекундах	Указывает время после отключения вызывающего абонента, в течение которого агент не сможет принять другой вызов. Используется, когда агенты должны выполнять функцию после каждого вызова (например, ввод сведений о вызове в журнал).
musiconhold	Класс музыки определенный в *misuconhold.conf*	Определяет прослушиваемые агентами класс музыки по умолчанию при входе в систему.
goodbye	Имя файла (относительно */var/lib/asterisk/sounds/<lang>*)	Определяет звук прощания по умолчанию, воспроизводимый агентам. По умолчанию *vm-goodbye*.
updatecdr	yes,no	Используется в CDRs для изменения поля исходного канала на *agent/agent_id*.
group	Целочисленное значение	Позволяет определить группы для наборов агентов. *Использование групп агентов является несколько устаревшей функциональностью, которую мы не рекомендуем использовать.* Если вы определяете group1, Вы можете использовать Agent/@1 в *queues.conf* для вызова этой группы агентов. Вызов будет подключен произвольно к одному из этих агентов. Если никакие агенты не доступны, то возвратится назад к очереди как любой другой неотвеченный вызов. Если вы используете Agent/:1, то он будет ждать пока участник группы станет доступным. *Использование стратегий не влияет на группы агентов. Не используйте их.*
recordagentcalls	yes,no	Включает/отключает запись вызовов агента. По умолчанию отключено.

Опция	Возможные значения	Описание
`recordformat`	Формат файла (`gsm`, `wav`, и тд)	Определяет формат, используемый для записи вызовов агента. По умолчанию `wav`.
`urlprefix`	Строка (URL)	Принимает строку в качестве аргумента. Строка может быть сформирована в виде URL и добавлена к началу текста, который будет добавлен к имени записи.
`savecallsin`	Путь файловой системы (напр. */var/calls*)	Принимает путь к файловой системе в качестве аргумента. Позволяет переопределить путь по умолчанию */var/spool/asterisk/monitor* одним из выбранных вами параметров.[a]
`custom_beep`	Имя файла (относительно */var/lib/asterisk/sounds/<lang>*)	Принимает имя файла в качестве аргумента. Может использоваться для определения пользовательского тона уведомления, чтобы сигнализировать всегда подключенному агенту о входящем вызове. В идеале, выбирайте короткую запись (не более нескольких сотен миллисекунд).

`agent`	Определение агента (см. описание)	Где резина попадает на дорогу. Этот параметр можно повторять так часто, как требуется. Каждый экземпляр определяет агента для использования `Queue()` и `AgentLogin()`. Это агенты, которые будут входить в систему и оставаться подключенными к системе, ожидая вызовов, которые будут доставлены приложением `Queue()`. Агенты определяются следующим образом:
		`agent => agent_id,agent_password,name`
		Пример определения агента:
		`agent => 1000,1234,Danielle Roberts`

[a] Поскольку для хранения вызовов потребуется большой объем места на жестком диске, вам потребуется определить стратегию для хранения и управления этими записями. Это место, вероятно, должно располагаться на отдельном томе с очень высокими характеристиками.

Расширенные очереди

В этом разделе мы рассмотрим некоторые более тонкие элементы управления очередью, такие как параметры для управления объявлениями и когда абоненты должны быть помещены (или удалены) в очередь. Мы также рассмотрим штрафы и приоритеты, исследуя методы управления агентами в нашей очереди, отдавая предпочтение пулу агентов для ответа на вызов и динамически увеличивая этот пул на основе времени ожидания в очереди. Наконец, мы рассмотрим использование локальных каналов в качестве участников очереди, что дает нам возможность выполнять функции диалплан до подключения вызывающего абонента к агенту.

Приоритетная очередь (взвешивание очереди)

Иногда вам нужно добавить людей в очередь с более высоким приоритетом, чем у других абонентов. Возможно, вызывающий абонент уже провел время в очереди, а агент получил некоторую информацию, но понял, что абонент должен быть переведен в другую очередь. В этом случае, чтобы минимизировать общее время ожидания абонента, может быть желательно перевести вызов в приоритетную очередь, которая имеет больший вес (и, следовательно, более высокое предпочтение), поэтому на этот вызов будет дан быстрый ответ.

Установка более высокого приоритета в очереди выполняется с помощью параметра `weight` (вес). Если у вас есть две очереди с разными весами (например, `support` и `support-priority`) агентам, назначенным в обе очереди, будут переданы вызовы из очереди с более высоким приоритетом, а не из очереди с низким приоритетом. Эти агенты не будут принимать вызовы из очереди с более низким приоритетом, пока очередь с высоким приоритетом не будет очищена. (Обычно есть некоторые агенты, которые назначаются только в очередь с более низким приоритетом чтобы гарантировать своевременную обработку этих вызовов.) Например, если мы поместим участника очереди Джеймса Шоу и в `support` и в `support-priority` звонки из очереди `support-priority` будут иметь предпочтительное положение для Джеймса по сравнению с вызывающими в очереди `support`.

Давайте посмотрим, как мы могли бы сделать эту работу. Во-первых, нам нужно создать две одинаковые очереди, за исключением опции `weght`. Мы можем использовать шаблон для этого, чтобы гарантировать, что две очереди останутся идентичными, если что-то должно будет изменяться в будущем:

```
[support_template](!)
musicclass=default
strategy=rrmemory
joinempty=no
leavewhenempty=yes
ringinuse=no

[support](support_template)
weight=0

[support-priority](support_template)
weight=10
```

С нашими настроенными очередями (и впоследствии перезагруженными с помощью *module reload app_queue.so* из консоли Asterisk) мы можем теперь создать два расширения для передачи абонентов. Это можно сделать в любом месте, где вы обычно размещаете логику диалплана для выполнения трансфера. Мы собираемся использовать контекст `LocalSets`, который мы ранее включили в качестве начального контекста для наших устройств:

```
[LocalSets]
include => Queue  ; разрешить прямой перевод вызовов в очереди

[Queues]
exten => 7000,1,Verbose(2,Entering the support queue)
   same => n,Queue(support)         ; стандартная очередь support доступна
                                    ; с номера 7000
   same => n,VoiceMail(7000@queues,u)     ; если нет участников в очереди мы
                                          ; выходим и отправляем абонента на
                                          ; голосовую почту
   same => n,Hangup()

exten => 8000,1,Verbose(2,Entering the priority support queue)
   same => n,Queue(support-priority)      ; приоритетная очередь доступна по
                                          ; номеру 8000
   same => n,VoiceMail(7000@queues,u)     ; если нет участников в очереди мы
                                          ; выходим и отправляем абонента на
                                          ; голосовую почту
   same => n,Hangup()
```

Вот и все: две очереди с разными весами. Мы настроили наши стандартные очереди для запуска с номера `7000`, а наши приоритетные очереди - с `8000`. Мы можем отразить это для нескольких очередей, просто сопоставив диапазоны `7XXX` и `8XXX`. Так, например, если у нас есть очередь `sales` с номером `7004`, наша очередь `priority-sales` (для постоянных клиентов) может быть помещена в зеркальную очередь на `8004`, которая имеет больший вес.

Единственная оставшаяся конфигурация - убедиться, что некоторые или все участники очереди помещены в обе очереди. Если у вас больше абонентов в очередях `7XXX`, возможно вы захотите, иметь больше участников в этой очереди, а часть участников вашей будет входить в обе очереди. Как именно вы хотите настроить свои очереди будет зависеть от вашей локальной политики и обстоятельств.

Приоритет участника очереди

В пределах очереди мы можем оштрафовать участников, чтобы снизить их предпочтение в вызовах, когда в определенной очереди ожидают люди. Например, мы можем оштрафовать участников очереди, когда хотим чтобы они были участниками очереди, но принимать вызовы только тогда, когда очередь заполнится настолько, что все наши предпочтительные агенты будут недоступны. Это означает что у нас может быть три очереди (скажем `support`, `sales` и `billing`), каждая из которых содержит тех же трех участников очереди: Джеймс Шоу, Кей Мэдсен и Даниэль Робертс.

Предположим, мы хотим, чтобы Джеймс Шоу был предпочтительным контактом в очереди `support`, Кей Мэдсен - в `sales`, а Даниэль Робертс - в `billing`. Наказывая Кея Мэдсена и Даниэля Робертса в `support`, мы гарантируем, что Джеймс Шоу будет предпочтительным участником вызываемой очереди. Точно так же мы можем наказать Джеймса Шоу и Даниэля Робертса в `sales`, поэтому Кей Мэдсен предпочтительнее, и наказать Джеймса Шоу и Кея Мэдсена в `billing`, поэтому Даниэль Робертс предпочтительнее.

Наказание участников очереди может быть выполнено либо в файле *queues.conf* (если вы статически указываете участников очереди), либо через приложение диалплана `AddQueueMember()`. Давайте посмотрим, как наши очереди будут настроены со статическими участниками в *queues.conf*. В нашем примере предполагается, что ваш *queues.conf* по-прежнему содержит шаблон `StandardQueue`, который мы определили ранее в этой главе:

```
[support](StandardQueue)
member => SIP/0000FFFF0001,0,James Shaw         ; предпочтительный
member => SIP/0000FFFF0002,10,Kay Madsen        ; второй предпочтительный
member => SIP/0000FFFF0003,20,Danielle Roberts  ; наименее предпочтительный

[sales](StandardQueue)
member => SIP/0000FFFF0002,0,Kay Madsen
member => SIP/0000FFFF0003,10,Danielle Roberts
member => SIP/0000FFFF0001,20,James Shaw

[billing](StandardQueue)
member => SIP/0000FFFF0003,0,Danielle Roberts
member => SIP/0000FFFF0001,10,James Shaw
member => SIP/0000FFFF0002,20,Kay Madsen
```

Определяя различные штрафы для каждого участника очереди,[7] мы можем помочь контролировать предпочтение, куда доставляются абоненты, но при этом гарантировать, что другие участники очереди будут доступны для ответа на вызовы если предпочтительный участник недоступен. Штрафы также могут быть определены с помощью `AddQueueMember()`, как показано в следующем примере:

```
exten => *54,1,Verbose(2,Logging In Queue Member)
   same => n,Set(MemberChannel=${CHANNEL(channeltype)}/${CHANNEL(peername)})

; *CLI> database put queue support/0000FFFF0001/penalty 0
   same => n,Set(QueuePenalty=${DB(queue/support/${CHANNEL(peername)}/penalty)})

; *CLI> database put queue support/0000FFFF0001/membername "James Shaw"
  same => n,Set(MemberName=${DB(queue/support/${CHANNEL(peername)}/membername)})

; AddQueueMember(queuename[,interface[,penalty[,options[,membername
; [,stateinterface]]]]])
   same => n,AddQueueMember(support,${MemberChannel},
${QueuePenalty},,${MemberName})
```

7 Подобно добавлению балласта жокею или гоночному автомобилю

Используя AddQueueMember() мы показали, как можно получить штраф, связанный с именем участника для конкретной очереди и присвоить это значение участнику когда он регистрируется в очереди. Для работы с несколькими очередями потребуется дополнительная абстракция; дополнительные сведения см. В разделе "Автоматическая регистрация в нескольких очередях".

Динамическое изменение штрафов (queuerules.conf)

Используя файл *queuerules.conf* можно указать правила для изменения значений переменных канала QUEUE_MIN_PENALTY и QUEUE_MAX_PENALTY. Переменные QUEUE_MIN_PENALTY и QUEUE_MAX_PENALTY используются для управления предпочтительными участниками очереди для обслуживания абонентов. Допустим, у нас есть очередь support и есть пять участников очереди с различными штрафами в диапазоне от 1 до 5. Если до того как вызывающий абонент войдет в очередь, для переменной канала QUEUE_MIN_PENALTY будет задано значение 2, а для QUEUE_MAX_PENALTY - значение 4, только те участники очереди, для которых установлены штрафы со значениями в диапазоне от 2 до 4 будут считаться доступными для ответа на этот вызов:

```
[Queues]
exten => 7000,1,Verbose(2,Entering the support queue)
same => n,Set(QUEUE_MIN_PENALTY=2)  ; устанавливает минимальный штраф участников
same => n,Set(QUEUE_MAX_PENALTY=4)  ; устанавливает максимальный штраф участников
same => n,Queue(support)            ; вход в очередь с использованием минимальных
                                    ; и максимальных штрафов участников
```

Более того, во время пребывания абонента в очереди мы можем динамически изменять значения QUEUE_MIN_PENALTY и QUEUE_MAX_PENALTY для этого абонента. Это позволяет использовать несколько или другой набор участников очереди в зависимости от того, как долго абонент ожидает в очереди. Например, в предыдущем примере мы могли бы изменить минимальный штраф на 1 и максимальный - на 5, если вызывающий должен ждать более 60 секунд в очереди. Правила определяются с помощью файла queuerules.conf. Несколько правил могут быть созданы для облегчения различных изменений штрафа на протяжении всего вызова. Давайте посмотрим, как мы определим изменения, описанные в предыдущем абзаце:

```
[more_members]
penaltychange => 60,5,1
```



Если вы вносите изменения в файл *queuerules.conf* и перезагружаете *app_queue* таким образом, новые правила будут влиять только на новых абонентов, входящих в очередь, а не на существующих, которые уже удерживаются.

Мы определили правило more_members в *queuerules.conf* и передали следующие значения для изменения штрафа:

60

Количество секунд ожидания перед изменением значений штрафа.

5

Новое значение QUEUE_MAX_PENALTY.

1

Новое значение QUEUE_MIN_PENALTY.

С нашим новым определенным правилом мы должны перезагрузить *app_queue*, чтобы применить изменения:

```
*CLI> module reload app_queue.so
    -- Reloading module 'app_queue.so' (True Call Queueing)
  == Parsing '/etc/asterisk/queuerules.conf':   == Found
```

Мы также можем проверить наши правила на консоли с помощью *queue show rules*:

```
*CLI> queue show rules
Rule: more_members
   After 60 seconds, adjust QUEUE_MAX_PENALTY to 5 and QUEUE_MIN_PENALTY to 1
```

Теперь, когда наше правило загружено в память, мы можем изменить диалплан, чтобы использовать его. Просто измените строку `Queue()` так, чтобы включить новое правило:

```
[Queues]
exten => 7000,1,Verbose(2,Entering the support queue)
   same => n,Set(QUEUE_MIN_PENALTY=2)     ; устанавливает миним. штраф участников
   same => n,Set(QUEUE_MAX_PENALTY=4)     ; устанавливает макс. штраф участников
; Queue(queuename[,options[,URL[,announceoverride[,timeout[,AGI[,macro
; [,gosub[,rule[,position]]]]]]]])
   same => n,Queue(support,,,,,,,,more_members)      ; вход в очередь с
                                                     ; использованием мин и
                                                  ; максимальных штрафов участников
```

Файл *queues.conf* достаточно гибкий. Мы можем определить наше правило, используя относительные, а не абсолютные значения штрафа и также можем определить несколько правил:

```
[more_members]
penaltychange => 30,+1
penaltychange => 45,,-1
penaltychange => 60,+1
penaltychange => 120,+2
```

Здесь мы изменили наше правило `more_members`, чтобы использовать относительные значения. Через 30 секунд мы увеличиваем максимальный штраф на `1` (что приведет нас к `5`, используя наш пример диалплана). После 45 секунд, мы уменьшаем минимальный штраф на `1` и так далее. Мы можем проверить изменения нашего нового правила после *module reload app_queue.so* в консоли Asterisk:

```
*CLI> queue show rules
Rule: more_members
  After 30 seconds, adjust QUEUE_MAX_PENALTY by 1 and QUEUE_MIN_PENALTY by 0
  After 45 seconds, adjust QUEUE_MAX_PENALTY by 0 and QUEUE_MIN_PENALTY by -1
  After 60 seconds, adjust QUEUE_MAX_PENALTY by 1 and QUEUE_MIN_PENALTY by 0
  After 120 seconds, adjust QUEUE_MAX_PENALTY by 2 and QUEUE_MIN_PENALTY by 0
```

Управление объявлениями

Asterisk имеет возможность воспроизводить несколько объявлений для абонентов, ожидающих в очереди. Например, вы можете объявить позицию вызывающего абонента в очереди, среднее время ожидания или сделать периодические объявления благодарящие абонентов за ожидание (или что бы там ни говорили ваши аудиофайлы). Важно тщательно настроить значения, контролирующие воспроизведение этих объявлений для вызывающих абонентов, потому что объявление их позиции, благодарность за ожидание и слишком частое информирование их о среднем времени удержания будет раздражать их, что не задумано этими функциями.

Воспроизведение объявлений между файлами музыки на удержание

Вместо обработки сложных объявлений для каждой из ваших очередей, вы могли бы альтернативно (или совместно) использовать функцию объявления, определенную в *musiconhold.conf*. Перед воспроизведением файла для музыки будет воспроизведен файл

объявления, и затем он воспроизведется снова между аудиофайлами. Скажем, например, у вас есть 5-минутный цикл аудио, но вы хотите воспроизводить сообщение «Спасибо за ожидание» каждые 30 секунд. Вы можете разделить аудиофайл на 30-секундные сегменты, установить их имена файлов, начиная с `00-`, `01-`, `02-` и т.д. (Чтобы они воспроизводились по порядку), а затем определить объявление. В *musiconhold.conf* класс может выглядеть примерно так:

```
[moh_jazz_queue]
mode=files
sort=alpha
announcement=queue-thankyou
directory=moh_jazz_queue
```

Есть несколько вариантов в файле *queues.conf*, которые можно использовать для тонкой настройки того, что и когда объявления воспроизводятся абонентам. Полный список параметров очереди доступен в "Файл queues.conf", но мы рассмотрим соответствующие здесь.

Таблица 13-7 содержит список параметров, которые вы можете использовать, чтобы контролировать, когда объявления воспроизводятся абоненту.

Таблица 13-7. Функции связанные с контролем времени в очереди

Опция	Возможные значения	Описание
`announce-frequency`	Значение в секундах	Определяет как часто мы должны объявлять позицию абонента и/или расчетное время удержания очереди. Установите это значение на ноль, чтобы отключить.
`min-announce-frequency`	Значение в секундах	Указывает минимальное количество времени, которое должно пройти прежде чем объявить позицию абонента в очереди снова. Используется когда позиция абонента может часто меняться, чтобы предотвратить прослушивание нескольких объявлений за короткий промежуток времени
`periodic-announce-frequency`	Значение в секундах	Указывает, как часто нужно делать периодические объявления вызывающей стороне.
`random-periodic-announce`	`yes,no`	Если установлено `yes`, будут воспроизводиться определенные периодические объявления в случайном порядке. См. `periodic-announce.`
`relative-periodic-announce`	`yes,no`	Если задано значение `yes`, таймер `periodic-announce-frequency` будет запускаться при достижении конца воспроизводимого файла, а не с самого начала. По умолчанию `no`
`announce-holdtime`	`yes,no,once`	Определяет, должно ли расчетное время ожидания воспроизводиться вместе с периодическими объявлениями. Можно установить значение `yes`, `no` или `once` (только один раз).
`announce-position`	`yes,no,limit,more`	Определяет, должна ли быть объявлена позиция вызывающего абонента в очереди. Если `no` - положение не будет объявлено. Если задано значение `yes`, позиция вызывающего абонента всегда будет объявлена. Если установлено значение `limit`, абонент будет слышать свое положение в очереди, только если оно находится в пределах лимита, определенного `announce-position-limit`. Если значение больше, то абонент будет слышать его позицию, только если он не входит в число определяемых `announce-position-limit`.
`announce-position-limit`	Число от нуля и выше	Используется, если вы определили announce-`position` как `limit` или `more`.
`announce-round-seconds`	Значение в секундах	Если это значение не равно нулю, объявляется также и число секунд, округленное до заданного значения.

Таблица 13-8 показывает файлы, которые будут использоваться при воспроизведении объявлений для абонента.

Таблица 13-8. Опции для управления воспроизведением подсказок в очереди

Опция	Возможные значения	Описание
`musicclass`	Класс музыки определенный в *musiconhold.conf*	Устанавливает класс музыки, который будет использоваться определенной очередью. Вы также можете переопределить это значение с помощью канальной переменной `CHANNEL(musicclass)`.
`queue-thankyou`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («Благодарим за терпение»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue-youarenext`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («Ваш звонок является первым в очереди и будет отвечен первым доступным оператором»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue-thereare`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («Ваша позиция в очереди»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue -callswaiting`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («Ожидайте, пожалуйста, ответа оператора»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue-holdtime`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («Прогнозируемое время ожидания составляет»).Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue-minutes`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («минут»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue-seconds`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («секунд»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`queue-reporthold`	Имя файла подсказки для воспроизведения	Если не определено, воспроизводится значение по умолчанию («Время ожидания»). Если установлено пустое значение, запрос не будет воспроизводиться вообще.
`periodic-announce`	Набор периодических объявлений для воспроизведения, разделенных запятыми	Подсказки воспроизводятся в порядке их определения. По умолчанию очередь-периодическое-объявление («Все наши операторы заняты, пожалуйста, оставайтесь на линии и Ваш звонок будет обслужен первым доступным оператором»).

Если количество параметров, предназначенных для воспроизведения объявлений для вызывающих абонентов, является показателем их важности, вероятно, в ваших же интересах использовать их в полной мере. Варианты в Таблице 13-7 помогут вам определить, когда воспроизводить объявления для абонентов, а варианты в Таблице 13-8 помогут контролировать то, что мы воспроизводим для наших абонентов. Имея эти таблицы давайте рассмотрим пример очереди, в которой мы определили некоторые значения. Мы будем использовать наш базовый шаблон очереди в качестве отправной точки:

```
[general]
autofill=yes      ; распределить всех ожидающих абонентов среди доступных
                  ; агентов
shared_lastcall=yes     ; уважайте время завершения для участников, вошедших в
                        ; более чем одну очередь

[StandardQueue](!)      ; шаблон для обеспечения общих функций
musicclass=default      ; воспроизводить музыку [default]
strategy=rrmemory       ; использовать стратегию Round Robin Memory
joinempty=yes           ; не присоединяться к очереди, когда нет доступных
```

```
                        ; участников
leavewhenempty=no       ; покинуть очередь, когда нет доступных участников
ringinuse=no            ; не звонить участникам, когда он InUse (предотвращает
                        ; многократные звонки оператору)

[sales](StandardQueue)  ; создать очередь sales используя параметры из шаблона
                        ; StandardQueue
[support](StandardQueue); создать очередь support используя параметры из шаблона
                        ; StandardQueue

Теперь мы изменим шаблон StardardQueue для управления нашими объявлениями:
[general]
autofill=yes      ; распределить всех ожидающих абонентов среди доступных
участников
shared_lastcall=yes     ; уважайте время завершения для участников, вошедших в
                        ; более чем одну очередь

[StandardQueue](!)      ; шаблон для обеспечения общих функций
musicclass=default      ; воспроизводить музыку [default]
strategy=rrmemory       ; использовать стратегию Round Robin Memory
joinempty=yes           ; не присоединяться к очереди, когда нет доступных
                        ; участников
leavewhenempty=no       ; покинуть очередь, когда нет доступных участников
ringinuse=no            ; не звонить участникам, когда он InUse (предотвращает
                        ; многократные звонки оператору)

; -------- Управление объявлениями --------
announce-frequency=30               ; объявляет время удержания и позицию
                                    ; звонящего каждые 30 секунд
min-announce-frequency=30           ; минимальное количество времени, которое
                                    ; должно пройти, прежде чем позиция
                                    ; вызывающего абонента будет объявлена
periodic-announce-frequency=45      ; определяет, как часто следует
                                    ; воспроизводить периодическое объявление
                                    ; для звонящего
random-periodic-announce=no         ; определяет, следует ли воспроизводить
                                    ; периодические объявления в случайном
                                    ; порядке или по порядку
relative-periodic-announce=yes      ; определяет, начинается ли таймер в
                                    ; конце воспроизведение файла (yes) или
                                    ; начале (no)
announce-holdtime=once              ; определяет, должно ли предполагаемое
                                    ; время удержания играться вместе с
                                    ; периодическим объявлением
announce-position=limit             ; определяет, должны ли мы объявить
                                    ; позицию звонящего в очереди
announce-position-limit=10          ; определяет предельное значение, в
                                    ; пределах которого мы объявляем позицию
                                    ; абонента (когда announce-position
                                    ; установлено на limit или more)
announce-round-seconds=30           ; округляет объявление времени удержания
                                    ; до ближайшего 30-секундного значения
```

Давайте опишем, что мы только что установили в нашем шаблоне `StandardQueue`.

Мы будем сообщать время удержания и позицию вызывающего абонента каждые 30 секунд (`announce-frequency`)[8] и убедитесь что минимальное количество времени, которое проходит до

8 Позиция и время удержания абонентов объявляются только в том случае, если в очереди находится более одного человека.

того, как мы объявим его снова, составляет не менее 30 секунд (`min-announce-frequency`).[9] Нам нужно ограничить частоту воспроизведения наших объявлений для вызывающих абонентов чтобы не раздражать их. Периодически мы будем воспроизводить звонящим объявление, в котором поблагодарим их за ожидание[10], и уверим что агент будет с ними в ближайшее время. (Объявление определяется настройкой `periodic-announcement`. Мы используем объявление по умолчанию, но вы можете определить одно или несколько объявлений самостоятельно, используя `periodic-announce`.)

Эти периодические объявления будут воспроизводиться каждые 45 секунд (`periodic-announce-frequency`) в порядке их определения (`random-periodic-announce`). Чтобы определить когда таймер `periodic-announce-frequency` должен запуститься, мы используем `relative-periodic-announce`. Настройка `yes` означает, что таймер запустится после окончания объявления, а не в начале. Проблема, с которой вы можете столкнуться, заключается в том, что если вы установите для него значение `no`, то если ваше периодическое объявление запускается в течение какого-либо значительного промежутка времени, например 30 секунд, то будет выглядеть так, будто оно играет каждые 15 секунд, а не 45, как на это может быть расчитано. Кроме того, если вы нашли какую-то полуприличную музыку на удержание, и ваши абоненты наслаждаются ею, прерывание для воспроизведения еще одного сообщения может привести к тому, что у них действительно закипит кровь. Когда им наконец ответят, ваши бедные агенты получат основную тяжесть их гнева, хотя на самом деле это ваша вина.[11]

Сколько раз мы объявляем время удержания для вызывающего абонента, контролируется с помощью параметра `announce-holdtime`, значение которого мы установили `once`. Установка значения `yes` будет объявлять время каждый раз, а установка на `no` отключит его.

Мы настраиваем, как и когда объявлять расчетное оставшееся время удержания звонящего через `announce-position`, который мы установили для ограничения. Использование значения `limit` для `announce-position` позволяет нам объявлять позицию вызывающего абонента, только если она находится в пределах, определенных в `announce-position-limit`. Таким образом, в этом случае мы объявляем позиции абонентов только если они находятся в первых 10 позициях очереди. Мы также могли бы использовать `yes`, чтобы объявлять позицию каждый раз, когда воспроизводится периодическое объявление; установив для нее значение `no` мы укажем никогда не объявлять ее или используя значение `more` укажем объявлять позицию только тогда, когда она превышает значение, установленное для `announce-position-limit`.

Наш последний параметр `announce-round-seconds` контролирует значение, к которому округляется объявляемое время удержания вызывающего абонента. В этом случае вместо того, чтобы сказать «1 минута и 23 секунды», значение будет округлено до ближайшего 30-секундного значения, что приведет к приглашению «1 минута и 30 секунд».

Переполнение

Переполнение очереди выполняется либо со значением тайм-аута, либо когда нет доступных участников очереди (как определено `joinempty` или `leftwhenempty`). В этом разделе мы обсудим, как контролировать переполнение.

9 Обратите внимание, что в продакшене может потребоваться установить значение более 30 секунд, особенно если среднее время удержания таково, что абонент может услышать это приглашение более одного раза во время удержания. Установите такое значение, чтобы большинство абонентов никогда не слышали это приглашение более двух раз, прежде чем им ответят.

10 Спасибо им, но не извиняйтесь, так как это звучит просто неискренне, так как по сути они извиняются перед машиной, и они это знают. Основная цель этих подсказок состоит в том, чтобы помочь абоненту понять, что система все еще держит их в очереди.

11 Просто говорю

Управление тайм-аутами

Приложение `Queue()` поддерживает два вида тайм-аутов: один для максимального периода времени, в течение которого вызывающий абонент находится в очереди, а другой - как долго звонить устройству при попытке соединить вызывающего абонента с участником очереди. Они не связаны, но могут влиять друг на друга. В этом разделе мы поговорим о максимальном периоде времени, в течение которого вызывающий абонент остается в приложении `Queue()` до того, как вызов переполнится, до следующего шага в диалплане, который может быть чем-то вроде `VoiceMail()` или даже другой очередью. После того, как вызов выпал из очереди, он может отправиться куда угодно, куда обычно может идти вызов, управляемый диалпланом.

Таймауты задаются в двух местах. Время ожидания, указывающее, как долго звонить участникам очереди, указывается в файле *queues.conf*. Абсолютное время ожидания (как долго абонент остается в очереди) управляется через приложение `Queue()`. Чтобы задать максимальное время пребывания абонентов в очереди, просто укажите его после имени очереди в приложении `Queue()`:

```
[Queues]
exten => 7000,1,Verbose(2,Joining the support queue for a maximum of 2
minutes)
   same => n,Queue(support,120)
   same => n,VoiceMail(support@queues,u)
   same => n,Hangup()
```

Конечно, мы можем определить другое назначение, но приложение `VoiceMail()` является общим местом назначения переполнения для очереди. Очевидно, что отправка звонков на голосовую почту не идеальна (они надеялись поговорить с кем-то вживую) поэтому убедитесь, что кто-то регулярно проверяет её и перезванивает вашим клиентам.

Теперь предположим, что мы установили наше абсолютное время ожидания равным 10 секундам, наше значение времени ожидания для звонков участникам очереди равным 5 секундам, а наше значение времени ожидания повторных попыток - 4 секунды. В этом случае мы будем звонить участнику очереди в течение 5 секунд, а затем ждать 4 секунды, прежде чем пытаться набрать другому участнику очереди. Это дает нам до 9 секунд нашего абсолютного тайм-аута в 10 секунд. В этот момент мы должны позвонить второму участнику очереди на 1 секунду и затем выйти из очереди, или мы должны позвонить этому участнику в течение полных 5 секунд перед выходом?

Мы контролируем, какое значение тайм-аута имеет приоритет с помощью параметра в `timeoutpriority` в *queues.conf*. Доступные значения `app` и `conf`. Если мы хотим, чтобы тайм-аут приложения (абсолютный тайм-аут) имел приоритет, что приведет к тому, что наш вызывающий будет выгнан через ровно 10 секунд, мы должны установить значение `timeoutpriority` в `app`. Если мы хотим, чтобы таймаут файла конфигурации имел приоритет и закончил звонить участнику очереди, что заставит вызывающего абонента оставаться в очереди немного дольше, мы должны установить `timeoutpriority` в `conf`. Значением по умолчанию является `app` (это поведение по умолчанию из предыдущих версий Asterisk).

Контроль времени присоединения и выхода из очереди

Asterisk предоставляет две опции, которые контролируют, когда вызывающие абоненты могут присоединиться и вынуждены покинуть очередь, в зависимости от статусов участников очереди. Первая опция - `joinempty`, используется для контроля могут ли абоненты входить в очередь. Опция `leftwhenempty` используется для управления тем, когда вызывающие абоненты, уже находящиеся в очереди, должны быть удалены из этой очереди (то есть, если все участники очереди становятся недоступными). Обе опции принимают список значений через запятую, которые управляют этим поведением. Факторы перечислены в Таблице 13-9.

Таблица 13-9. Опции, которые могут быть установлены для joinempty или leavewhenempty

Значение	Описание
paused	Пользователи считаются недоступными, если они приостановлены.
penalty	Участники считаются недоступными, если их штрафы меньше QUEUE_MAX_PENALTY.
inuse	Участники считаются недоступными, если их состояние устройства In Use.
ringing	Участники считаются недоступными, если их статус устройства Ringing.
unavailable	Относится в первую очередь к агентским каналам; если агент не вошел в систему, но является участником очереди, канал считается недоступным.
invalid	Участники считаются недоступными, если статус их устройства Invalid. Обычно это условие ошибки.
unknown	Участники считаются недоступными, если статус устройства неизвестен.
wrapup	Участники считаются недоступными, если они в настоящее время находятся во времени отдыха после завершения вызова.

Для joinempty прежде чем поместить вызывающего абонента в очередь, все участники проверяются на доступностьс помощью перечисленных в качестве критериев факторов. Если все участники считаются недоступными, абоненту не будет разрешен вход в очередь, и диалплан будет продолжен со следующего приоритета.[12] Для опции leavewhenempty статусы участников периодически проверяются условиями, перечисленными выше; если будет установлено, что участники готовы принять звонки, абонент удаляется из очереди, выполнение диалплана продолжится со следующего приоритета.

Примером использования joinempty может быть:

```
joinempty=paused,inuse,invalid
```

При такой конфигурации, до входа абонента в очередь, статусы всех участников очереди будут проверены и вызывающему абоненту не будет разрешено войти в очередь, если не будет найден по крайней мере один участник очереди не имеющий статуса paused,inuse,invalid.

Примером leavewhenempty может быть что-то вроде:

```
leavewhenempty=inuse,ringing
```

В этом случае статусы участников очереди будут периодически проверяться, а вызывающие абоненты будут удаляться из очереди, если не будет найдено ни одного участника очереди, без статуса inuse или ringing.

Предыдущие версии Asterisk использовали значения yes, no, strict и loose в качестве назначаемых значений. Сопоставление этих значений показано в Таблице 13-10.

Таблица 13-10. Сопоставление старых и новых значений для контроля входа и выхода абонентов из очередей

Значение	Сопоставление (joinempty)	Сопоставление (leavewhenempty)
yes	(пусто)	penalty,paused,invalid
no	penalty,paused,invalid	(пусто)
strict	penalty,paused,invalid,unavailable	penalty,paused,invalid,unavailable
loose	penalty,invalid	penalty,invalid

Использование локальных (Local) каналов

Использование локальных каналов в качестве участников очереди является мощным способом выполнения кода диалплана перед набором фактического номера агента. Например, он позволяет нам

12 Если приоритет n + 1 (из которого вызывалось приложение Queue()) не определен, вызов будет прерван. Другими словами, не используйте эту функцию, если ваш диалплан не делает что-то полезное на шаге, следующем сразу за Queue().

определять пользовательские переменные канала, записывать в файл журнала, устанавливать некоторое ограничение на длину вызова (например, если это платная услуга), отправлять сообщения всех видов повсеместно, выполнять транзакции базы данных и многие другие действия, которые мы могли бы выполнить в тот момент, когда приложение `Queue()` решило предоставить вызывающего абонента конкретному участнику (включая возврат `Congestion()` что приведет к возврату абонента в очередь, поскольку очередь не будет считать этот вызов успешно доставленным агенту).

При использовании локальных каналов для очередей они добавляются точно так же, как и любые другие каналы, как правило, либо определяя их в файле *queues.conf*, либо добавляя их динамически через приложение `AddQueueMember()`.

Если вы хотите определить их в файле *queues.conf*, добавление локального канала будет выглядеть примерно так:

```
; queues.conf
[support](StandardQueue)
member => Local/SIP-0000FFFF0001@MemberConnector      ; передайте технологию
                                                      ; дозвона и идентификатор устройства,
                                                      ; разделенные дефисом. Мы разобьем его
                                                      ; внутри контекста MemberConnector.
```



Обратите внимание, как мы передали тип технологии, которую хотим вызвать вместе с идентификатором устройства, в контекст `MemberConnector`. Мы просто использовали дефис (хотя мы могли бы использовать почти все в качестве аргумента-разделителя) в качестве маркера поля. Мы будем использовать функцию `CUT()` внутри контекста `MemberConnector` и назначим первое поле (`SIP`) одному каналу, переменную и второе поле (`0000FFFF0001`) - другой переменной канала, которая затем будет использоваться для вызова конечной точки.

Передача информации для последующего «разбиения» в контексте, используемом локальным каналом, является распространенным и полезным методом (по типу функции `explode()` в PHP).

Поскольку наш локальный канал ссылается на контекст `MemberConnector`, нам понадобится этот контекст в нашем диалплане для того, чтобы он содержал сведения, необходимые для фактического подключения вызывающего абонента к физическому каналу, который использует агент (в идеале какой-нибудь SIP-телефон):

```
[MemberConnector]
exten => _[A-Za-z0-9].,1,Verbose(2,Connect ${CALLERID(all)} to Agent at $
{EXTEN})

   ; фильтруем любые неверные символы, допустимы только цифры, буквы и дефис
   same => n,Set(QueueMember=${FILTER(A-Za-z0-9\-,${EXTEN})})

   ; назначить первое поле QueueMember - Technology; дефис как разделитель
   same => n,Set(Technology=${CUT(QueueMember,-,1)})

   ; назначить второе поле QueueMember - Device используя дефис как
разделитель
   same => n,Set(Device=${CUT(QueueMember,-,2)})

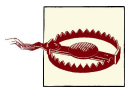
   ; вызываем агента
   same => n,Dial(${Technology}/${Device})
   same => n,Hangup()
```

Теперь у нас есть весь код, необходимый для передачи вызова очереди оператору через локальный канал. Однако, поскольку участник очереди является локальным каналом, а не каналом SIP, `Queue()` не обязательно будет знать состояние, в котором находится вызов, особенно когда локальный канал оптимизирован вне пути [см. Asterisk wiki для получения информации о модификаторе `/n`, который

приводит к тому, что локальный канал не будет оптимизирован вне пути]. Очередь будет контролировать состояние локального канала, а не устройства, которое мы действительно хотим отслеживать; таким образом, фактическое состояние канала, который использует агент, не будет известно очереди.

К счастью, мы можем предоставить `Queue()` реальное физическое устройство для мониторинга и связать его с локальным каналом, чтобы состояние участника очереди всегда соответствовало состоянию устройства, которое мы в конечном итоге вызываем. Наш участник очереди должен быть изменен в файле `queues.conf` следующим образом:

```
; queues.conf
[support](StandardQueue)
member => Local/SIP-0000FFFF0001@MemberConnector,,,SIP/0000FFFF0001
```



Только SIP-каналы способны отправлять обратно достоверную информацию о состоянии устройства, поэтому настоятельно рекомендуется использовать только эти каналы для локальных каналов в качестве участников очереди или, по крайней мере, знать, что информация о состоянии может быть ненадежной для других типов канальных технологий (таких как DAHDI, IAX2 и т.д.).

Помимо определения участников в файле *queues.conf*, вы также можете использовать приложения `AddQueueMember()` и `RemoveQueueMember()` для добавления и удаления локального канала участника из очереди, используя код диалплана (как вы можете с любым типом канала). `AddQueueMember()` также имеет возможность устанавливать состояния интерфейса, аналогично тому, что мы определили статически в файле *queues.conf*. Ниже приведен пример того, как вы можете это сделать:

```
[QueueMemberLogin]
exten => 500,1,Verbose(2,Logging in device ${CHANNEL(peername)} into the
support queue)

   ; Сохраняем технологию устройства в переменную канала MemberTech
   same => n,Set(MemberTech=${CHANNEL(channeltype)})

   ; Сохраняем идентификатор устройства в переменную канала MemberIdent
   same => n,Set(MemberIdent=${CHANNEL(peername)})

   ; Создаем имя интерфейса из типа канала и имени пира
   ; и назначаем его переменной канала Interface
   same => n,Set(Interface=${MemberTech}/${MemberIdent})

   ; Добавляем участника в очередь support, используя канал Local. Мы
   ; используем тот же формат, что и раньше, разделяя технологию и
   ; идентификатор устройства дефисом и передавая эту информацию в контекст
   ; MemberConnector.
   ; Затем мы используем функцию IF(), чтобы определить, является ли
   ; технология участника SIP и если да, то передаем содержимое переменной
   ; канала Interface в качестве значения в поле интерфейса состояния
   ; приложения AddQueueMember().
   ;
   ; При отладке может потребоваться жесткий код AddQueueMember с
   ; одним указанным интерфейсом, чтобы получить представление о синтаксисе,
   ; например:
   ; **** В этой строке не должно быть разрывов
   same => n,AddQueueMember(support,Local/SIP-0000FFFF0001
@MemberConnector,,,SIP/0000FFFF0001)
   ; однако ваш диалплан должен быть закодирован для обработки всех каналов и
   ; участников, как здесь:
   ; **** В этой строке не должно быть разрывов
```

```
   same => n,AddQueueMember(support,Local/${MemberTech}-${MemberIdent}
@MemberConnector,,,${IF($[${MemberTech} = SIP]?${Interface})})
   same => n,Playback(silence/1)
   ; Воспроизведение файла agent-loginok или agent-incorrect в зависимости от
   ; значения переменной AQMSTATUS.
   same => n,Playback(${IF($[${AQMSTATUS} = ADDED]?agent-loginok:agent-
incorrect)})
   same => n,Hangup()
```

Теперь, когда мы можем добавлять устройства в очередь, используя каналы Local, давайте посмотрим где это может быть полезно. В нашем примере мы собираемся контролировать количество вызовов на не-SIP-каналы или SIP-устройства с более чем одной линией (что типично для стандартного SIP-телефона). Мы можем использовать функции `GROUP()` и `GROUP_COUNT()` для отслеживания количества вызовов к конечной точке. Мы изменим наш контекст `MemberConnector`, чтобы принять это во внимание:

```
[MemberConnector]
exten => _[A-Za-z0-9].,1,Verbose(2,Connect ${CALLERID(all)} to Agent at $
{EXTEN})

   ; фильтруем любые неверные символы, допустимы только цифры, буквы и дефис
   same => n,Set(QueueMember=${FILTER(A-Za-z0-9\-,${EXTEN})})

   ; назначить первое поле QueueMember - Technology; дефис как разделитель
   same => n,Set(Technology=${CUT(QueueMember,-,1)})

   ; назначить второе поле QueueMember - Device используя дефис как
разделитель
   same => n,Set(Device=${CUT(QueueMember,-,2)})

   ; Увеличить значение группы внутри категории queue_members на 1
   same => n,Set(GROUP(queue_members)=${Technology}-${Device})
   ; Проверить если group@category больше 1, то вернуть Congestion()
   ; (слишком много каналов)
   ;
   ; **** В этой строке не должно быть разрывов
   same => n,ExecIf($[${GROUP_COUNT(${Technology}-${Device}@queue_members)} >
1]?Congestion())

   ; вызываем агента
   same => n,Dial(${Technology}/${Device})
   same => n,Hangup()
```

Возврат `Congestion()` приведет к тому, что вызывающий будет вернут в очередь (пока это происходит) звонящий не получает никаких признаков того, что что-то не так, и продолжает слушать музыку, пока на его звонок не ответит какой-то канал[13]). В идеале ваша очередь запрограммирована на попытку вызова другого агента; однако вы должны иметь в виду, что если `app_queue` определяет, что этот участник по-прежнему является первым для вызова, то вызов просто будет повторно направлен к тому же агенту (и снова получит перегрузку, и, таким образом, потенциально может создать цикл с перегрузкой ЦП). Чтобы избежать этого, вам нужно убедиться, что ваша очередь использует стратегию распределения, такую как `round_robin`, `random` или любую другую, которая гарантирует, что один и тот же участник не будет пробоваться снова и снова.

Мы также использовали этот метод для создания типа процесса бронирования. Если вы хотите вызвать агента напрямую (например, если вызывающему абоненту необходимо связаться с конкретным агентом), вы можете зарезервировать этого агента с помощью функций `GROUP()` и

13 Очевидно, что в локальном канале не стоит использовать код диалплана, который будет отвечать на вызов, например `Answer()`, `Playback()` и так далее.

GROUP_COUNT(), чтобы приостановить работу агента в очереди до тех пор, пока вызывающий не сможет подключиться. Это особенно полезно в ситуациях, когда вам необходимо воспроизвести какое-либо объявление для абонента до его соединения с агентом, но не хотите, чтобы агент подключался к другому вызывающему абоненту во время воспроизведения объявления.

Использование каналов Local для каналов участников не облегчит проектирование и отладку очереди, но даст вам гораздо больше возможностей для ваших очередей, чем простое использование app_queue, поэтому, если у вас сложные требования к очереди, использование каналов Local может дать вам уровень контроля, который вы иначе и не могли бы не иметь.

Статистика очереди: файл queue_log

Файл *queue_log* (обычно расположенный в */var/log/asterisk*) содержит накопительную информацию о событиях очередей, определенных в вашей системе (когда очередь перезагружается, когда добавляются участники очереди или удаляются и т.д.), а также некоторые детали вызова (например, их статус и к каким каналам были подключены абоненты). Журнал очереди включен по умолчанию, но им можно управлять через файл *logger.conf*. Существует три параметра, относящихся к файлу *queue_log*, а именно:

queue_log

Включен ли журнал очереди или нет. Допустимые значения: yes или no (по умолчанию yes).

queue_log_to_file

Определяет, должен ли журнал очереди записываться в файл, даже если присутствует серверная часть Realtime. Допустимые значения: yes или no (по умолчанию - no).

queue_log_name

Управляет именем журнала очереди. По умолчанию это queue_log.

Журнал очереди представляет собой разделенный на каналы список событий. Поля в файле *queue_log* следующие:

- Отметка времени события
- Уникальный идентификатор вызова
- Имя очереди
- Наименование мостового канала
- Тип события
- Ноль или более параметров события

Информация, содержащаяся в параметрах события, зависит от типа события. Типичный *queue_log* будет выглядеть примерно так:

```
1292281046|psy1-1292281041.87|7100|NONE|ENTERQUEUE||4165551212|1
1292281046|psy1-1292281041.87|7100|Local/9996@MemberConnector|RINGNOANSWER|0
1292281048|psy1-1292281041.87|7100|Local/9990@MemberConnector|CONNECT|2
|psy1-1292281046.90|0

1292284121|psy1-1292281041.87|7100|Local/9990@MemberCo|COMPLETECALLER|2|3073|1
1292284222|MANAGER|7100|Local/9990@MemberConnector|REMOVEMEMBER|
1292284222|MANAGER|7200|Local/9990@MemberConnector|REMOVEMEMBER|
1292284491|MANAGER|7100|Local/9990@MemberConnector|ADDMEMBER|
1292284491|MANAGER|7200|Local/9990@MemberConnector|ADDMEMBER|
1292284519|psy1-1292284515.93|7100|NONE|ENTERQUEUE||4165551212|1
1292284519|psy1-1292284515.93|7100|Local/9996@MemberConnector|RINGNOANSWER|0
1292284521|psy1-1292284515.93|7100|Local/9990@MemberConnector|CONNECT|2
|psy1-1292284519.96|0

1292284552|MANAGER|7100|Local/9990@MemberConnector|REMOVEMEMBER|
```

```
1292284552|MANAGER|7200|Local/9990@MemberConnector|REMOVEMEMBER|
1292284562|psy1-1292284515.93|7100|Local/9990@MemberCo|COMPLETECALLER|2|41|1
```

Как видно из этого примера, не всегда может быть уникальный идентификатор для события. Внешние сервисы, такие как Asterisk Manager Interface (AMI), могут выполнять действия в очереди и в этих случаях вы увидите что-то вроде MANAGER в поле Уникальный идентификатор.

Доступные события и информация, которую они предоставляют, описаны в Таблице 13-11.

Таблица 13-11. События в журнале очереди Asterisk

Событие	Предоставляемая информация
ABANDON	Записывается, когда абонент в очереди вешает трубку, прежде чем на его вызов отвечает агент. Три параметра предназначены для отказа: позиция звонящего на момент отбоя, исходное положение абонента при поступлении в очередь, и количество времени, которое абонент ожидал до отбоя.
ADDMEMBER	Записывается при добавлении участника в очередь. Имя канала моста будет заполнено именем канала, добавленного в очередь.
AGENTDUMP	Указывает, что агент сбросил вызов абонента во время воспроизведения объявления очереди, до того как они были соединены вместе.
AGENTLOGIN	Записывается при входе агента в систему. Поле канала моста будет содержать что-то вроде `Agent/9994` при входе с помощью `chan_agent`, а первое поле параметра будет содержать вход канала (например, `SIP/0000FFFF0001`).
AGENTLOGOFF	Регистрируется при выходе агента из системы вместе с параметром, указывающим, как долго агент был в системе. Обратите внимание, что поскольку `RemoveQueueMember()` часто используется для выхода агента из системы, этот параметр может не записываться. Смотрите `REMOVEMEMBER` вместо этого.
COMPLETEAGENT	Записывается, когда вызов соединен мостом с агентом, и агент вешает трубку, наряду с параметрами, указывающими количество времени, в течение которого абонент удерживался в очереди, длину вызова с агентом и исходной позицией, с которой абонент вошел в очередь.
COMPLETECALLER	То же самое, что и `COMPLETEAGENT`, за исключением того, что звонивший повесил трубку, а не агент.
CONFIGURELOAD	Указывает, что конфигурация очереди была перезагружена (например, через *module reload app_queue.so*).
CONNECT	Записывается когда абонент и агент были соединены. Также записываются три параметра: время ожидания абонента в очереди, уникальный идентификатор канала участника очереди, к которому абонент был присоединен, и время, в течении которого звонил телефон участника очереди до того, как он ответил.
ENTERQUEUE	Записывается, когда абонент входит в очередь. Также записываются два параметра: URL (если указан) и CallerID абонента.
EXITEMPTY	Записывается, когда абонент удаляется из очереди из-за отсутствия доступных агентов для ответа на вызов (как указано в параметре `leavewhenempty`). Также записываются три параметра: позиция абонента в очереди, исходная позиция, с которой абонент вошел в очередь, и время нахождения абонента в очереди.
EXITWITHKEY	Записывается, когда абонент выходит из очереди, нажав одну клавишу DTMF на своем телефоне, чтобы выйти из очереди и продолжить в диалплане (как разрешено параметром `context` в *queues.conf*). Записываются четыре параметра: клавиша, используемая для выхода из очереди, позиция абонента в очереди при выходе, исходная позиция абонента с которой он поступил в очередь и время ожидания абонентом в очереди.
EXITWITHTIMEOUT	Записывается, когда абонент удаляется из очереди из-за тайм-аута (как указано в параметре `timeout` для `Queue()`). Записываются также три параметра: положение абонента при выходе из очереди, исходное положение абонента при входе в очередь и время ожидания абонента в очереди.
PAUSE	Записывается при приостановке участника очереди.
PAUSEALL	Записывается при приостановке всех участников очереди.

Событие	Предоставляемая информация
UNPAUSE	Записывается при возобновлении участника очереди.
UNPAUSEALL	Записывается при возобновлении всех участников очереди.
PENALTY	Записывается при изменении штрафа участника. Штраф можно изменить несколькими способами, такими как функция QUEUE_MEMBER_PENALTY(), AMI или Asterisk CLI.
REMOVEMEMBER	Записывается, когда участник очереди удаляется из очереди. Поле канал моста будет содержать имя участника, удаленного из очереди.
RINGNOANSWER	Регистрируется, когда участник очереди вызывается в течение определенного периода времени и превышено значение времени ожидания для вызова участника очереди. Также будет записан один параметр, указывающий время, в течение которого звонил добавочный номер участника.
TRANSFER	Записывается при переводе абонента на другой внутренний номер. Дополнительные параметры, которые также записываются: расширение и контекст, в который была переведен абонент, время удержания абонента в очереди, количество времени, в течении которого абонент разговаривал с участником очереди, и исходное положение абонента, когда он вошел в очередь.[a]
SYSCOMPAT	Записывается, если агент пытается ответить на вызов, но вызов не может быть настроен из-за несовместимости в настройке медианосителя.

[a] Обратите внимание, что при передаче вызывающего абонента с использованием SIP-трансфера (а не встроенного трансфера, запускаемого через DTMF и настраиваемого в файле *functions.conf*), событие TRANSFER может не записываться.

Вывод

Мы начали эту главу с рассмотрения простых очередей вызовов, обсуждения того, что они из себя представляют, как они работают и когда вы, возможно, захотите их использовать. После построения простой очереди мы изучили как управлять участниками очереди с помощью различных средств (включая использование каналов Local, которые обеспечивают возможность выполнения некоторой логики диалплана непосредственно перед подключением к участнику очереди). Мы также изучили все доступные нам опции в файлах *queues.conf*, *agents.conf* и *queuerules.conf*, которые предоставляют нам детальный контроль над любыми настраиваемыми очередями. Конечно, нам нужна возможность отслеживать, что делают наши очереди, поэтому мы быстро просмотрели файл журнала очереди и различные поля, записанные в результате событий, происходящих в наших очередях.

Благодаря знаниям, приведенным в этой главе, вы должны быть на пути к реализации успешного набора очередей для вашей компании.

Состояния устройств

Из беспорядка найти простоту.
-Альберт Эйнштейн

Часто бывает полезно определить состояние устройств, подключенных к телефонной системе. Например, у администратора может потребоваться возможность видеть статусы каждого в офисе, чтобы определить, может ли кто-нибудь позвонить по телефону. Сам Asterisk нуждается в такой же информации. В качестве другого примера, если вы построили очередь вызовов, как описано в Главе 13, Asterisk должен знать, когда агент доступен, чтобы можно было отправить другой вызов. В этой главе обсуждаются концепции состояния устройств в Asterisk, а также способы использования и доступа к этим устройствам и приложениям.

Состояния устройств

Существует два типа устройств, к которым относятся состояния устройств: реальные устройства и виртуальные. Реальные устройства - это конечные точки телефонии, которые могут совершать или принимать вызовы, такие как SIP-телефоны. Виртуальные устройства включают объекты, которые находятся внутри Asterisk, но предоставляют полезную информацию о состоянии. Таблица 14-1 содержит список доступных виртуальных устройств в Asterisk.

Таблица 14-1. Виртуальные устройства в Asterisk

Виртуальное устройство	Описание
`MeetMe:<conference bridge>`	Состояние конференц-моста MeetMe. Статус будет отражать, есть ли в настоящее время участники в конференции. Более подробную информацию об использовании `MeetMe()` для конференц-связи можно найти в разделе “Конференц-связь с MeetMe()” в Главе 10.
`SLA:<shared line>`	Общие сведения о состоянии внешней линии. Это состояние управляется приложениями `Trunk()` и `SLAStation()`. Более подробную информацию можно найти в разделе “Общие внешние линии”.
`Custom:<custom name>`	Пользовательские состояния устройств. Эти состояния имеют пользовательские имена и изменяются с помощью функции `DEVICE_STATE()`. Пример использования можно найти в разделе "Использование пользовательских состояний устройств".
`Park:<exten@context>`	Состояние парковочного места. Сведения о состоянии будут отражать, припаркован ли абонент в данный момент на этом дополнительном номере. Более подробную информацию о парковке вызовов в Asterisk можно найти в разделе “Парковочные места” в Главе 11.
`Calendar:<calendar name>`	Состояние календаря. Asterisk будет использовать содержимое именованного календаря для установки состояния `available` или `busy`. Более подробную информацию об интеграции календаря в Asterisk можно найти в Главе 18.

Состояние устройства - это простое однозначное сопоставление с устройством. Рисунок 14-1 показывает это сопоставление.

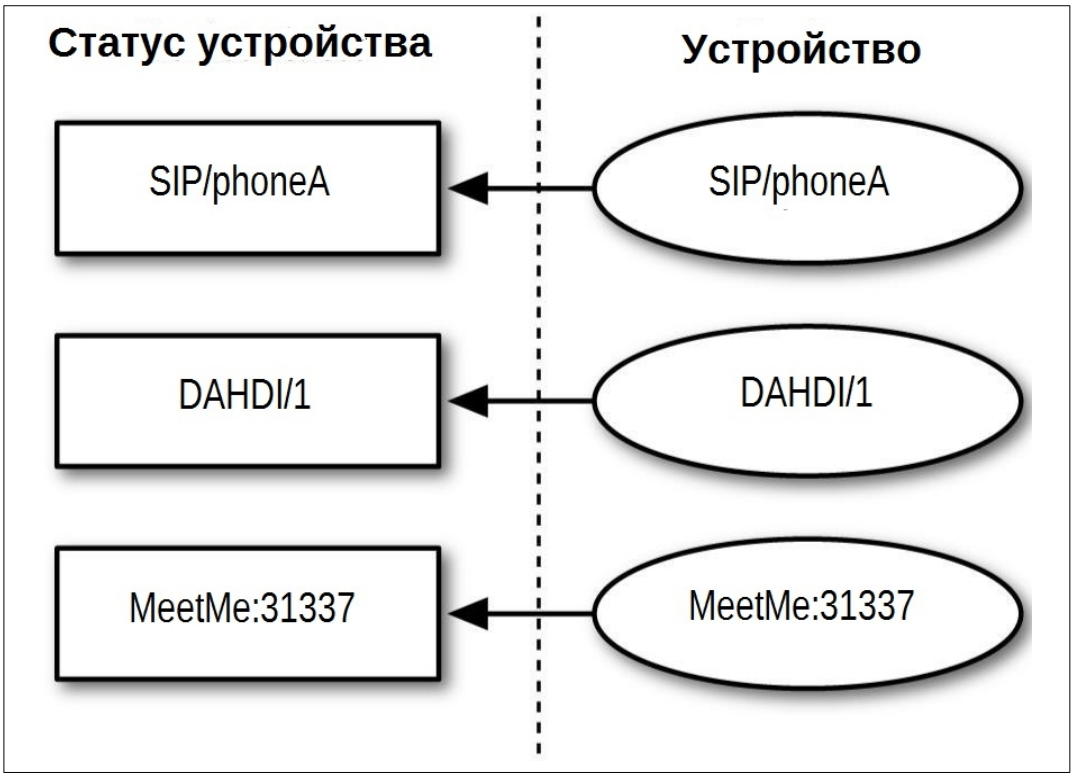


Рисунок 14-1. Отображение состояний устройства.

Проверка состояния устройства

Функция диалплана `DEVICE_STATE()` может использоваться для чтения текущего состояния устройства.

Вот простой пример её использования в диалплане:

```
exten => 7012,1,Answer()

; *** Эта строка не должна иметь разрывов
   same => n,Verbose(3,The state of SIP/0004F2060EB4 is
${DEVICE_STATE(SIP/0004F2060EB4)})
   same => n,Hangup()
```

Если мы вызываем номер 7012 из того же устройства, на котором мы проверяем состояние, на консоли Asterisk появляется следующее подробное сообщение:

```
-- The state of SIP/0004F2060EB4 is INUSE
```



Глава 20 обсуждает Asterisk Manager Interface (AMI). Действие диспетчера `GetVar` может использоваться для извлечения значений состояний устройства во внешнюю программу. Вы можете использовать его для получения значения как нормальной переменной, так и функции диалплана, такой как `DEVICE_STATE()`.

Следующий список содержит возможные значения, которые возвращаются из функции DEVICE_STATE():

- `UNKNOWN`
- `NOT_INUSE`
- `INUSE`
- `BUSY`

- INVALID
- UNAVAILABLE
- RINGING
- RINGINUSE
- ONHOLD

Статусы внутренних номеров

Внутренние номера являются еще одним важным понятием в Asterisk. Статусы внутренних номеров - это то, на что подписываются SIP-устройства для получения информации о присутствии. (SIP-присутствие более подробно обсуждается в разделе "SIP-присутствие"). Состояние внутреннего номера определяется путем проверки состояния одного или нескольких устройств. Список устройств, отображающих состояния внутр.номеров, определен в диалплане Asterisk */etc/asterisk/extensions.conf* с использованием специальной директивы `hint`. На Рисунке 14-2 показано сопоставление между устройствами, состояниями устройств и состояниями внутренних номеров.



Рисунок 14-2. Сопоставления состояний внутренних номеров.

Хинты

Чтобы определить хинт (подсказку) состояния номера в диалплане, используется ключевое слово hint вместо приоритета. Вот простой пример диалплана, который относится к Рисунку 14-2:

```
[default]

exten => 1234,hint,SIP/phoneA&SIP/phoneB&SIP/phoneC

exten => 5555,hint,DAHDI/1

exten => 31337,hint,MeetMe:31337
```

Обычно хинты просто определяются вместе с остальной частью внутреннего номера. В следующем примере добавляются простые записи номеров для отслеживания событий при вызове каждого из этих номеров:

```
[default]

exten => 1234,hint,SIP/phoneA&SIP/phoneB&SIP/phoneC
exten => 1234,1,Dial(SIP/phoneA&SIP/phoneB&SIP/phoneC)

exten => 5555,hint,DAHDI/1
exten => 5555,1,Dial(DAHDI/1)

exten => 31337,hint,MeetMe:31337
exten => 31337,1,MeetMe(31337,dM)
```

В нашем примере мы установили прямую корреляцию между добавочным номером хинта и набираемым добавочным номером, хотя нет необходимости в том, чтобы это было так.

Проверка статуса внутреннего номера

Самый простой способ проверить текущий статус номера - CLI Asterisk. Команда *core show hints* покажет вам все сконфигурированные хинты. Рассмотрим следующее определение хинта:

```
[phones]

exten => 7001,hint,SIP/0004F2060EB4
```

Когда core show hints выполняются в CLI Asterisk, при использовании устройства отображается следующий вывод:

```
*CLI> core show hints

   -= Registered Asterisk Dial Plan Hints =-
    7001@phones : SIP/0004F2060EB4 State:InUse Watchers 0
----------------
- 1 hints registered
```

В дополнение к отображению статуса внутреннего номера, вывод *core show hints* также обеспечивает количество наблюдателей. *Watcher (наблюдатель)* - это кто-то в Asterisk, кто подписался на получение обновлений о статусе этого номера. Если SIP-телефон подписывается на состояние номера, счетчик будет увеличен.

Состояние внутренних номеров также можно получить с помощью функции диалплана `EXTENSION_STATE()`. Эта функция работает так же, как функция `DEVICE_STATE()`, описанная в предыдущем разделе. В следующем примере показан номер, который будет печатать текущее состояние другого номера в консоли Asterisk:

```
exten => 7013,1,Answer()
   same => n,Verbose(3,The state of 7001@phones is
${EXTENSION_STATE(7001@phones)})
   same => n,Hangup()
```

Когда этот номер вызывается, это подробное сообщение, отображается в консоли Asterisk:

```
-- The state of 7001@phones is INUSE
```

Следующий список включает возможные значения, которые могут быть возвращены функцией `EXTENSION_STATE()`:

- `UNKNOWN`

- NOT_INUSE
- INUSE
- BUSY
- UNAVAILABLE
- RINGING
- RINGINUSE
- HOLDINUSE
- ONHOLD

SIP-присутствие

Asterisk предоставляет устройствам возможность подписываться на состояние номеров с использованием SIP-протокола. Эта функция часто упоминается как BLF (Busy Lamp Field).[1]

Конфигурация Asterisk

Чтобы сделать эту работу, хинты должны быть определены в */etc/asterisk/extensions.conf* (см. "Хинты" для большей информации о настройке хинтов в диалплане). Дополнительно, некоторые важные параметры должны быть установлены в файле конфигурации для драйвера канала SIP, тоесть в */etc/asterisk/sip.conf*. В следующем списке обсуждаются такие параметры:

`callcounter`

Включает/отключает счетчики вызовов. Он должен быть включен для Asterisk чтобы иметь возможность получать статус для SIP-устройств. Этот параметр может быть установлен либо в `[general]`, либо или в секциях пиров в *sip.conf*.



Если вы хотите, чтобы состояния устройств работали на устройствах SIP, то должны, по крайней мере, установить для параметра `callcounter` значение `yes`. В противном случае драйвер SIP-канала не будет отслеживать звонки на устройства и с них и не будет предоставлять информацию о состоянии.

`busylevel`

Устанавливает количество вызовов, которые должны выполняться для Asterisk, чтобы сообщить, что устройство занято. Этот параметр может быть установлен только в секциях пиров в *sip.conf*. По умолчанию эта опция не установлена. Это означает, что Asterisk сообщит, что устройство используется, но не покажет что занято.

`call-limit`

Этот параметр устарел в пользу использования функций `GROUP()` и `GROUP_COUNT()` в диалплане Asterisk. Вы можете найти более старую документацию, которая предполагает, что этот параметр необходим для работы SIP-присутствия. Это имело место, но этот вариант был заменен опцией `callcounter` для этой цели.

`allowubscribe`

Позволяет отключить поддержку подписки. Если этот параметр не установлен, подписки будут включены. Чтобы отключить поддержку подписки полностью, установить `allowsubscribe` в значение `no` в секции `[general]` файла *sip.conf*.

`subscribecontext`

Позволяет установить определенный контекст для подписок. Без этой установки будет использоваться контекст, определяемый параметром `context`. Эта опция может быть задана либо в разделе `[general]`, либо в секциях пиров в *sip.conf*.

1 Некоторые также любят называть их "мигающие лампы” или"мигающие огни" для своих телефонов. Гики и их светодиоды…

`notifyingringing`

Контролирует, будет ли отправлено уведомление, когда номер переходит в состояние вызова. Этот параметр имеет значение `yes` по умолчанию. Он влияет только на подписки, которые используют пакет событий *dialog-info*. Эта опция может быть установлена только глобально в секции `[general]` в *sip.conf*.

`notifyhold`

Позволяет `chan_sip` устанавливать состояния SIP-устройств в `ONHOLD`. По умолчанию установлено значение `yes`. Эта опция может быть установлена только глобально в секции `[general]` в *sip.conf*.

`notifycid`

Включает/отключает отправку информации CallerID абонента в расширение. Этот параметр применяется к устройствам, которые подписываются на уведомления о состоянии номеров, основанные на `dialog-info+xml`, таких телефонов как Snom. Отображение CallerID вызывающего абонента может быть полезно, чтобы помочь агенту решить, следует ли выполнять ответ на входящий вызов. Этот параметр установлен в `no` по умолчанию.



Этот волшебный пикап работает только в том случае, если номер и контекст хинта совпадают с номером и контекстом входящего вызова. Следует отметить, что использование опции `subscribecontext` обычно ломает эту опцию. Эта опция также может быть установлена в значение `ignorecontext`. Это позволит обойти проблему контекста, но должно использоваться только в среде, где существует только один экземпляр номера, на который была подписка. В противном случае вы можете случайно ответить на звонки, на которые не собирались отвечать.

Использование пользовательских состояний устройств

Asterisk предоставляет возможность создавать пользовательские состояния устройств, которые позволяют разрабатывать различные интересные пользовательские приложения. Мы начнем с отображения основного синтаксиса для управления пользовательскими состояниями устройств, а затем создадим пример, который их использует.

Состояния пользовательских устройств начинаются с префикса `Custom:`. Текст, который появляется после префикса, может быть любым, какой хотите. Чтобы установить или прочитать значение настраиваемого состояния устройства, используйте функцию диалплана `DEVICE_STATE()`. Например, чтобы установить пользовательское состояние устройства:

```
exten => example,1,Set(DEVICE_STATE(Custom:example) = BUSY)
```

Аналогично, чтобы прочитать текущее значение состояния пользовательского устройства:

```
exten => Verbose(1,The state of Custom:example is
${DEVICE_STATE(Custom:examle)})
```

Пользовательские состояния устройств могут использоваться как способ прямого управления состоянием, отображаемым на устройстве, которое подписано на состояние номера. Просто сопоставьте расширение с пользовательским состоянием устройства, используя хинт в диалплане:

```
exten => example,hint,Custom:example
```

Как пример

Для пользовательских состояний устройства существует ряд интересных вариантов использования. В этом разделе мы построим пример, который реализует пользовательскую кнопку «Не беспокоить» (DND) на SIP-телефоне. Этот же подход может быть применен ко многим другим вещам, которые вы хотели бы переключать одним нажатием кнопки. Например, этот подход может использоваться чтобы сообщить участникам вошли они в очередь или нет.

Первая часть примера - это хинт в диалплане. Он необходим, таким образом, BLF может быть настроен на SIP-телефоне для подписки на этот номер. В этом случае телефон должен быть настроен для подписки на состояние DND_7015:

```
exten => DND_7015,hint,Custom:DND_7015
```

Далее мы создадим внутр.номер, который будет вызываться когда пользователь нажимает клавишу назначенную на пользовательскую функцию DND. Интересно отметить, что этот номер ничего не делает с аудио. Фактически, пользователь телефона, скорее всего, даже не узнает, что происходит звонок когда он нажимает кнопку. Что касается пользователя, нажатие этой клавиши просто включает или выключает индикацию рядом с кнопкой, которая отражает, включен ли DND. Расширение должно выглядеть так:

```
exten => DND_7015,1,Answer()
   same => n,GotoIf($["${DEVICE_STATE(Custom:DND_7015)}"="BUSY"]?
turn_off:turn_on)
   same => n(turn_off),Set(DEVICE_STATE(Custom:DND_7015)=NOT_INUSE)
   same => n,Hangup()
   same => n(turn_on),Set(DEVICE_STATE(Custom:DND_7015)=BUSY)
   same => n,Hangup()
```

В заключительной части этого примера показано, как состояние DND используется в диалплане. Если DND включен, для вызывающего абонента воспроизводится сообщение о том, что агент недоступен. Если он отключен, на SIP-устройство будет сделан вызов:

```
exten => 7015,1,GotoIf($["${DEVICE_STATE(Custom:DND_7015)}"="BUSY"]?
busy:available)
   same => n(available),Verbose(3,DND is currently off for 7015.)
   same => n,Dial(SIP/exampledevice)
   same => n,Hangup()
   same => n(busy),Verbose(3,DND is on for 7015.)
   same => n,Playback(vm-theperson)
   same => n,Playback(digits/7&digits/0&digits/1&digits/5)
   same => n,Playback(vm-isunavail)
   same => n,Playback(vm-goodbye)
   same => n,Hangup()
```

Пример 14-1 показывает полный набор номеров, поскольку они появятся в */etc/asterisk/extensions.conf*.

Пример 14-1. Пользовательская функция «Не беспокоить» с использованием пользовательских состояний устройства

```
;
; Хинт чтобы телефон мог использовать BLF для сигнализации о статусе DND.
;
exten => DND_7015,hint,Custom:DND_7015

;
; Номер для набора при нажатии пользовательской клавиши DND
; на своем телефоне. Это переключит состояние и приведет к включению или
; выключению индикатора на телефоне.
;
exten => DND_7015,1,Answer()
   same => n,GotoIf($["${DEVICE_STATE(Custom:DND_7015)}"="BUSY"]?
turn_off:turn_on)
   same => n(turn_off),Set(DEVICE_STATE(Custom:DND_7015)=NOT_INUSE)
   same => n,Hangup()
   same => n(turn_on),Set(DEVICE_STATE(Custom:DND_7015)=BUSY)
   same => n,Hangup()
```

```
;
; Пример использования состояния DND.
;
exten => 7015,1,GotoIf($["${DEVICE_STATE(Custom:DND_7015)}"="BUSY"]?
busy:available)
   same => n(available),Verbose(3,DND is currently off for 7015.)
   same => n,Dial(SIP/exampledevice)
   same => n,Hangup()
   same => n(busy),Verbose(3,DND is on for 7015.)
   same => n,Playback(vm-theperson)
   same => n,Playback(digits/7&digits/0&digits/1&digits/5)
   same => n,Playback(vm-isunavail)
   same => n,Playback(vm-goodbye)
   same => n,Hangup()
```

Распространение состояний устройств

Asterisk в основном предназначен для работы в одной системе. Однако по мере увеличения требований к масштабируемости для развертываний обычно требуется несколько серверов Asterisk. Поскольку это становится все более распространенным, некоторые функции были добавлены чтобы упростить координацию нескольких серверов Asterisk. Одна из этих функций - поддержка *распространения состояний устройств*.

Это означает, что если устройство находится в состоянии вызова на одном сервере Asterisk, состояние этого устройства отражается на всех серверах. Чтобы быть более конкретным, способ, которым это работает, заключается в том, что каждый сервер знает состояние каждого устройства с точки зрения каждого сервера. Используя этот набор состояний, каждый сервер будет вычислять какое общее значение состояния устройства должно сообщаться остальной части серверов Asterisk.

Для выполнения распространения состояний устройств необходимо использовать какой-то механизм обмена сообщениями, чтобы серверы взаимодействовали друг с другом. По состоянию на Asterisk 11 поддерживаются два таких механизма: Corosync и XMPP.

Использование Corosync

Corosync предоставляет набор API-интерфейсов, полезных для кластеризованных приложений. Asterisk использует группу коммуникаций Corosync API для распространения событий между кластерами серверов Asterisk.

Corosync построен таким образом, что узлы должны располагаться в одной высокоскоростной локальной сети с низкой латентностью. Если ваше развертывание географически распределено, вы должны использовать поддержку распространения состояния устройства на основе XMPP, о чем говорится в "Использование XMPP".

Установка

Далее установите Corosync. Самый простой способ установить его - из системы управления пакетами дистрибутива. В RHEL:

```
$ sudo yum install corosync corosynclib corosynclib-devel
```

В Ubuntu:

```
$ sudo apt-get install corosync corosync-dev
```

Если вы установили Asterisk перед установкой Corosync, вам нужно будет перекомпилировать и переустановить Asterisk, чтобы получить поддержку Corosync. Начните с запуска Asterisk скрипта *configure*. Скрипт *configure* отвечает за проверку системы, для выяснения, какие дополнительные зависимости необходимо найти, чтобы система сборки знала, какие модули могут быть собраны:

```
$ cd /путь/до/asterisk
$ ./configure
```

После запуска скрипта *configure* запустите *menuselect* - инструмент для указания сборки модуля `res_corosync` (этот модуль может быть найден в разделе *Resourse Modules* меню *menuselect*):

```
$ make menuselect
```

Наконец, скомпилируйте и установите Asterisk:

```
$ make
$ sudo make install
```



Это довольно быстрый и грубый набор инструкций для компиляции и установки Asterisk. Для получения более полного набора инструкций см. Главу 3.

Конфигурация Corosync

Теперь, когда Corosync установлен, его необходимо настроить. Для Corosync есть файл конфигурации, который должен быть установлен. Проверьте, существует ли файл */etc/corosync/corosync.conf*. Если нет, то в */etc/corosync* должны быть образцы для начала работы.

При написании этого раздела были использованы следующие конфигурации *corosync.conf* для простой настройки между двумя виртуальными машинами. Эта настройка использует транспорт `udpu`, что означает выполнение одноадресной рассылки (unicast) между двумя узлами вместо многоадресной рассылки (multicast), которая является традиционным транспортом Corosync:

```
# Пожалуйста, прочтите страницу руководства corosync.conf.5
compatibility: whitetank
totem {
  version: 2
  secauth: off
  interface {
            member {
                    memberaddr: 192.168.122.250
            }
            member {
                    memberaddr: 192.168.122.94
            }
            ringnumber: 0
            bindnetaddr: 192.168.122.0
            mcastport: 5405
            ttl: 1
  }
  transport: udpu
}
logging {
   fileline: off
   to_logfile: yes
   to_syslog: yes
   debug: on
   logfile: /var/log/corosync/corosync.log
   debug: off
   timestamp: on
   logger_subsys {
            subsys: AMF
            debug: off
   }
```

```
}
```

Для подробной документации о параметрах в этом файле конфигурация, см связанной справочной страницы:

```
$ man corosync.conf
```

Для Corosync должен быть создан ещё один файл. Corosync необходимо сообщить, что приложениям, запущенным от имени пользователя `asteriskpbx` разрешено подключаться к нему. Поместите следующее содержимое в новый файл с именем */etc/corosync/uidgid.d/asterisk*:

```
uidgid {
    uid: asteriskpbx
    gid: asteriskpbx
}
```

Как только конфигурация завершена, запустите службу Corosync:

```
$ sudo service corosync start
```



Если у вас возникли проблемы с синхронизацией узлов друг с другом, проверьте, нет ли правил брандмауэра на узлах, которые блокируют многоадресный трафик, используемый для обмена данными между узлами.

Конфигурация Asterisk

Модуль `res_corosync` для Asterisk имеет один файл конфигурации */etc/asterisk/res_corosync.conf*. В этом файле требуется один короткий раздел чтобы включить распространение состояний устройств в кластере Corosync. Поместите следующее содержимое в файл */etc/asterisk/res_corosync.conf* :

```
[general]

publish_event = device_state
subscribe_event = device_state
```

Команда CLI Asterisk может использоваться чтобы гарантировать правильность загрузки этой конфигурации:

```
*CLI> corosync show config

============================================================
=== res_corosync config ====================================
============================================================
===
=== ==> Publishing Event Type: device_state
=== ==> Subscribing to Event Type: device_state
=== ==> Publishing Event Type: ping
=== ==> Subscribing to Event Type: ping
===
============================================================
```

Другая полезная команда Asterisk CLI, предоставленная модулем res_corosync, используется для перечисления участников кластера Corosync:

```
*CLI> corosync show members
============================================================
=== Cluster members ========================================
============================================================
===
```

```
=== Node 1
=== --> Group: asterisk
=== --> Address 1: 192.168.122.94
=== Node 2
=== --> Group: asterisk
=== --> Address 1: 192.168.122.250
===
============================================================
```

Тестирование изменения состояний устройств

Теперь, когда вы установили и сконфигурировали распространение состояния устройства с помощью Corosync, есть несколько простых тестов, которые можно выполнить с помощью пользовательских состояний устройств, чтобы обеспечить связь состояний устройств между серверами. Начните с создания тестового хинта в диалплане Asterisk */etc/asterisk/extensions.conf*:

```
[devstate_test]

exten=> foo,hint,Custom:abc
```

Теперь вы можете настроить пользовательское состояние устройства из командной строки Asterisk с помощью команды CLI *dialplan set global*, а затем проверить состояние на каждом сервере с помощью команды *core show hints*. Например, мы можем использовать эту команду для установки состояния на одном сервере:

```
pbx1*CLI> dialplan set global DEVICE_STATE(Custom:abc) INUSE

    -- Global variable 'DEVICE_STATE(Custom:abc)' set to 'INUSE'
```

а затем, проверьте состояние на другом сервере с помощью этой команды:

```
*CLI> core show hints
-= Registered Asterisk Dial Plan Hints =-
   foo@devstatetest     : Custom:abc      State:InUse       Watchers 0
```

Если вы хотите глубже погрузиться в обработку изменения распределения состояния устройства, могут быть включены некоторые полезные отладочные сообщения. Во-первых, включите отладку на консоли Asterisk в /etc/asterisk/logger.conf. Затем включите отладку в командной строке Asterisk CLI:

```
*CLI> core set debug 1
```

Когда вывод debug будет включен, вы увидите сообщения, которые показывают, как Asterisk обрабатывает каждое изменение состояния. Когда состояние устройства изменяется на одном сервере, Asterisk проверяет информацию о состоянии этого устройства на всех серверах и определяет общее состояние устройства. Следующие примеры иллюстрируют:

```
*CLI> dialplan set global DEVICE_STATE(Custom:abc) NOT_INUSE

    -- Global variable 'DEVICE_STATE(Custom:abc)' set to 'NOT_INUSE'

[Nov 13 13:27:12] DEBUG[14801]: devicestate.c:652
handle_devstate_change: Processing device state change for 'Custom:abc'
[Nov 13 13:27:12] DEBUG[14801]: devicestate.c:602
process_collection: Adding per-server state of 'Not in use' for 'Custom:abc'
[Nov 13 13:27:12] DEBUG[14801]: devicestate.c:602
process_collection: Adding per-server state of 'Not in use' for 'Custom:abc'
[Nov 13 13:27:12] DEBUG[14801]: devicestate.c:609
process_collection: Aggregate devstate result is 'Not in use' for 'Custom:abc'
[Nov 13 13:27:12] DEBUG[14801]: devicestate.c:631
process_collection: Aggregate state for device 'Custom:abc' has changed to
```

```
'Not in use'

*CLI> dialplan set global DEVICE_STATE(Custom:abc) INUSE

    -- Global variable 'DEVICE_STATE(Custom:abc)' set to 'INUSE'

[Nov 13 13:29:30] DEBUG[14801]: devicestate.c:652 handle_devstate_change:
Processing device state change for 'Custom:abc'
[Nov 13 13:29:30] DEBUG[14801]: devicestate.c:602 process_collection:
Adding per-server state of 'Not in use' for 'Custom:abc'
[Nov 13 13:29:30] DEBUG[14801]: devicestate.c:602 process_collection:
Adding per-server state of 'In use' for 'Custom:abc'
[Nov 13 13:29:30] DEBUG[14801]: devicestate.c:609 process_collection:
Aggregate devstate result is 'In use' for 'Custom:abc'
[Nov 13 13:29:30] DEBUG[14801]: devicestate.c:631 process_collection:
Aggregate state for device 'Custom:abc' has changed to 'In use'
```

Использование XMPP

eXtensible Messaging and Presence Protocol (XMPP) (расширяемый протокол обмена сообщениями о присутствии) ранее (и до сих пор) известный как *Jabber*, является протоколом связи, стандартизированным Internet Engineering Task Force (IETF). Он чаще всего известен как протокол IM, но его можно использовать и для ряда других интересных приложений. Фонд стандартов XMPP (XSF) работает для стандартизации расширений к протоколу XMPP. Одно из таких расширений, называемое PubSub, обеспечивает механизм публикации/подписки.

Asterisk имеет возможность использовать XMPP PubSub для распространения информации о состоянии устройства. Одна из приятных особенностей использования XMPP заключается в том, что он отлично работает для географически распределенных серверов Asterisk.

Установка

Чтобы распространять состояния устройств с помощью XMPP вам понадобится сервер XMPP, поддерживающий PubSub. Одним из таких серверов, который был успешно протестирован с Asterisk, является Tigase.

На веб-сайте Tigase есть инструкции по установке и настройке сервера Tigase. Мы предлагаем вам следовать этим инструкциям (или инструкциям, предоставленным для любого другого сервера, который вы можете использовать) и вернуться к этой книге, когда вы будете готовы работать с определенными компонентами Asterisk.

На стороне Asterisk вам нужно будет убедиться, что вы установили модуль `res_jabber`. Вы можете проверить, загружен ли он в CLI Asterisk:

```
*CLI> module show like jabber
Module      Description                          Use   Count
res_jabber.so     AJI - Asterisk Jabber Interface     0
1 modules load
```

Если вы используете пользовательский /etc/asterisk/modules.conf, в котором перечислены только определенные модули для загрузки, вы также можете проверить файловую систему, чтобы узнать, был ли скомпилирован и установлен модуль:

```
$ cd /usr/lib/asterisk/modules
$ ls -l res_jabber.so

-rwxr-xr-x 1 root root 837436
2010-11-12 15:33 res_jabber.so
```

Если у вас еще нет `res_jabber`, вам нужно будет установить `iksemel` и библиотеки OpenSSL. Затем вам нужно будет пересобрать и переустановить Asterisk. Начните с запуска скрипта Asterisk *configure*, который отвечает за проверку системы и определение дополнительных зависимостей, чтобы система сборки знала, какие модули могут быть собраны:

```
$ cd /путь/до/asterisk
$ ./configure
```

После запуска скрипта *configure*, запустите *menuselect* чтобы гарантировать, что Asterisk было сказано собрать модуль `res_jabber`. Этот модуль можно найти в разделе *Resourse Modules* меню *menuselect*:

```
$ make menuselect
```

Наконец, скомпилируйте и установите Asterisk:

```
$ make
$ sudo make install
```



Это довольно быстрый и грубый набор инструкций для сборки и установки Asterisk. Для получения более полного набора инструкций см. Главу 3.

Создание учетных записей XMPP

К сожалению, Asterisk в настоящее время не может регистрировать новые учетные записи на сервере XMPP. Вам нужно будет создать учетную запись для каждого сервера через какой-либо другой механизм. Метод, который мы использовали во время тестирования заключался в том, чтобы завершить регистрацию аккаунта через клиент XMPP, такой как Pidgin. После завершения регистрации учетной записи клиент XMPP больше не нужен. Для остальных примеров мы будем использовать следующих двух приятелей, которые оба находятся на сервере *jabber.shifteight.org*:

- `server1@jabber.shifteight.org/astvoip1`
- `server2@jabber.shifteight.org/astvoip2`

Конфигурация Asterisk

Файл */etc/asterisk/jabber.conf* необходимо будет настроить на каждом сервере. Покажем конфигурацию для установки двух серверов здесь, но конфигурация может быть легко расширена на нескольких серверах при необходимости. Пример 14-2. показывает содержимое файла конфигурации для сервера 1, а Пример 14-3 - для сервера 2. Дополнительные сведения о параметрах *jabber.conf*, связанных с распределением состояний устройств, см. в файле *configs/jabber.conf.sample*, который включен в исходное дерево Asterisk.

Пример 14-2. jabber.conf для server1

```
[general]
autoregister = yes

[asterisk]
type = client
serverhost = jabber.shifteight.org
pubsub_node = pubsub.jabber.shifteight.org
username = server1@jabber.shifteight.org/astvoip1
secret = mypassword
distribute_events = yes
status = доступно
usetls = no
usesasl = yes
```

```
buddy = server2@jabber.shifteight.org/astvoip2
```

Пример 14-3. jabber.conf для server2

```
[general]
autoregister = yes

[asterisk]
type = client
serverhost = jabber.shifteight.org
pubsub_node = pubsub.jabber.shifteight.org
username = server2@jabber.shifteight.org/astvoip2
secret = mypassword
distribute_events = yes
status = доступно
usetls = no
usesasl = yes
buddy = server1@jabber.shifteight.org/astvoip1
```

Тестирование

Чтобы все работало правильно, начните с выполнения проверки параметров *jabber.conf* на каждом сервере. Пара соответствующих команд CLI Asterisk может быть использована. Первая - это *jabber show connected*, которая проверит, что Asterisk успешно вошел в систему с учетной записью на *jabber* сервере. Вывод этой команды на первом сервере показывает:

```
*CLI> jabber show connected

Jabber Users and their status:
 User: server1@jabber.shifteight.org/astvoip1   - Connected
----
   Number of users: 1
```

Между тем, если *jabber show connect* выполняется на втором сервере, это покажет:

```
*CLI> jabber show connected

Jabber Users and their status:
 User: server2@jabber.shifteight.org/astvoip2   - Connected
----
   Number of users: 1
```

Следующая полезная команда для проверка настройки - это *jabber show buddies*. Эта команда позволяет проверить, правильно ли указан другой сервер в списке приятелей. Она также позволяет увидеть, доступен ли другой сервер в настоящее время. Если вы запустите эту команду на первом сервере без запущенной Asterisk на втором сервере, вывод будет выглядеть следующим образом:

```
*CLI> jabber show buddies

Jabber buddy lists
Client: server1@jabber.shifteight.org/astvoip1
    Buddy: server2@jabber.shifteight.org
           Resource: None
    Buddy: server2@jabber.shifteight.org/astvoip2
           Resource: None
```

Затем запустите Asterisk на втором сервере и выполните *jabber show buddies* на этом сервере. Вывод будет содержать больше информации, так как второй сервер увидит первый сервер онлайн:

```
*CLI> jabber show buddies
```

```
Jabber buddy lists
Client: server2@jabber.shifteight.org/astvoip2
    Buddy: server1@jabber.shifteight.org
           Resource: astvoip1
               node: http://www.asterisk.org/xmpp/client/caps
               version: asterisk-xmpp
               Jingle capable: yes
           Status: 1
           Priority: 0
    Buddy: server1@jabber.shifteight.org/astvoip1
           Resource: None
```

На данный момент вы должны быть готовы проверить распределение состояний устройств. Процедура такая же, как и для тестирования состояний устройств с помощью Corosync, которые можно найти в "Тестирование изменения состояний устройств".

Общие внешние линии

В Asterisk, Shared Line Appearances (SLA) (Общие внешние линии - ОВЛ) - иногда также упоминаемые в отрасли как Bridged Line Appearances (BLA) (Общие соединительные линии - ОСЛ) - могут быть использованы. Эта функциональность может использоваться для удовлетворения двух основных потребностей, которые включают эмуляцию простой системы клавиш и создание общих внутренних номеров на УАТС.

Создание системы эмуляции клавиш - это то, для чего в первую очередь предназначались эти приложения. В этой среде у вас есть небольшое количество транков, входящих в УАТС, таких как аналоговые телефонные линии, и каждый телефон имеет специальную кнопку для вызовов на этом транке. Например, вы можете ссылаться на эти транки, например как line1,line2 и line3.

Другой вариант использования - создание общих внутренних номеров вашей УАТС. Этот прецедент является наиболее распространенным в наши дни. Есть множество причин, по которым вы можете это сделать; например, вы можете захотеть, чтобы номер появлялся на телефонах как исполнителя, так и его помощника по административным вопросам. Другим примером может быть, если вы хотите, чтобы так же номер отображался на всех телефонах в той же лаборатории.

Хотя эти варианты использования поддерживаются в определенной степени, существуют ограничения. В Asterisk еще предстоит проделать большую работу, чтобы эти функции работали действительно хорошо для того, что люди хотят с ними делать. Это обсуждается в разделе "Ограничения".

Установка приложений SLA

Приложения SLA построены на двух ключевых технологиях Asterisk. Первый - это обработка состояний устройств, а второй - конференц-связь. В частности, конференц-связь, используемая этими приложениями, является приложением `MeetMe()`. Приложения SLA поставляются с тем же модулем, что и приложение `MeetMe()`, поэтому необходимо установить модуль `app_meetme`.

Вы можете проверить в CLI Asterisk, чтобы увидеть, имеется ли этот модуль:

```
pbx*CLI> module show like app_meetme.so
ModuleDescription Use Count
0 modules loaded
```

В этом случае модуль отсутствует. Наиболее распространенная причина, по которой система Asterisk не будет иметь модуль `app_meetme` - не был установлен DAHDI. Приложение `MeetMe()` использует DAHDI для микширования конференций.



Дополнительные сведения о конференц-связи `MeetMe()` см. в разделе “Конференц-связь с MeetMe()” в Главе 10. Вам потребуется `res_timing_dahdi` и DAHDI для установки, так как `MeetMe()` требует его для выполнения микширования. К сожалению, в Asterisk 11, в приложение `ConfBridge()` еще не добавлена необходимая функциональность для SLA.

После установки DAHDI (см. Главу 3 для информации об установке) повторно запустите скрипт *configure*, перекомпилируйте с помощью *make* и переустановите с помощью *sudo make install*. После того, как модуль был правильно установлен (вы можете использовать *module load app_meetme.so* из консоли Asterisk), вы должны увидеть его состояние в CLI:

```
*CLI> module show like app_meetme.so

Module      Description                    Use Count
app_meetme.so           MeetMe conference bridge           0
1 modules loaded
```

Как только модуль `app_meetme` загружен, вы должны иметь оба приложения `SLAStation()` и `SLATrunk()`:

```
*CLI> core show applications like SLA

    -= Matching Asterisk Applications =-
            SLAStation: Shared Line Appearance Station.
              SLATrunk: Shared Line Appearance Trunk.
    -= 2 Applications Matching =-
```

Обзор конфигурации

Для настройки SLA необходимо отредактировать два основных файла конфигурации: */etc/asterisk/extensions.conf* и */etc/asterisk/sla.conf*. Файл *sla.conf* используется для определения транков и станций. Станция - это любой SIP-телефон, который будет использовать SLA. Транки - это буквально транки или общие внутренние номера, которые будут отображаться на двух или более станциях. Диалплан Asterisk *extensions.conf*, обеспечивает некий важный клей, который объединяет конфигурацию SLA. Вложения (includes) диалплана включают некоторые хинты состояний номеров и номера, которые определяют, как вызовы входят и выходят из установки SLA. Следующие несколько разделов содержат подробные примеры конфигурации для нескольких различных случаев использования.

Пример системы клавиш с аналоговыми транками

Это использование SLA поставляется с самой простой конфигурацией.[2] Этот сценарий обычно использовался для довольно небольшого количества установок, где у вас есть несколько аналоговых линий и SIP-телефонов, у которых есть клавиши линии, непосредственно связанные с аналоговыми линиями. Для целей этого примера мы предположим что имеем две аналоговые линии и четыре SIP-телефона. Каждый SIP-телефон будет иметь кнопку для `line1` и кнопку для `line2`. В этом разделе предполагается, что вы предварительно выполнили некоторые настройки, в том числе:

- Настроенные четыре SIP-телефона. Для получения дополнительной информации о настройке SIP-телефонов см. Главу 5.
- Настроенные две аналоговые линии. Более подробную информацию о настройке аналоговых линий с помощью Asterisk см. в Главе 7.

В этом примере мы будем использовать следующие имена устройств для SIP-телефонов и аналоговых линий. Не забудьте адаптировать примеры в соответствии с вашей конфигурацией:

2 Следует признать, что ни одна из конфигураций для SLA не является простой.

- SIP/station1
- SIP/station2
- SIP/station3
- SIP/station4
- DAHDI/1
- DAHDI/2

sla.conf

Как упоминалось ранее, *sla.conf* содержит настройки сопоставления устройств для транков и станций. В этом примере мы начнем с определения двух транков:

```
[line1]
type = trunk
device = DAHDI/1

[line2]
type = trunk
device = DAHDI/2
```

Далее мы настроим определение станций. У нас есть четыре SIP-телефона, каждый из которых будет использовать оба транка. Обратите внимание, что имена разделов в *sla.conf* для станций не должны совпадать с именами SIP-устройств, но это сделано таким образом здесь для удобства:

```
[station1]
type = station
device = SIP/station1
trunk = line1
trunk = line2

[station2]
type = station
device = SIP/station2
trunk = line1
trunk = line2

[station3]
type = station
device = SIP/station3
trunk = line1
trunk = line2

[station4]
type = station
device = SIP/station4
trunk = line1
trunk = line2
```

Конфигурация станций немного повторяющаяся. Секции шаблона конфигурационного файла Asterisk пригодятся здесь, чтобы немного уменьшить конфигурацию. Вот конфигурация станций снова, но на этот раз с использованием шаблона:

```
[station](!)
type = station
trunk = line1
trunk = line2
```

```
[station1](station)
device = SIP/station1

[station2](station)
device = SIP/station2

[station3](station)
device = SIP/station3

[station4](station)
device = SIP/station4
```

После внесения изменений в *sla.conf* и сохранения их, не забудьте перезагрузить модуль `app_meetme` чтобы изменения вступили в силу. Вы можете сделать это с помощью *module reload app_meetme* из консоли Asterisk.

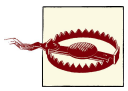
extensions.conf

Следующий файл конфигурации, необходимый для этого примера - */etc/asterisk/extensions.conf*. Существует три контекста. Во-первых, у нас есть контексты `line1` и `line2`. Когда вызов поступает на одну из аналоговых линий, он входит в одном из этих контекстов в диалплан и выполняет приложение `SLATrunk()`. Это приложение будет заботиться о вызове всех соответствующих станций:

```
[line1]
exten => s,1,SLATrunk(line1)

[line2]
exten => s,1,SLATrunk(line2)
```

Следующий раздел диалплана - контекст `sla_stations`. Все вызовы от телефонов SIP должны быть переданы этому контексту. Далее SIP-телефоны должны быть настроены так, чтобы как только поднимается трубка, они немедленно делают вызов к номеру `station1` (или `station2`, `station3`, и т.д. в соответствующих случаях). Если на телефоне нажата клавиша `line1`, то вызов должен быть отправлен на добавочный номер `station 1_line1` (или `station 2_line1` и т.д.).



Формат `station1_line1` имеет особое значение для приложения `SLAStation()`. Разделитель подчеркивание (_) определяет, какую станцию и транк использовать для вызова. Помните об этом при именовании станций или транков, так как использование подчеркивания в их именах нарушит функциональность.

Каждый раз, когда поднимается телефонная трубка или нажата клавиша line, сделанный вызов немедленно соединит его с одной из аналоговых линий. Для линии, которая еще не используется, аналоговая линия будет обеспечивать сигнал готовности вызова, и пользователь сможет набрать цифры, чтобы сделать вызов. Если пользователь нажимает клавишу line для линии, которая уже используется, то пользователь будет соединен с существующим вызовом на той линии. Контекст sla_stations выглядит так:

```
[sla_stations]
exten => station1,1,SLAStation(station1)
exten => station1_line1,hint,SLA:station1_line1
exten => station1_line1,1,SLAStation(station1_line1)
exten => station1_line2,hint,SLA:station1_line2
exten => station1_line2,1,SLAStation(station1_line2)

exten => station2,1,SLAStation(station2)
exten => station2_line1,hint,SLA:station2_line1
```

```
exten => station2_line1,1,SLAStation(station2_line1)
exten => station2_line2,hint,SLA:station2_line2
exten => station2_line2,1,SLAStation(station2_line2)

exten => station3,1,SLAStation(station3)
exten => station3_line1,hint,SLA:station3_line1
exten => station3_line1,1,SLAStation(station3_line1)
exten => station3_line2,hint,SLA:station3_line2
exten => station3_line2,1,SLAStation(station3_line2)

exten => station4,1,SLAStation(station4)
exten => station4_line1,hint,SLA:station4_line1
exten => station4_line1,1,SLAStation(station4_line1)
exten => station4_line2,hint,SLA:station4_line2
exten => station4_line2,1,SLAStation(station4_line2)
```

После внесения изменений обязательно перезагрузите диалплан с помощью *dialplan reload*.

Дополнительные задачи по настройке телефона

В предыдущем разделе описана схема набора номеров для транков и станций. При настройке телефонов для использования с этой опцией необходимо учитывать некоторые особенности. Во-первых, каждый телефон должен быть настроен на отправку вызова как только снята трубка.

Другим важным пунктом является настройка клавиш линии. Asterisk использует подписки на состояния номеров для управления светодиодами рядом с кнопками линий. Кроме того, каждая кнопка линии должна быть настроена как быстрый набор. Используйте следующий контрольный список для конфигурации клавиш линии (выполнение этих задач будет зависеть от конкретных телефонов, которые вы используете):

- Установите метку клавиши на *Line 1* (и т.д.) или как вы считаете нужным.
- Настройте клавиши так, чтобы клавиша *Line 1* на `station1` подписывалась на состояние `station1_line1` и так далее. Это необходимо чтобы Asterisk мог заставить светодиоды отражать состояния линий.
- Убедитесь что если клавиша *Line 1* на `station1` нажата, вызов направляется в добавочный номер `station1_line1` и так далее.

Пример системы клавиш с SIP-транками

Этот пример по функциональности идентичен предыдущему примеру. Разница в том, что вместо использования аналоговых линий в качестве транков мы будем использовать соединение с SIP-провайдером, который будет терминировать вызовы ТфОП. Для получения дополнительной информации о настройке Asterisk для подключения к SIP-провайдеру см. Главу 7.

sla.conf

Файл *sla.conf* для этого сценария будет немного сложнее.[3] Вы можете ожидать, что в линии устройства в конфигурации транка будет указан канал SIP, но вместо этого мы будем использовать локальный канал. Это позволит нам использовать дополнительную логику диалплана для обработки вызовов. Назначение локального канала станет более понятным в следующем разделе, когда обсуждается пример диалплана. Вот конфигурации транка:

```
[line1]
type = trunk
device = Local/disa@line1_outbound
```

3 Читайте: взлом

```
[line2]
type = trunk
device = Local/disa@line2_outbound
```

Конфигурация станции идентична последнему примеру, поэтому давайте вернемся к ней:

```
[station](!)
type = trunk
trunk = line1
trunk = line2

[station1](station)
device = SIP/station1

[station2](station)
device = SIP/station2

[station3](station)
device = SIP/station3

[station4](station)
device = SIP/station4
```

extensions.conf

Как и в предыдущем примере, вам необходимы контексты `line1` и `line2` для обработки входящих вызовов на этих транках:

```
[line1]
exten => s,1,SLATrunk(line1)
;
; Если провайдер указывает ваш номер телефона при отправке вызова, то
; потребуется другое правило в диалплане чтобы соответствовать этому.
;
exten => _X.,1,Goto(s,1)

[line2]
exten => s,1,SLATrunk(line2)

exten => _X.,1,Goto(s,1)
```

В этом примере так же потребуется контекст `sla_stations`. Это для всех звонков с телефонов. Это то же самое, что и в последнем примере:

```
[sla_stations]
exten => station1,1,SLAStation(station1)
exten => station1_line1,hint,SLA:station1_line1
exten => station1_line1,1,SLAStation(station1_line1)
exten => station1_line2,hint,SLA:station1_line2
exten => station1_line2,1,SLAStation(station1_line2)

exten => station2,1,SLAStation(station2)
exten => station2_line1,hint,SLA:station2_line1
exten => station2_line1,1,SLAStation(station2_line1)
exten => station2_line2,hint,SLA:station2_line2
exten => station2_line2,1,SLAStation(station2_line2)

exten => station3,1,SLAStation(station3)
exten => station3_line1,hint,SLA:station3_line1
exten => station3_line1,1,SLAStation(station3_line1)
```

```
exten => station3_line2,hint,SLA:station3_line2
exten => station3_line2,1,SLAStation(station3_line2)

exten => station4,1,SLAStation(station4)
exten => station4_line1,hint,SLA:station4_line1
exten => station4_line1,1,SLAStation(station4_line1)
exten => station4_line2,hint,SLA:station4_line2
exten => station4_line2,1,SLAStation(station4_line2)
```

Последний требуемый кусок диалплана - это реализация контекстов `line1_outbound` и `line2_outbound`. Это то, что приложения SLA используют когда хотят отправить вызов SIP-провайдеру. Ключом к этой настройке является использование приложения `Read()`. В последнем примере телефоны были напрямую подключены к аналоговой линии, что позволяло комутатору предоставлять тональный сигнал вызова, получать цифры и завершать вызов. В этом примере мы используем приложение `Read()` локально для предоставления тонального сигнала и получения цифр. Обычно `Read()` воспроизводит звуковой файл, но с помощью параметра `i` мы можем указать, какую индикацию воспроизводить из файла *indications.conf*. В этом случае мы будем использовать `dial`, который воспроизводит сигнал набора. Набранный номер будет сохранен в переменной канала `Request` и мы прекратим получать цифры либо по истечении времени ожидания, либо после набора максимума из 11 цифр.

После того, как будет набран полный номер, вызов будет продолжен и передан SIP-провайдеру:

```
[line1_outbound]
exten => disa,1,NoOp()
   same => n,Read(Request,dial,11,i)
   same => n,Goto(${Request},1)

; Добавьте внутренние номера для любых внешних номеров, набор которых вы
; хотите разрешить.
;
exten => _1NXXNXXXXXX,1,Dial(SIP/${EXTEN}@myprovider)

[line2_outbound]
exten => disa,1,NoOp()
   same => n,Read(Request,dial,11,i)
   same => n,Goto(${Request},1)

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

Пример системы альтернативных клавиш с SIP-транками

Чтобы продемонстрировать некоторую гибкость в отношении конфигурации линии, мы покажем пример реализации функциональности клавиш линии с помощью SIP-транков и станций. Мы будем использовать те же понятия, что и в предыдущих разделах, поэтому может быть полезно ознакомиться с ними, прежде чем продолжить. То, что мы будем создавать, можно охарактеризовать как своего рода гибридную модель, в которой у нас будет одна строка регистрации, которую можно использовать как стандартную строку в вашей системе Asterisk. У неё не будет никаких ограничений по функциональности, включая возможность переадресации звонков. Дополнительные линии, которые мы определили, будут действовать больше как традиционная система SLA, в которой трансферы невозможны, но совместное использование транков будет возможно.

В нашем примере мы будем определять отдельные строки для каждой из наших станций в *sla.conf*. Этот альтернативный метод позволяет нам подписывать клавиши линий на нашем телефоне на каждую из определенных линий. Во-первых, мы определяем наши транки как и ранее (однако, слегка модифицированные):

```
[T1]  ; Line 1
```

```
type=trunk
device=Local/trunk1@sla_trunks/n

[T2]  ; Line 2
type=trunk
device=Local/trunk2@sla_trunks/n
```

Затем нам нужно определить наши линии, которые мы свяжем с телефонами. Чтобы не быть слишком многословным, мы определим линии для двух станций, причем две линии для каждой:

```
[L1-0000FFFF0005] ; Line 1
type=station
device=SIP/0000FFFF0005
trunk=T1

[L2-0000FFFF0005] ; Line 2
type=station
device=SIP/0000FFFF0005
trunk=T2

[L1-0000FFFF0006] ; Line 1
type=station
device=SIP/0000FFFF0006
trunk=T1

[L2-0000FFFF0006] ; Line 2
type=station
device=SIP/0000FFFF0006
trunk=T2
```

Как вы можете видеть, с точки зрения Asterisk определены четыре станции. Мы логически разделили каждый из транков на отдельные клавиши линий, на которые можно подписаться с любого устройства. Здесь подразумевается, что нашими станциями являются `SIP/0000FFFF0005` и `SIP/0000FFFF0006`, и каждая из них будет содержать две клавиши линий, которые связаны с транками `T1` и `T2`. После внесения наших изменений в *sla.conf* мы должны обязательно перезагрузить *app_meetme.so* с помощью *module reload app_meetme.so* из консоли Asterisk.

Далее нужно настроить наш диалплан. Мы добавим к существующему контексту `LocalSets`, так как хотим чтобы телефоны могли нормально выполнять вызовы в дополнение к использованию клавиш линий SLA. В нашем примере мы просто покажем части, относящиеся к SLA:

```
[LocalSets]
include => SLA_Outbound
exten => L1-0000FFFF0005_T1,hint,SLA:L1-0000FFFF0005_T1
exten => L2-0000FFFF0005_T2,hint,SLA:L2-0000FFFF0005_T2
exten => L1-0000FFFF0006_T1,hint,SLA:L1-0000FFFF0006_T1
exten => L2-0000FFFF0006_T2,hint,SLA:L2-0000FFFF0006_T2
```

Здесь мы включили новый контекст под названием `SLA_Outbound`, который будет обрабатывать запросы, сделанные телефоном, когда клавиша линии нажата. Мы также включили наши хинты SLA, которые будут использоваться приложениями абонентской группы SLA для управления подсветкой клавиш линии.

Теперь давайте добавим контекст `SLA_Outbound`:

```
[SLA_Outbound]
exten => _LX!,1,NoOp()
    same => n,SLAStation(${EXTEN})
    same => n,Hangup()
```

И нам также понадобится контекст `sla_trunks`, который используется для фактического размещения исходящих звонков. Мы определили, что используется этот контекст в файле *sla.conf*, когда создавали наши транки.



Причина, по которой мы имеем `[n]` в сопоставлении с образцом `_tru[n]kX`, заключается в том, что буква `n` будет соответствовать не букве, а цифре от 2 до 9. Поскольку мы хотим, чтобы соответствовало слову `trunk`, мы можем сделать `n` буквой, заключив её в квадратные скобки. Буква `X` будет соответствовать цифрам от 0 до 9.

```
[sla_trunks]
exten => _tru[n]kX,1,NoOp()
    same => n,Read(Request,dial,10,i)
    same => n,Dial(SIP/my_itsp/${Request})
```

Последний бит диалплана - нам нужно добавить для входящих вызовов для SLA-транков. Мы можем добавить это следующим образом:

```
[SLA_Inbound]
exten => _LX!,1,NoOp()
    same => n,Set(slaLineId=${EXTEN:1:1})
    same => n,SLATrunk(T${slaLineId})
    same => n,Hangup()
```

После внесения изменений в *extensions.conf* вы должны выполнить *dialplan reload* из консоли Asterisk.

К этому моменту мы завершили настройку Asterisk для использования SLA, но настройка телефонов - действительно самая сложная часть. Мы не можем углубиться в конфигурацию каждого производителя, но мы можем, по крайней мере, дать вам отправную точку. На телефонах Polycom (протестировано с Soundpoint IP450 и Soundpoint IP650) вы можете использовать конфигурацию оператора для создания клавиш линий, которые соответствуют состоянию созданных нами хинтов SLA. При совершении звонков по этим линиям они используют клавишу первой линии, определенную для телефона, поэтому у вас фактически будут настроены три клавиши линии. Клавиша первой линии будет стандартной, которая может выполнять трансферы и другие стандартные функции. Две другие клавиши линий будут действовать как линии SLA.

Пример конфигурации, который мы использовали в лаборатории, выглядит следующим образом. Это может дать вам достаточно данных для начала работы:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Generated reg-basic.cfg Configuration File -->
<polycomConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="polycomConfig.xsd">
  <call call.callsPerLineKey="24">
  </call>
  <reg reg.1.address="0000FFFF0005"
       reg.1.auth.password="welcome"
       reg.1.auth.userId="0000FFFF0005"
       reg.1.label="ATDG5"
       reg.1.lineKeys="1"
       reg.1.outboundProxy.address="">
  </reg>

  <attendant
    attendant.reg="1"
    attendant.behaviors.display.spontaneousCallAppearances.normal="1"
    attendant.behaviors.display.remoteCallerID.normal="1"
```

```
    attendant.resourceList.1.address="L1-0000FFFF0005_T1"
    attendant.resourceList.1.label="L1"
    attendant.resourceList.1.type="normal"

    attendant.resourceList.2.address="L2-0000FFFF0005_T2"
    attendant.resourceList.2.label="L2"
    attendant.resourceList.2.type="normal"
  />
</polycomConfig>
```

Пример общего внутреннего номера

Предыдущие два примера были для эмуляции малых системных клавиш. Для этого примера мы попробуем что-то совсем другое. Многие поставщики УАТС предлагают возможность совместного использования одного и того же внутреннего номера несколькими телефонами. Это не просто вопрос вызова нескольких телефонов при вызове одного добавочного номера - это более глубокая интеграция. Поведение клавиши линии для общего номера аналогично поведению клавиши линии как системной. Например, вы можете просто приостановить вызов с одного телефона и принять его с другого. Кроме того, если несколько телефонов нажимают клавишу для общего добавочного номера, все они будут соединены в один и тот же вызов. Вот почему эта функциональность часто также упоминается как Bridged Line Appearances (BLA) - внешняя соединительная линия.

В предыдущих двух примерах у нас было два транка и четыре станции. Для этого примера мы собираемся установить один общий номер на двух телефонах. Общий номер будет упоминаться как номер `5001`.

sla.conf

Каждое использование приложений SLA требует определения транка и станции. Этот пример, как и предыдущие, будет использовать приложение `Read()`, и файл *sla.conf* будет выглядеть очень похоже:

```
[5001]
type = trunk
device = Local/disa@5001_outbound

[5001-phone1]
device = SIP/5001_phone1
trunk = 5001

[5001-phone2]
device = SIP/5001_phone2
trunk = 5001
```

extensions.conf

Первая часть требуемого диалплана - это то, что будет выполняться при наборе добавочного номера `5001` на УАТС. Обычно для вызова телефона используется приложение `Dial()`. В этом случае мы будем использовать приложение `SLATrunk()`. Это позаботится о том, чтобы звонить на оба телефона и соединять их вместе:

```
exten => 5001,1,SLATrunk(5001)
```

Далее нам понадобится контекст, который будет использоваться для исходящих вызовов с этого общего добавочного номера. Здесь предполагается что `5001_phone1` и `5001_phone2` были настроены с параметрами `context`, установленными на `5001` в *sip.conf*:

```
[5001]
;
```

```
; Этот номер необходим если вы хотите, чтобы общий номер использовался по
; умолчанию. В этом случае наберите этот добавочный номер,
; когда трубка будет снята.
;
exten => 5001-phone1,1,SLAStation(5001-phone1)
;
; Это добавочный номер, который должен быть набран при нажатии клавиши 5001 на
; 5001_phone1.
;
exten => 5001-phone1_5001,hint,SLA:5001-phone1_5001
exten => 5001-phone1_5001,1,SLAStation(5001-phone1_5001)

exten => 5001-phone2,1,SLAStation(5001-phone2)
exten => 5001-phone2_5001,hint,SLA:5001-phone2_5001
exten => 5001-phone2_5001,1,SLAStation(5001-phone2_5001)
```

Наконец, нам нужна реализация контекста `5001_outbound`. Он будет использоваться для предоставления тонального сигнала и сбора цифр для соединительной линии:

```
[5001_outbound]

exten => disa,1,Read(Request,dial,,i)
    same => n,Goto(${Request},1)
;
; Этот контекст также должен быть в состоянии видеть любые внутренние номера,
; которые вы хотели бы сделать доступными из этого добавочного номера.
;
include => pbx_extensions
```

Расширенная конфигурация

В файле */etc/asterisk/sla.conf* есть несколько необязательных параметров конфигурации, которые не использовались ни в одном из примеров в этой главе. Чтобы дать вам представление о том, какое другое поведение можно настроить, параметры описаны здесь. Этот файл имеет раздел `[general]`, который зарезервирован для глобальных параметров конфигурации. В настоящее время в этом разделе может быть указан только один параметр:

`attemptcallerid = yes`

Этот параметр указывает, должны ли приложения SLA пытаться передавать информацию о callerID. По умолчанию установлено значение `no`. Если же включено, отображение телефонов может не соответствовать ожиданиям в некоторых ситуациях.

В определениях транков в предыдущих примерах указаны только тип и устройство. Вот некоторые дополнительные параметры, которые можно указать для транка:

`autocontext = line1`

Если этот параметр установлен, Asterisk автоматически создаст контекст диалплана используя это имя. Контекст будет содержать добавочный номер `s`, который выполняет приложение `SLATrunk()` с соответствующим аргументом для этого транка. По умолчанию все записи диалплана должны быть созданы вручную.

`ringtimeout = 20`

Этот параметр позволяет указать количество секунд, в течение которых входящий вызов в этом транке будет звонить до того, как приложение `SLATrunk()` завершит работу и будет считать этот вызов неотвеченным. По умолчанию эта опция не установлена.

`barge = no`

Параметр `barge` указывает, разрешено ли другим станциям присоединяться к вызову, который выполняется в этом транке, нажатием той же кнопки линии. Баржинг по транку разрешен по умолчанию.

`hold = private`

Параметр `hold` указывает разрешения на удержание для этого транка. Если он установлен на `open`, любая станция может перевести этот транк в режим ожидания, а любой другой станции разрешено снять его с удержания. Если для этого параметра установлено значение `private`, только тем станциям, которые перевели транк в режим ожидания, разрешено снимать его с удержания. Значение по умолчанию `open`.

Когда мы определили станции в предыдущих примерах, мы предоставили только тип, устройство и список транков. Тем не менее, определения станций также принимают некоторые дополнительные параметры конфигурации. Они перечислены здесь:

`autocontext = sla_stations`

Если этот параметр указан, Asterisk автоматически создаст номера, необходимые для вызовов, поступающих с этой станции, в указанном контексте. Это отключено по умолчанию, что означает, что все номера должны быть указаны вручную.

`ringtimeout = 20`

Можно указать время ожидания в секундах, в течение которого станция будет звонить, прежде чем вызов будет считаться неотвеченным. Тайм-аут не установлен по умолчанию.

`ringdelay = 5`

Задержка вызова в секундах может быть указана для станции. Если задержка указана, то станция не начнет звонить в течение этого количества секунд после начала вызова на этом транке. Задержка не установлена по умолчанию.

`hold = private`

Разрешение на удержание также может быть указано для конкретной станции. Если для этого параметра установлено значение `private`, любые транки, удерживаемые этой станцией, могут быть получены только этой станцией. По умолчанию установлено `open`.

`trunk = line1,ringtimeout = 20`

`ringtimeout` может быть применен к вызовам, поступающим только из конкретного транка.

`trunk = line1,ringdelay = 5`

`ringdelay` также может быть применен к вызовам из конкретного транка.

Ограничения

В то время как Asterisk упрощает многие вещи, SLA не является одной из них. Эта функциональность была предназначена для эмуляции простых функций, но конфигурация, необходимая для ее работы, довольно сложна. Кто-то, кто является новичком в Asterisk и хочет только простую настройку системы клавиш, должен будет изучить много сложных концепций Asterisk и SIP-телефонов, чтобы заставить её работать.

Еще одна особенность, которая все еще нуждается в доработке, прежде чем она будет бесперебойно работать с SLA - это CallerID. На момент написания этой функции у Asterisk не было соответствующей инфраструктуры, чтобы иметь возможность обновлять информацию об CallerID на протяжении всего разговора. В зависимости от того, как реализована эта функциональность, эта инфраструктура необходима для того, чтобы сделать дисплей на телефонах полезным. Он существует начиная с Asterisk 1.8, но приложения SLA еще не были обновлены для его использования. Конечным результатом является то, что у вас может вообще не быть никакой информации о CallerID или вы

можете включить ее и понять, что дисплей телефона не всегда будет отображаться правильно, так как изменения происходят во время разговора.

Другое ограничение, наиболее актуальное для использования общих номеров, заключается в том, что трансферы не работают. Основная причина заключается в том, что трансферы обычно предполагают приостановку вызова на короткое время. Удержание вызова обрабатывается особым образом с помощью SLA, так как удерживаемый вызов не контролируется телефоном, который инициировал удержание. Это нарушает обработку трансфера.

Таким образом, SLA не так прост в настройке и имеет ряд существенных ограничений. С учетом сказанного, если то, что существует, соответствует вашим потребностям, обязательно соглашайтесь на это.

Создание службы обратного звонка

После сбоя вызова из-за того, что адресат занят или иным образом недоступен, служба обратного вызова позволяет предоставить вызывающему абоненту возможность автоматического обратного вызова, когда адресат становится доступным. Дополнительные услуги по завершению вызова (Call Completion Supplementary Services - CCSS) позволяют вам запросить Asterisk перезвонить вам после попытки вызова без ответа или занятости. В этом разделе показано, как включить эту функцию для использования между телефонами, подключенными к одной и той же системе Asterisk.

Для этого примера мы включим эту услугу для двух SIP-телефонов. Для этого добавьте вследующее в */etc/asterisk/sip.conf*:

```
[phone1]
...
cc_agent_policy = generic
cc_monitor_policy = generic

[phone2]
...
cc_agent_policy = generic
cc_monitor_policy = generic
```

Теперь добавьте следующие номера в диалплан:

```
[phones]
;
; Номера для набора номера телефона менять не нужно.
; Это просто просто примеры набора номера телефона с 20-секундным таймаутом.
;
exten => 7101,1,Dial(SIP/phone1,20)
   same => n,Hangup()

exten => 7102,1,Dial(SIP/phone2,20)
   same => n,Hangup()

;
; Наберите *30, чтобы запросить услуги завершения вызова для последней
; попытки вызова.
;
exten => *30,1,CallCompletionRequest()
   same => n,Hangup()
;

; Для отмены запроса на завершение вызова наберите *31.
;
exten => *31,1,CallCompletionCancel()
```

```
same => n,Hangup()
```

В этом примере мы использовали политики `generic` для `agent` и `monitor`. Этот способ настройки является самым простым для начала работы, но он работает только для вызовов между телефонами, подключенными к одной и той же системе Asterisk. `Agent` является частью системы, которая действует от имени абонента, запрашивающего услуги завершения вызова. `Monitor` является частью системы, отвечающий за мониторинг устройства (или устройств) которые были вызваны, чтобы определить когда они становятся доступными.

Используя этот диалплан, давайте рассмотрим пример, где используется CCSS. Мы начнем с того, что `7001` вызовет `7002`, но дадим тайм-аут - 20 секунд. В этом случае, вызов не состоялся из-за отсутствия ответа. Теперь абонент `7001` может запросить CCSS, набрав `*30`. Это называется завершением вызова без ответа (Call Completion No Response - CCNR). Вы можете проверить запрос CCNR в Asterisk CLI:

```
*CLI> cc report status
1 Call completion transactions
Core ID           Caller                    Status
------------------------------------------------------------------------------
20                SIP/phone1                CC accepted by callee
             |-->7102@phones
             |-->SIP/phone2(CCNR)
```

На данный момент Asterisk использует свою стандартную реализацию generic monitor для ожидания доступности `SIP/phone2`. В случае CCNR, он определяет доступность, ожидая когда телефон сделает следующий вызов. Когда этот вызов заканчивается, Asterisk инициирует вызов между `SIP/phone1` и `SIP/phone2` и запрос CCNR будет выполнен.

Другим сценарием является завершение вызова занятым абонентом (Call Completion Busy Subscriber - CCBS). Это тот случай, когда вызываемый абонент уже разговаривает по телефону. Запрос услуг завершения вызова в этом сценарии работает так же, как и раньше. Generic monitor будет определять доступность, ожидая завершения вызова на устройстве.

Asterisk также поддерживает расширение CCSS на нескольких серверах с использованием SIP или ISDN (в частности, ISDN в Европе). Однако настройка и работа специфичных для протокола методов выходит за рамки этого рецепта. Для получения дополнительной информации об использовании CCSS с Asterisk см. примеры файлов конфигурации Asterisk, а также информацию в вики Asterisk.

Вывод

В этой главе обсуждаются многие аспекты обработки состояний устройств в Asterisk. Мы начали с обсуждения основных понятий состояний устройств и состояний внутренних номеров, а затем создали их. Мы рассмотрели, как SIP-телефоны могут подписываться на состояния, инструменты для создания пользовательских состояний и два механизма, которые можно использовать для распространения состояний между несколькими серверами. Наконец, мы рассмотрели одну из функций в Asterisk, как общие внешние линии, которая в значительной степени зависит от инфраструктуры состояний устройств в Asterisk.

Автосекретарь

Я не отвечаю на телефонные звонки. У меня такое чувство, что каждый раз, когда я это делаю, на другом конце провода кто-то есть.
- Фред Коуплз

Во многих УАТС часто используется система меню для автоматического ответа на входящие вызовы, которая позволяет вызывающим абонентам перенаправлять себя на различные добавочные номера и ресурсы в системе с помощью выбора меню. Эта система известна в телекоммуникационной отрасли как автосекретарь (АС). Автосекретарь обычно предоставляет следующие функции:

- Перевод на добавочный номер
- Перевод на голосовую почту
- Перевод в очередь
- Воспроизведение сообщения (например, «наш адрес…»).
- Подключение к подменю (например, «для просмотра списка наших отделов ...»)
- Подключение к приемной
- Повтор выбора

Для чего-либо еще, особенно если требуется внешняя интеграция, например, поиск в базе данных, обычно требуется интерактивный голосовой ответ (IVR).

Автосекретарь не является IVR

В opensource телекоммуникационном сообществе вы часто будете слышать термин IVR, используемый для описания автосекретаря. Однако в сфере телекоммуникаций IVR отличается от автосекретаря. По этой причине, когда вы разговариваете с кем-либо о каком-либо телекоммуникационном меню, вам следует убедиться, что вы говорите об одном и том же. Для специалиста в области телекоммуникаций - термин IVR подразумевает относительно комплексную и сложную в разработке (и последующе затратную), тогда как автосекретарь - простая и недорогая вещь, которая характерна для большинства АТС.

В этой главе мы поговорим о создании автосекретаря. IVR мы обсудим в Главе 17.[1]

Проектирование вашего автосекретаря

Самая распространенная ошибка, которую допускают новички при разработке АС - излишняя сложность. В то время как при создании многоуровневого АС может быть много радости и чувства

1 Следует отметить, что Asterisk-отличный инструмент для создания IVR. Это неплохо для создания автосекретаря.

выполненного долга с десятками отличных опций и кучей действительно интересных подсказок, ваши абоненты имеют другую повестку дня. Причина, по которой люди звонят по телефону, в первую очередь потому, что они хотят с кем-то поговорить. В то время как люди привыкли к реальности автосекретарей (в некоторых случаях они могут ускорить процесс), по большей части люди предпочитают разговаривать с кем-то живым. Это означает, что есть два основных правила, которых должен придерживаться каждый автосекретарь:

1. Не усложняйте.
2. Убедитесь, что вы всегда включаете обработчик для людей, которые собираются нажать `0` каждый раз, когда они слышат автосекретаря. Если вы не хотите иметь опцию `0`, имейте в виду, что многие люди будут оскорблены этим, и повесят трубку и не перезвонят. В бизнесе это вообще плохо.

Прежде чем вы начнете создавать свой АС, разумно его спроектировать. Вам нужно будет определить поток вызовов, и нужно будет указать подсказки, которые будут воспроизводиться на каждом шаге. Для этого могут быть полезны программные средства построения диаграмм, но не нужно фантазировать. Таблица 15-1 предоставляет хороший шаблон для базового автосекретаря, который будет делать то, что вам нужно.

Таблица 15-1. Базовый автосекретарь

Шаг или выбор	Образец приглашения	Примечание	Имя файла
Приветствие-рабочее время	Спасибо, что позвонили в компанию АБВ.	Дневное приветствие. Воспроизводится сразу после ответа системы на вызов.	*daygreeting.wav*
Приветствие-нерабочее время	Спасибо, что позвонили в компанию АВС. Наш офис сейчас закрыт.	Ночное приветствие. Как и выше, но играет в нерабочее время.	*nightgreeting.wav*
Главное меню	Если вы знаете добавочный номер человека, с которым хотите связаться, пожалуйста, введите его сейчас. Для отдела продаж, пожалуйста, нажмите кнопку 1; для отдела поддержки, нажмите 2; для перехода в каталог нашей компании, нажмите #; для информации о нашем адресе и факсе, пожалуйста, нажмите 3. Для повтора меню - нажмите клавишу 9, или вы можете остаться на линии или нажмите 0 для соединения с оператором.	Приглашение главного меню. Играет немедленно после приветствия. Для вызывающего абонента, приветствие и главное меню звучат как одно приглашение, однако в системе полезно сохранить эти приглашения отдельно.	*mainmenu.wav*
`1`	Пожалуйста ожидайте пока мы соединяем ваш звонок.	Перевод в очередь продаж.	*holdwhileweconnect.wav*
`2`	Пожалуйста ожидайте пока мы соединяем ваш звонок.	Перевод в очередь поддержки.	*holdwhileweconnect.wav*
`#`	н/а	Запуск приложения `Directory()`	н/а
`3`	Наш адрес [адрес]. Наш номер факса [номер факса]., и тд.	Воспроизведение записи, содержащей адрес и информацию о факсе. Возврат абонента в меню подсказок по окончанию.	*faxandaddress.wav*
`0`	Перевод на секретаря. Пожалуйста, ожидайте.	Перевод на ресепшен/оператору.	*transfertoreception.wav*

Шаг или выбор	Образец приглашения	Примечание	Имя файла
`9`	н/а	Повтор. Воспроизвести главное меню (но не приветствие.)	н/а
`t`	н/а	Таймаут. Если вызывающий не выбрал ничего, рассматривать вызов, как если бы абонент набрал `0` (или в некоторых случаях, повторить запрос).	
`i`	Вы сделали неверный выбор. Пожалуйста, попробуйте еще раз.	Абонент нажал недопустимую цифру: воспроизвести главное меню (но не приветствие).	invalid.wav
`_XXX`[a]	н/а	Перевод вызова на добавочный номер.	holdwhileweconnect.wav

[a] Это совпадение с шаблоном должно соответствовать вашему диапазону номеров.

Давайте рассмотрим различные компоненты этого шаблона. Затем мы покажем вам код набора номера, необходимый для его реализации, а также как создать для него подсказки.

Приветствие

Первое, что слышит звонящий, - это два приглашения.

Первая подсказка - это приветствие. Единственное, что должно сделать приветствие, это поприветствовать звонящего. Примерами приветствия могут быть «Спасибо, что позвонили Брайанту, Ван Меггелену и партнерам», «Добро пожаловать в Школу мудрости и дизайна футболок Лейфа» или «Вы позвонили в офис адвокатов Дьюи, Читума и Хоу». Вот и все - выбор для звонящего будет позднее. Это позволяет записывать различные приветствия без необходимости записывать новое меню. Например, в течение нескольких недель в году вы можете пожелать, чтобы в вашем приветствии говорилось «Привет сезона» или что-то в этом роде, но ваше меню менять будет не нужно. Кроме того, если вы хотите воспроизвести другую запись в нерабочее время («Спасибо, что позвонили. Наш офис закрыт»), вы можете использовать разные приветствия, но суть меню остается прежней. Наконец, если вы хотите иметь возможность возвращать вызывающих абонентов в меню из другой части системы, вы не захотите, чтобы они снова услышали приветствие.

Главное меню

Подсказка главного меню - это место, где вы сообщаете своим абонентам о доступных для них вариантах выбора. Вы должны проговаривать их как можно быстрее (без спешки).[2] Когда вы записываете варианты выбора, *всегда сообщайте пользователям о действиях, которые будут предприняты, прежде чем дать им цифровые опции для выполнения этого действия*. Поэтому не говорите «нажмите 1 для отдела продаж», а говорите «для отдела продаж, нажмите 1». Причина в том, что большинство людей не будут обращать должное внимание на подсказку, пока не услышат интересующий их выбор. Как только они услышат свой выбор, вы будете иметь их полное внимание и можете сообщить какую кнопку нажать, чтобы направить их туда, куда они хотят.

Другой момент, который следует учитывать - это порядок размещения выбора. Например, типичный бизнес захочет, чтобы продажи были первым пунктом меню, и большинство абонентов ожидают этого. Важно думать о своих клиентах. Например, большинство людей не будут интересоваться адресом и факсом, поэтому не делайте его первым пунктом.[3] Подумайте о том, чтобы как можно

2 При необходимости, вы можете использовать программу редактирования аудио, такую как Audacity, чтобы удалить тишину и даже немного ускорить запись.

3 На самом деле, мы обычно не рекомендуем это в AC, потому что это добавляется к тому, что абонент будет слышать, и большинство людей отправятся на веб-сайт для получения такого рода информации.

быстрее доставить абонентов в назначенные места при проектировании выбора. Безжалостно вырезайте все, что не является абсолютно необходимым.

Выбор 1

Опция 1 в нашем примере будет простым трансфером. Обычно это относится к ресурсу, расположенному в другом контексте и он, как правило, имеет внутренний добавочный номер, чтобы внутренние пользователи также могли передавать ему вызовы. В этом примере мы собираемся использовать эту опцию для отправки абонентов в очередь под названием `sales`, которая была создана в Главе 13.

Выбор 2

Опция 2 будет технически идентичнан варианту 1. Только пункт назначения будет другим. Этот выбор переведет звонящих в очередь `support`.

Выбор

Хорошо иметь опцию для каталога как можно ближе к началу записи. Многие люди будут использовать каталог, если они знают что он там есть, но не желают слушать всё меню, чтобы узнать о нём. Нетерпеливые люди будут нажимать `0`, поэтому чем раньше вы расскажете им о каталоге, тем выше вероятность того, что они будут им пользоваться, и тем самым уменьшите нагрузку на вашего регистратора/секретаря.

Выбор 3

Если у вас есть опция, которая ничего не делает, кроме воспроизведения записи для вызывающего абонента (например, адрес и информация о факсе), вы можете оставить весь код для этого в том же контексте, что и меню, и просто вернуть вызывающего абонента в подсказку главного меню в конце записи. В общем, такие варианты не так полезны, как хотелось бы думать, поэтому в большинстве случаев вам, вероятно, захочется об этом забыть.

Выбор 9

Очень важно дать звонящему возможность услышать пункты меню снова. Многие люди не будут обращать внимания на все меню, и если вы не дадите им возможность снова услышать возможный выбор, они, скорее всего, нажмут `0`.

Обратите внимание, что вам не нужно воспроизводить приветствие снова, только подсказку главного меню.

Выбор 0

Как указывалось ранее - нравится вам это или нет, это выбор, который предпочтут многие (возможно, большинство) ваших абонентов. Если вы действительно не хотите, чтобы кто-то обрабатывал эти вызовы, вы можете отправить этот добавочный номер в почтовый ящик, но мы не рекомендуем это. Если вы владеете бизнесом, многие из ваших абонентов будут вашими клиентами. Вы хотите, чтобы им было легко связаться с вами. Доверьтесь нам.

Тайм-аут

Многие люди звонят по номеру и не обращают внимание на происходящее. Они знают, что если они просто будут ждать на линии, то в конечном итоге будут переведены на оператора. Или, возможно, они в машинах, и, на самом деле, не могут нажимать кнопки на своих телефонах. В любом случае, сделайте им одолжение. Если они не делают никакого выбора, не беспокойте их и не заставляйте их делать его. Подключите их к оператору.

Неверный (Invalid)

Люди делают ошибки. Это нормально. Неверный обработчик сообщит им, что пункт, который они выбрали, не является допустимым параметром, и вернет в подсказку главного меню, чтобы они могли повторить попытку. Обратите внимание - вы не должны воспроизводить приветствие снова, только подсказку главного меню.

Набор добавочного номера

Если кто-то звонит в вашу систему и знает к какому добавочному номеру он хочет обратиться, ваш автосекретарь должен иметь код для обработки этого.



Хотя Asterisk может обрабатывать наложения между вариантами меню и добавочными номерами (т.е. у вас может быть пункт меню 1 и добавочные номера от 100 до 199), обычно лучше избегать этого перекрытия. В противном случае диалплану всегда придется выжидать межцифровой таймаут всякий раз, когда кто-то нажимает 1, потому что он не будет знать, планируют ли они набрать добавочный номер 123. Тайм-аут между цифрами - это задержка, которую система допускает между цифрами, прежде чем она предположит что введено все число. Этот таймер позволяет вызывающим абонентам иметь достаточно времени для набора многозначного добавочного номера, но также вызывает задержку в обработке однозначных вводов.

Создание вашего автосекретаря

После того, как вы спроектировали автосекретаря, вам нужно сделать три вещи, чтобы заставить его работать должным образом:

- Записать подсказки.
- Построить диалплан для меню.
- Направить входящие каналы в контекст автосекретаря.

Начнем с разговоров о записях.

Запись подсказок

Запись подсказок для телефонной системы является критически важной задачей. Это то, что ваши абоненты услышат, когда будут взаимодействовать с вашей системой и качество и профессионализм этих подсказок отразится на вашей организации.

Asterisk очень гибок в этом отношении и может работать со многими различными аудиоформатами. Мы обнаружили, что в целом наиболее выгодным форматом является WAV. Файлы, сохраненные в этом формате, могут быть разных типов, но с Asterisk будет работать только один тип файлов WAV: файлы должны быть закодированы в 16-битном, 8000 Гц, монофоническом формате.

Рекомендуемый формат файла подсказки

Рекомендуемый нами формат файла WAV полезен для системных подсказок, поскольку его можно легко конвертировать в любой другой, который ваши телефоны могут использовать без искажений, и практически любой компьютер может воспроизводить его без какого-либо специального программного обеспечения. Таким образом, не только Asterisk может легко обработать файл, но также легко работать с ним на ПК (что может быть полезно). Asterisk может обрабатывать и другие форматы файлов и в некоторых случаях они могут быть более подходящими для ваших потребностей, но в целом мы считаем, что с 16-битными 8 кГц файлами WAV проще всего работать и, в большинстве случаев, имеют лучшее возможное качество.

По сути, есть два способа получить подсказки в систему. Одним из них является запись звуковых файлов в студии или на ПК, а затем перемещение этих файлов в систему. Второй способ заключается в записи подсказок непосредственно в систему с помощью телефонного аппарата. Мы предпочитаем второй метод.

Наш совет таков: не зацикливайтесь на сложностях записи звука через ПК или в студии.[4] Это вообще необязательно. Телефонный аппарат будет делать записи превосходного качества, и причины просты: микрофон и электроника в телефоне тщательно разработаны для захвата человеческого голоса в формате, который идеально подходит для передачи по телефонным сетям, и, следовательно, телефонный аппарат также идеально подходит для подсказок. Устройство будет захватывать звук в правильном формате, отфильтровывать фоновые шумы и нормализовать уровень децибел.



Да, должным образом подготовленная студийная подсказка будет лучше, чем подсказка, записанная по телефону, но если у вас нет оборудования или опыта, воспользуйтесь нашим советом и воспользуйтесь телефоном для записи, потому что плохо изготовленная студийная подсказка будет намного хуже, чем подсказка, записанная через телефонный аппарат.

Использование диалплана для создания записей

Простейшим способом записи подсказок является использование приложения `Record()`. Например:

```
[UserServices] ❶

exten => 500,1,Playback(vm-intro)
   same => n,Record(daygreeting.wav)
   same => n,Wait(2)
   same => n,Playback(daygreeting)
   same => n,Hangup

exten => 501,1,Playback(vm-intro)
   same => n,Record(mainmenu.wav)
   same => ... etc ... (создайте код диалплана для каждого приглашения,
                        которое необходимо записать)
```

❶ Для использования в этом контексте вам нужно будет включить его в контекст, в котором ваши варианты наборов входят в диалплан. Итак, в вашем контексте `[LocalSets]` вы должны добавить строку **`include => UserServices`**. В продакшене вам, вероятно, понадобится пароль, чтобы не каждый мог записывать подсказки.

Этот добавочный номер воспроизводит подсказку, издает звуковой сигнал, делает запись и воспроизводит эту запись.[5] Примечательно, что приложение `Record()` принимает в качестве аргумента все имя файла, а `Playback()` исключает расширение типа файла (*.wav*, *.gsm* и т.д.). Это связано с тем, что приложение `Record()` должно знать, в каком формате должна быть сделана запись, а `Playback()` - нет. Вместо этого, `Playback()` автоматчески выбирает лучший доступный формат аудио, основанный на используемом кодеке вашего телефона и форматов, доступных в папке *sounds* (например, если у вас есть файлы *daygreeting.wav* и *daygreeting.gsm* в вашей папке *sounds*, `Playback(daygreeting)` будет выбирать тот, который требует наименьшую нагрузку ЦП для воспроизведения звонящему).

Вам, вероятно, понадобится отдельный добавочный номер для записи каждого из приглашений, возможно, скрытый от ваших обычных внутренних номеров, чтобы избежать стирания ваших текущих подсказок меню. Если количество ваших запросов велико, повторять этот номер с

4 Если вы не являетесь экспертом в этих областях, в таком случае пойдите на это!

5 Приглашение *vm-intro* неидеально (оно просит вас оставить сообщение), но оно достаточно близко для наших целей. По крайней мере, инструкции по использованию верны: нажмите **#**, чтобы завершить запись. После того, как вы освоили запись приглашений, вы можете вернуться назад, записать настраиваемое приглашение и изменить приоритет 1, чтобы отразить более подходящие инструкции для записи собственных приглашений.

небольшими изменениями для каждого будет утомительно, но есть способы обойти это. Мы покажем вам, как сделать вашу оперативную запись более разумной в Главе 17, но пока метод, описанный выше, будет отвечать нашим непосредственным потребностям.

Диалплан

Вот код, необходимый для создания автосекретаря, который мы разработали ранее. Мы часто будем использовать пустые строки перед метками, чтобы облегчить чтение диалплана, но учтите, что наличие пустой строки не означает, что имеется другой номер:

```
[main_menu]

exten => s,1,Verbose(1, Caller ${CALLERID(all)} has entered the auto
attendant)
    same => n,Answer()

;устанавливаем таймер между набором цифр
    same => n,Set(TIMEOUT(digit)=2)

; ожидание в 1 секунду до установки аудио
    same => n,Wait(1)

; Если Пон-Пятн 9-17 переходим на метку daygreeting
    same => n,GotoIfTime(9:00-17:00,mon-fri,*,*?daygreeting:afterhoursgreeting)

; ПРИВЕТСТВИЕ НЕРАБОЧЕГО ВРЕМЕНИ
    same => n(afterhoursgreeting),Background(nightgreeting)
    same => n,Goto(menuprompt)

; ДНЕВНОЕ ПРИВЕТСТВИЕ
    same => n(daygreeting),Background(daygreeting)
    same => n,Goto(menuprompt)

; ПРИВЕТСТВИЕ ГЛАВНОГО МЕНЮ
    same => n(menuprompt),Background(mainmenu)
    same => n,WaitExten(4)    ; больше 4 секунд - уже слишком много
    same => n,Goto(0,1)       ; Если абонент нажал '0'

exten => 1,1,Verbose(1, Caller ${CALLERID(all)} has entered the sales queue)
    same => n,Goto(Queues,7002,1)         ; очередь sales - см. Главу 13

exten => 2,1,Verbose(1, Caller ${CALLERID(all)} has entered the service queue)
    same => n,Goto(Queues,7001,1)         ; очередь service - см Главу 13

exten => 3,1,Verbose(1, Caller ${CALLERID(all)} has requested address and fax
info)
    same => n,Background(faxandaddress)   ; Инф-я - адрес и факс
    same => n,Goto(s,menuprompt)          ; Вернуть абонента в главное меню

exten => #,1,Verbose(1, Caller ${CALLERID(all)} is entering the directory)
    same => n,Directory(default)          ; Направить абонента в каталог.
                         ; Используйте контекст InternalSets для вызова

exten => 0,1,Verbose(1, Caller ${CALLERID(all)} is calling the operator)
    same => n,Dial(SIP/operator)          ; Operator - номер/очередь

exten => i,1,Verbose(1, Caller ${CALLERID(all)} has entered an invalid
selection)
```

```
    same => n,Playback(invalid)
    same => n,Goto(s,menuprompt)

exten => t,1,Verbose(1, Caller ${CALLERID(all)} has timed out)
    same => n,Goto(0,1)

; Вы захотите иметь соответствие шаблону для различных номеров, которые
; позволите набирать внешним абонентам
; НО ТОЛЬКО НЕ ВКЛЮЧАЙТЕ КОНТЕКСТ LOCALSETS ИЛИ ВНЕШНИЕ АБОНЕНТЫ СМОГУТ
; СОВЕРШАТЬ ВЫЗОВЫ ИЗ ВАШЕЙ СИСТЕМЫ, ЧТО БЫ ВЫ ЗДЕСЬ НИ ДЕЛАЛИ, ТЩАТЕЛЬНО
; ПРОТЕСТИРУЙТЕ ЕГО, ЧТОБЫ УБЕДИТЬСЯ, ЧТО ВНЕШНИЕ АБОНЕНТЫ НЕ СМОГУТ НИЧЕГО
; ДЕЛАТЬ, КРОМЕ НАБОРА ВНУТРЕННИХ НОМЕРОВ

exten => _1XX,1,Verbose(1,Call to an extension starting with '1')
    same => n,Goto(InternalSets,${EXTEN},1)
```

Доставка входящих вызовов

Любой вызов в систему будет входить в диалплан в контексте, определенном для любого канала, на который поступил вызов. Во многих случаях это будет контекст с именем `incoming` или `from-pstn` или что-то подобное. Звонки будут поступать либо с добавочным номером (как в случае с DID) или без него (как в случае с традиционной аналоговой линией).

Независимо от названия контекста и от имени внутреннего номера, вы будете отправлять каждый входящий вызов в меню. Вот несколько примеров:

```
[from-pstn] ; для аналоговой линии context=from-pstn
            ; (обычно канал DAHDI)
exten => s,1,Goto(main_menu,s,1)

[incoming]  ; DID, входящий в канал с context=incoming
            ; (PRI, SIP или IAX)
exten => 4169671111,1,Goto(main_menu,s,1)
```

В зависимости от того, как вы конфигурируете свои входящие каналы, вы, как правило, используете приложение `Goto()` если хотите отправлять вызов автосекретарю. Это гораздо лучше чем использование автосекретаря во входящем контексте.

IVR

Мы рассмотрим Интерактивное голосовое меню (IVR) более подробно в Главе 17, но прежде чем мы это сделаем, поговорим о том, что важно для любого IVR. Интеграция базы данных является предметом следующей главы.

Заключение

Автосекретарь может предоставить очень полезную услугу абонентам. Однако, если он не спроектирован и не реализован должным образом, он также может стать барьером для ваших абонентов, который может их отпугнуть. Потратьте время, чтобы тщательно спланировать вашего автосекретаря, и сделать его простым.

Интеграция реляционной базы данных

Нет ничего более раздражающего, чем хороший пример.
- Марк Твен

В этой главе мы рассмотрим интеграцию некоторых возможностей и функций Asterisk в базу данных. Для Linux доступно несколько баз данных, но мы решили ограничить наше обсуждение двумя самыми популярными: PostgreSQL и MySQL.

Мы также объясним, как настроить Linux для подключения к базе данных Microsoft SQL через ODBC; однако конфигурация части Windows/Microsoft выходит за рамки этой книги.

Независимо от того, какую базу данных вы используете, эта глава фокусируется в первую очередь на ODBC-коннекторе, поэтому, если у вас есть некоторое знакомство с подготовкой вашей любимой базы данных ODBC, у вас не должно быть никаких проблем с этой главой.

Интеграция Asterisk с базами данных является одним из фундаментальных аспектов построения крупной кластерной или распределенной системы. Мощь базы данных позволит вам использовать динамически изменяющиеся данные в ваших диалпланах для таких задач, как обмен информацией по массиву систем Asterisk или интеграция с веб-сервисами. Наша любимая функция диалплана, которую мы рассмотрим далее в этой главе - `func_odbc`.

Хотя не для всех развертываний Asterisk потребуются реляционные базы данных, понимание того, как их использовать, открывает сундук с сокровищами, полный новых способов разработки вашего телекоммуникационного решения.

Установка и настройка PostgreSQL и MySQL

В следующих разделах мы покажем, как устанавливать и настраивать PostgreSQL и MySQL как для RHEL, так и для Ubuntu.[1] Рекомендуется устанавливать только одну базу данных одновременно, работая через этот раздел. Выберите базу данных, которая вам наиболее удобна, так как нет неправильного выбора.

Установка PostgreSQL для RHEL

Следующая команда может быть использована для установки сервера PostgreSQL и его зависимостей с консоли RHEL:

```
$ sudo yum install postgresql-server
```

1 В большой, загруженной системе лучше установить базу данных на совершенно отдельный блок от вашей системы Asterisk.

```
...
Install       3 Package(s)
Upgrade       0 Package(s)
Total download size: 6.3 M
Installed size: 29 M
Is this ok [y/N]: y
```

После установки базы данных вы должны ее инициализировать:

```
$ sudo service postgresql initdb
```

Затем запустите базу данных:

```
$ sudo service postgresql start
```

Теперь перейдите в раздел "Настройка PostgreSQL" для получения инструкций о том, как выполнить первоначальную настройку.

Установка PostgreSQL для Ubuntu

Чтобы установить PostgreSQL на Ubuntu выполните следующую команду. Вам также будет предложено установить любые дополнительные пакеты зависимостей. Нажмите Enter чтобы принять список зависимостей, после чего пакеты будут установлены, и PostgreSQL будет автоматически запущен и инициализирован:

```
$ sudo apt-get install postgresql
...
After this operation, 19.1MB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

Теперь перейдите в раздел "Настройка PostgreSQL" для получения инструкций о том, как выполнить первоначальную настройку.

Установка MySQL для RHEL

Чтобы установить MySQL на RHEL, выполните следующую команду. Вам будет предложено установить несколько зависимостей. Нажмите Enter чтобы принять, и будет установлен MySQL сервер и зависимости пакетов:

```
$ sudo yum install mysql-server
...
Install       3 Package(s)
Upgrade       0 Package(s)
Total download size: 9.6 M
Installed size: 27 M
Is this ok [y/N]: y
```

Затем запустите базу данных MySQL, запустив:

```
$ sudo service mysqld start
```

Теперь перейдите в "Настройка MySQL" для выполнения начальной конфигурации.

Установка MySQL для Ubuntu

Для установки MySQL на Ubuntu выполните следующую команду. Вам будет предложено установить несколько зависимостей. Нажмите Enter чтобы согласиться, и MySQL сервер и его пакеты зависимостей будут установлены:

```
$ sudo apt-get install mysql-server
Need to get 24.0MB of archives.
After this operation, 60.6MB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

Во время установки вы будете помещены в мастер настройки, чтобы помочь вам с начальной конфигурацией базы данных. Вам будет предложено ввести новый пароль для пользователя *root*. Введите сложный пароль и нажмите Enter. Затем вас попросят подтвердить пароль. Введите свой сложный пароль еще раз, а затем Enter. Затем вы вернетесь в консоль где будет завершена установка. Теперь будет запущен сервис MySQL.

Теперь перейдите в "Настройка MySQL" для выполнения начальной конфигурации.

Настройка PostgreSQL

Далее создайте пользователя с именем *asterisk*, которого вы будете использовать для подключения к базе данных и управления ею. Вы можете переключиться на пользователя *postgres*, используя следующую команду:

```
$ sudo su - postgres
```



На момент написания этой статьи в RHEL и Ubuntu используется PostgreSQL версии 8.4.x.

Затем запустите следующие команды для создания пользователя *asterisk* в базе данных и установите разрешения:

```
$ createuser -P
Enter name of user to add: asterisk
Enter password for new user:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
CREATE ROLE
```

Теперь отредактируйте файл *pg_hba.conf*, чтобы разрешить пользователю *asterisk*, которого вы только что создали, подключение к серверу PostgreSQL через сокет TCP/IP.

В RHEL этот файл будет расположен по адресу */var/lib/pgsql/data/pg_hba.conf*. На Ubuntu вы найдете его на */etc/postgresql/8.4/main/pg_hba.conf*.

В конце файла замените все строки ниже этой:

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
```

на следующее:

```
# TYPE  DATABASE USER      CIDR-ADDRESS  METHOD
host    all      asterisk  127.0.0.1/32  md5
local   all      asterisk                trust
```

Настройка доступа к базе данных PostgreSQL через IPv6 localhost

Кроме того, на Ubuntu вам, скорее всего, придется добавить следующую строку:

```
host  all   asterisk    ::1/128     md5
```

Без него при “Проверке ODBC-коннектора" может возникнуть следующая ошибка при подключении:

```
[28000][unixODBC]FATAL:no pg_hba.conf entry for host "::1", user "asterisk",
database "asterisk", SSL off
[ISQL]ERROR: Could not SQLConnect
```

Теперь вы можете создать базу данных, которую мы будем использовать в этой главе. Вызовите базу данных *asterisk* и установите владельцем вашего пользователя *asterisk*:

```
$ createdb --owner = asterisk asterisk
CREATE DATABASE
```

Вы можете установить пароль для пользователя *asterisk* следующим образом:

```
$ psql -d template1
template1=# "ALTER USER asterisk WITH PASSWORD 'password'"
template1=# \q
```

Выход из пользователя *postgres*:

```
$ exit
```

Затем перезапустите сервер PostgreSQL. В RHEL:

```
$ sudo service postgresql restart
```



Вам необходимо перезапустить службу PostgreSQL, потому что вы внесли изменения в *pg_hba.conf*, а не потому что вы добавили нового пользователя или сменили пароль.

В Ubuntu:

```
$ sudo /etc/init.d/postgresql restart
```

Вы можете проверить свое соединение с сервером PostgreSQL через TCP/IP, например:

```
$ psql -h 127.0.0.1 -U asterisk
Пароль для пользователя asterisk:

Добро пожаловать в psql 9.1 , интерактивный терминал PostgreSQL.

Тип:  \copyright для условий распространения
      \h для справки по командам SQL
      \? для получения помощи по командам psql
      \g или terminate с точкой с запятой для выполнения запроса
      \q для выхода

asterisk =>
```

Теперь вы готовы перейти к разделу "Установка и настройка ODBC".

Настройка MySQL

Теперь, когда база данных MySQL работает, вы должны защитить свою инсталляцию. Удобно, для этого есть скрипт, который вы можете выполнить, что позволит ввести новый пароль[2] пользователя

2 Если вы установили Ubuntu, значит вы уже установили пароль `root`. Вам нужно будет ввести этот пароль во время выполнения скрипта, после чего он скажет, что вы уже установили пароль `root`, поэтому вам не нужно его менять.

root наряду с некоторыми дополнительными опциями. Скрипт довольно прост, и после ввода и подтверждения пароля *root* вы можете продолжать выбирать значения по умолчанию, если у вас нет конкретной причины.

Выполните следующий скрипт:

```
$ sudo /usr/bin/mysql_secure_installation
```

Затем подключитесь к консоли базы данных, чтобы вы могли создать своего пользователя asterisk и установить разрешения:

```
$ mysql -u root -p
Enter password:
```

После ввода пароля вам будет представлена подсказка `mysql` в консоли. Теперь вы можете создать своего пользователя *asterisk*, выполнив команду `CREATE USER`. `%` - символ шаблона с указанием пользователя *asterisk* может подключаться с любого хоста и `IDENTIFIED BY` паролем *`some_secret_password`* (который вы явно должны изменить). Обратите внимание на конечную точку с запятой:

```
mysql> CREATE USER 'asterisk'@'%' IDENTIFIED BY 'some_secret_password';
Query OK, 0 rows affected (0.00 sec).
```

Давайте также создадим исходную базу данных, которую вы будете использовать в этой главе:

```
mysql> CREATE DATABASE asterisk;
Query OK, 1 rows affected (0.00 sec.)
```

Теперь, когда вы создали своего пользователя и базу данных, необходимо назначить разрешения для пользователя *asterisk* для доступа к базе данных *asterisk*:

```
mysql> GRANT ALL PRIVILEGES ON asterisk.* TO 'asterisk'@'%';
Query OK, 0 rows affected (0.00 sec)
```

Наконец, выйдите из консоли и убедитесь, что ваши права верны, выполнив вход в базу данных *asterisk* в качестве пользователя *asterisk*:

```
mysql> exit
Bye
# mysql -u asterisk -p asterisk
Enter password:

mysql>
```

Этот пароль вам понадобится при настройке и тестировании ODBC-коннектора, поэтому сохраните его. Теперь вы готовы перейти к следующему разделу.

Установка и настройка ODBC

ODBC-коннектор представляет собой уровень абстракции базы данных, который позволяет Asterisk обмениваться данными с широким спектром баз данных, не требуя от разработчиков создания отдельного коннектора для каждой поддерживаемой Asterisk базы данных. Это экономит много усилий по разработке и обслуживанию кода. Существует небольшая стоимость производительности, потому что мы добавляем еще один прикладной уровень между Asterisk и базой данных, но это можно смягчить с помощью правильной конструкции и стоимости этого, когда вам нужны мощные и гибкие возможности базы данных в вашей системе Asterisk.

Перед установкой коннектора в Asterisk вам необходимо установить ODBC в сам Linux.

Чтобы установить драйверы ODBC, используйте одну из следующих команд.

На RHEL:

```
$ sudo yum install unixODBC unixODBC-devel libtool-ltdl libtool-ltdl-devel
```

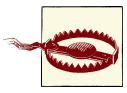
На Ubuntu:

```
$ sudo apt-get install unixODBC unixODBC-dev
```



См. Главу 3 для просмотра матрицы пакетов, которые вы должны установить.

Вам также потребуется установить *unixODBC* пакет разработки, поскольку Asterisk использует его для создания модулей ODBC, которые мы будем использовать в этой главе.



Драйверы *unixODBC*, поставляемые с дистрибутивами, часто отличаются от официально выпущенных версий, доступных на веб-сайте unixODBC.
Если у вас есть проблемы со стабильностью при использовании *unixODBC*, вам может потребоваться установить из исходников. Просто сначала удалите драйверы *unixODBC* через диспетчер пакетов, а затем обновите пути в файле */etc/odbcinst.ini*.

Чтобы установить ODBC-коннектор MySQL на RHEL:

```
$ sudo yum install mysql-connector-odbc
```

Для установки ODBC-коннектора PostgreSQL на RHEL:

```
$ sudo yum install postgresql-odbc
```

Для установки ODBC-коннектора PostgreSQLна Ubuntu:

```
$ sudo apt-get install odbc-postgresql
```

Или установить ODBC-коннектор MySQL на Ubuntu:

```
$ sudo apt-get install libmyodbc
```

Настройка ODBC для PostgreSQL

Конфигурация для драйвера ODBC PostgreSQL выполняется в файле */etc/odbcinst.ini*.

В RHEL файл по умолчанию уже содержит некоторые данные, в том числе для PostgreSQL, поэтому просто убедитесь, что данные существуют. Файл будет выглядеть следующим образом:

```
[PostgreSQL]
Description       = ODBC for PostgreSQL
Driver            = /usr/lib/psqlodbc.so
Setup             = /usr/lib/libodbcpsqlS.so
Driver64          = /usr/lib64/psqlodbc.so
Setup64           = /usr/lib64/libodbcpsqlS.so
FileUsage         = 1
```

В Ubuntu файл */etc/odbcinst.ini* будет пустым, поэтому вам нужно будет добавить данные в этот файл конфигурации. Добавьте в *odbcinst.ini* следующее:

```
[PostgreSQL]
Description       = ODBC for PostgreSQL
Driver            = /usr/lib/odbc/psqlodbca.so
Setup             = /usr/lib/odbc/libodbcpsqlS.so
FileUsage         = 1
```



В 64-битных системах , вам нужно будет изменить путь библиотек из */usr/lib* на */usr/lib64* , чтобы получить доступ к файлам с правильной библиотекой.

В любом случае вы можете использовать *cat > /etc/odbcinst.ini* для записи чистого файла конфигурации, как это было сделано в других главах. Просто используйте Ctrl + D, чтобы сохранить файл, как только закончите.

Убедитесь, что система видит драйвер, выполнив следующую команду. Она должна вернуть имя метки `PostgreSQL`, если все хорошо:

```
$ odbcinst -q -d
[PostgreSQL]
```

Затем настройте файл */etc/odbc.ini*, который используется для создания идентификатора, который Asterisk будет использовать для ссылки на эту конфигурацию. Если в любой момент в будущем вам нужно изменить базу данных на что-то еще, то нужно будет просто изменить этот файл, чтобы Asterisk продолжал указывать на то же место:[3]

```
[asterisk-connector]
Description       = PostgreSQL connection to 'asterisk' database
Driver            = PostgreSQL
Database          = asterisk
Servername        = localhost
Port              = 5432
Protocol          = 8.1
ReadOnly          = No
RowVersioning     = No
ShowSystemTables  = No
ShowOidColumn     = No
FakeOidIndex            = No
ConnSettings            =
```

Настройка ODBC для MySQL

Конфигурация для MySQL ODBC-драйвера находится в файле */etc/odbcinst.ini*.

В RHEL файл по умолчанию уже содержит некоторые данные, в том числе для MySQL, но он должен быть раскоментирован и требует нескольких изменений. Замените существующий текст следующим:

```
[MySQL]
Description       = ODBC for MySQL
Driver            = /usr/lib/libmyodbc5.so
Setup             = /usr/lib/libodbcmyS.so
Driver64          = /usr/lib64/libmyodbc5.so
Setup64           = /usr/lib64/libodbcmyS.so
FileUsage         = 1
```

В Ubuntu */etc/odbcinst.ini* будет пустым, поэтому вам нужно будет добавить данные в этот файл конфигурации. Добавьте в *odbcinst.ini* следующее:

```
[MySQL]
Description       = ODBC for MySQL
Driver            = /usr/lib/odbc/libmyodbc.so
Setup             = /usr/lib/odbc/libodbcmyS.so
FileUsage         = 1
```

3 Да, это слишком многословно. Единственные записи, которые вам действительно нужны, - это `Driver`, `Database` и `Servername`



В 64-битных системах , вам нужно будет изменить путь библиотек из */usr/lib* на */usr/lib64*, чтобы получить доступ к файлам с правильной библиотекой.

В любом случае вы можете использовать *cat > /etc/odbcinst.ini* для записи чистого файла конфигурации, как это было сделано в других главах. Просто используйте Ctrl + D, чтобы сохранить файл, как только вы закончите.

Убедитесь, что система видит драйвер, выполнив следующую команду. Она должна вернуть имя метки `MySQL`, если все хорошо:

```
$ odbcinst -q -d
[MySQL]
```

Затем настройте файл */etc/odbc.ini*, который используется для создания идентификатора, который Asterisk будет использовать для ссылки на эту конфигурацию. Если в любой момент в будущем вам нужно изменить базу данных на что-то еще, то нужно будет просто перенастроить этот файл, чтобы Asterisk продолжал указывать на то же место:

```
[asterisk-connector]
Description       = MySQL connection to 'asterisk' database
Driver            = MySQL
Database          = asterisk
Server            = localhost
Port              = 3306
Socket            = /var/lib/mysql/mysql.sock  (1)
```

(1) Местоположение этого файла может быть другим, поэтому вам может потребоваться найти его в вашей системе.



На Ubuntu местоположение сокета - */var/run/mysqld/mysqld.sock*.

Настройка ODBC для Microsoft SQL

Подключение к Microsoft SQL (MS SQL) аналогично подключению к MySQL или PostgreSQL, как мы обсуждали ранее. Конфигурация MS SQL выходит за рамки этой книги, но следующая информация позволит настроить ваш Asterisk для подключения к базе данных MS SQL после включения соответствующих разрешений для этой базы данных.

Чтобы подключиться к MS SQL, вам необходимо установить драйверы FreeTDS с помощью диспетчера пакетов (или путем компиляции исходных файлов, доступных по адресу http://www.freetds.org).

На RHEL:

```
$ sudo yum install freetds
```

On Ubuntu:

```
$ sudo apt-get install tdsodbc
```

После установки драйверов вам необходимо поправить файл */etc/odbcinst.ini*, в котором сообщается системе, где находятся файлы драйверов.

Вставьте следующий текст в */etc/odbcinst.ini* помощью вашего любимого текстового редактора или с помощью следующей команды:

```
$ sudo cat> /etc/odbcinst.ini
```

```
[FreeTDS]
Description = ODBC для Microsoft SQL
Driver= / usr / lib / i386-linux-gnu / odbc / libtdsodbc.so
UsageCount  = 1
Threading   = 2
Ctrl+D
```



Значение драйвера должно быть */usr/lib/x86_64-linux-gnu/* если вы используете 64-разрядную установку. Если вы скомпилировали из исходных кодов, файлы могут быть расположены в каталоге */usr/local/lib/* или (если вы скомпилировали в 64-разрядной системе) */usr/local/lib64/*.

Убедитесь, что система может видеть драйвер, выполнив следующую команду. Она должна вернуть имя метки `FreeTDS`, если все в порядке:

```
$ odbcinst -q -d
[FreeTDS]
```

После настройки драйверов вам необходимо изменить */etc/odbc.ini* для управления подключением к базе данных:

```
[asterisk-connector]
Description       = MS SQL connection to 'asterisk' database
Driver            = FreeTDS
Database          = asterisk
Server            = 192.168.100.1
Trace             = No
TDS_Version       = 7.0
Port              = 1433
```

В следующем разделе вы сможете проверить свое соединение с MS SQL-сервером.

Проверка ODBC-коннектора

Теперь убедитесь, что вы можете подключиться к своей базе данных с помощью приложения *isql*. В приложении *echo* выведите `select 1` и перенаправьте в *isql*, который затем будет подключаться с использованием секции `asterisk-connector`, добавленной в */etc/odbc.ini*. Вы должны получить следующий результат (или по крайней мере что-то подобное; мы ищем результат `1 rows fetched`):

```
$ echo "select 1" | isql -v asterisk-connector asterisk some_secret_password ❶
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
+------------+
| ?column?   |
+------------+
| 1          |
+------------+
SQLRowCount returns 1
1 rows fetched
```

❶ Пароль, который вы использовали при создании пользователя `'asterisk'@` в разделе "Настройка MySQL".

Теперь, когда unixODBC установлен, настроен и проверен на работоспособность, вам необходимо перекомпилировать Asterisk, чтобы модули ODBC были созданы и установлены. Вернитесь в исходный каталог Asterisk и запустите скрипт *./configure*, чтобы он знал, что вы установили unixODBC:

```
$ cd ~ / src / asterisk-complete / asterisk / 11
$ ./configure
$ make menuselect
$ make install
```



Почти все в этой главе включено по умолчанию. Вы должны запустить *make menuselect*, чтобы убедиться, что модули, связанные с ODBC, включены. К ним относятся `cdr_odbc`, `cdr_adaptive_odbc`, `func_odbc`, `func_realtime`, `pbx_realtime`, `res_config_odbc` и `res_odbc`. Для голосовой почты, хранящейся в базе данных ODBC, обязательно выберите `ODBC_STORAGE` в меню *Voicemail Build Options* (Параметры сборки голосовой почты). Вы можете проверить, что модули существуют в каталоге */usr/lib/asterisk/modules*.

Как только Linux сможет использовать ODBC, Asterisk также сможет использовать ODBC. Суть в том, что ваш ODBC должен работать под Linux без ошибок, прежде чем вы погрузитесь в часть процесса настройки Asterisk.

Компиляция модулей ODBC для Asterisk

Теперь, когда ODBC установлен и протестирован, вам необходимо скомпилировать соответствующие модули для Asterisk.

Самый простой способ сделать это - просто вернуться в каталог установки («Asterisk»), повторно запустить *sudo ./configure*, а затем *sudo make install* и система распознает что зависимости для ODBC теперь выполнены и автоматически скомпилирует и установит соответствующие модули для Asterisk. Если хотите, то можете запустить *make menuselect* и убедиться, что функция диалплана `func_odbc` теперь отображается как `[*] func_odbc`, а модуль ресурсов `res_odbc` также указан как `[*]res_odbc`.

Настройка res_odbc для подключения Asterisk к ODBC

Подключения Asterisk к ODBC настраиваются в файле *res_odbc.conf*, расположенном в */etc/asterisk*. В *res_odbc.conf* устанавливаются параметры, которые различные модули Asterisk будут использовать для подключения к базе данных.



Опции `pooling` и `limit` являются весьма полезными для баз данных MS SQL и Sybase. Они позволяют устанавливать несколько соединений (вплоть до `limit` подключений) с базой данных, гарантируя при этом, что каждое соединение имеет только одну инструкцию, выполняемую одновременно (это связано с ограничением в протоколе, используемом этими серверами баз данных).

Измените *res_odbc.conf* следующим образом:

```
[asterisk]
enabled => yes
dsn => asterisk-connector
username => asterisk
password => welcome
pooling => no
limit => 1
pre-connect => yes
```

Параметр `dsn` указывает на подключение к базе данных, которое вы настроили в */etc/odbc.ini*, а параметр `pre-connect` сообщает Asterisk открыть и поддерживать соединение с базой данных при загрузке модуля *res_odbc.so*. Это снижает некоторые издержки, связанные с многократной установкой и разрывом соединения с базой данных.

После того как вы настроили *res_odbc.conf*, запустите Asterisk и проверьте соединение с базой данных с командой CLI *odbc show*:

```
*CLI> odbc show
ODBC DSN Settings
-----------------
  Name: asterisk
  DSN: asterisk-connector
    Last connection attempt: 1969-12-31 19:00:00
  Pooled: No
  Connected: Yes
```

Управление базами данных

Хотя в этой книге не описывается управление базами данных, по крайней мере стоит кратко отметить некоторые приложения, которые можно использовать для управления базами данных. Некоторые из них существуют, некоторые из них являются локальными клиентскими приложениями, запущенными с Вашего компьютера и подключенными к базе данных, а другие - веб-приложениями, которые могут обслуживаться с того же компьютера, на котором запущена сама база данных, что позволяет подключаться удаленно.

Некоторые из тех, которые мы использовали, включают в себя:

- phpMyAdmin
- MySQL Workbench
- pgAdmin
- Navicat (коммерческий)

Устранение неполадок с базой данных

При работе с соединениями базы данных ODBC и Asterisk важно помнить, что соединение ODBC абстрагирует часть информации, переданной между Asterisk и базой данных. В случаях, когда все работает не так, как ожидалось, вам может потребоваться включить логирование на вашей платформе базы данных, чтобы узнать, что Asterisk отправляет в базу (например, какие инструкции `SELECT`, `INSERT` или `UPDATE` запускаются из Asterisk), что база данных видит и почему может отклонять инструкции.

Например, одной из наиболее распространенных проблем, возникающих при интеграции базы данных ODBC, является некорректно определенная таблица или отсутствующий столбец, существование которого ожидает Asterisk. Хотя большие успехи были сделаны в виде адаптивных модулей, не все части Asterisk являются адаптивными. В случае хранилища голосовой почты ODBC вы, возможно, пропустили столбец, такой как `flag`, который является новым столбцом, не найденным в версиях Asterisk до 11.[4] Чтобы выяснить, почему ваши данные не записываются в базу данных, как ожидалось, вы должны включить логирование операторов на стороне базы данных, а затем определить, какая инструкция выполняется и почему база данных отклоняет ее.

4 На самом деле это была проблема, с которой столкнулся один из авторов во время работы над этой книгой, и столбец флага был найден путем просмотра журнала операторов во время тестирования PostgreSQL.

Инъекция SQL

Безопасность всегда учитывается при создании сетевых приложений, и безопасность базы данных не является исключением.

В случае с Asterisk вы должны подумать о том, какие ввходные данные принимаете от пользователей (как правило то, что они могут отправить в диалплан), и работать над дезинфекцией этих данных, чтобы убедиться, что вы разрешаете только символы, которые допустимы для вашего приложения. Например, типичный телефонный вызов допускал бы только цифры в качестве ввода (и, возможно, символы * и #), поэтому не было бы причин принимать какие-либо другие символы. Имейте в виду, что протокол SIP допускает больше, чем просто числа как часть адреса, поэтому не предполагайте, что кто-то, пытающийся скомпрометировать вашу систему, ограничится одними цифрами.

Небольшое дополнительное время, затрачиваемое на дезинфекцию ваших разрешенных входных данных, улучшит безопасность вашего приложения.

Мощь вашего диалплана с func_odbc

Функция диалплана `func_odbc`, возможно крутейшая[5] и самая мощная функция диалплана в Asterisk. Она позволяет вам определять и использовать относительно простые функции в вашем диалплане, которые будут извлекать информацию из баз данных по мере обработки вызовов. Существуют всевозможные способы их использования, такие как управление пользователями или возможность совместного использования динамической информации в кластере машин Asterisk. Мы не будем утверждать, что это упростит проектирование и создание кода диалплана, но мы пообещаем, что это позволит вам добавить совершенно новый уровень мощности в ваш диалплан, особенно если вы любите работать с базами данных. До сих пор мы не можем думать о тех, кто не любит `func_odbc`.

`func_odbc` позволяет определять SQL-запросы, которыми назначаются имена функций. По сути, вы создаете пользовательские функции, которые получают результаты, выполняя запросы к базе данных. В *func_odbc.conf* вы указываете отношения между именами функций, которые вы создаете, и операторами SQL, которые хотите выполнить. Обращаясь к названным функциям в диалплане, вы можете получать и обновлять значения в базе данных.

Чтобы вы были в хорошем настроении для дальнейшего, мы хотим, чтобы вы представили себе сэндвич Dagwood.[6]

Можете ли вы передать общий опыт такой вещи, показывая кому-то изображение помидора или размахивая кусочком сыра? Вряд ли. Это та головоломка, с которой мы столкнулись, когда пытались дать полезные примеры того, почему `func_odbc` настолько силен. Итак, мы решили создать целый сэндвич для вас. Это довольно полный рот, но после нескольких укусов это арахисовое масло и желе никогда не будут одинаковыми.

Отношения файлов конфигурации ODBC

5 Лучшая из лучших.

6 И если вы не знаете, что такое Dagwood, для этого есть Википедия. Я не настолько стар.

Чтобы Asterisk мог использовать ODBC из диалплана, несколько файлов должны выстроиться в линию. Рисунок 16-1 пытается дать визуальное понимание этого. Вы, вероятно, найдете эту диаграмму более полезной, как только поработаете с примерами в следующих разделах.

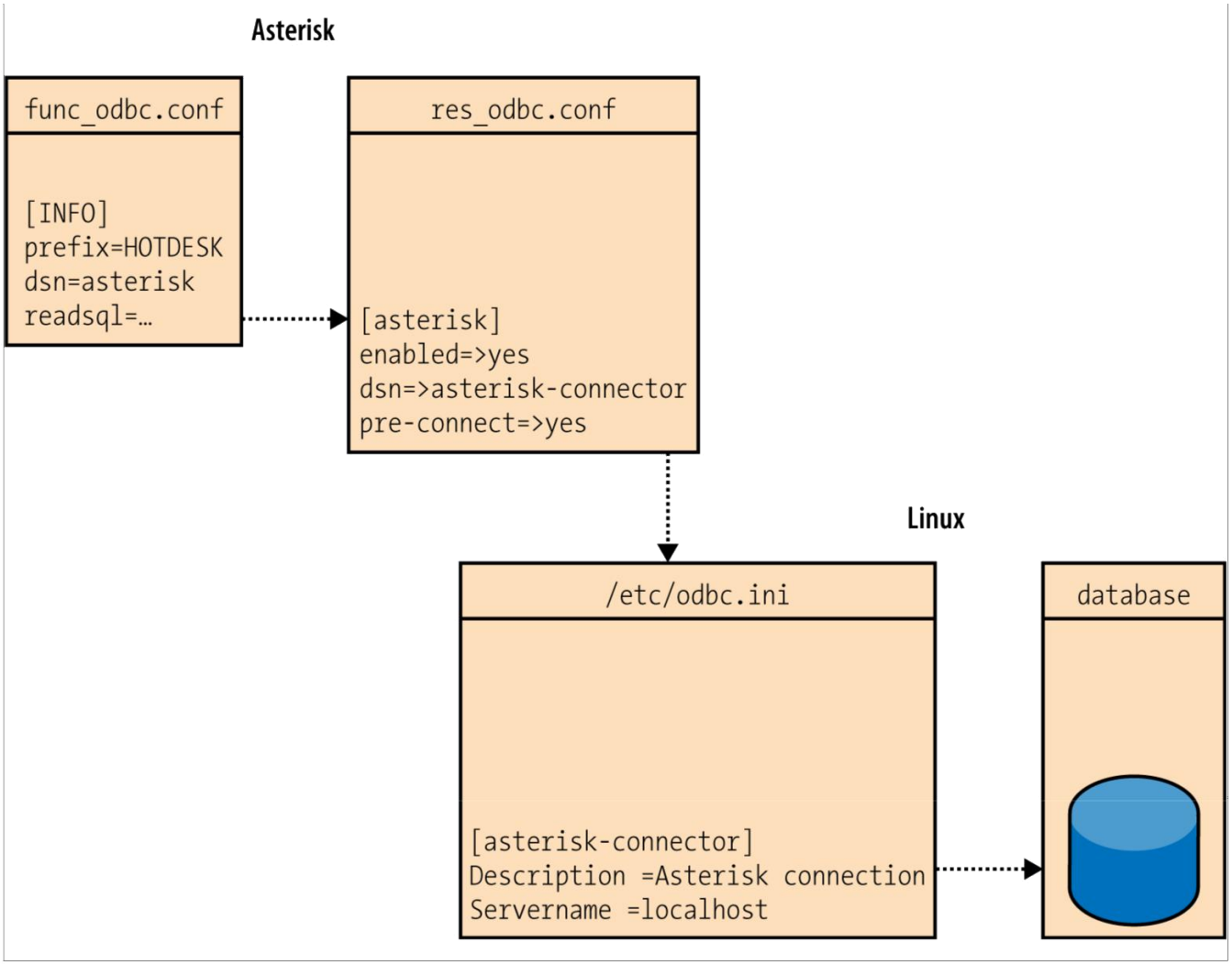


Рисунок 16-1. Взаимоотношения между func_odbc.conf, res_odbc.conf, /etc/odbc.ini (unixODBC) и соединением с базой данных

Нежное введение в func_odbc

Прежде чем мы углубимся в `func_odbc`, мы чувствуем, что история в порядке.

Самое первое использование `func_odbc`, которое случилось когда его автор еще писал его, также является хорошим введением в его использование. Клиент одного из авторов модуля отметил, что некоторые люди, вызывающие его коммутатор, нашли способ совершать бесплатные звонки с его системы. Хотя его конечной целью было изменить свой диалплан, чтобы избежать этих проблем, он нуждался в черном списке определенных номеров вызывающих абонентов, в то же время, он хотел использовать для этого базы данных сервера Microsoft SQL.

За некоторыми исключениями это фактический диалплан:

```
[span3pri]
exten => _50054XX,1,NoOp()
    same => n,Set(CDR(accountcode)=pricall)
    ; Этот callerID присутствует в базе данных?
    same => n,GotoIf($[${ODBC_ANIBLOCK(${CALLERID(number)})}]?busy)
    same => n(dial),Dial(DAHDI/G1/${EXTEN})
    same => n(busy),Busy(10) ; Да, вы находитесь в черном списке.
    same => n,Hangup
```

Этот диалплан, в двух словах, передает все вызовы другой системе для целей маршрутизации, за исключением вызовов, чьи CallerID находятся в черном списке. Вызовы, входящие в эту систему, имеют блок из 100 семизначных идентификаторов DID. Вы заметите, что используется функция диалплана, которую вы не найдете ни в одной из функций, которые поставляются с Asterisk: `ODBC_ANIBLOCK()`. Эта функция была определена в другом файле конфигурации *func_odbc.conf*:

```
[ANIBLOCK]
dsn=telesys
readsql=SELECT IF(COUNT(1)>0, 1, 0) FROM Aniblock WHERE NUMBER='${ARG1}'
```

Итак, ваша функция `ODBC_ANIBLOCK()`[7] подключается к источнику данных в *res_odbc.conf* с именем `telesys` и выбирает количество записей, которые имеют `NUMBER`, указанный в аргументе, который (ссылаясь на наш диалплан выше) является CallerID. Номинально эта функция должна возвращать либо `1` (указывающий CallerID присутствует в таблице Aniblock), либо `0` (если это не так). Это значение также напрямую оценивается как `true` или `false`, что означает что нам не нужно использовать выражение в нашем диалплане для усложнения логики.

И это, в двух словах, о `func_odbc`: написание пользовательских функций диалплана, которые возвращают результат из базы данных. Далее, более подробный пример того, как можно использовать `func_odbc`.

Веселимся с func_odbc: горячий стол

Хорошо, вернемся к сэндвичу Dagwood, который мы обещали.

Мы полагаем, что значение `func_odbc` станет для вас очень ясным если вы выполните следующий пример, который приведет к созданию новой функции в вашей системе Asterisk, которая сильно зависит от `func_ocbc`.

Представьте себе небольшую компанию с отделом продаж из пяти человек, которым приходится делить два стола. Это не так жестоко, как кажется, потому что эти люди проводят большую часть своего времени в дороге и каждый из них находится в офисе не более одного дня в неделю.

Тем не менее, когда они попадают в офис, то хотели бы чтобы система знала, за каким столом они сидят, чтобы их звонки могли быть направлены туда. Кроме того, босс хочет иметь возможность отслеживать когда они находятся в офисе и контролировать привилегии вызова с этих телефонов, когда никого нет.

Эта потребность обычно решается тем, что называется горячий стол. Мы создали один для вас, чтобы показать вам силу `func_odbc`.

Давайте начнем с простых вещей и создадим два настольных телефона в файле *sip.conf*:

```
;sip.conf
;HOT DESK ТЕЛЕФОНЫ
[hot_desk_phone](!) ;шаблон
type = friend
host = dynamic
context = hotdesk
qualify = yes

[0000FFFF0003](hot_desk_phone)      ; первый стол
secret = this_phone_needs_a_good_password

[0000FFFF0004] (hot_desk_phone)     ; второй стол
secret = this_phone_also_needs_a_secret
```

7 Мы используем функцию SQL `IF()`, чтобы убедиться, что мы возвращаем значение `0` или `1`. Это работает на MySQL 5.1 или выше. Если она не работает в вашей установке SQL, вы также можете проверить возвращаемый результат в диалплане, используя функцию `IF()`.

```
; КОНЕЦ ТЕЛЕФОНОВ HOT DESK
```

Эти два настольных телефона входят в диалплан в контексте `hotdesk` в *extensions.conf*, который мы вскоре определим. Если вы хотите, чтобы эти устройства действительно работали, вам, конечно же, нужно будет установить соответствующие параметры в самих устройствах, но мы это рассмотрели в Главе 5. Если вы создаете эти телефоны в своей лаборатории, имена устройств, которые мы используем, нужно заменить на то, что вам лучше всего подходит для ваших нужд (мы рекомендуем использовать MAC-адрес).

Это все для *sip.conf*. У нас есть два куска хлеба, которые пока не сэндвич.

Теперь давайте создадим часть базы данных (мы предполагаем, что у вас есть база данных ODBC, созданная и работающая, как описано в предыдущих частях этой главы). Сначала подключитесь к консоли базы данных.

Для PostgreSQL:

```
$ sudo su - postgres
$ psql -U asterisk -h localhost asterisk
Password:
```

Затем создайте таблицу со следующим битом SQL:

```
CREATE TABLE ast_hotdesk
(
  id serial NOT NULL,
  extension int8,
  first_name text,
  last_name text,
  cid_name text,
  cid_number varchar(10),
  pin int4,
  context text,
  status bool DEFAULT false,
  "location" text,
  CONSTRAINT ast_hotdesk_id_pk PRIMARY KEY (id)
)
WITHOUT OIDS;
```

Для MySQL:

```
$ mysql -u asterisk -p asterisk
Enter password:
```

Затем создайте таблицу со следующим битом SQL:

```
CREATE TABLE ast_hotdesk
(
  id serial NOT NULL,
  extension int8,
  first_name text,
  last_name text,
  cid_name text,
  cid_number varchar(10),
  pin int4,
  context text,
  status bool DEFAULT false,
  location text,
  CONSTRAINT ast_hotdesk_id_pk PRIMARY KEY (id)
);
```

Информация о таблице приведена в Таблице 16-1.

Таблица 16-1. Резюме таблицы ast_hotdesk

Имя столбца	Тип столбца
id	серийный, автоувеличивающийся
extension	целочисленный
first_name	текст
last_name	текст
cid_name	текст
cid_number	10-символьная переменная
pin	целочисленная
context	текст
status	логическая, по-умолчанию `false`
location	текст

После этого заполните базу данных следующей информацией (некоторые из значений, которые вы увидите, на самом деле изменятся только после выполнения работы диалплана, но мы включим их здесь в качестве примера).

На консоли PostgreSQL запустите следующие команды:

```
asterisk => INSERT INTO ast_hotdesk ('extension', 'first_name', 'last_name', \
'cid_name', 'cid_number', 'pin', 'context', 'location') \
VALUES (1101, 'Leif', 'Madsen', 'Leif Madsen', '4165551101', '555',\
'longdistance' , '0000FFFF0003');
```

На консоли MySQL запустите следующие команды:

```
mysql> INSERT INTO ast_hotdesk (extension, first_name, last_name, cid_name,
cid_number, pin, context, status, location)
VALUES (1101, 'Leif', 'Madsen', 'Leif Madsen',
'4165551101', '555', 'longdistance', 1, '0000FFFF0003');


mysql> INSERT INTO ast_hotdesk (extension, first_name,
last_name, cid_name, cid_number, pin, context)
VALUES (1104,'Mark','Spencer','Mark Spencer',
'4165551104','558','international');
```

Повторите эти команды, изменив значения `VALUES` по мере необходимости, для всех записей, которые вы хотите иметь в базе данных.[8] После ввода данных образца вы можете просмотреть данные в таблице `ast_hotdesk`, запустив простой оператор `SELECT` из консоли базы данных:

```
mysql> SELECT * FROM ast_hotdesk;
```

которая может дать вам что-то вроде следующего вывода:

```
+----+-----------+------------+--------------+------------------+
| id | extension | first_name | last_name    | cid_name         |
+----+-----------+------------+--------------+------------------+
|  1 |      1101 | Leif       | Madsen       | Leif Madsen      |
|  2 |      1104 | Mark       | Spencer      | Mark Spencer     |
|  3 |      1105 | Matt       | Jordan       | Matt Jordan      |
|  4 |      1102 | Jim        | Van Meggelen | Jim Van Meggelen |
```

8 Обратите внимание, что в первом примере пользователю присваивается статус `1` и местоположение, в то время как во втором примере пользователю не задается значение для этих полей.

```
|  5 |      1103 | Russell    | Bryant       | Russell Bryant   |
+----+-----------+------------+--------------+------------------+
+------------+------+--------------+--------+--------------+
| cid_number | pin  | context      | status | location     |
+------------+------+--------------+--------+--------------+
| 4165551101 |  555 | longdistance |      1 | 0000FFFF0003 |
| 4165551104 |  558 | international|      0 | NULL         |
| 4165551105 |  559 | local        |      0 | NULL         |
| 4165551102 |  556 | longdistance |      0 | NULL         |
| 4165551103 |  557 | local        |      0 | NULL         |
+------------+------+--------------+--------+--------------+
```

У нас есть приправы, так что давайте перейдем к нашему диалплану. Здесь и будет происходить волшебство.

Где-то в *extensions.conf* нам необходимо создать контекст `hotdesk`. Для начала давайте определим внутерний номер-шаблон, который позволит пользователям входить в систему:

```
; extensions.conf
; Возможность горячего стола
[hotdesk]
; Вход в систему горячего стола
exten => _99110[1-5],1,NoOp()
   same => n,Set(E=${EXTEN:2})      ; отделить ведущие 99
   same => n,Verbose(1,Hot Desk Extension ${E} is changing status)
   same => n,Verbose(1,Checking current status of extension ${E})
   same => n,Set(${E}_STATUS=${HOTDESK_INFO(status,${E})})
   same => n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
```

Мы еще не закончили писать этот код, но нам нужно отойти на несколько страниц, чтобы обсудить, где мы находимся до сих пор.

Когда торговый агент садится за стол, он регистрируется, набрав 99 плюс его собственный добавочный номер. В этом случае мы разрешили номерам с 1101 по 1105 войти в систему совпадением шаблонов `_99110[1-5]`. Вы можете так же легко сделать это менее ограничительным, используя `_9911XX` (с 1100 по 1199). Это расширение использует `func_odbc` для выполнения поиска с помощью функции диалплана `HOTDESK_INFO()`. Эта настраиваемая функция (которую мы определим в файле *func_odbc.conf*) выполняет оператор SQL и возвращает все, что извлекается из базы данных.

Мы бы определили новую функцию `HOTDESK_INFO()` в *func_odbc.conf* следующим образом:

```
[INFO]
prefix=HOTDESK
dsn=asterisk
readsql=SELECT ${ARG1} FROM ast_hotdesk WHERE extension = '${ARG2}'
```

Это много всего за несколько строк. Давайте быстро прикроем их, прежде чем двигаться дальше.[9]

Прежде всего, `prefix` является необязательным (префикс по умолчанию - `'ODBC'`). Это означает, что если вы не определяете префикс, Asterisk добавляет `'ODBC'` к имени функции (в данном случае `INFO`), что означает, что эта функция станет `ODBC_INFO()`. Это не очень хорошо описывает, что делает функция, поэтому может быть полезно назначить префикс, который помогает связать ваши функции ODBC с задачами, которые они выполняют. Мы выбрали `'HOTDESK'`, что означает, что эта настраиваемая функция будет называться `HOTDESK_INFO()` в диалплане.

9 Вы должны перезагрузить диалплан (*dialplan reload*) и `func_odbc` (*module reload func_odbc.so*) и протестировать диалплан до сих пор (наберите **`991101`** с одного из аппаратов, которые вы назначили этому контексту). Убедитесь, что уровень детализации консоли равен по крайней мере `3` (*core set verbose 3*), так вы сможете увидеть как этот диалплан работает в консоли (вызов этого диалплана будет возвращать быструю занятость, даже если он работает успешно).



Причина, по которой `prefix` отделен, заключается в том, что автор модуля хотел уменьшить возможные коллизии с существующими функциями диалплана. Цель префикса состояла в том, чтобы разрешить несколько копий одной и той же функции, подключенной к различным базам данных, для многозадачных систем Asterisk. Мы, авторы, были немного более либеральны в использовании префикса, чем первоначально предполагал разработчик.

Атрибут `dsn` указывает Asterisk какое соединение использовать от *res_odbc.conf*. Поскольку в базе данных можно настроить несколько подключений к базе данных *res_odbc.conf*, мы укажем какой из них следует использовать здесь. На Рисунке 16-1 мы показываем взаимосвязь между различными конфигурациями файлов и тем, как они ссылаются на цепочку для подключения к базе данных.



Файл *func_odbc.conf.sample* в исходниках Asterisk содержит дополнительную информацию о том, как работать с несколькими базами данных и контролировать чтение и запись информации для различных соединений DSN. В частности, аргументы `readhandle`, `writehandle`, `readsql` и `writeql` предоставят вам большую гибкость для интеграции и управления базами данных.

Наконец, мы определяем наш оператор SQL с атрибутом `readsql`. Функции диалплана можно вызывать в двух разных форматах: один для извлечения информации и один для настройки информации. Атрибут `readsql` используется при вызове функции `HOTDESK_INFO()` с форматом извлечения (мы могли бы выполнить отдельный оператор SQL с атрибутом `writesql`, мы обсудим формат для этого атрибута немного позже в этой главе).

Чтение значений из этой функции примет этот формат в диалплане:

```
exten => s,n,Set(RETURNED_VALUE=${HOTDESK_INFO(status,1101)})
```

Это вернет значение, находящееся в базе данных в столбце `status`, где столбец `extension` равен `1101`. `status` и `1101`, которые мы передаем функции `HOTDESK_INFO()` затем помещаются в оператор SQL, присвоенный атрибуту `readsql`, доступный как `${ARG1}` и `${ARG2}` соответственно. Если бы мы допустили третий вариант, он был бы доступен как `${ARG3}`.

После выполнения инструкции SQL возвращаемое значение (если есть) назначается канальной переменной `RETURNED_VALUE`.

Использование функции ARRAY()

В нашем примере мы используем два отдельных вызова базы данных и присваиваем эти значения паре переменных канала, `${E}_STATUS` и `${E}_PIN`. Это было сделано чтобы упростить пример:

```
exten => _110 [1-5],1,NoOp()
    same => n,Dial(${E}_STATUS=${HOTDESK_INFO(status,${E})})
    same => n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
```

В качестве альтернативы мы могли бы вернуть несколько столбцов и сохранить их в отдельные переменные, используя функцию набора номера `ARRAY()`. Если бы мы определили наш оператор SQL в файле *func_odbc.conf* следующим образом:

```
readsql=SELECT pin,status FROM ast_hotdesk WHERE extension = '${E}'
```

мы могли бы использовать функцию `ARRAY()` для сохранения каждого столбца информации из строки в свою переменную с одним вызовом базы данных:

```
exten => _110[1-5],1,Set(ARRAY(${E}_PIN,${E}_STATUS)=${HOTDESK_INFO(${E})})
```

Использование `ARRAY()` удобно в любое время, когда вы можете получить значения, разделенные запятыми и хотите присвоить значения отдельным переменным, например `CURL()`.

Итак, в первых двух строках следующего блока кода мы передаем значения `status` и значение, содержащееся в переменной `${E}` (например, `1101`), в функцию `HOTDESK_INFO()`. Эти два значения затем заменяются в операторе SQL с помощью `${ARG1}` и `${ARG2}` соответственно и выполняется инструкция SQL. Наконец, возвращаемое значение присваивается переменной канала `${E}_STATUS`.

Итак, давайте закончим писать внутренний номер-шаблон:

```
same => n,Set(${E}_STATUS=${HOTDESK_INFO(status,${E})})
same => n,Set(${E}_PIN=${HOTDESK_INFO(pin,${E})})
same => n,GotoIf($[${ODBCROWS} < 0]?invalid_user,1)
same => n,GotoIf($[${${E}_STATUS} = 1]?logout,1:login,1)
```

После присвоения значения столбца `status` в переменную `${E}_STATUS` (если пользователь идентифицирует сам себе как номер `1101`, имя переменной будет `1101_STATUS`), мы проверяем если получили значение из базы данных (проверка ошибок), используя переменную канала `${ODBCROWS}`.

Последняя строка в блоке проверяет состояние телефона и, если агент в настоящий момент вошел в систему, отключает его. Если агент еще не вошел в систему, он перейдет на расширение `login`, с приоритетом 1 в том же контексте.



Помните, что в традиционной телефонной системе все расширения должны быть номерами, но в Asterisk расширения также могут иметь имена. Возможным преимуществом использования расширения, которое не является числом, является то, что пользователю будет намного сложнее набрать его со своего телефона и, следовательно, более безопасно. В этом примере мы будем использовать несколько именованных расширений. Если вы хотите быть абсолютно уверены, что злоумышленник не сможет получить доступ к этим именованным расширениям, просто используйте трюк, который использует загрузчик AEL: *запустите с приоритетом, отличным от 1*. Вы можете получить доступ к первой строке расширения присвоив ему метку приоритета и ссылаясь на нее с помощью комбинации имя расширения/метка приоритета.

Расширение `login` выполняет некоторые первоначальные проверки, чтобы проверить PIN-код, введенный агентом. (Кроме того, мы использовали функцию `FILTER()`, чтобы убедиться что были введены только цифры во избежание некоторых проблем с SQL-инъекциями.) Мы разрешаем ему три попытки на введение правильного PIN-кода, и если все попытки неудачны, то отправляем вызов на расширение `login_fail` (которое мы опишем позже):

```
exten => login,1,NoOp()             ; установить начальные значения счетчика
   same => n,Set(PIN_TRIES=1)       ; счетчик попыток pin
   same => n,Set(MAX_PIN_TRIES=3)   ; установка макс числа попыток входа
   same => n,Playback(silence/1)    ; воспроизвести некоторое молчание, так
                                    ; что первая подсказка не отрезана
   same => n(get_pin),NoOp()
   same => n,Set(PIN_TRIES=$[${PIN_TRIES} + 1]) ; увеличение счетчика pin
   same => n,Read(PIN_ENTERED,enter-password,${LEN(${${E}_PIN})})
   same => n,Read(PIN_ENTERED=${FILTER(0-9,${PIN_ENTERED})})
   same => n,GotoIf($["${PIN_ENTERED}" = "${${E}_PIN}"]?valid_login,1)
   same => n,Playback(pin-invalid)
   same => n,GotoIf($[${PIN_TRIES} <= ${MAX_PIN_TRIES}]?get_pin:login_fail,1)
```

Если введен PIN-код, мы проверяем логин с расширением `valid_login`. Сначала используем переменную `CHANNEL`, чтобы выяснить, какое телефонное устройство вызывается агентом. Переменная `CHANNEL` обычно заполняется чем-то вроде `SIP/0000FFFF0001-ab4034c`, поэтому мы используем функцию `CUT()`, чтобы сначала убрать часть строки `SIP/` и назначить ее `LOCATION`. Затем мы удаляем часть строки `-ab4034c`, отбрасываем ее и назначаем оставшуюся часть (`0000FFFF0001`) переменной `LOCATION`:

```
exten => valid_login,1,NoOp()
; отрезаем технологию канала и назначаем переменной LOCATION
```

```
   same => n,Set(LOCATION=${CUT(CHANNEL,/,2)})
; отрезаем уникальный идентификатор и сохраняем остальное в переменной
; LOCATION
   same => n,Set(LOCATION=${CUT(LOCATION,-,1)})
```

Мы используем еще одну пользовательскую функцию, созданную в файле *func_odbc.conf* - `HOTDESK_CHECK_PHONE_LOGINS()`, чтобы проверить, были ли ранее зарегистрированы другие пользователи под этим телефоном и просто забыли выйти из системы. Если количество неавторизованных пользователей больше `0` (оно никогда не должно быть больше `1`, но мы всё равно проверяем на большие значения и сбрасываем их тоже), он запускает логику в расширении `logout_login`:

```
; func_odbc.conf
[CHECK_PHONE_LOGINS]
prefix=HOTDESK
dsn=asterisk
readsql=SELECT COUNT(status) FROM ast_hotdesk WHERE status = '1'
readsql+=AND location = '${ARG1}'
```



Из-за ограничений длины строк в книге мы разбили команду `readsql` на несколько строк, используя синтаксис `+=`, который сообщает Asterisk добавлять содержимое после `readsql+=` к последнему определенному значению `readsql=`. Использование `+=` применимо не только к параметру `readsql`, но также может использоваться и в других местах в других *.conf*-файлах в Asterisk. Например, что-то вроде `callerid+=` можно использовать в *sip.conf*.

Если нет других агентов вошедших в устройство, мы обновляем состояние входа для данного пользователя функцией `HOTDESK_STATUS()`:

```
; Продолжение расширения valid_login ниже
   same => n,Set(USERS_LOGGED_IN=${HOTDESK_CHECK_PHONE_LOGINS(${LOCATION})})
   same => n,GotoIf($[${USERS_LOGGED_IN} > 0]?logout_login,1)
   same => n(set_login_status),NoOp()
; Устанавливаем состояние для телефона в '1' и где агент вошел в систему
   same => n,Set(HOTDESK_STATUS(${E})=1,${LOCATION})
   same => n,GotoIf($[${ODBCROWS} < 1]?error,1)
   same => n,Playback(agent-loginok)
   same => n,Hangup()
```

Мы создаем функцию записи в *func_odbc.conf* следующим образом:

```
[STATUS]
prefix=HOTDESK
dsn=asterisk
writeql=UPDATE ast_hotdesk SET status = '${SQL_ESC(${VAL1})}', writeql+=
location='${SQL_ESC(${VAL2})}'
writeql+=WHERE extension = '${SQL_ESC(${ARG1})}
```

Синтаксис очень похож на `readsql`, рассмотренный ранее в этой главе, но здесь есть несколько новых вещей, поэтому давайте обсудим их, прежде чем двигаться дальше.

Первое что вы, возможно, заметили, это то, что теперь мы имеем переменные `${VALx}` и `${ARGx}` в нашем операторе SQL.



Мы также обернули значения `${VALx}` и `${ARGx}` в функцию `SQL_ESC()`, которая будет экранировать символы, такие как обратные кавычки, которые могут быть использованы при атаке SQL-инъекции.

Они содержат значения, которые мы передаем функции из диалплана. В этом случае у нас есть две переменные VAL и одна ARG, которые были установлены из диалплана через этот оператор:

```
Set(HOTDESK_STATUS(${E})=1,${LOCATION})
```

Обратите внимание, что синтаксис немного отличается от синтаксиса функции в стиле чтения. Это сигнализирует Asterisk, что вы хотите выполнить запись (это тот же синтаксис, что и для других функций диалплана).

Мы передаем значение переменной ${E} функции HOTDESK_STATUS(), значение которой затем доступно в операторе SQL в *func_odbc.conf* с переменной ${ARG1}. Затем мы передаем два значения: 1 и ${LOCATION}. Они доступны оператору SQL в переменных ${VAL1} и ${VAL2} соответственно.

Использование SQL непосредственно в диалплане.

Некоторые люди предпочли бы напрямую писать свои SQL-запросы в диалплане, а не создавать пользовательскую функцию для каждого типа транзакций базы данных, которые они могут выполнять.

Теоретически вы можете создать только одну функцию в *func_odbc.conf* следующим образом:

```
[SQL]
prefix=GENERIC
dsn=asterisk
readsql=${SQL_ESC(${ARG1})}
writeql=${SQL_ESC(${VAL1})}
```

Тогда, в диалплане вы могли бы написать практически любой тип SQL, который захотели (при условии, что ODBC-коннектор мог бы обрабатывать его, что не имеет ничего общего с Asterisk). Затем эта функция должна передать любую строку, указанную вами, в базу данных через ODBC-коннектор.[10]

Некоторые утверждают, что это приводит к большей детализации в вашем диалплане; другие будут настаивать на том, что полезно иметь гораздо более простой файл *func_odbc.conf*:

```
[odbc_hacking]
exten => 8811,1,Goto(odbcreadtest,1)
exten => 8822,1,Goto(odbcwritetest,1)
exten => odbcreadtest,1,Noop()
        same => n,Set(result=${GENERIC_SQL(SELECT first_name
                                FROM ast_hotdesk WHERE id=4)})
        same => n,Verbose(1,${result})
        same => n,Hangup()
; обратите внимание на необходимость экранирования символов кавычек
exten => odbcwritetest,1,Noop()
        same => n,Set(GENERIC_SSQL()
                = UPDATE ast_hotdesk SET
                first_name=\'${EXTEN}\'WHERE id=4)
        same => n,Verbose (1,ODBC_RESULT - ${OBDBC_RESULT})
        same => n,Hangup()
```

Мы склонны думать, что обычно лучше строить функции с помощью *func_odbc.conf*, для обработки запросов, которые вы будете выполнять из вашего диалплана; однако нельзя отрицать соблазна использовать одну функцию для обработки всех SQL-запросов и написать весь запрос непосредственно в диалплане. Попробуйте оба метода и посмотрите, что работает для вас (вы даже можете использовать комбинацию обоих типов).

Суть в том, что func_odbc - это очень гибкий модуль Asterisk, который является немалой частью того, почему мы его любим.

10 Это также может создать ненужную угрозу безопасности.

Как уже упоминалось ранее, если нам нужно было выйти из одного или нескольких агентов перед входом в этот, мы проверили бы это в расширении `logout_login`. Эта логика диалплана будет использовать функцию `ODBC_FETCH()`, чтобы вывести информацию из стека, возвращаемого функцией `HOTDESK_LOGGED_IN_USER()`. Скорее всего, она выполнит только один цикл, но это хороший пример того, как вы можете обновлять или анализировать несколько строк в базе данных.[11]

Первая часть нашего диалплана возвращает идентификационный номер, который мы можем использовать с функцией `ODBC_FETCH()`, для перебора возвращаемых значений. Мы напишем раздел диалплана, чтобы присвоить этот идентификатор переменной канала `LOGGED_IN_ID`:

```
same => n,Set(LOGGED_IN_ID=${HOTDESK_LOGGED_IN_USER(${LOCATION})})
```

Вот расширение `logout_login`, которое потенциально может циклически проходить через несколько строк:

```
exten => logout_login,1,NoOp()
; установите для всех пользователей, вошедших в систему на этом устройстве,
; состояние выхода из системы
   same => n,Set(LOGGED_IN_ID=${HOTDESK_LOGGED_IN_USER(${LOCATION})})
   same => n(start_loop),NoOp()
   same => n,Set(WHO=${ODBC_FETCH(${LOGGED_IN_ID})})
   same => n,GotoIf($["${ODBC_FETCH_STATUS}" = "FAILURE"]?cleanup)
   same => n,Set(HOTDESK_STATUS(${WHO})=0)      ; выйти из телефона
   same => n,Goto(start_loop)
   same => n(cleanup),ODBCFinish(${LOGGED_IN_ID})
   same => n,Goto(valid_login,set_login_status) ; возврат ко входу в систему
```

И вот функция, которую мы добавили бы в *func_odbc.conf* (не забудьте перезагрузить модуль *func_odbc.so*):

```
[LOGGED_IN_USER]
prefix=HOTDESK
dsn=asterisk
mode=multirow
readsql=SELECT extension FROM ast_hotdesk
readsql+= WHERE status = '1'
readsql+= AND location = '${SQL_ESC(${ARG1})}'
```

Мы назначаем первое значение, возвращаемое из базы данных (например, номер `1101`) в канал `WHO`. Прежде чем что-либо делать, мы проверяем, была ли функция `ODBC_FETCH()` успешной при возврате данных. Если переменная канала `ODBC_FETCH_STATUS` содержит `FAILURE`, у нас нет данных для работы, поэтому мы переходим к метке приоритета `cleanup`.

Если у нас есть данные, мы передаем значение `${WHO}` в качестве аргумента функции `HOTDESK_STATUS()`, которая содержит значение `0`. Это первое переданное значение в `HOTDESK_STATUS()` и отображается как `${VAL1}` в *func_odbc.conf*, где объявлена функция.

Если вы посмотрите на функцию `HOTDESK_STATUS()` в *func_odbc.conf*, вы увидите, что мы также можем передать второе значение, но не делаем этого, поскольку хотим удалить любые значения из этого столбца, чтобы вывести пользователя из системы, который не создает никакой ценности.

После использования `HOTDESK_STATUS()` для выхода пользователя из системы, мы возвращаемся к метке приоритета `start_loop` чтобы выполнить цикл по всем значениям, который просто выполняет `NoOp()`. После попытки получить значение, мы снова проверяем `ODBC_FETCH_STATUS` на `FAILURE`. Если это значение найдено, мы переходим на метку приоритета `cleanup`, где выполняем приложение диалплана `ODBCFinish()` для выполнения очистки. Затем возвращаемся к расширению `valid_login` на метку приоритета `set_login_status`.

11 Дополнительные сведения и примеры анализа нескольких строк, возвращаемых из базы данных, см. также в разделе “Мультистроковая функциональность с func_odbc”.

Мультистроковая функциональность с func_odbc

Существует режим, позволяющий Asterisk обрабатывать несколько строк данных, возвращаемых из базы данных. Например, если бы нам нужно было создать функцию диалплана в *func_odbc.conf*, которая вернула бы все доступные внутренние номера, нам нужно было бы включить мультивывод для этой функции. Это приведет к тому, что функция будет работать несколько иначе, возвращая идентификационный номер, который затем может быть передан функции `ODBC_FETCH()`, чтобы возвращать каждую строку по очереди.

Давным-давно нам нужно было использовать функции SQL `LIMIT` и `OFFSET` для управления данными, возвращаемыми в Asterisk для итерации. Это было ресурсоемким (по крайней мере, в отношении многострочного режима), поскольку для каждой строки требовалось несколько запросов к базе данных.

Далее следует простой пример. Предположим, что у нас есть следующий *func_odbc.conf*:

```
[ALL_AVAIL_EXTENS]
prefix=GET
dsn=asterisk-connector
mode=multirow
readsql=SELECT extension FROM ast_hotdesk WHERE status = '${ARG1}'
```

и диалплан в *extensions.conf*, который выглядит примерно так:

```
[multirow_example]
exten => start,1,Verbose(1,Looping example)
   same => n,Set(ODBC_ID=${GET_ALL_AVAIL_EXTENS(1)})
   same => n,GotoIf($[${ODBCROWS} < 1]?no_rows,1)
   same => n,Set(COUNTER=1)
   same => n,While($[${COUNTER} <= ${ODBCROWS}])
   same => n,Set(AVAIL_EXTEN_${COUNTER}=${ODBC_FETCH(${ODBC_ID})})
   same => n,Set(COUNTER=$[${COUNTER + 1])
   same => n,EndWhile()
   same => n,ODBCFinish()

exten => no_rows,1,Verbose(1,No rows returned)
   same => n,Playback(silence/1&invalid)
   same => n,Hangup()
```

Функция `ODBC_FETCH()` по существу будет обрабатывать информацию как стек и каждый вызов ее с переданным `ODBC_ID` будет выводить следующую строку информации из стека. У нас также есть возможность использовать канальную переменную `ODBC_FETCH_STATUS`, которая устанавливается после функции `ODBC_FETCH()` (которая возвращает `SUCCESS`, если доступны дополнительные строки или `FAILURE`, если нет дополнительных строк). Это позволяет вам написать диалплан как показано ниже, который не использует счетчик, но по-прежнему циклически проходится по данным. Это может быть полезно, если мы ищем что-то конкретное и нет надобности просматривать все данные. Как только мы закончим, необходимо вызвать приложение `ODBCFinish()` для очистки всех оставшихся данных.

Вот еще один пример *extensions.conf*:

```
[multirow_example_2]
exten => start,1,Verbose(1,Looping example with break)
   same => n,Set(ODBC_ID=${GET_ALL_AVAIL_EXTENS(1)})
   same => n(loop_start),NoOp()
   same => n,Set(ROW_RESULT=${ODBC_FETCH(${ODBC_ID})})
   same => n,GotoIf($["${ODBC_FETCH_STATUS}" = "FAILURE"]?cleanup,1)
   same => n,GotoIf($["${ROW_RESULT}" = "1104"]?good_exten,1)
   same => n,Goto(loop_start)
```

```
exten => cleanup,1,Verbose(1,Cleaning up after all iterations)
   same => n,Verbose(1,We did not find the extension we wanted)
   same => n,ODBCFinish(${ODBC_ID})
   same => n,Hangup()

exten => good_exten,1,Verbose(1,Extension we want is available)
   same => n,ODBCFinish(${ODBC_ID})
   same => n,Verbose(1,Perform some action we wanted)
   same => n,Hangup()
```

Остальная часть контекста должна быть довольно простой (если некоторые из них не имеют смысла, мы предлагаем вам вернуться и обновить память в Главах 6 и 10). Единственным трюком, с которым вы не знакомы, может быть использование переменной канала `${ODBCROWS}`, которая задается функцией `HOTDESK_STATUS()`. Она говорит нам сколько строк было затронуто в SQL `UPDATE`, которуе мы считаем равной `1`. Если значение `${ODBCROWS}` меньше `1`, мы принимаем ошибку и обрабатываем ее соответствующим образом:

```
exten => logout,1,NoOp()
   same => n,Set(HOTDESK_STATUS(${E})=0)
   same => n,GotoIf($[${ODBCROWS} < 1]?error,1)
   same => n,Playback(silence/1&agent-loggedoff)
   same => n,Hangup()

exten => login_fail,1,NoOp()
   same => n,Playback(silence/1&login-fail)
   same => n,Hangup()

exten => error,1,NoOp()
   same => n,Playback(silence/1&connection-failed)
   same => n,Hangup()

exten => invalid_user,1,NoOp()
   same => n,Verbose(1,Hot Desk extension ${E} does not exist)
   same => n,Playback(silence/2&invalid)
   same => n,Hangup()
```

Мы также включаем контекст `hotdesk_outbound`, который будет обрабатывать наши исходящие вызовы после того, как мы зарегистрировали агента в системе:

```
include => hotdesk_outbound
```

В контексте `hotdesk_outbound` используются многие из тех же принципов, которые уже обсуждались. В этом контексте используется сопоставление шаблонов, чтобы уловить любые номера, набранные с телефонов с поддержкой «горячего стола». Сначала мы устанавливаем переменную `LOCATION` с использованием переменной `CHANNEL`, затем определяем, какой внутренний номер (агент) заносится в систему и присваиваем это значение переменной `WHO`. Если эта переменная равна `NULL` мы отклоняем исходящий вызов. Если это не `NULL` мы получаем информацию об агенте с помощью функции `HOTDESK_INFO()` и присваем ее нескольким переменным `CHANNEL`, включая контекст для обработки вызова, где мы выполняем `Goto()` в контекст, который назначили (который управляет нашим исходящим вызовом).

Мы будем использовать функцию диалплана `HOTDESK_PHONE_STATUS()`, которую вы можете определить в `func_odbc.conf` следующим образом:

```
[PHONE_STATUS]
prefix=HOTDESK
dsn=asterisk
```

```
readsql=SELECT extension FROM ast_hotdesk WHERE status = '1'
readsql+= AND location = '${SQL_ESC(${ARG1})}'
```

Если мы попытаемся набрать номер, который не обрабатывается нашим контекстом (или одним из транзитных контекстов, т.е. международный включает междугородний, который содержит локальный), будет выполнено встроенное расширение `i`, которое воспроизводит сообщение о том, что действие не может быть выполнено, и завершает вызов:

```
[hotdesk_outbound]
exten => _X.,1,NoOp()
   same => n,Set(LOCATION=${CUT(CHANNEL,/,2)})
   same => n,Set(LOCATION=${CUT(LOCATION,-,1)})
   same => n,Set(WHO=${HOTDESK_PHONE_STATUS(${LOCATION})})
   same => n,GotoIf($[${ISNULL(${WHO})}]?no_outgoing,1)
   same => n,Set(${WHO}_CID_NAME=${HOTDESK_INFO(cid_name,${WHO})})
   same => n,Set(${WHO}_CID_NUMBER=${HOTDESK_INFO(cid_number,${WHO})})
   same => n,Set(${WHO}_CONTEXT=${HOTDESK_INFO(context,${WHO})})
   same => n,Goto(${${WHO}_CONTEXT},${EXTEN},1)

[international] ; как набирается из NANP
exten => _011.,1,NoOp()
   same => n,Set(E=${EXTEN})
   same => n,Goto(outgoing,call,1)

exten => i,1,NoOp()
   same => n,Playback(silence/2&sorry-cant-let-you-do-that2)
   same => n,Hangup()

include => longdistance

[longdistance] ; содержит NANP
exten => _1NXXNXXXXXX,1,NoOp()
   same => n,Set(E=${EXTEN})
   same => n,Goto(outgoing,call,1)

exten => _NXXNXXXXXX,1,Goto(1${EXTEN},1)

exten => i,1,NoOp()
   same => n,Playback(silence/2&sorry-cant-let-you-do-that2)
   same => n,Hangup()

include => local

[local] ; в пределах NANP NPA 416
exten => _416NXXXXXX,1,NoOp()
   same => n,Set(E=${EXTEN})
   same => n,Goto(outgoing,call,1)

exten => i,1,NoOp()
   same => n,Playback(silence/2&sorry-cant-let-you-do-that2)
   same => n,Hangup()
```



Приведенный выше пример не является полным диалпланом, а скорее демонстрирует концепцию того, как направить пользователей через различные части диалплана на основе информации, хранящейся в базе данных. Мы использовали очень упрощенные совпадения шаблонов здесь, и в продакшене вам, скорее всего, понадобится более подробный диалплан. В качестве примера, фактический NPA 416 содержит наложение (647), а также есть обмены в пределах NPA 905/289, которые обычно набираемы как местные NPA 416. Междугородние ограничения маршрутизации могут быть сложными.

Если вызов разрешен для выполнения, он отправляется в контекст `[outgoing]` для обработки, а имя и номер идентификатора абонента задаются с помощью функции `CALLERID()`. Затем вызов отправляется через канал SIP с помощью `service_provider`, который мы создали в файле *sip.conf*:

```
[outgoing]
exten => call,1,NoOp()
   same => n,Set(CALLERID(name)=${${WHO}_CID_NAME})
   same => n,Set(CALLERID(number)=${${WHO}_CID_NUMBER})
   same => n,Dial(SIP/service_provider/${E})
   same => n,Playback(silence/2&pls-try-call-later)
   same => n,Hangup()
```

Наш `service_provider` может выглядеть в *sip.conf* примерно так:

```
[service_provider]
type=friend
host=switch1.service_provider.net
defaultuser=my_username
fromuser=my_username
secret=welcome
context=incoming
canreinvite=no
disallow=all
allow=ulaw
```

Теперь, когда мы внедрили довольно сложную функцию в диалплан с помощью `func_odbc` для извлечения и хранения данных в удаленной реляционной базе данных, вы можете начать видеть, что с помощью нескольких самоопределяющихся функций диалплана в файле *func_odbc.conf* и нескольких таблиц в базе данных вы можете создать несколько мощных приложений!

Сколько вещей вы только что придумали, к которым можно применить `func_odbc`?

Использование Realtime

Архитектура Asterisk Realtime[12] (ARA) позволяет хранить все параметры, обычно хранящиеся в файлах конфигурации Asterisk (обычно в каталоге */etc/asterisk*) в базе данных. Существует два типа Realtime: статический и динамический.

Статическая версия похожа на традиционный метод чтения файла конфигурации (информация загружается только при запуске из CLI), за исключением того, что данные считываются из базы данных.[13]

Динамический метод Realtime, который загружает и обновляет информацию, когда она используется живой системой, обычно используется для таких вещей, как пользователи и пиры SIP (или IAX2 и т.д.), а также ящики голосовой почты.

Для внесения изменений в статическую информацию требуется перезагрузка, как если бы вы изменили текстовый файл в системе, но динамическая информация была опробована Asterisk по мере необходимости, поэтому при внесении изменений в эти данные не требуется перезагрузка. Realtime настроен в *extconfig.conf*, расположенный в каталоге */etc/asterisk*. Этот файл сообщает Asterisk, что загружать из базы данных и откуда, позволяя загружать определенные файлы из базы данных и другие файлы из стандартных файлов конфигурации.

> Другой (возможно более старый) способ хранения конфигурации Asterisk состоял в использовании внешнего скрипта, который бы взаимодействовал с базой данных и генерировал соответствующие плоские файлы (или *.conf* файлы), а затем перезагружал

12 Конфигурация реального времени, далее везде используется *Realtime*.

13 Да, вызов этого “реального времени” несколько вводит в заблуждение, так как обновления данных не повлияют ни на что происходящее в реальном времени (пока не будет выполнена перезагрузка соответствующего модуля).



соответствующий модуль после записи нового файла. У него есть преимущество (если база данных падает, ваша система будет продолжать функционировать, скрипт просто не будет обновлять файлы до восстановления связи с базой данных), но она также имеет недостатки. Одним из главных недостатков является то, что любые изменения, внесенные пользователем, не будут доступны до запуска скрипта обновления. Это, вероятно, не самая большая проблема для небольших систем, но в больших системах ожидания вступления изменений в силу могут вызвать проблемы, такие как приостановка вызова во время загрузки и анализа большого файла.

Некоторые из них можно устранить, используя реплицированную систему баз данных. Asterisk предоставляет возможность перехода на другую систему баз данных. Таким образом, вы можете объединять базы данных, используя связь master-master (для PostgreSQL, pgcluster, или Postgre-R;[14] для MySQL он родной[15]) или master-slave (для PostgreSQL или Slony-I; для MySQL он родной) систему репликации.

Наш неофициальный обзор таких вещей показывает, что использование скриптов для записи плоских файлов из баз данных не так популярен, как запрос к базе данных в режиме реального времени (и обеспечение достаточной отказоустойчивости базы данных, чтобы справиться с тем фактом, что от нее зависит живая телекоммуникационная система).

Статический Realtime

Статическая конфигурация Realtime полезна, когда вы хотите загрузить из базы данных конфигурацию, которую обычно размещаете в файлах конфигурации в */etc/asterisk*. Те же правила, которые применяются к файлам конфигурации в вашей системе, по-прежнему применяются при использовании Static Realtime. Например, после внесения изменений в конфигурацию вы должны либо запустить глобальную команду *reload* из Asterisk CLI, либо перезагрузить определенный модуль, связанный с файлом конфигурации (например, *module reload chan_sip.so*).

При использовании Static Realtime, мы говорим Asterisk какие файлы мы хотим загрузить из базы данных используя следующий синтаксис в файле *extconfig.conf*:

```
; /etc/asterisk/extconfig.conf
[settings]
filename.conf => driver,database[,table]
```



Нет файла конфигурации с именем *filename.conf*. Вместо этого используйте фактическое имя файла конфигурации, хранящегося в базе данных. Если имя таблицы не указано, Asterisk будет использовать имя файла в качестве имени таблицы (за исключением части *.conf*). Кроме того, все настройки внутри файла *extconfig.conf* должны находиться под заголовком `[settings]`. Имейте в виду, что вы не можете загружать определенные файлы из RealTime, включая *asterisk.conf*, *extconfig.conf* и *logger.conf*.

Модуль Static Realtime использует специально определенную форматированную таблицу, позволяющую Asterisk читать различные статические файлы из базы данных. Таблица 16-2 иллюстрирует столбцы, поскольку они должны быть определены в вашей базе данных:

Таблица 16-2. Раскладка таблицы и описание ast_config

Имя столбца	Тип столбца	Описание
`id`	серийный, автоувеличивающийся	Автоувеличивающееся уникальное значение для каждой строки в таблице.
`cat_metric`	целочисленный	Вес категории в файле. Более низкая метрика означает что она отображается выше в файле (см. врезку Слово о метриках).
`ver_metric`	целочисленный	Вес предмета в категории. Более низкий показатель означает, что он отображается выше в списке (см. врезку Слово о метриках). Это полезно для таких вещей, как порядок кодеков в *sip.conf* или *iax.conf*, где вы хотите,

14 *pgcluster* кажется мертвым проектом, а Postgres-R, похоже, находится в зачаточном состоянии, поэтому в настоящее время не может быть хорошего решения для репликации master-master с использованием PostgreSQL.

15 Есть несколько учебников в Интернете, описывающих, как настроить репликацию с MySQL.

Имя столбца	Тип столбца	Описание
		чтобы сначала отображалось `disallow = all` (метрика 0), затем `allow = ulaw` (метрика 1), затем `allow = gsm` (метрика 2).
`filename`	символьный (до 128)	Имя файла, которое модуль обычно считывает с жесткого диска вашей системы (например, *musiconhold.conf*, *sip.conf*, *iax.conf*).
`category`	символьный (до 128)	Имя раздела в файле, например, `[general]`. Не заключайте имя в квадратные скобки при сохранении в базу данных.
`var_name`	символьный (до 128)	Опция слева от знака равенства (например, `disallow` - это `var_name` в `disallow=all`).
`var_val`	символьный (до 128)	Значение опции справа от знака равенства (например, `all` - это `var_val` в `disallow=all`).
`commented`	целочисленный	Любое значение, отличное от `0`, будет оцениваться так, как если бы оно было с префиксом точки с запятой в плоском файле (закомментировано).

Слово о метриках

Метрики в статическом Realtime используются для управления порядком считывания объектов в память. Считайте `cat_metric` и `var_metric` исходными номерами строк в файле конфигурации. Сначала обрабатывается более высокое значение `cat_metric`, поскольку Asterisk сопоставляет категории снизу вверх. Внутри категории сначала обрабатывается более низкая переменная `var_metric`, поскольку Asterisk обрабатывает параметры снизу вверх (например, `disallow=all` должно быть установлено значение, меньшее, чем значение `allow` в категории, чтобы убедиться, что оно обработано первым).

Простым файлом, который мы можем загрузить из Static Realtime, является файл *musiconhold.conf*.[16] Начнем с перемещения этого файла во временное место:

```
$ cd /etc/asterisk
$ mv musiconhold.conf musiconhold.conf.old
```

Для того, чтобы классы были удалены из памяти, нам необходимо перезапустить Asterisk. Затем мы можем проверить, что наши классы пустые, запустив *moh show classes*:

```
* CLI> core restart now
* CLI> moh show classes
* CLI>
```

ОК, теперь войдите в свою базу данных и создайте следующую таблицу:

```
mysql> create table ast_config (
    -> id int (8) primary key auto_increment,
    -> cat_metric int (8),
    -> var_metric int (8),
    -> filename varchar (128),
    -> category varchar (128),
    -> var_name varchar (128),
    -> var_val varchar (128),
    -> commented int default 0
    -> );
```

Давайте вставим класс `[default]` в Asterisk, но теперь мы загрузим его из базы данных. Подключитесь к своей базе данных и выполните следующие инструкции `INSERT`:

16 Файл musiconhold.conf также может быть загружен через dynamic realtime (и это может быть лучшим выбором для производственной системы), но мы используем его статически, поскольку это простой файл, который является хорошим примером.

```
> INSERT INTO ast_config

(cat_metric,var_metric,filename,category,var_name,var_val)
  VALUES
  (1,1,'musiconhold.conf','default','mode','files');

> INSERT INTO ast_config
(cat_metric,var_metric,имя_файла,category,var_name,var_val)
  VALUES
  (1,2,'musiconhold.conf','default','directory','/var/lib/asterisk/moh');
```

Можно проверить, что значения попали в базу данных, выполнив инструкцию SELECT:

```
asterisk=# SELECT filename,category,var_name,var_val FROM ast_config;

 filename         | category       | var_name     | var_val
------------------+----------------+--------------+------------------------
 musiconhold.conf | default        | mode         | files
 musiconhold.conf | default        | directory    | /var/lib/asterisk/moh
(2 rows)
```

Есть еще одна модификация - необходимо изменить файл *extconfig.conf* в каталоге */etc/asterisk* чтобы сообщить Asterisk получать данные для *musiconhold.conf* из базы данных, используя ODBC-коннектор. В первом столбце указано что мы используем драйверы ODBC для подключения (*res_odbc.conf*) и имя соединения `asterisk` (как определено в `[asterisk]` в *res_odbc.conf*). Добавьте следующую строку в конец файла *extconfig.conf* и затем сохраните его:

```
[settings]
musiconhold.conf => odbc,asterisk,ast_config
```

Затем подключитесь к консоли Asterisk и выполните перезагрузку:

```
* CLI> module reload res_musiconhold.so
```

Теперь вы можете проверить, загружается ли ваша музыка на удержание с базы данных, запустив *moh show classes*:

```
*CLI> moh show classes
Class: general
        Mode: files
        Directory: /var/lib/asterisk/moh
```

И вот вы видите: *musiconhold.conf* загружен из базы данных. Если у вас есть проблемы с перезагрузкой модуля, загружающего данные в память, попробуйте перезапустить Asterisk. Вы можете выполнить те же действия для загрузки других файлов конфигурации из базы данных, если это необходимо.

Когда использовать preload в modules.conf для Realtime модулей

В *modules.conf* иногда вам может потребоваться загрузка модулей подключения к базам данных на ранней стадии с использованием директивы `preload`. Причина в том, что определенные файлы, такие как *manager.conf*, *cdr.conf* и *rtp.conf*, не являются внешними модулями, а скорее загружаются изнутри когда запускается Asterisk. Кроме того, если эти файлы, такие как *manager.conf*, хотят использовать именованные списки ACL из базы данных, вам также потребуется предзагрузка в данной ситуации.

Если бы мы использовали ODBC в качестве интерфейса базы данных в реальном времени, то нам нужно было бы добавить следующие строки в modules.conf сразу после опции autoload:

```
; modules.conf
[modules]
autoload=yes
preload => res_odbc.so
preload => res_odbc_config.so
```

Динамический Realtime

Система динамического Realtime используется для загрузки объектов, которые могут часто меняться, таких как SIP/IAX2 пользователи и пиры, очереди и их участники, а также сообщения голосовой почты.

Точно так же, когда новые записи, вероятно, будут добавляться на регулярной основе, мы можем использовать возможности базы данных чтобы позволить нам загружать эту информацию по мере необходимости.

Все для Realtime настроено в файле */etc/asterisk/extconfig.conf*; однако динамический Realtime имеет явно определенные имена конфигурации. Все предопределенные имена должны быть настроены под заголовком `[settings]`. Например, определение пиров SIP выполняется в следующем формате:

```
; extconfig.conf
[settings]
sippeers => driver,database[,table]
```

Имя таблицы является необязательным. Если оно опущено, Asterisk будет использовать предопределенное имя (например `sippeers`), чтобы идентифицировать таблицу для поиска данных.

В нашем примере мы будем использовать таблицу `ast_sippeers` для хранения информации о SIP-пирах. Таким образом, чтобы настроить Asterisk для загрузки всех SIP-пиров из нашей базы данных в реальном времени, мы бы определили что-то вроде этого:

```
; extconfig.conf
[settings]
sippeers => odbc,asterisk,ast_sippeers
```

Предопределенные динамические имена для Realtime в extconfig.conf

Следующие имена предопределены для динамического Realtime:

- `iaxusers` - IAX пользователи
- `iaxpeers` - IAX пиры
- `sippeers` - SIP пиры и пользователи
- `sipregs` - регистрации SIP
- `voicemail` - ящики голосовой почты
- `extensions` - диалплан
- `meetme` - MeetMe конференции
- `queues` - очереди
- `queues_members` - участники очередей
- `acls` - списки доступа управления
- `musiconhold` - музыка на удержание
- `queue_log` - логирование очередей

Пользователи и пиры SIP загружаются из одной таблицы, определяемой помощью спекции `sippeers` из *extconfig.conf*. Это связано с тем, что будет поле `type` (точно так же, как если бы мы определяли

тип в файле *sip.conf*), который позволит нам определить тип `user`, `peer` или `friend`. Если вы выгрузите *chan_sip.so* и затем загрузите его обратно в память (т.е. используя *module unload chan_sip.so*, а затем *module load chan_sip.so*) после настройки *extconfig.conf*, вас будут приветствовать некоторые предупреждения, сообщающие вам, какие столбцы вам не нужны для таблицы в Realtime. Если бы вы загружались из Realtime, то получили бы следующий аналогичный вывод в консоли (который был обрезан из-за требований к пространству):

```
WARNING: Realtime table ast_sippeers@asterisk requires column
'name', but that column does not exist![17]
WARNING: Realtime table ast_sippeers@asterisk requires column
'ipaddr', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'port', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'regseconds', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'defaultuser', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'fullcontact', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'regserver', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'useragent', but that column does not exist!
WARNING: Realtime table ast_sippeers@asterisk requires column
'lastms', but that column does not exist!
```

Как вы можете видеть - нам не хватает нескольких столбцов из таблицы `ast_sipfriends`, которую мы определили для подключение к объекту `asterisk`, как определено в *res_odbc.conf*. Следующий шаг - создать нашу таблицу `ast_sipfriends` со всеми столбцами, перечисленными в предупреждающих сообщениях, в дополнение к следующим: столбец `type`, который требуется для определения пользователей, пиров и друзей; столбец `secret`, который используется для установки пароля; и столбец `host`, который позволяет нам определить, динамически ли регистрируется пир или имеет статический IP-адрес. Таблица 16-3 перечисляет все столбцы, которые должны отображаться в нашей таблице, и их типы.

Таблица 16-3. Минимальная realtime таблица sippeers/sipusers

Имя столбца	Тип столбца
`type`	символьный (до 6)
`name`	символьный (до 128)
`secret`	символьный (до 128)
`context`	символьный (до 128)
`host`	символьный (до 128)
`ipaddr`	символьный (до 128)
`port`	символьный (до 5)
`regseconds`	больше-целочисленный
`defaultuser`	символьный (до 128)
`fullcontact`	символьный (до 128)
`regserver`	символьный (до 128)
`useragent`	символьный (до 128)
`lastms`	целочисленный

17 Предупреждение: таблица Realtime ast_sippeers@asterisk требует столбец "*", но этот столбец не существует!

Код SQL для создания этой таблицы должен выглядеть примерно так:

```
create table ast_sippeers
(
type varchar(6)
name varchar(128),
secret varchar(128),
context varchar(128),
host varchar(128),
ipaddr varchar(128),
port varchar(5),
regseconds bigint,
defaultuser varchar(128),
fullcontact varchar(128),
regserver varchar(128),
useragent varchar(128),
lastms integer
);
```

Для каждого пира, которого хотите зарегистрировать, нужно вставить данные в `type`, `name`, `secret`, `context`, `host`, и `defaultuser`. Остальные столбцы будут заполнены автоматически при регистрации пира.
Поля `port`, `regseconds` и `ipaddr` необходимы чтобы позволить Asrterisk хранить регистрационную информацию для пира, таким образом он может определить куда направлять вызовы. (Обратите внимание, что если пир `static` - вам нужно будет самостоятельно заполнить поле `ipaddr`.) Поле `port` является необязательным и по умолчанию используется стандартный порт, определенный в разделе `[general]`, а `regseconds` останется пустым. В Таблице 16-4 перечислены некоторые примеры значений, которые мы будем использовать для заполнения нашей таблицы `ast_sipfriends`.

Таблица 16-4. Пример информации, используемой для заполнения таблицы ast_sipfriends:

Имя столбца	Значение
`type`	`friend`
`name`	`0000FFFF0008`
`defaultuser`	`0000FFFF0008`
`host`	`dynamic`
`secret`	`dont-use-this-password-pick-another`[18]
`context`	`LocalSets`

Однако, перед регистрацией ваших пиров вы должны включить кэширование в режиме Realtime в *sip.conf*. В противном случае пир не будет загружен в память, и регистрация не будет запомнена. Если ваши пиры только совершают вызовы и не нуждаются в регистрации в вашей системе, вам не нужно включать кэширование в Realtime, потому что пиры будут проверяться по базе базы данных каждый раз, когда они совершают вызов. Однако, если вы загружаете своих пиров в память, с базой данных необходимо будет связаться только при первоначальной регистрации и по истечении срока регистрации.

Дополнительные параметры в *sip.conf* существуют для пиров в Realtime. Они определены в разделе `[general]` и описаны в Таблице 16-5.

Таблица 16-5. Параметры Realtime в sip.conf

Параметр конфигурации	Описание
`rtcachefriends`	Кэш пиров в памяти по мере необходимости после того, как они связались с сервером. То есть, при запуске Asterisk пиры не загружаются в память автоматически; только после того,

18 не-используйте-этот-пароль-выберите-другой

Параметр конфигурации	Описание
	как пир связался с сервером (например, через регистрацию или телефонный звонок) загружается в память. Значения `yes` или `no`.
`rtsavesysname`	Когда пир регистрируется в системе, сохраняется `systemname` (как определено в *asterisk.conf*) в поле `regserver` в базе данных. (Дополнительные сведения см. в разделе "Установка имени системы для глобальных уникальных идентификаторов".) Использование `regserver` полезно при наличии нескольких серверов, регистрирующих пиров в одной таблице. Значения `yes` или `no`.
`rtupdate`	Отправляет регистрационную информацию, такую как IP-адрес, порт отправителя, период регистрации и имя пользователя `user-agent` в базу данных, когда пир регистрируется на Asterisk. Значения: `yes` или `no`, по умолчанию - `yes`.
`rtautoclear`	Автоматически истекает срок действия пиров по тому же расписанию, как если бы они только что зарегистрировались. Это приводит к тому, что пир удаляется из памяти, когда период регистрации истек, пока этот узел не будет запрошен снова (например, через регистрацию или совершение вызова). Значения `yes`, `no` или целое значение, которое приводит к удалению пиров из памяти после этого числа секунд вместо интервала регистрации.
`ignoreregexpire`	Если этот параметр включен, пиры не удаляются из памяти по истечении периода регистрации. Вместо этого информация остается в памяти, так что при запросе вызова конечной точки с истекшей регистрацией будет использоваться последняя известная информация (IP-адрес, порт и т.д.).

После включения `rtcachefriends=yes` в *sip.conf* и перезагрузки *chan_sip.so* (с помощью *module reload chan_sip.so*) вы можете зарегистрировать своего пира Asterisk, используя Realtime, и пир должен быть занесен в память. Вы сможете проверить это, выполнив команду *sip show peers* в консоли Asterisk:

```
Name/username                Host          Dyn Port Status      Realtime
0000FFFF0008/0000FFFF0008    172.16.0.160   D  5060 Unmonitored Cached RT
```

Если бы вы посмотрели на таблицу непосредственно в базе данных, вы бы увидили что-то вроде этого:

```
+--------+--------------+---------+-----------+---------+--------------+
| type   | name         | secret  | context   | host    | ipaddr       |
+--------+--------------+---------+-----------+---------+--------------+
| friend | 0000FFFF0008 | welcome | LocalSets | dynamic | 172.16.0.160 |
+--------+--------------+---------+-----------+---------+--------------+
+------+------------+--------------+-------------------------------------+
| port | regseconds | defaultuser  | fullcontact                         |
+------+------------+--------------+-------------------------------------+
| 5060 | 1283928895 | 0000FFFF0008 | sip:0000FFFF0008@172.16.0.160:52722 |
+------+------------+--------------+-------------------------------------+
+-----------+-----------------+--------+
| regserver | useragent       | lastms |
+-----------+-----------------+--------+
| NULL      | Zoiper rev.6739 | 0      |
+-----------+-----------------+--------+
```

Есть еще много вариантов, которые мы можем определить для SIP friends, такие как CallerID; добавление этой информации так же просто, как добавление столбца `callerid` в таблицу. См. файл *sip.conf.sample* для получения дополнительных параметров, которые могут быть определены для SIP friends.

Хранение записей деталей вызовов (CDR)

Записи деталей вызовов (CDR) содержат информацию о вызовах, прошедших через вашу систему Asterisk. Они рассматриваются далее в Главе 24. Хранение CDR - популярное использование баз

данных в Asterisk, поскольку упрощает управление ими (например, вы можете отслеживать несколько систем Asterisk в одной таблице). Кроме того, размещая записи в базе данных, вы открываете множество возможностей, включая создание собственного веб-интерфейса для отслеживания статистики, например, использование вызовов и наиболее часто вызываемых местоположений, биллинга или проверки счетов в телефонной компании.

Мы склонны предполагать, что поскольку хранение CDR в базе данных настолько полезно, вы всегда должны внедрять хранение CDR в базу данных на любой производственной системе (вы всегда можете хранить CDR в файле, так что ничего не потеряете).

Установка имени системы для глобальных уникальных идентификаторов

CDR состоит из уникального идентификатора и нескольких полей информации о вызове (включая исходный и целевой канал, продолжительность вызова, последнее приложение и т.д.). В кластерном наборе Asterisk теоретически возможно дублирование между уникальными идентификаторами, поскольку каждая система Asterisk рассматривает только себя. Чтобы решить эту проблему, мы можем автоматически добавлять системный идентификатор в начало уникальных идентификаторов, добавив параметр */etc/asterisk/asterisk.conf*. Для каждого из ваших Asterisk задайте идентификатор, добавив что-то вроде:

```
[options]
systemname = toronto
```

Лучший способ сохранить ваши данные о деталях звонков - через модуль `cdr_adaptive_odbc`. Этот модуль позволяет вам выбрать какие столбцы данных, встроенные в Asterisk, будут храниться в вашей таблице и это позволит вам добавить дополнительные столбцы, которые могут быть заполнены функцией диалплана `CDR()`. Вы можете даже хранить разные части данных CDR в разных таблицах и базах данных, если это необходимо.[19]

Более подробная информация о стандартных столбцах CDR в Asterisk приведена в Таблице 24-2. На момент написания этой статьи здесь приведен пример команды SQL для создания значений столбцов по умолчанию:

```
CREATE TABLE cdr (
  calldate datetime NOT NULL default '0000-00-00 00:00:00',
  clid varchar(80) NOT NULL default '',
  src varchar(80) NOT NULL default '',
  dst varchar(80 ) NOT NULL default '',
  dcontext varchar(80) NOT NULL default '',
  channel varchar(80) NOT NULL default '',
  dstchannel varchar(80) NOT NULL default '',
  lastapp varchar(80) NOT NULL default '' ,
  lastdata varchar(80) NOT NULL default '',
  duration int(11) NOT NULL default '0',
  billsec int(11) NOT NULL default '0',
  disposition varchar(45) NOT NULL default '',
  amaflags int( 11) NOT NULL default '0',
  accountcode varchar(20) NOT NULL default '',
  uniqueid varchar(32) NOT NULL default '',
  userfield varchar(255) NOT NULL default '',
  peeraccount varchar(20) NOT NULL default '',
  linkedid varchar(32) NOT NULL default '',
  sequence int(11) NOT NULL default '0'
);
```

19 Мы не уверены, что это хорошая идея, но приятно знать, что это возможно.

Вы можете определить все или любую часть этих записей в базе данных, и Asterisk будет работать над тем, что доступно. Вы также можете добавить больше столбцов для хранения других данных, относящихся к вызовам. Например, если вы хотите внедрить маршрутизацию с наименьшей стоимостью (least-cost routing - LCR), вы можете добавить столбцы для маршрута, стоимость минуты и поминутную тарификацию. После того, как вы добавили эти столбцы, они могут быть заполнены с помощью диалплана с помощью функции `CDR()` (например, `Set(CDR(per_minute_rate)=0.01)`[20]).

После создания таблицы с именем `cdr` в базе данных вы должны настроить *cdr_adaptive_odbc.conf* в каталоге */etc/asterisk*. В следующем примере используется соединение `asterisk`, которое мы определили в *res_odbc.conf* и сохраним данные в таблице `cdr`:

```
; cdr_adaptive_odbc.conf
[adaptive_connection]
connection=asterisk
table=cdr
```

Да, действительно, это все, что вам нужно. После настройки *cdr_adaptive_odbc.conf* просто перезагрузите модуль *cdr_adaptive_odbc.so* из консоли Asterisk, запустив *module reload cdr_adaptive_odbc.so*. Вы можете проверить, загружен ли адаптер Adaptive ODBC путем запуска *cdr show status*:[21]

```
*CLI> cdr show status
Call Detail Record (CDR) settings
----------------------------------
  Logging: Enabled
  Mode: Simple
  Log unanswered calls: No
  Log congestion: No

* Registered Backends
  -------------------
  cdr-syslog
  Adaptive ODBC
  cdr-custom
  csv
  cdr_manager
```

Теперь совершите вызов, на который будет дан ответ (например, с помощью функции `Playback()` или `Dial()` на другой канал с ответом на него). Вы должны получить некоторые CDR, хранящиеся в вашей базе данных. Вы можете проверить, запустив `SELECT * FROM CDR;` из консоли базы данных.

С основной информацией CDR, хранящейся в базе данных, вы можете добавить дополнительную информацию в таблицу `cdr`, такую как рейтинг маршрута (route rate). Вы можете использовать директиву `ALTER TABLE`, чтобы добавить столбец с именем `route_rate` в таблицу:

```
sql> ALTER TABLE cdr ADD COLUMN route_rate varchar(10);
```

Теперь перезагрузите модуль *cdr_adaptive_odbc.so* из консоли Asterisk:

```
*CLI> module reload cdr_adaptive_odbc.so
```

и заполните новый столбец из диалплана Asterisk, используя функцию `CDR()`, например:

```
exten => _NXXNXXXXXX,1,Verbose(1,Example of adaptive ODBC usage)
   same => n,Set(CDR(route_rate)=0.01)
   same => n,Dial(SIP/my_itsp/${EXTEN})
   same => n,Hangup()
```

20 Которая не будет работать, если вы не создали в таблице CDR поле `per_minute_rate`.

21 Вы можете увидеть различные зарегистрированные бэкенды, в зависимости от того, какую конфигурацию вы сделали с другими компонентами различных модулей CDR.

После изменения в вашей базе данных и диалплане, вы можете позвонить, а затем посмотреть на свои CDR. Вы должны увидеть что-то вроде следующего:

```
+--------------+----------+---------+------------+
| src          | duration | billsec | route_rate |
+--------------+----------+---------+------------+
| 0000FFFF0008 | 37       | 30      | 0.01       |
+--------------+----------+---------+------------+
```

На самом деле сохранение рейтинга в записи вызовов может быть неидеальным (CDR обычно используется в качестве исходного ресурса, и такие вещи как рейтинг, добавляются ниже с помощью программного обеспечения для биллинга). Возможность добавления настраиваемых полей в CDR очень полезна, но будьте осторожны чтобы не использовать ваши записи вызовов для замены правильной биллинговой платформы. Лучше всего сохранить свой CDR в чистоте и делать дальнейшую обработку ниже.

Дополнительные параметры конфигурации для cdr_adaptive_odbc.conf

Некоторые дополнительные параметры конфигурации существуют в файле *cdr_adaptive_odbc.conf*, который может быть нам полезен. Во-первых, вы можете определить несколько баз данных или таблиц для хранения информации, поэтому если у вас есть несколько баз данных, которые нуждаются в одной и той же информации, вы можете просто определить их в *res_odbc.conf*, создать таблицы в базах данных, а затем обратиться к ним в отдельных разделах конфигурации:

```
[mysql_connection]
connection=asterisk_mysql
table=cdr

[mssql_connection]
connection=production_mssql
table=call_records
```



Если вы укажете несколько разделов с использованием одного и того же соединения и таблицы, вы получите дубликаты записей.

Помимо простой настройки нескольких соединений и таблиц (которые, конечно же, могут содержать или не содержать одну и ту же информацию, модуль CDR, который мы используем, адаптивен к подобным ситуациям), мы можем определить псевдонимы для встроенных переменных, таких как `accountcode`, `src`, `dst`, `billsec` и т.д.

Если бы мы добавили псевдонимы для имен столбцов для нашего MS SQL-соединения, мы могли бы изменить наше определение соединения следующим образом:

```
[mssql_connection]
connection=production_mssql
table=call_records
alias src => Source
alias dst => Destination
alias accountcode => AccountCode
alias billsec => BillableTime
```

В некоторых ситуациях вы можете указать соединение, в котором хотите только регистрировать вызовы из определенного источника или для определенного адресата. Мы можем сделать это с

помощью фильтров:

```
[logging_for_device_0000FFFF0008]
connection=asterisk_mysql
table=cdr_for_0000FFFF0008
filter src => 0000FFFF0008
```

Если вам нужно заполнить определенный столбец информацией на основе имени раздела, его можно задать статически с помощью параметра `static`, который можно использовать с параметром `filter`:

```
[mysql_connection]
connection=asterisk_mysql
table=cdr

[filter_mysql_connection]
connection=asterisk_mysql
table=cdr
filter src => 0000FFFF0008
static «DoNotCharge» => accountcode
```



В предыдущем примере вы получите дубликаты записей в той же таблице, но вся информация будет одинаковой, за исключением заполненного столбца `accountcode`, поэтому вы сможете отфильтровать её с помощью SQL.

Хранение сообщений голосовой почты ODBC

Asterisk позволяет хранить голосовую почту внутри базы данных с помощью ODBC-коннектора. Это полезно в кластерной среде, где вы хотите абстрагировать данные голосовой почты из локальной системы, чтобы несколько блоков Asterisk имели доступ к одним и тем же данным. Конечно, вы должны принять во внимание что вы централизуете часть Asterisk и необходимо защитить эти данные от потерь, например, делать регулярные резервные копии и, возможно, кластеризовать базу данных с помощью репликации.

Asterisk хранит каждое сообщение голосовой почты внутри Binary Large OBject (BLOB). При извлечении данных он извлекает информацию из BLOB и временно сохраняет ее на жестком диске, пока она воспроизводится пользователю. Затем Asterisk удаляет BLOB и запись из базы данных, когда пользователь удаляет голосовую почту. Многие базы данных, такие как MySQL, содержат встроенную поддержку BLOB, но, как вы увидите, с PostgreSQL требуется несколько дополнительных шагов для использования этой функции. После завершения этого раздела вы сможете записывать, воспроизводить и удалять данные голосовой почты из базы данных так же, как если бы они хранились на локальном жестком диске.



Этот раздел основывается на предыдущих разделах конфигурации в этой главе. Если вы еще этого не сделали, обязательно выполните действия, описанные в разделах «Установка PostgreSQL для RHEL» и «Установка и настройка ODBC» прежде чем продолжать. В последнем убедитесь, что вы включили `ODBC_STORAGE` в системе *menuselect* в разделе Voicemail Options.

Альтернативный метод централизации

Сохранение голосовой почты в базе данных - один из способов централизовать голосовую почту. Другой метод - запустить автономный сервер голосовой почты, как мы обсуждали в Главе 8.



В этом разделе мы обсудим хранение голосовых сообщений в базе данных. Это не имеет никакого отношения к хранению фактических конфигураций почтовых ящиков (т.е. каждого почтового ящика). Если вы хотите это сделать, вам нужно будет создать таблицу для

обработки записей *voicemail.conf* с использованием статического или динамического Realtime (обсуждалось ранее в этой главе).

Компиляция модуля app_voicemail для поддержки хранилища ODBC

Чтобы поддерживать запись голосовых сообщений в базу данных ODBC, эта возможность должна быть скомпилирована в модуле голосовой почты.



Чтобы все это работало, необходимо уже скомпилировать поддержку ODBC в Asterisk. Дополнительные сведения см. в разделе «Установка и настройка ODBC».

Перейдите в каталог, в который вы загрузили исходный код Asterisk. Если вы следовали Главе 3, вы должны найти его где-то здесь:

```
cd ~/src/asterisk-complete/asterisk/11
```

Запустить `make` с аргументом *menuselect*:

```
$ sudo make menuselect
```

Это запустит интерфейс `Asterisk Module and Build Selection`. Вам нужно будет перейти к `Voicemail Build Options` и выбрать `ODBC_STORAGE`:

```
Asterisk Module and Build Option Selection

Voicemail Build Options

    --- core ---
( ) FILE_STORAGE
[*] ODBC_STORAGE
XXX IMAP_STORAGE
```

Затем просто сохраните и запустите:

```
$ make install
```

и перезапустите Asterisk. Ваша голосовая почта готова к записи в базу данных!

Создание типа Large Object для PostgreSQL

Хотя MySQL имеет тип BLOB (Binary Large OBject), мы должны сообщить PostgreSQL, как обращаться с большими (large) объектами.[22] Это включает создание триггера для очистки данных, когда мы удаляем из базы данных запись, которая ссылается на large-объект.

Подключитесь к базе данных как пользователь *asterisk* из консоли:

```
$ psql -h localhost -U asterisk asterisk
Password:
```



Вы должны быть *superuser* для выполнения следующего кода. Кроме того, если вы используете пользователя *postgres* для создания таблицы - вам нужно будет использовать SQL-директиву `ALTER TABLE`, чтобы изменить владельца на пользователя *asterisk*.

22 Если вы не можете хранить голосовые сообщения как BLOB-данные, но вы являетесь пользователем Postgres, можем ли мы предложить вам создать выделенный сервер баз данных для хранения голосовых сообщений и установить MySQL именно для этой цели? Здесь много сложностей, и мы не можем обещать, что это не будет стоить усилий.

В консоли PostgreSQL запустите следующий скрипт для создания объекта типа large:

```
CREATE FUNCTION loin (cstring) RETURNS lo AS 'oidin' LANGUAGE internal
IMMUTABLE STRICT;

CREATE FUNCTION loout (lo) RETURNS cstring AS 'oidout' LANGUAGE internal
IMMUTABLE STRICT;

CREATE FUNCTION lorecv (internal) RETURNS lo AS 'oidrecv' LANGUAGE internal
IMMUTABLE STRICT;

CREATE FUNCTION losend (lo) RETURNS bytea AS 'oidrecv' LANGUAGE internal
IMMUTABLE STRICT;

CREATE TYPE lo ( INPUT = loin, OUTPUT = loout, RECEIVE = lorecv, SEND =
losend,
INTERNALLENGTH = 4, PASSEDBYVALUE );

CREATE CAST (lo AS oid) WITHOUT FUNCTION AS IMPLICIT;
CREATE CAST (oid AS lo) WITHOUT FUNCTION AS IMPLICIT;
```

Мы будем использовать процедурный язык PostgreSQL, называемый pgSQL/PL, для создания функции. Эта функция будет вызываться из триггера, который запускается всякий раз, когда мы изменяем или удаляем запись в таблице, используемой для хранения сообщений голосовой почты. Это значит, что данные очищаются и не остаются сиротой в базе данных:

```
CREATE FUNCTION vm_lo_cleanup() RETURNS "trigger"
    AS $$
    declare
      msgcount INTEGER;
    begin
    -- raise notice 'Starting lo_cleanup function for large object with oid
       %',old.recording;
    -- If it is an update action but the BLOB (lo) field was not changed,
       don't do anything
    if (TG_OP = 'UPDATE') then
      if ((old.recording = new.recording) or (old.recording is NULL)) then
        raise notice 'Not cleaning up the large object table,
        as recording has not changed';
        return new;
      end if;
    end if;
    if (old.recording IS NOT NULL) then
      SELECT INTO msgcount COUNT(*) AS COUNT FROM voicemessages WHERE
recording
      = old.recording;
      if (msgcount > 0) then
        raise notice 'Not deleting record from the large object table, as
object
        is still referenced';
        return new;
      else
        perform lo_unlink(old.recording);
        if found then
          raise notice 'Cleaning up the large object table';
          return new;
        else
          raise exception 'Failed to clean up the large object table';
```

```
            return old;
          end if;
        end if;
      else
        raise notice 'No need to clean up the large object table,
        no recording on old row';

        return new;
      end if;
    end$$
    LANGUAGE plpgsql;
```

Мы создадим таблицу `voicemessages`, в которой будет храниться информация голосовой почты:

```
CREATE TABLE voicemessages
(
  uniqueid serial PRIMARY KEY,
  msgnum int4,
  dir varchar(80),
  context varchar(80),
  macrocontext varchar(80),
  callerid varchar(40),
  origtime varchar(40),
  duration varchar(20),
  mailboxuser varchar(80),
  mailboxcontext varchar(80),
  recording lo,
  label varchar(30),
  "read" bool DEFAULT false,
  flag varchar(10)
);
```

И теперь нам нужно связать триггер с нашей вновь созданной таблицей, чтобы выполнять очистку при каждом изменении или удалении записи в таблице `voicemessages`:

```
CREATE TRIGGER vm_cleanup AFTER DELETE OR UPDATE ON voicemessages FOR EACH ROW
EXECUTE PROCEDURE vm_lo_cleanup();
```

Макет таблицы хранения голосовой почты ODBC

Мы будем использовать таблицу `voicemessages` для хранения информации голосовой почты в базе данных через ODBC-подключение. Таблица 16-6 описывает конфигурацию таблицы для хранения голосовой почты ODBC. Если вы используете базу данных PostgreSQL, определение таблицы и поддержка large-объектов были настроены в предыдущем разделе.

Таблица 16-6. Макет таблицы хранения голосовой почты ODBC

Имя столбца	Тип столбца
`uniqueid`	последовательный, первичный ключ
`dir`	символьный (до 80)
`msgnum`	целочисленный
`recording`	BLOB (Binary Large OBject)
`context`	символьный (до 80)
`macrocontext`	символьный (до 80)
`callerid`	символьный (до 40)
`origtime`	символьный (до 40)

Имя столбца	Тип столбца
duration	символьный (до 20)
mailboxuser	символьный (до 80)
mailboxcontext	символьный (до 80)
label	символьный (до 30)
read[a]	логический, по умолчанию false
flag	символьный (до 10)

[a] read является зарезервированным словом как в MySQL, так и в PostgreSQL (и, вероятно, в других базах данных), что означает, что вам нужно будет этого избегать имени столбца при его создании. В MySQL это делается с помощью обратных кавычек (`) вокруг слова read при создании таблицы, а в PostgreSQL с двойными кавычками ("). В MS SQL вы будете использовать квадратные скобки, например, [read].

... вы абсолютно уверены, что вам нужно хранить ваши сообщения в таблице базы данных?

Вот пример того, как создать эту таблицу в MySQL:

```
CREATE TABLE voicemessages
(
  uniqueid serial PRIMARY KEY,
  msgnum int(4),
  dir varchar(80),
  context varchar(80),
  macrocontext varchar(80),
  callerid varchar(40),
  origtime varchar(40),
  duration varchar(20),
  mailboxuser varchar(80),
  mailboxcontext varchar(80),
  recording blob,
  label varchar(30),
  `read` bool DEFAULT false,
  flag varchar(10)
);
```

пример для PostgreSQL приведен в предыдущем разделе.

Настройка voicemail.conf для ODBC-хранилища

Особо нечего добавить к файлу *voicemail.conf* для включения хранилища голосовой почты ODBC. На самом деле, всего три строчки! У вас, вероятно, есть несколько типов форматов, определенных в разделе [general] *voicemail.conf*, но нам нужно установить один формат, потому что мы можем сохранить только один файл (формат) в базу данных. Формат WAV49 представляет собой сжатый формат файла WAV, который должен воспроизводиться на настольных компьютерах Linux и Microsoft Windows.

Опция odbcstorage ссылается на имя, указанное вами в *res_odbc.conf* (если вы следовали этой главе, то мы назвали его *asterisk*). Параметр odbctable ссылается на таблицу, в которой должна храниться информация голосовой почты. В примерах в этой главе мы используем таблицу с именем voicemessages.

Измените раздел [general] вашего *voicemail.conf* так, чтобы были установлены следующие значения:

```
[general]
format=wav49
odbcstorage=asterisk
odbctable=voicemessages
```

Для создания пользователей вы можете либо разделить контекст голосовой почты, либо просто использовать раздел голосовой почты `default`. Кроме того, вы можете пропустить создание нового пользователя и использовать существующего пользователя, например `0000FFFF0001`. Мы определим почтовый ящик в разделе по умолчанию для *voicemail.conf* следующим образом:

```
[default]
1000 => 1000,J.P. Wiser
```



Вы также можете использовать голосовую почту, определенную в файле *extconfig.conf* для загрузки своих пользователей из базы данных. См. "Динамический Realtime" для получения дополнительной информации о настройке определенных параметров конфигурации модуля в базе данных и "Статический Realtime" для получения дополнительной информации о загрузке остальной части файла конфигурации.

Теперь подключитесь к консоли Asterisk и выгрузите, затем загрузите модуль *app_voicemail.so*:

```
*CLI> module unload app_voicemail.so
  == Unregistered application 'VoiceMail'
  == Unregistered application 'VoiceMailMain'
  == Unregistered application 'MailboxExists'
  == Unregistered application 'VMAuthenticate'

*CLI> module load app_voicemail.so
 Loaded /usr/lib/asterisk/modules/app_voicemail.so =>
(Comedian Mail (Voicemail System))
  == Registered application 'VoiceMail'
  == Registered application 'VoiceMailMain'
  == Registered application 'MailboxExists'
  == Registered application 'VMAuthenticate'
  == Parsing '/etc/asterisk/voicemail.conf': Found
```

Затем убедитесь, что ваш новый почтовый ящик загружен успешно:

```
*CLI> voicemail show users for default
Context     Mbox   User                 Zone        NewMsg
default     1000   J.P. Wiser 0
```

Тестирование хранилища голосовых сообщений ODBC

Давайте создадим простую логику диалплана, чтобы оставить и получить голосовую почту из нашего тестового ящика голосовой почты. Вы можете добавить простую логику диалплана в файл *extensions.conf* (или, конечно же, любые функции доставки и извлечения голосовой почты, определенные ранее в этой книге):

```
[odbc_vm_test]
exten => 100,1,VoiceMail(1000@default)          ; оставить сообщение
exten => 200,1,VoiceMailMain(1000@default)      ; получить сообщение
```



В приведенном выше примере диалплан будет доступен только устройствам, которым доступен контекст `[odbc_vm_test]` или любой другой контекст в диалплане, который имеет оператор `include => odbc_vm_test` для разрешения вызова этого контекста из другого.

После обновления файла *extensions.conf* обязательно перезагрузите диалплан:

```
* CLI> dialplan reload
```

Затем настройте свой телефон или клиент с именем пользователя *odbc_test_user* и паролем *<supersecret>* совершите вызов на номер `100` и оставьте голосовую почту. В случае успеха вы должны увидеть нечто вроде:

```
  -- Executing VoiceMail("SIP/odbc_test_user-10228cac", "1000@default") in new
     stack
  -- Playing 'vm-intro' (language 'en')
  -- Playing 'beep' (language 'en')
  -- Recording the message
  -- x=0, open writing: /var/spool/asterisk/voicemail/default/1000/tmp/dlZunm
     format: wav49, 0x101f6534
  -- User ended message by pressing #
  -- Playing 'auth-thankyou' (language 'en')
== Parsing '/var/spool/asterisk/voicemail/default/1000/INBOX/msg0000.txt':
Found
```



На этом этапе вы можете проверить базу данных, чтобы убедиться, что ваши данные были успешно записаны. Дополнительную информацию см. в следующих разделах.

Теперь, когда вы удостоверились, что все правильно сохранилось в базе данных, то можете попробовать прослушать почту через приложение `VoiceMailMain()`, набрав `200`:

```
*CLI>
    -- Executing VoiceMailMain("SIP/odbc_test_user-10228cac",
       "1000@default") in new stack
    -- Playing 'vm-password' (language 'en')
    -- Playing 'vm-youhave' (language 'en')
    -- Playing 'digits/1' (language 'en')
    -- Playing 'vm-INBOX' (language 'en')
    -- Playing 'vm-message' (language 'en')
    -- Playing 'vm-onefor' (language 'en')
    -- Playing 'vm-INBOX' (language 'en')
    -- Playing 'vm-messages' (language 'en')
    -- Playing 'vm-opts' (language 'en')
    -- Playing 'vm-first' (language 'en')
    -- Playing 'vm-message' (language 'en')
  == Parsing '/var/spool/asterisk/voicemail/default/1000/INBOX/msg0000.txt':
Found
```

Проверка двоичных данных, хранящихся в PostgreSQL

Чтобы убедиться что запись действительно сделана в базу данных, используйте приложение *psql*:

```
$ psql -h localhost -U asterisk asterisk
Password:
```

Далее запустить оператор `SELECT` чтобы убедиться, что у вас есть данные в таблице `voicemessages`:

```
localhost=# SELECT uniqueid,dir,callerid,mailboxcontext,
recording FROM voicemessages;
uniqueid | dir                                                | callerid
---------+----------------------------------------------------+--------------
1        | /var/spool/asterisk/voicemail/default/1000/INBOX | +18005551212
| mailboxcontext | recording |
+----------------+-----------+
| default        | 47395     |
(1 row)
```

Если запись была помещена в базу данных, вы должны получить строку возврата. Вы заметите, что столбец `recording` содержит число (которое, безусловно, будет отличаться от того, что указано здесь), которое на самом деле является идентификатором large-объекта, хранящегося в системной таблице. Вы можете убедиться в том что large-объект существует в этой системной таблице с помощью команды `lo_list`:

```
localhost=# \lo_list
    Large objects
  ID   | Description
-------+-------------
 47395 |
(1 row)
```

Как вы убедились - идентификатор объекта в таблице `voicemessages` соответствует тому, что указано в large-объекте системной таблицы. Можно также извлечь данные из базы данных и сохранить их на жестком диске:

```
localhost=# \lo_export 47395 /tmp/voicemail-47395.wav
lo_export
```

Затем проверьте аудио в вашем любимом звуковом приложении, таком как *play*:

```
$ play /tmp/voicemail-47395.wav

Input Filename    : /tmp/voicemail-47395.wav
Sample Size       : 8-bits
Sample Encoding: wav
Channels    : 1
Sample Rate : 8000

Time: 00:06.22 [00:00.00] of 00:00.00 (   0.0%) Output Buffer: 298.36K

Done.
```

Проверка двоичных данных, хранящихся в MySQL

Чтобы убедиться, что ваши данные записаны правильно, вы можете использовать приложение *mysql* для входа в вашу базу данных и экспорта записи голосовой почты в файл:

```
$ mysql -u asterisk -p asterisk
Enter password:
```

после входа в систему в базе данных вы можете использовать оператор `SELECT` для сброса содержимого записи в файл. Во-первых, убедитесь, что в вашей таблице `voicemessages` есть хотя бы одна запись:[23]

```
mysql> SELECT uniqueid, msgnum, callerid, mailboxuser, mailboxcontext, `read`
    -> FROM voicemessages;
+----------+--------+-----------------------------+-------------
| uniqueid | msgnum | callerid                    | mailboxuser
+----------+--------+-----------------------------+-------------
|        1 |      0 | "Leif Madsen" <100>         | 100
|        2 |      1 | "Leif Madsen" <100>         | 100
|        3 |      2 | "Leif Madsen" <100>         | 100
|        5 |      0 | "Julie Bryant" <12565551111> | 100
+----------+--------+-----------------------------+-------------
```

23 Если вы привыкли использовать `SELECT * FROM` при перечислении табличных данных, имейте в виду, что данные, хранящиеся в BLOB, могут не отображать ничего или, по крайней мере, ничего, что можно было бы прочитать. Запрос, предназначенный для создания текста в качестве выходных данных, не сможет обрабатывать двоичные данные должным образом.

```
+----------------+------+
| mailboxcontext | read |
+----------------+------+
| shifteight.org |    0 |
| shifteight.org |    0 |
| shifteight.org |    0 |
| default        |    0 |
+----------------+------+
```



Вы также можете добавить столбец `recording` в оператор `SELECT`, но в итоге получите много тарабарщины на экране.

После проверки наличия данных в таблице `voicemessages` можно экспортировать одну из записей и воспроизвести ее из консоли:

```
mysql> SELECT recording FROM voicemessages WHERE uniqueid = '5'
    -> DUMPFILE '/tmp/voicemail_recording.wav';
```



Пользователю, под которым вы экспортируете данные, должен иметь разрешение `FILE` в MySQL, что означает, что ему должен быть предоставлен доступ `ALL`. Если вы не предоставили пользователю *asterisk* права доступа `ALL`, для экспорта файла необходимо использовать пользователя *root*.

Теперь выйдите из консоли MySQL и используйте приложениес *play* из консоли (при условии, что у вас есть динамики и звуковая карта, настроенная в вашей системе Asterisk, которую вы можете использовать если собираетесь использовать ее для служебного пейджинга) или скопировать файл в другую ситсему и слушать его там:

```
$ play /tmp/voicemail_recording.wav

voicemail_recording.wav:

 File Size: 7.28k Bit Rate: 13.1k
  Encoding: GSM
  Channels: 1 @ 16-bit
Samplerate: 8000Hz
Replaygain: off
  Duration: 00:00:04.44

In:100% 00:00:04.44 [00:00:00.00] Out:35.5k [ | ] Hd:4.4 Clip:0
Done.
```

Интеграция базы данных с очередями

Одним из преимуществ хранения конфигурации в базе данных является упрощение создания интерфейсов для управления данными (теоретически это можно сделать и с конфигурационными файлами, однако большинство веб-фреймворков или сред разработки интерфейсов предполагают, что данные конфигурации будут храниться в базе данных).

В колл-центре (часто называемым очередями ACD) может быть очень полезно иметь возможность настроить параметры очереди без необходимости редактировать и перезагружать конфигурационные файлы. Управление колл-центром может быть сложной задачей, а упрощение настройки параметров может сделать жизнь каждого человека намного проще.

Хранение queues.conf в базе данных

Сохранение *queues.conf* в базе данных работает так же, как и с любым другим файлом конфигурации. У вас есть выбор - использование статического или динамического Realtime.

Хранение queues.conf с использованием статического Realtime

Для статического Realtime[24] параметры хранятся точно так же, как для любого другого файла конфигурации, как описано в разделе "Статический Realtime".

Во-первых, в *extconfig.conf* вам нужен раздел, который указывает, что *queues.conf* будет храниться в базе данных:

```
vim /etc/asterisk/extconfig.conf

[settings]
queues.conf => odbc,asterisk,ast_config
```

Во-вторых, в самой таблице вам нужно сохранить параметры очереди, как они появляются в файле *queues.conf*.

Пример синтаксиса для однострочной записи будет выглядеть примерно так:

```
mysql> insert into ast_config
(cat_metric,var_metric,filename,category,var_name,var_val,commented)
VALUES
(2,1,'queues.conf','firstqueue','strategy','rrmemory',0);
```

Это приведет к записи в таблице, выглядящей как Таблица 16-7.

Таблица 16-7. Пример записи таблицы

id	cat_metric	var_metric	filename	category	var_name	var_val	commented
3	2	1	*queues.conf*	first queue	strategy	rrmemory	0

Вам потребуется несколько десятков записей чтобы правильно воспроизвести файл *queues.conf*.

Мы рекомендуем использовать динамический Realtime для хранения ваших параметров очереди.

Хранение queues.conf с использованием динамического Realtime

Хранение параметров очереди в таблице имеет гораздо больший смысл, если вы используете динамический Realtime. Созданная вами таблица будет немного легче для глаз (хотя она может содержать много столбцов[25]) и каждая очередь будет определена в одной записи:

```
CREATE TABLE `Queues` (
  `QueueID` mediumint(8) unsigned NOT NULL auto_increment,
  `name` varchar(128) NOT NULL COMMENT 'Asterisk's name for the queue',
  `description` varchar(128) default NULL,
  `maxlen` tinyint(4) default NULL,
  `reportholdtime` varchar(3) default 'no',
  `periodic_announce_frequency` varchar(4) default NULL,
  `periodic_announce` varchar(128) default NULL,
  `strategy` varchar(20) NOT NULL default 'rrmemory',
  `joinempty` varchar(35) default 'no',
  `leavewhenempty` varchar(35) default 'no',
  `autopause` varchar(3) default 'no',
```

24 Рекомендуется использовать динамический Realtime, как описано в следующем разделе.

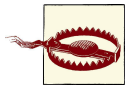
25 Если столбец не определен, в очередь будет загружено значение по умолчанию для этого столбца. Если `app_queue` не распознает столбец, он будет проигнорирован.

```
  `announce_round_seconds` varchar(4) default NULL,
  `retry` varchar(4) default NULL,
  `wrapuptime` varchar(4) default NULL,
  `announce_holdtime` varchar(3) default 'no',
  `announce_frequency` varchar(4) default '0',
  `timeout` varchar(4) default '60',
  `context` varchar(128) NOT NULL,
  `musicclass` varchar(128) default 'default',
  `autofill` varchar(3) default 'yes',
  `ringinuse` varchar(45) default 'no',
  `musiconhold` varchar(128) default 'yes',
  `monitor_type` varchar(128) default 'MixMonitor',
  `monitor_format` varchar(128) default 'wav',
  `servicelevel` varchar(4) default '60',
  `queue_thankyou` varchar(128) default '',
  `queue_youarenext` varchar(128) default '',
  `queue_thereare` varchar(128) default '',
  `queue_callswaiting` varchar(128) default '',
  `queue_holdtime` varchar(128) default '',
  `queue_minutes` varchar(128) default '',
  `queue_seconds` varchar(128) default '',
  `queue_lessthan` varchar(128) default '',
  `queue_reporthold` varchar(128) default '',
  `relative_periodic_announce` varchar(4) default 'yes',
  PRIMARY KEY (`QueueID`),
  UNIQUE KEY `name_UNIQUE` (`name`),
  UNIQUE KEY `UniqueID_UNIQUE` (`QueueID`)
)
```

Затем вы просто редактируете файл *extconfig.conf* чтобы сообщить Asterisk о своем намерении использовать эту таблицу для хранения конфигураций очередей:[26]

```
vim /etc/asterisk/extconfig.conf

[settings]
queues => odbc,asterisk,queue_table
queue_members => odbc,asterisk,queue_member_table
```



В некоторых версиях Asterisk существует ошибка из-за которой очередь не загружается если нет ссылки на `queue_members` (даже если вы не планируете жестко привязывать своих участников очереди в таблице).

```
CREATE TABLE queue_member_table (
uniqueid INT(10) UNSIGNED PRIMARY KEY AUTO_INCREMENT,
membername varchar(40),
queue_name varchar(128),
interface varchar(128),
penalty INT(11),
paused INT(11),
UNIQUE KEY queue_interface (queue_name, interface)
);
```

`queue_member_table` не должна иметь никаких данных, хранящихся в ней. Она просто должна существовать.

26 В Realtime вы не можете выбрать какое имя будет использоваться для ссылки на модуль. Например, хранение *queues.conf* в базе данных Realtime требует чтобы вы ссылались на него по имени `queues =>`.

После перезапуска Asterisk, ваши очереди теперь должны быть доступны для просмотра через команду *queue show* и, конечно, вы также сможете использовать их в вашем диалплане.

Хранение параметров диалплана для очереди в базе данных

Приложение диалплана `Queue()` позволяет передавать несколько параметров. Команда CLI *core show application Queue* определяет следующий синтаксис:

```
[Syntax]
Queue(queuename[,options[,URL[,announceoverride[,timeout[,AGI[,macro[,gosub[,
  rule[,position]]]]]]]])
```

Поскольку мы сохраняем нашу очередь в базе данных, почему бы не сохранить параметры, которые вы хотите передать в очередь, аналогичным образом?

```
CREATE TABLE `QueueDialplanParameters` (
  `QueueDialplanParametersID` mediumint(8) NOT NULL auto_increment,
  `Description` varchar(128) NOT NULL,
  `QueueID` mediumint(8) unsigned NOT NULL COMMENT 'Pointer to queues_table',
  `options` varchar(45) default 'n',
  `URL` varchar(256) default NULL,
  `announceoverride` bit(1) default NULL,
  `timeout` varchar(8) default NULL,
  `AGI` varchar(128) default NULL,
  `macro` varchar(128) default NULL,
  `gosub` varchar(128) default NULL,
  `rule` varchar(128) default NULL,
  `position` tinyint(4) default NULL,
  `queue_tableName` varchar(128) NOT NULL,
  PRIMARY KEY (`QueueDialplanParametersID`)
);
```

Используя `func_odbc` вы можете написать функцию, которая вернет параметры диалплана, относящиеся к этой очереди:

```
[QUEUE_DETAILS]
prefix=GET
dsn=asterisk
readsql=SELECT * FROM QueueDialplanParameters
readsql+= WHERE QueueDialplanParametersID='${ARG1}'
```

Затем передайте эти параметры в приложение `Queue()` по мере поступления вызовов:

```
exten => s,1,Verbose(1,Call entering queue named ${SomeValidID)
   same => n,Set(QueueParameters=${GET_QUEUE_DETAILS(SomeValidID)})
   same => n,Queue(${QueueParameters})
```

В то время как несколько сложнее разработать, чем просто написать соответствующий диалплан, преимущество состоит в том, что вы сможете управлять большим количеством очередей с более широким спектром параметров используя диалплан, который достаточно гибок, чтобы обрабатывать любые параметры, принимаемые приложением очередей в Asterisk. Мы думаем что для чего-то большего, чем простая очередь, вы найдете что использование базы данных для всего этого будет стоить усилий.

Запись queue_log в базу данных.

Наконец, мы можем хранить наш журнал `queue_log` в базе данных, что упростит внешним приложениям извлечение данных о производительности очереди из системы:

```
CREATE TABLE queue_log (
  id int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  time char(26) default NULL,
  callid varchar(32) NOT NULL default '',
  queuename varchar(32) NOT NULL default '',
  agent varchar(32) NOT NULL default '',
  event varchar(32) NOT NULL default '',
  data1 varchar(100) NOT NULL default '',
  data2 varchar(100) NOT NULL default '',
  data3 varchar(100) NOT NULL default '',
  data4 varchar(100) NOT NULL default '',
  data5 varchar(100) NOT NULL default '',
  PRIMARY KEY (`id`)
);
```

Отредактируйте файл *extconfig.conf*, чтобы обратиться к таблице `queue_log`:

```
[settings]
queue_log => odbc, asterisk, queue_log
```

Перезапускаем Asterisk и ваша очередь теперь будут записывать информацию в базу данных. Например, вход агента в очередь продаж должно приводить к следующему:

```
mysql> select * from queue_log;
+----+----------------------------+----------------------+-----------+
| id | time                       | callid               | queuename |
+----+----------------------------+----------------------+-----------+
|  1 | 2013-01-22 15:07:49.772263 | NONE                 | NONE      |
|  2 | 2013-01-22 15:07:49.809028 | toronto-1358885269.1 | support   |
+----+----------------------------+----------------------+-----------+
+------------------+------------+-------+-------+-------+-------+-------+
| agent            | event      | data1 | data2 | data3 | data4 | data5 |
+------------------+------------+-------+-------+-------+-------+-------+
| NONE             | QUEUESTART |       |       |       |       |       |
| SIP/0000FFFF0001 | ADDMEMBER  |       |       |       |       |       |
+------------------+------------+-------+-------+-------+-------+-------+
```

Если вы разрабатываете какое-либо внешнее приложение, нуждающееся в доступе к статистике очередей, то хранение данных будет намного превосходить использование */var/log/asterisk/queue_log*.

Заключение

В этой главе мы узнали о нескольких областях, в которых Asterisk может интегрироваться с реляционной базой данных. Это полезно для систем где необходимо масштабирование путем кластеризации нескольких блоков Asterisk, работающих с одной и той же централизованной информацией или когда вы хотите начать создавать внешние приложения для изменения информации, не требующей перезагрузки системы (т.е. не требуя изменения файлов конфигурации).

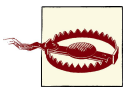
Интерактивное голосовое меню

«Однажды Алиса подошла к развилке дорог и увидела Чеширского Кота.
Скажите, пожалуйста, какую дорогу мне выбрать? - спросила она.
А куда ты хочешь попасть? - ответил Кот вопросом на вопрос.
Не знаю, - призналась Алиса.
Тогда все равно, куда идти, - изрек Кот»
- Льюис Кэрролл

В этой главе мы поговорим об IVR. Если вам нужен автосекретарь мы также написали главу для этого (Глава 15). Термин IVR часто неправильно используется для обозначения автосекретаря, но это две совершенно разные вещи.[1]

Что такое IVR?

Цель системы интерактивного голосового меню (IVR) - принимать входные данные от вызывающего абонента, выполнять действие на основе этого ввода (обычно, искать данные во внешней системе, такой как база данных) и возвращать результат вызывающему.[2] Традиционно системы IVR были сложными, дорогими и раздражающими в реализации. Asterisk меняет все это.



Asterisk размывает линии между традиционными УАТС и системами IVR. Сила и гибкость диалплана Asterisk приводят к системе, в которой почти каждый внутренний номер можно рассматривать как IVR в традиционном смысле этого термина.

Компоненты IVR

Самые основные элементы IVR очень похожи на те, что у автосекретаря, хотя цель другая. Нам нужно, по крайней мере, передать одно уведомление вызывающему, в котором сообщить что ожидает от него IVR; метод получения ввода от вызывающего, логика проверки, что ответ вызывающего абонента является допустимым вводом, логика определения того, каким должен быть следующий шаг IVR, и, наконец, механизм хранения ответов, если они действительны. Мы можем думать об IVR как о дереве решений, хотя у него не должно быть ветвей. Например, опрос может предоставлять точно такой же набор подсказок каждому вызывающему абоненту, независимо от того, какие варианты выбирают вызывающие, и единственная логика маршрутизации связана с тем, являются ли набранные ответы действительными для вопросов.

С точки зрения абонента, каждый IVR должен начинаться с подсказки. Это начальное приглашение сообщит вызывающему абоненту для чего нужен IVR и попросит вызывающего абонента сделать

1 Мы подозреваем, что это потому, что ”ИВР“ намного легче сказать, чем "автосекретарь".
2 В отличие от автосекретаря, целью которого является лишь маршрутизация вызовов.

первый ввод. Мы обсудили подсказки в службе автосекретаря в Главе 15. Позже мы создадим диалплан, который позволит вам лучше управлять несколькими голосовыми подсказками.

Второй компонент IVR представляет собой способ приема ввода от абонента. Напомним, что в Главе 15 мы обсудили приложения `Background()` и `WaitExten()` для получения нового номера. Хотя вы можете создать IVR, используя `Background()` и `WaitExten()` гораздо проще и практичнее использовать приложение `Read()`, которое обрабатывает как приглашение, так и захват ввода. Приложение `Read()` было разработано специально для использования с системами IVR. Его синтаксис выглядит следующим образом:

```
Read(variable[,filename[&filename2...]][,maxdigits][,option][,attempts]
[,timeout])
```

Аргументы описаны в Таблице 17-1.

Таблица 17-1. Приложение Read()

Аргумент	Назначение
`variable`	Переменная, в которой хранится ответ абонента. Рекомендуется присвоить каждой переменной в IVR имя, аналогичное приглашению, связанному с этой переменной. Это поможет позже, если по каким-то причинам или из-за простоты использования потребуется изменить порядок шагов IVR. Именование переменных `var1`, `var2` и т.д., может показаться легким в краткосрочной перспективе, но позже в вашем жизненном цикле это сделает исправление ошибок более сложным.
`prompt`	Файл (или список файлов, разделенных символом &) воспроизводимый абоненту с запросом ввода. Не забудьте опустить расширение формата в конце каждого имени файла.
`maxdigits`	Максимальное количество символов для ввода. В случае вопросов "да/нет" и "множественный выбор" рекомендуется ограничить это значение `1`. В случае более длинных значений, абонент всегда может прекращать ввод, нажимая клавишу #.

`options`	`s` (*skip - пропустить*)
	Немедленно выйти из канала, если на него не ответили.
	`i` (*indication - индикация*)
	Вместо приглашения воспроизводить сигнал индикации (например, гудок).
	`n` (*no answer - нет ответа*)
	Читать ввод цифр абонента, даже если еще не ответили.
	`attempts`
	Количество раз для воспроизведения подсказки. Если абонент не может ничего ввести, приложение `Read()` может автоматически запросить пользователя. По умолчанию используется одна попытка.
	`timeout`
	Количество секунд, в течение которых абонент должен ввести свои данные. Значение по умолчанию в Asterisk равно 10 секундам, хотя его можно изменить для одного приглашения с помощью этой опции или для всего сеанса, присвоив значение с помощью функции диалплана `TIMEOUT(response)`.

Как только ввод получен - он должен быть проверен. Если вы не подтвердите ввод, вы, скорее всего, обнаружите абонентов, жалующихся на нестабильное приложение. Недостаточно обрабатывать вводимые вами значения; вам также необходимо обрабатывать значения, которые вы не ожидаете. Например, вызывающие могут расстраиваться и набирать `0`, находясь в вашем IVR; если вы хорошо поработали, вы будете обрабатывать его грамотно и связывать их с кем-то, кто сможет им помочь или предоставить полезную альтернативу. Хорошо спроектированное IVR (как и любая программа) будет пытаться предугадать все возможные входные данные и предоставить механизмы для грамотного управления этим вводом.

Как только ввод подтвержден, вы можете отправить его на внешний ресурс для обработки. Это можно сделать с помощью запроса к базе данных, представления в URI, программы AGI или многих других. Это внешнее приложение должно привести к результату, который вы передадите обратно абоненту. Это может быть подробный результат, например «Баланс вашего счета ...» или простое подтверждение, например «Ваша учетная запись была обновлена». Мы не можем думать о каком-либо реальном случае, когда какой-то результат возвращать абоненту не требуется.

Иногда IVR может иметь несколько этапов, поэтому результат может включать запрос дополнительной информации от абонента, чтобы перейти к следующему шагу IVR.

Можно разработать очень сложные системы IVR с десятками или даже сотнями возможных путей. Мы уже говорили об этом раньше, и мы скажем это снова: *людям не нравится разговаривать с вашей телефонной системой, независимо от того, насколько она умна*. Держите свою IVR простой для ваших абонентов, и они, скорее всего, получат от этого какую-то выгоду.

Совершенно вкусная IVR

Отличный пример IVR, который люди любят использовать - это тот, который используют многие службы для доставки пиццы: когда вы звоните, чтобы разместить свой заказ, IVR ищет ваш номер телефона и говорит: “Если вы хотите точно такой же заказ, как в прошлый раз, нажмите 1.”

Это все, что оно делает, и это прекрасно.

Очевидно, что эти компании могли бы разработать более сложные IVR, которые позволили бы вам выбрать каждую деталь вашего пирога (”для семеричной корочки нажмите 7"), но сколько нетрезвых, голодных клиентов могли бы успешно перемещаться по чему-то подобному в 3 часа ночи?

Лучшие IVR - это те, которые требуют наименьшего ввода от вызывающего абонента. Жми кнопку `1` и твоя ...ца уже в пути! О, е!

Соображения по проектированию IVR

При разработке собственного IVR есть некоторые важные вещи, которые необходимо иметь в виду. Мы собрали список вещей, которые нужно делать, и вещей, которые не нужно делать в вашем IVR.

Делать

- Держать его простым.
- Иметь возможность набрать `0`, чтобы связаться с живым человеком.
- Корректно обрабатывать ошибки

Не делать

- Думать что IVR может полностью заменить людей.
- Использовать IVR чтобы показать людям насколько вы умны.
- Попробовать воспроизвести свой веб-сайт с помощью IVR.
- Беспокоиться о создании IVR если вы не можете принять числовой ввод. Никто не хочет писать свое имя на клавиатуре телефона.[3]
- Заставлять абонентов слушать рекламу. Помните, что они могут повесить трубку в любой момент, когда пожелают.

Модули Asterisk для построения IVR

«Фронтенд» IVR (части, которые взаимодействуют с абонентами) могут обрабатываться в диалплане. Можно построить систему IVR используя только диалплан (возможно, с помощью *astdb* для хранения и извлечения данных); однако вам, как правило, нужно будет общаться с чем-то внешним по отношению к Asterisk («бекенд» IVR).

3 Особенно если это что-то вроде Ван Меггелен.

CURL

Функция диалплана `CURL()` в Asterisk позволяет охватывать целые веб-приложения одной строкой кода диалплана. Мы будем использовать его в нашем примере IVR в этой главе позже.

Хотя вы обнаружите, что `CURL()` сама по себе проста в использовании, для создания веб-приложения потребуется опыт работы с веб-разработкой.

func_odbc

Используя `func_odbc` в Asterisk можно разрабатывать чрезвычайно сложные приложения, используя не что иное, как код диалплана и поиск в базе данных. Если вы не сильный программист, но хорошо разбираетесь в диалплане и базах данных Asterisk, вам понравится `func_odbc` так же как и нам. Проверьте это в Главе 16.

AGI

Интерфейс шлюза Asterisk (Asterisk Gateway Interface) является такой важной частью интеграции внешних приложений с Asterisk, что мы предоставили ему свою собственную главу. Более подробную информацию вы можете найти в Главе 21.

AMI

Интерфейс Asterisk Manager (Asterisk Manager Interface) - это интерфейс сокета, который вы можете использовать для получения информации о конфигурации и статусе, запросить действия, которые необходимо выполнить и получать уведомления о событиях, происходящих с вызовами. Мы также написали целую главу о AMI. Дополнительную информацию вы можете найти в Главе 20.

Простой IVR используя CURL

Программа GNU/Linux cURL полезна для извлечения данных из URI. В Asterisk `CURL()` является функцией диалплана.

Мы будем использовать `CURL()` в качестве примера того, как может выглядеть очень простой IVR.Мы собираемся запросить наш внешний IP-адрес с http://www.whatismyip.org.



В действительности большинство приложений IVR будут намного сложнее. Даже большинство применений `CURL()` будут иметь тенденцию быть сложными, поскольку URI может возвращать массивный и очень переменный объем данных, подавляющее большинство которых будет непонятно Asterisk. Дело в том, что IVR - это не только диалплан; это также внешние приложения, которые запускаются диалпланом, которые выполняют настоящую работу IVR.

Прежде чем сможете использовать `CURL()`, вы должны убедиться что он установлен.

Установка модуля cURL

Установка cURL проста. Если он не был в вашей системе, когда вы в последний раз компилировали Asterisk, после его установки вам необходимо перекомпилировать Asterisk чтобы он мог найти зависимости cURL и скомпилировать модуль *func_curl.so*.

На RHEL:

```
$ sudo yum -y install libcurl-devel
```

В Ubuntu:

```
$ sudo apt-get install libcurl4-openssl-dev
```

Диалплан

Диалплан для нашего примера IVR очень прост. Функция `CURL()` будет извлекать наш IP-адрес с http://www.whatismyip.org, а затем `SayAlpha()` будет сообщать результаты вызывающему:

```
exten => *764,1,Verbose(2, Run CURL to get IP address from whatismyip.org)
    same => n,Answer()
    same => n,Set(MyIPAddressIs=${CURL(http://www.whatismyip.org/)})
    same => n,SayAlpha(${MyIPAddressIs})
    same => n,Hangup()
```

Просто до невозможности. В традиционной IVR-системе такие вещи могут занять несколько дней.

Приложение для записи приглашений

В Главе 15 для быстрой записи приглашений мы создали простой кусок диалплана. Он был довольно ограниченным, так как записывал только одно имя файла, и, таким образом, для каждого приглашения файл нужно было скопировать до того, как будет записано новое приглашение. Здесь мы расширим до полного меню для записи приглашений:

```
[prompts]
exten => s,1,Answer
exten => s,n,Set(step1count=0) ; Инициализация счетчиков

; Если мы не получаем ответа после 3 раз - прекращаем спрашивать
    same => n(beginning),GotoIf($[${step1count} > 2]?end)
    same => n,Read(which,prompt-instructions,3)
    same => n,Set(step1count=$[${step1count} + 1])

; Все подсказки должны быть длиной 3 цифры
    same => n,GotoIf($[${LEN(${which})} != 3]?beginning)
    same => n,Set(step1count=0) ; Успешный ответ; сброс счетчиков
    same => n,Set(step2count=0)

    same => n(step2),Set(step2count=$[${step2count} + 1])
    same => n,GotoIf($[${step2count} > 2]?beginning)  ; Нет ответа после 3
                                                      ;попыток

; Если файл не существует, то не спрашивать воспроизводить ли его
    same => n,GotoIf($[${STAT(f,${which}.wav)} = 0]?recordonly)
    same => n,Background(prompt-tolisten)

    same => n(recordonly),Background(prompt-torecord)
    same => n,WaitExten(10) ; Ожидать 10 секунд для ответа
    same => n,Goto(step2)

exten => 1,1,Set(step2count=0)
    same => n,Background(${which})
    same => n,Goto(s,step2)

exten => 2,1,Set(step2count=0)
    same => n,Playback(prompt-waitforbeep)
    same => n,Record(${CHANNEL(uniqueid)}.wav)

    same => n(listen),Playback(${CHANNEL(uniqueid)})
    same => n,Set(step3count=0)
    same => n,Read(saveornot,prompt-1tolisten-2tosave-3todiscard,1)
```

```
same => n,GotoIf($["${saveornot}" = "1"]?listen)
same => n,GotoIf($["${saveornot}" = "2"]?saveit)
same => n,System(rm -f /var/lib/asterisk/sounds/${CHANNEL(uniqueid)}.wav)
same => n,Goto(s,beginning)

same => n(saveit),System(mv -f ${CHANNEL(uniqueid)}.wav ${which}.wav)
same => n,Playback(prompt-saved)
same => n,Goto(s,beginning)
```

В этой системе имя приглашения больше не является описательным; вместо этого оно является числом. Это означает что вы можете записать гораздо больше разнообразных приглашений используя тот же механизм, но компромисс состоит в том, что ваши приглашения больше не будут иметь описательных имен.

Распознавание речи и преобразование текста в речь

Хотя в большинстве случаев система IVR представляет собой предварительно записанные подсказки вызывающему абоненту и принимает входные данные посредством клавиатуры, также возможно: a) генерировать подсказки искусственно, общеизвестно как преобразование текста в речь; и b) принимать устные вводы через механизм распознавания речи.
Хотя концепция умения вести интеллектуальную беседу с машиной - это то, что писатели-фантасты обещали нам много лет назад, реальная наука об этом остается сложной и подверженной ошибкам. Несмотря на свои удивительные возможности, компьютеры плохо подходят для понимания тонких нюансов человеческой речи.
Сказав это, следует отметить, что за последние 50 лет или около того были достигнуты удивительные успехи как в области преобразования текста в речь, так и в области распознавания речи. Хорошо продуманная система, созданная для очень конкретной цели, может работать очень хорошо.
Несмотря на то, что говорят специалисты по маркетингу, ваш компьютер по-прежнему не может с вами разговаривать, и вам нужно помнить об этом, если вы рассматриваете любую систему, которая сочетает вашу телефонную систему с этими технологиями.

Text-to-Speech

Text-to-speech (также известный как синтез речи) требует, чтобы система могла искусственно строить речь из сохраненных данных. Хотя было бы хорошо, если бы мы могли просто назначить звук букве и заставить компьютер воспроизводить каждый звук, когда он читает буквы, письменный английский язык не является полностью фонетическим.[4]

Хотя на первый взгляд идея говорящего компьютера очень привлекательна, на самом деле она имеет ограниченную полезность. Более подробную информацию об интеграции текста в речь с Asterisk можно найти в Главе 18.

Распознавание речи

Как только мы убедили компьютеры говорить с нами, мы, естественно, захотим поговорить с ними.[5] Любой, кто пытался выучить иностранный язык, может начать осознавать сложность обучения компьютера понимать слова; однако распознавание речи также должно учитывать тот факт, что прежде чем компьютер сможет попытаться понять слова, он должен сначала преобразовать аудио в цифровой формат. Это больший вызов чем можно было бы подумать на первый взгляд. Например, как люди, мы, естественно, способны распознавать речь в отличие от, скажем, звука лающей собаки или автомобильного гудка. Для компьютера это очень сложная вещь. Кроме того, для системы распознавания речи на основе телефона звук, который принимается, всегда будет иметь очень низкую точность, и, таким образом, компьютер будет иметь гораздо меньше информации для работы. [6]

4 И многие другие письменные языки вовсе не являются фонетическими.
5 На самом деле, большинство из нас разговаривает со своими компьютерами, но редко когда вежливо.
6 Если распознавание речи должно происходить с мобильного телефона в шумном конференц-зале, то это становится практически невозможным.

Asterisk не имеет встроенного распознавания речи, но есть много сторонних пакетов распознавания речи, которые интегрируются с Asterisk. Многое из этого выходит за рамки данной книги, поскольку эти приложения являются внешними по отношению к Asterisk.

Заключение

Asterisk стал чрезвычайно популярным в качестве платформы IVR. Вся эта книга, во многих отношениях, учит вас навыкам, которые могут быть применены к развитию IVR. В то время как средства массовой информации действительно обращают внимание только на Asterisk как на “свободную УАТС”, реальность такова, что Asterisk спокойно штурмует индустрию IVR. В любой организации солидного размера очень вероятно, что системные администраторы Linux используют Asterisk для решения телекоммуникационных проблем, которые ранее были неразрешимыми или невероятно дорогими. Это тайная революция, но не менее значительная по своей относительной неизвестности.

Если вы находитесь в бизнесе IVR, вам нужно узнать Asterisk.

Внешние службы

Поправьте меня, если я ошибаюсь - эта штуковина подключена к флинфлангу, подключенному к ватзи, ватзи, связанному с ду-папой, подключенным к дин-донгу.
- Патрик Б. Олифант.

Asterisk довольно изящен сам по себе, но один из самых мощных, меняющихся в отрасли, революционных аспектов Asterisk - это огромное количество замечательных способов подключения к внешним приложениям и службам. Это поистине беспрецедентно в мире телекоммуникаций. В этой главе мы рассмотрим некоторые популярные сервисы и приложения, которые можно интегрировать с системой Asterisk. Вот некоторые из внешних связей, которые мы решили охватить (Asterisk может больше, но наш редактор ждет нас, чтобы закончить это издание, которое уже является самой большой книгой Asterisk):

- Если вы используете Lightweight Directory Access Protocol (LDAP) в вашей сети (например, с Active Directory) мы покажем вам как загружать пользователей SIP из ваших служб LDAP.
- Для человека, находящегося в пути с динамически изменяющимся календарем, мы попробуем некоторые идеи того, как вы можете интегрировать Asterisk с сервером календаря (что позволяет автоматически перенаправлять вызовы в зависимости от вашего текущего статуса).
- Если вы фанат обмена мгновенными сообщениями - есть раздел о том, как общаться с Asterisk по протоколу XMPP (Jabber).
- Если вы хотите привязать свою голосовую почту к вашему серверу Internet Message Access Protocol (IMAP) мы познакомим вас с основами.
- Хотите научить свою телефонную систему читать? Мы рассмотрим основы преобразования текста в речь.

Есть еще много внешних сервисов, к которым Asterisk может подключиться, но мы полагаем, что именно они помогут вам лучше понять, как интегрировать внешние службы с Asterisk.

Интеграция календаря

Asterisk может быть интегрирован с несколькими различными типами форматов календаря, такими как iCal, CalDAV, MS Exchange (Exchange 2003) и веб-службы MS Exchange (Exchange 2007 и более поздние версии). Интеграция Asterisk с вашим календарем дает возможность управлять маршрутизацией вызовов на основе текущей информации календаря. Например, если вы не собираетесь находиться в офисе днем, возможно, имеет смысл направить абонентов, звонящих на ваш настольный телефон, прямо на вашу голосовую почту.

Еще одним преимуществом интеграции календаря является возможность инициировать вызовы на основе информации календаря. Например, если вы назначаете собрание на своем сервере конференции, то можете организовать напоминание за пять минут до начала собрания, после чего вы попадете в конференц-зал. Мы думаем, что этот тип гибкости и интеграции довольно изящен и весьма полезен.

Компиляция поддержки календаря в Asterisk

Поскольку существует несколько модулей для поддержки календаря (что позволяет нам предоставлять поддержку для различных бэкэндов, таких как MS Exchange, CalDAV, iCal и т.д.), Вам необходимо установить зависимости для бэкэндов, которые вы хотите поддерживать. Эта модульная установка имеет преимущество, поскольку вам нужно устанавливать зависимости только для тех модулей, которые вам нужны; кроме того, другие бэкэнды могут быть легко интегрированы с основным бэкэндом календаря в будущем.

Из-за различных зависимостей каждого модуля нам нужно проверить *menuselect* на предмет того, что необходимо установить для каждого из модулей календаря, которые мы хотим поддерживать. Все модули требуют библиотеку разработки neon. Для `res_calendar_ews` (веб-службы Exchange) требуется версия 0.29 или более поздняя, что означает, что в некоторых дистрибутивах вам потребуется компилировать библиотеку neon из исходного кода вместо использования предварительно скомпилированного пакета, доступного в дистрибутиве.

Хотя конфигурация для всех модулей календаря схожа, мы будем обсуждать интеграцию CalDAV специально, поскольку она широко поддерживается рядом программных средств и серверов календаря.[1]

Зависимости RHEL

Поскольку для всех модулей требуется библиотека neon, сначала мы установим ее:

```
$ sudo yum install neon-devel
```



Если вы планируете скомпилировать модуль `res_calendar_ews` вам потребуется `neon` версии 0.29 или новее. В настоящее время RHEL 6.x поставляется с 0.29. Если вы используете более старую версию RHEL, то придется скомпилировать библиотеку neon и сделать ссылку на нее из скрипта *configure*. Это можно сделать через *./configure --with-neon29=<путь к neon>*.

Следующим шагом является установка зависимости libical-devel:

```
$ sudo yum install libical-devel
```



Версии RHEL до 6.x требуют стороннего репозитория (см. "Сторонние репозитории" в Главе 3). В этом случае для старых версий RHEL необходимо установить `libical-devel` из репозитория EPEL (Extra Packages for Enterprise Linux):

```
$ sudo yum --enablerepo=epel install libical-devel
```

После установки наших зависимостей мы можем запустить скрипт *configure* в нашем каталоге исходников Asterisk и включить оба модуля `res_calendar` и `res_calendar_caldav` в разделе Resource Modules из *menuselect*.

Зависимости Ubuntu

Поскольку для всех модулей требуется библиотека разработки `neon`, нам необходимо сначала установить ее. Мы собираемся установить последнюю доступную нам версию:

1 И потому авторы этой книги не имеют доступа к серверам Exchange для тестирования. :)

```
$ sudo apt-get install libneon27-dev
```



Если вы планируете скомпилировать модуль `res_calendar_ews` вам потребуется `neon` 0.29 или выше. В настоящее время Ubuntu поставляется с 0.27, поэтому вам придется скомпилировать библиотеку `neon` и сделать ссылку на нее из скрипта *configure*. Это можно сделать через *./configure --with-neon29=<путь к neon>*.

С установленной *libneon* теперь мы можем установить пакет *libical-dev* и его зависимости через *apt-get*:

```
$ sudo apt-get install libical-dev
```

После установки наших зависимостей мы можем запустить скрипт *configure* в нашем каталоге исходников Asterisk и включить оба модуля `res_calendar` и `res_calendar_caldav` из раздела Resource Modules *menuselect*.

Настройка поддержки календаря для Asterisk

В этом разделе мы обсудим как подключить систему Asterisk к календарю Google. Мы используем календари от Google по той простой причине, что им не требуются никакие другие настройки (например, настройка сервера календарей), что позволит нам быстрее запуститься и работать. Конечно, как только вы освоите настройку поддержки календарей в Asterisk, вы можете подключить ее к любому календарному серверу, который пожелаете.

Первый шаг - убедиться, что у вас есть аккаунт Gmail у Google, который даст вам доступ к серверу календаря. После того, как вы вошли в свою учетную запись Gmail, в верхнем левом углу (возможно уже в другом месте - прим. переводчика) должна быть ссылка на ваш календарь. Нажмите на ссылку Календарь и вставьте пару элементов, происходящих в течение следующего часа или двух. Когда мы будем настраивать наш *calendar.conf*, то будем инструктировать Asterisk проверять наличие новых событий каждые 15 минут и получать данные за 60 минут.



Не забудьте проверить время на вашем сервере. Если время не синхронизировано с остальным миром - например, если оно не обновляется через Network Time Protocol (NTP) - ваши события могут не отображаться или могут появляться в неправильное время. Этот совет является результатом столкновения с этой самой проблемой при тестировании и документировании. :)

Следующим шагом является настройка файла *calendar.conf* для опроса календарного сервера.



В файле *calendar.conf.sample* есть несколько примеров серверов календарей, например, серверы календарей на базе Microsoft Exchange, iCal и CalDAV.

Следующая конфигурация будет подключаться к серверу календаря Google и каждые 15 минут опрашивать новые события, получая данные за 60 минут. Не стесняйтесь изменять эти настройки по мере необходимости, но помните, что получение большего количества данных (особенно если у вас есть несколько календарей для сотрудников вашей компании) будет использовать больше памяти:

```
$ cat >> calendar.conf
[myGoogleCal]
type=caldav
url=https://www.google.com/calendar/dav/<Gmail Email Address>/events/
user=<Gmail Email Address>
secret=<Gmail Password>
refresh=15
timeframe=60
Ctrl+D
```

Если *calendar.conf* настроен, давайте загрузим модули календаря в Asterisk. Сначала мы загрузим модуль *res_calendar.so* в память, а затем проследим за ним, выполнив *module reload*, которая загрузит дочерние модули (например, *res_calendar_caldav.so*) правильно:[2]

```
$ asterisk -r
*CLI> module load res_calendar.so
*CLI> module reload
```

После загрузки модулей мы можем убедиться, что наш календарь подключен к серверу и правильно загружен в память, выполнив *calendar show calendars*:

```
*CLI> calendar show calendars
Calendar                Type        Status
--------                ----        ------
myGoogleCal             caldav      busy
```

Наш статус в настоящее время установлен на `busy` (что не имеет никакого отношения к нашей абонентской группе на данный момент, это просто означает, что у нас есть событие, которое пометило нас в календаре как занятого) и мы можем увидеть загруженные в данный момент события для нашего временного диапазона, запустив *calendar show calendar <myGoogleCal>* из консоли Asterisk:

```
*CLI> calendar show calendar <myGoogleCal>
Name              : myGoogleCal
Notify channel    :
Notify context    :
Notify extension  :
Notify applicatio :
Notify appdata    :
Refresh time      : 15
Timeframe         : 60
Autoreminder      : 0
Events
------
Summary     : Awesome Call With Russell
Description :
Organizer   :
Location    :
Cartegories :
Priority    : 0
UID         : hlfhcpi0j360j8fteop49cvk68@google.com
Start       : 2010-09-28 08:30:00 AM -0400
End         : 2010-09-28 09:00:00 AM -0400
Alarm       : 2010-09-28 04:20:00 AM -0400
```

Первое поле в верхнем разделе - это название нашего календаря. После этого есть несколько полей `Notify`, которые используются для набора адресата в начале собрания, которое мы обсудим более подробно в ближайшее время. Поля `Refresh time` и `Timeframe` являются значениями, которые мы сконфигурировали для частоты проверки наличия новых событий и долготы просмотра диапазона данных, соответственно. Поле `Autoreminder` определяет, за сколько до события мы должны выполнять опции `Notify` .



Если вы не настроили ни один из параметров `Notify`, но в календаре установлено оповещение, вы можете получить следующее `WARNING` сообщение:

```
WARNING[5196]: res_calendar.c:648 do_notify: Channel should be in
```

2 На момент написания статьи в процессе загрузки модулей календаря после запуска Asterisk возникла ошибка. Она была подана как ошибка 18067 на https://issues.asterisk.org и, надеюсь, будет решена к тому времени, когда вы будете читать это. В противном случае следует помнить, что для правильной загрузки модулей в память может потребоваться перезапуск Asterisk.

```
form Tech/Dest (was '')
```

Предупреждение вызвано тем, что для уведомления о начале собрания было установлено оповещение, но Asterisk не удалось сделать вызов из-за значений, не настроенных для совершения вызова. Это предупреждение можно проигнорировать, если вы не планируете выполнять вызовы для уведомлений о событиях.

Остальная часть вывода на экран - это список событий, доступных на нашем `Timeframe` вместе с информацией о событиях. Далее мы посмотрим на некоторые примеры диалплана, которые можем выполнить сейчас, когда у нас есть информация календаря в Asterisk, и настроим уведомления о вызовах для напоминаний о предстоящих встречах.

Запуск напоминаний календаря на вашем телефоне

В этом разделе мы обсудим, как настроить файл *calendar.conf* для выполнения некоторого простого диалплана, который будет звонить на ваш телефон до события календаря. Хотя представленный нами диалплан может быть не готов к полноценной работе, он, безусловно, предлагает обзор возможностей для совершения вызова на основе события календаря.

Запуск вызова будильника

В нашем первом примере мы собираемся вызвать устройство и воспроизвести напоминание для определенного события календаря. Может быть полезно получить напоминания такого типа, если вы, вероятно, дремлете за столом в то время, когда проходит еженедельная встреча в понедельник. Чтобы настроить напоминание о пробуждении, нам просто нужно добавить следующие строки в конфигурацию нашего календаря в *calendar.conf*:

```
channel=SIP/0000FFFF0001
app=Playback
appdata=this-is-yr-wakeup-call
```



В календаре необходимо убедиться, что с добавляемым событием связан сигнал тревоги или напоминание. В противном случае Asterisk не будет пытаться создать вызов.

После внесения этого изменения перезагрузите модуль *res_calendar.so* из консоли Asterisk:

```
*CLI> module reload res_calendar.so
```

Когда событие наступит Asterisk сгенерирует вам звонок и воспроизведет звуковой файл *this-is-yr-wakeup-call*. Вывод в консоль будет выглядеть следующим образом:

```
  -- Dialing SIP/0000FFFF0001 for notification on calendar myGoogleCal
== Using SIP RTP CoS mark 5
  -- Called 0000FFFF0001
  -- SIP/0000FFFF0001-00000001 is ringing
  -- SIP/0000FFFF0001-00000001 connected line has changed, passing it to
     Calendar/myGoogleCal-5fd3c52
  -- SIP/0000FFFF0001-00000001 answered Calendar/myGoogleCal-5fd3c52
  -- <SIP/0000FFFF0001-00000001> Playing 'this-is-yr-wakeup-call.ulaw'
     (language 'en')
```



Если вы измените событие календаря таким образом, что оно начнется всего через пару минут в будущем, вы можете быстро инициировать события, выгрузив, а затем загрузив модуль *res_calendar_caldav.so* из консоли Asterisk. Сделав это, вы заставите Asterisk немедленно сгенерировать вызов.

Помните, что наша частота обновления установлена на 15 минут и мы собираем события за 60 минут. Возможно, вам придется изменить эти цифры, если вы захотите проверить это на своем сервере.

Планирование вызовов между двумя участниками

В этом примере мы покажем, как можно использовать комбинацию простого диалплана и функции `CALENDAR_EVENT()` для создания вызова между двумя участниками на основе информации в поле местоположения. Мы собираемся заполнить поле местоположение как `0000FFFF0002`, которое является SIP-устройством, которое мы хотим вызвать после ответа на наше напоминание.



Мы не указали `SIP/0000FFFF0002` непосредственно в событии календаря, потому что мы хотим быть более осторожными с тем, что принимаем. Поскольку мы будем отфильтровывать все, кроме буквенно-цифровых символов, мы не сможем принять косую черту в качестве разделителя между технологией и местоположением (например, `SIP/0000FFFF0001`). Мы могли бы, конечно, позволить это, но подвергаемся риску совершения людбми дорогих исходящих звонков, особенно если пользователь открывает свой календарь публично или если он был скомпрометирован. С помощью метода, который собираемся использовать, мы просто ограничиваем наш риск.

Добавим следующий диалплан в наш *extensions.conf*:

```
[AutomatedMeetingSetup]
exten => start,1,Verbose(2,Triggering meeting setup for two participants)
   same => n,Set(DeviceToDial=${FILTER(0-9A-Za-z,$
{CALENDAR_EVENT(location)})})
   same => n,Dial(SIP/${DeviceToDial},30)
   same => n,Hangup()
```

Когда время события наступит - наше устройство получит вызов и при ответе на этот вызов другой вызов будет направлен на конечную точку, с которой мы хотим провести нашу встречу. Вывод в консоль выглядит следующим образом:

```
Здесь наш календарь запускает вызов на наше устройство

    -- Dialing SIP/0000FFFF0001 for notification on calendar myGoogleCal
  == Using SIP RTP CoS mark 5
    -- Called 0000FFFF0001
    -- SIP/0000FFFF0001-00000004 is ringing

И вот мы ответили на звонок Астериска, вызванный событием

    -- SIP/0000FFFF0001-00000004 connected line has changed, passing it to
       Calendar/myGoogleCal-347ec99
    -- SIP/0000FFFF0001-00000004 answered Calendar/myGoogleCal-347ec99

После ответа мы инициируем некоторый диалплан, который ищет оконечную точку
для вызова

    -- Executing [start@AutomatedMeetingSetup:1]
       Verbose("SIP/0000FFFF0001-00000004", "2,
       Triggering meeting setup for two participants") in new stack
  == Triggering meeting setup for two participants

Здесь мы использовали CALENDAR_EVENT(location) для удаленного устройства

   -- Executing [start@AutomatedMeetingSetup:2] Set("SIP/0000FFFF0001-
00000004",
      "DeviceToDial=0000FFFF0002") in new stack

И теперь мы вызываем эту конечную точку
```

```
  -- Executing [start@AutomatedMeetingSetup:3] Dial("SIP/0000FFFF0001-
00000004",
"SIP/0000FFFF0002,30") in new stack
  == Using SIP RTP CoS mark 5
    -- Called 0000FFFF0002
    -- SIP/0000FFFF0002-00000005 is ringing
```

Другой конец ответил на звонок и Asterisk соединил нас вместе

```
-- SIP/0000FFFF0002-00000005 answered SIP/0000FFFF0001-00000004
-- Locally bridging SIP/0000FFFF0001-00000004 and SIP/0000FFFF0002-00000005
```

Конечно, диалплан мог бы быть расширен, чтобы заставить первоначального абонента подтвердить готовность к встрече до вызова другого участника. Аналогично, мы могли бы добавить диалплан, который воспроизводит подсказку для другого абонента, которая сообщает ему что он запланировал встречу и если он нажмет 1, то будет немедленно соединен с другой стороной. Мы могли бы даже создать диалплан, который позволил бы исходной стороне записать сообщение для воспроизведения другому абоненту.

Просто для удовольствия мы покажем вам пример функциональности, которую мы только что описали. Не стесняйтесь изменять его в соответствии с вашими пожеланиями:

```
[AutomatedMeetingSetup]
exten => start,1,Verbose(2,Triggering meeting setup for two participants)

; *** Эта строка не должна содержать разрывов
   same => n,Read(CheckMeetingAcceptance,to-confirm-wakeup&press-1&otherwise
&press-2,,1)

   same => n,GotoIf($["${CheckMeetingAcceptance}" != "1"]?hangup,1)
   same => n,Playback(silence/1&pls-rcrd-name-at-tone&and-prs-pound-whn-
finished)

; Мы устанавливаем случайное число и присваиваем его по окончанию записи,
; чтобы у нас было уникальное имя файла, если оно используется
; несколькими людьми одновременно.
;
; Мы также добавили префикс - двойное подчеркивание, потому что канальная
; переменная также должна быть доступна каналу, который мы собираемся вызвать
;
   same => n,Set(__RandomNumber=${RAND()})
   same => n,Record(/tmp/meeting-invite-${RandomNumber}.ulaw)

   same => n,Set(DeviceToDial=${FILTER(0-9A-Za-z,$
{CALENDAR_EVENT(location)})})
   same => n,Dial(SIP/${DeviceToDial},30,M(CheckConfirm))
   same => n,Hangup()

exten => hangup,1,Verbose(2,Call was rejected)
   same => n,Playback(vm-goodbye)
   same => n,Hangup()

[macro-CheckConfirm]
exten => s,1,Verbose(2,Allowing called party to accept or reject)
   same => n,Playback(/tmp/meeting-invite-${RandomNumber})

; *** Эта строка не должна содержать разрывов
   same => n,Read(CheckMeetingAcceptance,to-confirm-wakeup&press-1&otherwise
&press-2,,1)
```

```
   same => n,GotoIf($["${CheckMeetingAcceptance}" != "1"]?hangup,1)

exten => hangup,1,Verbose(2,Call was rejected by called party)
   same => n,Playback(vm-goodbye)
   same => n,Hangup()
```

Мы надеемся, что вы Вы сможете использовать этот простой пример диалплана в качестве отправной точки. С небольшим творческим потенциалом и некоторыми навыками диалплана, возможности безграничны!

Вызов участников собрания и размещение их в конференции

Чтобы расширить функциональные возможности, описанные в предыдущем разделе, мы рассмотрим логическую проблему размещения нескольких участников собрания. Наша цель состоит в том, чтобы использовать наш календарь для вызовов нам когда планируется начать собрание, а затем, когда мы отвечаем, делать звонки всем другим участникам конференции. По мере того как другие участники будут отвечать на телефонные звонки, они будут помещены в виртуальный конференц-зал, где станут ждать присоединения организатора собрания. После того, как все участники были вызваны и ответили (или, возможно, не ответили), будет вызван организатор, после чего начнется собрание.

Этот тип функциональности увеличивает вероятность того, что собрание начнется вовремя, и это означает, что организатору собрания не нужно постоянно выполнять перекличку, поскольку новые участники продолжают присоединяться после того, как вызов должен начаться (что неизменно происходит с людьми, чьи графики обычно довольно загружены).

Диалплан, который мы покажем вам, не обязательно является отлаженной, готовой к продакшену установкой (например, данные, возвращаемые из календаря, поступающие из поля описания, имеют дело только с именами устройств и предполагают технологию SIP). Тем не менее, мы проделали тяжелую работу для вас, разработав использование локального канала вместе с использованием флага `M()` (макроса) с `Dial()`. После некоторого тестирования и настроек этот код, безусловно, можно было бы доработать более полно для вашей конкретной конфигурации, но мы придерживались его общего порядка, чтобы он мог использоваться для большего количества людей в более сложных ситуациях. Пример диалплана выглядит следующим образом:

```
[AutomatedMeetingSetup]
exten => start,1,Verbose(2,Calling multiple people and placing into a
conference)

; Получить информацию из календаря и сохранить ее. Префикс CalLocation с
; подчеркиванием чтобы он был доступен локальному каналу (наследование
; переменных).
;
   same => n,Set(CalDescription=${CALENDAR_EVENT(description)})
   same => n,Set(_CalLocation=${CALENDAR_EVENT(location)})
   same => n,Set(X=1)

; Наш разделитель это карет (^), поэтому описание будет в формате:
; 0000FFFF0001^0000FFFF0002^итд...
;
   same => n,Set(EndPoint=${CUT(CalDescription,^,${X})})

; Этот цикл используется для построения переменной ${Toroidal}, которая
; содержит список локальных каналов для набора, тем самым вызывая несколько
; действий Originate() одновременно, а не линейно
;
   same => n,While($[${EXISTS(${EndPoint})}])

; Этот оператор должен находиться в одной строке
```

```
   same => n,Set(ToDial=${IF($[${ISNULL(${ToDial})}]?
                    :${ToDial}&)}Local/${EndPoint}@MeetingOriginator)
   same => n,Set(X=$[${X} + 1])
   same => n,Set(EndPoint=${CUT(CalDescription,^,${X})})
   same => n,EndWhile()

; Если никакие значения не возвращаются, то не утруждайте себя набором номера
   same => n,GotoIf($[${ISNULL(${ToDial})}]?hangup)
   same => n,Dial(${ToDial})

; После возврата оператора Dial() нас следует поместить в конференц-зал.
; Мы маркированы, поэтому конференция может начаться
; (что обозначается флагом 'A' в MeetMe).
;
   same => n,MeetMe(${CalLocation},dA)
   same => n(hangup),Hangup()

[MeetingOriginator]
exten => _[A-Za-z0-9].,1,NoOp()
   same => n,Set(Peer=${FILTER(A-Za-z0-9,${EXTEN})})

; Originate вызывает пира, переданного из канала Local. После ответа
; вызываемая сторона должна выполнить диалплан, расположенный в расширении
; _meetme-XXXX, где XXXX номер конференц-зала.
;
   same => n,Originate(SIP/${Peer},exten,MeetingOriginator,meetme-$
{CalLocation},1)
   same => n,Hangup()

; Присоединиться к собранию; используя флаг 'w', что означает
; 'ждать присоединения маркированного пользователя для начала'
;
exten => _meetme-XXXX,1,Verbose(2,Joining a meeting)
   same => n,Answer()
   same => n,MeetMe(${EXTEN:7},dw)
   same => n,Hangup()
```

Управление вызовами на основе информации календаря

Иногда полезно автоматически перенаправлять вызовы, например, когда вы находитесь на собрании или в отпуске. В этом разделе мы будем использовать функцию диалплана `CALENDAR_BUSY()`, которая позволяет нам проверять текущее состояние нашего календаря чтобы определить заняты мы или нет. Простым примером этого может быть отправка всех вызовов на голосовую почту с использованием сообщения «занято», когда запланировано событие, которое помечает нас как «занятых».

В следующем диалплане показан простой пример, в котором мы проверяем наш календарь на занятость перед отправкой вызова на устройство. Обратите внимание, что большая часть информации в этом примере является статической; потребовалось бы больше усилий, чтобы сделать её динамической и пригодной для продакшена:

```
exten => 3000,1,Verbose(2,Simple calendar busy check example)
   same => n,Set(CurrentExten=${EXTEN})
   same => n,Set(CalendarBusy=${CALENDAR_BUSY(myGoogleCal)})
   same => n,GotoIf($["${CalendarBusy}" = "1"]?voicemail,1)
   same => n,Dial(SIP/0000FFFF0002,30)
   same => n,Goto(voicemail,1)
```

```
exten => voicemail,1,Verbose(2,Caller sent to voicemail)

; *** Эта строка не должна содержать разрывов
   same => n,GotoIf($["${DIALSTATUS}" = "BUSY" |
"${CalendarBusy}" = "1"]?busy:unavail)

   same => n(busy),VoiceMail(${CurrentExten}@shifteight,b)
   same => n,Hangup()

   same => n(unavail),VoiceMail(${CurrentExten}@shifteight,u)
   same => n,Hangup()
```

А вот немного более сложный раздел диалплана, в котором используются некоторые из инструментов, которые мы изучили на протяжении всей книги, включая функции `DB_EXISTS()`, `GotoIf()` и `IF()`:

```
exten => _3XXX,1,Verbose(2,Simple calendar busy check example)
   same => n,Set(CurrentExten=${EXTEN})
   same => n,GotoIf($[${DB_EXISTS(extension/${CurrentExten}/device)}]?
:no_device,1)
   same => n,Set(CurrentDevice=${DB_RESULT})
   same => n,GotoIf($[${DB_EXISTS(extension/${CurrentExten}/calendar)}]?
:no_calendar)
   same => n,Set(CalendarBusy=${CALENDAR_BUSY(${DB_RESULT})})
   same => n,GotoIf($[${CalendarBusy}]?voicemail,1)
   same => n(no_calendar),Verbose(2,No calendar was found for this user)
   same => n,Dial(SIP/${CurrentDevice},30)
   same => n,Goto(voicemail,1)

exten => voicemail,1,Verbose(2,Sending caller to voicemail)

; *** Эта строка не должна содержать разрывов строк
   same =>
n,GotoIf($[${DB_EXISTS(extension/${CurrentExten}/voicemail_context)}]
?:no_voicemail)

   same => n,Set(VoiceMailContext=${DB_RESULT})

; *** Эта строка не должна содержать разрывов строк
   same => n,Set(VoiceMailStatus=${IF($["${DIALSTATUS}" = "BUSY" |
0${CalendarBusy}]?b:u)})
   same => n,VoiceMail(${CurrentExten}@${VoiceMailContext},${VoiceMailStatus})
   same => n,Hangup()

   same => n(no_voicemail),Playback(number-not-answering)
   same => n,Hangup()

exten => no_device,1,Verbose(2,No device found in the DB)
   same => n,Playback(invalid)
   same => n,Hangup()
```

Запись информации о вызове в календарь

С помощью функции `CALENDAR_WRITE()` открываются некоторые другие возможности в плане интеграции календаря. Из абонентской группы Asterisk мы можем вставить в календарь информацию, которую могут использовать другие устройства и приложения. Наш следующий пример - календарь, который отслеживает журналы вызовов. Для любого, кто изрядно разговаривает по телефону, кому

нужно отслеживать время обзвона клиентов, записывать все звонки в календарь для наглядной справки может быть полезна при проверке дел в конце дня.

Мы собираемся снова использовать веб-календарь Google для этого примера, но теперь собираемся создать новый отдельный календарь только для отслеживания вызовов. Для записи в календарь нам нужно настроить файл *calendar.conf* немного по-другому, используя формат календаря CalDAV. Однако сначала нам нужно создать наш новый календарь.

С левой стороны интерфейса календаря Google будет ссылка с надписью *Add (Добавить)*. Нажатие на нее откроет новое окно, где мы можем создать календарь. Мы назвали наш «Phone Calls» (Телефонные звонки).

Теперь нам нужно включить синхронизацию календаря CalDAV для нашего календаря. Информация о том, как это сделать, находится на странице поддержки Google. На этой странице отмечается, что только ваш основной календарь будет синхронизирован с устройством, но мы хотим чтобы наши звонки регистрировались в отдельном календаре, чтобы их можно было легко скрыть (и чтобы наш смартфон не синхронизировал звонки телефона, что может вызвать путаницу). В нижней части страницы есть две ссылки: одна для обычных пользователей календаря Google, а другая для пользователей Google Apps. Когда мы нажимаем на соответствующую ссылку, появляются наши календари. Затем мы видим страницу, которая содержит наши календари. Мы выбираем календарь *Phone Calls* (Телефонные звонки) и затем выбираем *Save* (Сохранить).

Далее настраивается наш *calendar.conf* для Asterisk. Одним из необходимых нам параметров является ссылка на календарь CalDAV. Существует `Calendar ID` - значение, которое нужно нам, оно будет определять наш календарь. Чтобы найти идентификатор календаря, нажмите стрелку вниз рядом с именем календаря в левой части страницы календаря и выберите *Calendar Settings* (Настройки календаря). В нижней части настроек календаря будут две строки, которые содержат значки для совместного использования календаря (XML, ICAL, HTML). Рядом с первым набором значков внутри поля *Calendar Address* (Адрес календаря) будет идентификатор календаря. Он будет выглядеть следующим образом:

```
(Calendar ID: 2hfb6p5974gds924j61cmg4gfd@group.calendar.google.com)
```

Если вы настраиваете его с помощью Google Apps перед идентификатором календаря будет стоять префикс вашего имени домена и подчеркивания (например `shiftteight.org_`). Запишите эту строку, так как мы собираемся использовать ее далее.

Откройте файл *calendar.conf* и добавьте новый раздел календаря. В нашем случае мы назвали его `[phone_call_calendar]`. Вы узнали форматирование календаря ранее, поэтому мы не будем проходить через все настройки снова. Ключевой настройкой для заметки является параметр `url`. Формат этого параметра:

```
https://www.google.com/calendar/dav/<calendar_id>/events/
```

Нам нужно заменить *<calendar_id>* идентификатором календаря, который мы недавно нашли. Полная конфигурация в итоге выглядит следующим образом:

```
[phone_call_calendar]
type=caldav

; URL-адрес должен быть в одну строку
url=https://www.google.com/calendar/dav/
   shifteight.org_2hfb6p5974gds924j61cmg4gfd@group.calendar.google.com/events/

user = leif@shifteight.org
secret = my_secret_password
refresh=15
timeframe=120
```

Теперь, когда у нас настроен календарь, нам нужно загрузить его в память, что можно сделать, перезагрузив модуль *res_calendar.so*:

```
*CLI> module reload res_calendar.so
```

Убедитесь, что календарь был успешно загружен в память командой *calendar show*:

```
*CLI> calendar show calendars
Calendar                Type        Status
--------                ----        ------
phone_call_calendar     caldav      free
```

С помощью нашего календаря, успешно загруженного в память, мы можем написать некоторый диалплан для команды `Dial()`, чтобы сохранить информацию о вызове в календарь с помощью функции `CALENDAR_WRITE()`:

```
[LocalSets]
exten => _NXXNXXXXXX,1,Verbose(2,Outbound calls)
   same => n,Set(CalendarStart=${EPOCH})  ; Используется CALENDAR_WRITE()
   same => n,Set(X=${EXTEN})              ; Используется CALENDAR_WRITE()
   same => n,Dial(SIP/ITSP/${EXTEN},30)
   same => n,NoOp(Handle standard voicemail stuff here)
   same => n,Hangup()

exten => h,1,Verbose(2,Call cleanup)

; Всё, что следует далее должно быть в одной строке
   same => n,Set(CALENDAR_WRITE(phone_call_calendar,summary,description,
start,end)=
      OUTBOUND: ${X},Phone call to ${X} lasted for ${CDR(billsec)} seconds.,
      ${CalendarStart},${EPOCH})
```

В нашем диалплане мы создали простой сценарий, в котором осуществляем исходящий вызов через нашего провайдера интернет-телефонии (ITSP), но перед выполнением вызова сохраняем epoch[3] в переменной канала (поэтому мы можем использовать ее позже, когда сделаем нашу запись в календаре в конце вызова). После вызова мы делаем запись в наш календарь `phone_call_calendar` с помощью функции `CALENDAR_WRITE()` во встроенном расширении `h`. Есть несколько параметров, которые мы можем передать в календарь, такие как сводка, описание, время начала и окончания. Вся эта информация затем сохраняется в календаре.

Мы также использовали функцию `CDR()` в нашем описании, чтобы показать количество секунд, в течение которых длился разговор, чтобы получить более точную оценку того, был ли дан ответ на вызов, и, если да, то как долго длился разговор. Мы также могли бы быть умнее и писать в календарь только в том случае, если `${CDR(billsec)}` был больше `0`, обрабатывая приложение `Set()` в приложении `ExecIf()`; например, `same => n,ExecIf($[${CDR(billsec)} > 0]? Set(CALENDAR_WRITE ...))`.

Существует много возможностей для функции `CALENDAR_WRITE()`, но это только то, что мы реализовали и наслаждаемся.

Дополнительные функции

Доступны другие функции календаря, такие как `CALENDAR_QUERY()`, которая позволяет получить список событий за определенный период времени для определенного календаря, и `CALENDAR_QUERY_RESULT()`, которая позволяет получить доступ к особенностям событий этого календаря. Кроме того, вы можете создать функцию, которая записывает события в ваш календарь с помощью `CALENDAR_WRITE()`: например, вы можете разработать некоторый диалплан, который

3 В Unix, epoch - это количество секунд, прошедших с 1 января 1970 года, не считая високосных секунд.

позволит вам выделять блоки времени в вашем календаре с телефона, когда вы находитесь в дороге без доступа к ноутбуку. Существует много возможностей, и все, что нужно, это немного творчества.

Интеграция голосовой почты с IMAP

“Единая система обмена сообщениями” была модным словом в телекоммуникационной отрасли на протяжении веков. Все дело в интеграции служб, чтобы пользователи могли получать доступ к одним и тем же типам данных в нескольких местах, используя разные методы. Одним из самых рекламируемых приложений является интеграция электронной и голосовой почты. Asterisk делал это в течение многих лет, но многие крупные компании все еще пытаются научиться этому. Asterisk имеет возможность отправлять пользователям голосовые сообщения по электронной почте, используя почтовый транспортный агент (Mail Transport Agent - MTA) в вашем дистрибутиве Linux (это всегда был *sendmail*, но и Postfix становится все более популярным как MTA). Голосовая почта на электронную является одной из старейших функций Asterisk, и обычно она работает без какой-либо конфигурации вообще.[4]

Интеграция Internet Message Access Protocol (IMAP) существует в Asterisk (и постоянно развивается) с версии 1.4. Интеграция голосовой почты IMAP означает, что пользователи могут получать доступ к голосовой почте через папку в своих учетных записях электронной почты, что дает им возможность прослушивать, пересылать и отмечать сообщения голосовой почты с той же гибкостью, что и приложение диалплана `VoiceMail()`. Asterisk будет знать о состоянии этих сообщений при следующем входе пользователей через телефонную систему.

По мере увеличения числа администраторов, интегрирующих Asterisk со своими IMAP-серверами, количество зарегистрированных и исправленных ошибок сначала увеличилось, а затем уменьшилось до такой степени, что интеграцию IMAP можно считать достаточно стабильной для использования в рабочей среде. В этом разделе мы обсудим, как скомпилировать поддержку голосовой почты IMAP и подключить систему голосовой почты к серверу IMAP.

Компиляция поддержки голосовой почты IMAP в Asterisk

Чтобы получить поддержку голосовой почты через IMAP в Asterisk, нам нужно скомпилировать библиотеку IMAP Университета Вашингтон. Набор инструментов UW IMAP даст нам функциональность для подключения к нашему серверу IMAP. Однако перед компиляцией программного обеспечения нам необходимо установить некоторые зависимости.

Зависимости для создания библиотеки IMAP включают инструменты, необходимые для сборки Asterisk, но способ, которым мы ее создаем, также требует библиотек разработки для OpenSSL и подключаемых модулей аутентификации (Pluggable Authentication Modules - PAM). Мы включили инструкции для RHEL и Ubuntu.

Зависимости RHEL

Установка библиотек разработки OpenSSL и PAM на RHEL может быть выполнена следующей командой:

```
$ sudo yum install openssl-devel pam-devel
```

Зависимости Ubuntu

Установка библиотек разработки OpenSSL и PAM на Ubuntu может быть выполнена следующей командой:

```
$ sudo apt-get install libssl-dev libpam0g-dev
```

4 Когда мы говорим “это работает", то имеем в виду, что Asterisk составит электронное письмо и отправит его в MTA, и электронное письмо будет успешно передано из системы. То, что происходит с ним после того, как оно покидает систему, немного сложнее и часто включает в себя спам-фильтры, рассматривающие почту как подозрительную и фактически не доставляющие ее. На самом деле это не вина Asterisk, но это то, с чем вам придется иметь дело.



Если вы попытаетесь установить *libpam-dev* на Ubuntu, он предупредит вас, что *libpam-dev* является виртуальным пакетом и вы должны явно установить один из пакетов в списке, который он представляет вам (который в нашем случае содержал только один пакет). Если *libpam0g-dev* не подходит для вашей версии Ubuntu, попробуйте установить виртуальный пакет. Это должно дать вам список допустимых пакетов для библиотеки разработки PAM.

Компиляция библиотеки IMAP

Теперь, когда наши зависимости удовлетворены, мы можем скомпилировать библиотеку IMAP, которую Asterisk будет использовать для подключения к серверу IMAP.

Первое, что нужно сделать, это перейти в каталог *thirdparty*, расположенный в каталоге *asterisk-complete*. Если вы еще не создали этот каталог, сделайте это сейчас:

```
$ cd ~/src/asterisk-complete
$ mkdir thirdparty
$ cd thirdparty
```

Далее идет загрузка набора инструментов IMAP и его компиляция. В следующих шагах будет получена последняя версия инструментария IMAP с сервера Университета Вашингтона (более подробная информация о наборе инструментов доступна по адресу http://www.washington.edu/imap/):

```
$ wget ftp://ftp.cac.washington.edu/mail/imap.tar.Z
$ tar zxvf imap.tar.Z
$ cd imap-2007e
```



Имя каталога *imap-2007e* может изменяться по мере появления новых версий набора инструментов.

Есть несколько опций, которые мы должны передать команде *make* при создании библиотеки IMAP, и значений, которые вы должны передать, зависящие от того, на какой платформе вы собираете (32- или 64-разрядная), если вам нужна поддержка OpenSSL, и нужна ли вам поддержка IPv6 или просто IPv4. Таблица 18-1 показывает некоторые из различных вариантов, которые вы могли бы выбрать на разных платформах.

Таблица 18-1. Опции времени компиляции библиотеки IMAP

Опция	Описание
`EXTRACFLAGS="-fPIC"`	Требуется при сборке на 64-разрядных платформах.
`EXTRACFLAGS="-I/usr/include/openssl"`	Используется для сборки с поддержкой OpenSSL.
`IP6=4`	Многие платформы, поддерживающие IPv6, предпочитают такой способ подключения, который может быть нежелателен для всех серверов. Если вы хотите принудительно использовать IPv4 в качестве предпочтительного способа подключения установите этот параметр.

Если вы посмотрите в *Makefile*, поставляемом с библиотекой IMAP, то найдете список платформ, для которых библиотека может быть скомпилирована. В нашем случае мы будем компилировать для RHEL или Ubuntu с поддержкой PAM. Если вы компилируете на других системах, посмотрите в *Makefile* трехбуквенный код, который сообщает библиотеке как компилироваться для вашей платформы.

Компиляция для 64-разрядной платформы с поддержкой OpenSSL и предпочтением подключения через IPv4:

```
$ make lnp EXTRACFLAGS="-fPIC -I/usr/include/openssl" IP6=4
```

Компиляция для 32-разрядной платформы с поддержкой OpenSSL и предпочтением подключения через IPv4:

```
$ make lnp EXTRACFLAGS="-I/usr/include/openssl" IP6=4
```

Если вы не хотите компилироваться с поддержкой OpenSSL, просто удалите `-I/usr/include/openssl` из дополнительного параметра `EXTRACFLAGS`. Если вы предпочитаете подключение по IPv6 по умолчанию, просто не указывайте параметр `IP6=4`.



При установке поддержки IMAP мы всегда компилировали библиотеку `c-client` из исходного кода. Однако она может быть доступна в виде пакета для вашего дистрибутива. Например, Ubuntu имеет доступный пакет *libc-client-dev*. Это может сработать и избавить вас от некоторых неприятностей, но мы не тестировали его.

Компиляция Asterisk

После компиляции библиотеки IMAP нам нужно перекомпилировать модуль *app_voicemail.so* с поддержкой IMAP. Первый шаг должен выполнить скрипт *configure* и передать ему параметр *--with-imap*, чтобы сообщить где находится библиотека IMAP:

```
$ cd ~/src/asterisk-complete/asterisk/11
$ ./configure --with-imap=~/src/asterisk-complete/thirdparty/imap-2007e/
```

После завершения выполнения скрипта *configure* необходимо включить поддержку голосовой почты IMAP в *menuselect*:

```
$ make menuselect
```

Из интерфейса *menuselect* зайдите в *Voicemail Build Options*. В этом меню вы должны иметь возможность выбрать `IMAP_STORAGE`.



Если у вас нет возможности выбрать эту опцию - убедитесь, что ваша библиотека IMAP была успешно собрана (т.е. что у вас установлены все необходимые зависимости и при сборке не произошло ошибки) и что вы правильно указали путь к библиотеке IMAP при запуске скрипта *configure*. Можно также проверить что библиотека IMAP была найдена правильно, посмотрев файл *config.log* (находится в каталоге сборки Asterisk) для IMAP.

После выбора `IMAP_STORAGE`, сохраните и выйдите из *menuselect* и запустите *make install*, который будет перекомпилировать модуль *app_voicemail.so* и установит его в соответствующее место. Следующий шаг - настройка файла *voicemail.conf* находящегося в */etc/asterisk*.

Настройка Asterisk

Теперь, когда мы скомпилировали поддержку IMAP в Asterisk, нам нужно включить ее, подключившись к серверу с поддержкой IMAP. Есть много IMAP-серверов, которые мы могли бы использовать, в том числе те, которые поставляются с серверами Microsoft, Dovecot и Cyrus на Unix, или веб-сервер IMAP такой, как тот, который поставляется Gmail от Google.[5] Наши инструкции покажут, как подключить Asterisk к учетной записи Gmail с включенным IMAP так как это требует наименьшего количества усилий чтобы поднять и запустить голосовую почту с IMAP, но эти инструкции могут быть легко адаптированы для использования с любым существующим сервером IMAP.

Включение IMAP в Gmail. Включение поддержки IMAP в вашей учетной записи Gmail является простым (см. Рисунок 18-1). После входа в учетную запись выберите *Settings* в правом верхнем углу. Затем выберите *Forwarding and POP/IMAP* в строке меню под заголовком *Settings*. В разделе *IMAP Access* выберите *Enable IMAP*. После включения IMAP нажмите кнопку *Save Changes* в нижней части экрана.

5 Недавно мы также проверили open source проект roundcube для веб-почты и были очень впечатлены.

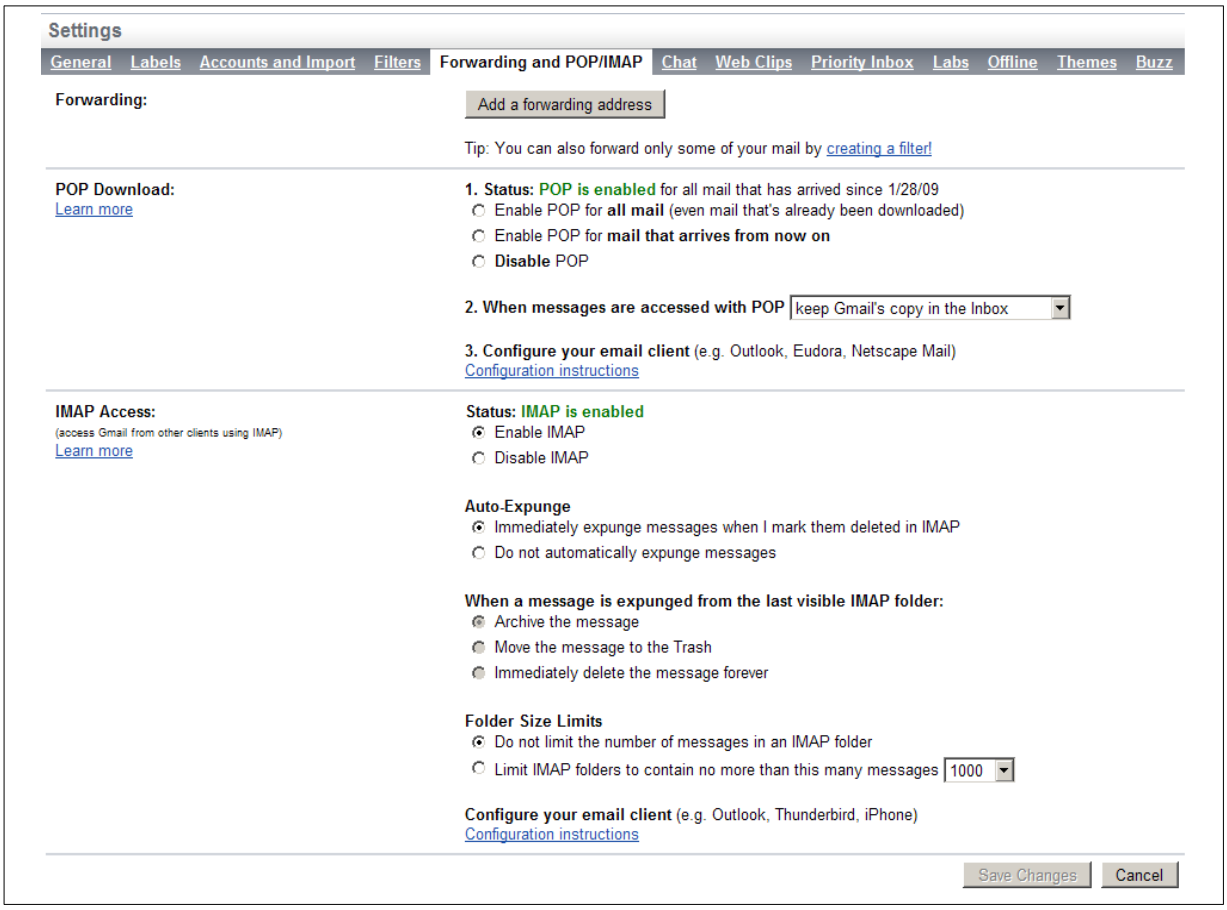


Рисунок 18-1. Включение IMAP в Gmail.

Настройка voicemail.conf для IMAP. Чтобы наша система голосовой почты могла подключаться к системе IMAP - нам необходимо убедиться, что поддержка IMAP встроена в модуль *app_voicemail.so* в соответствии с инструкцией в "Компиляция Asterisk". С поддержкой IMAP, скомпилированной в Asterisk, нам просто нужно указать модулю голосовой почты, как подключаться к нашему серверу IMAP.

Мы продемонстрируем как подключиться к учетной записи Gmail с поддержкой IMAP и использовать ее для хранения и получения сообщений голосовой почты. Если вы еще этого не сделали, прочитайте раздел "Включение IMAP в Gmail", прежде чем продолжить. Последний шаг - настройка *voicemail.conf* для подключения к серверу.

В *voicemail.conf* добавьте следующие строки в раздел `[general]`. Убедитесь, что вы указываете только один формат (мы рекомендуем `wav49`) для записей голосовой почты и удалите все ссылки на хранилище голосовой почты ODBC, если вы включали их ранее:

```
[general]
format=wav49                ; формат хранения файлов
imapserver=imap.gmail.com   ; расположение IMAP-сервера
imapport=993                ; порт IMAP-сервера
imapflags=ssl               ; флаги, необходимые для подключения
expungeonhangup=yes         ; удалять сообщения когда кладут трубку
pollmailboxes=yes           ; используется для индикации ожидающего сообщения
pollfreq=30                 ; как часто проверять изменения сообщений
```

Прежде чем настроить нашего пользователя для подключения к серверу IMAP Gmail, давайте обсудим параметры, которые мы только что установили в разделе `[general]`. Это основные параметры, которые помогут нам начать; мы сделаем еще несколько настроек в ближайшее время, но давайте посмотрим, что мы сделали уже.

Во-первых, опция `format=wav49` объявила, что мы собираемся сохранить наши файлы как GSM с заголовком WAV, который можно воспроизводить на большинстве настольных компьютеров (включая Microsoft Windows), сохраняя при этом небольшой размер файла.

Затем мы настроили Asterisk для подключения к `imapserver`, расположенному в `imap.gmail.com` на `imapport 993`. Мы также установили `imapflags` на `ssl`, так как Gmail требует безопасного соединения. Без флага `ssl` IMAP сервер отклонит наши попытки подключения (поэтому было важно, чтобы мы скомпилировали нашу библиотеку IMAP с поддержкой OpenSSL). Другой вариант, который может потребоваться на частных серверах IMAP, таких как Dovecot - указать `novalidate-cert` для `imapflags`, когда требуется SSL-соединение, но сертификат не создается Центром сертификации.

Далее, мы установили `expungeonhangup=yes`, в результате чего сообщения, помеченные на удаление, удаляются с сервера когда разрывают соединение с приложением `VoiceMail()`. Без этой опции сообщения просто помечаются как прочитанные и остаются на сервере, пока не будут удалены через почтовое приложение или веб-интерфейс.

Для корректного получения обновлений индикации ожидания сообщения (Message Waiting Indication - MWI) необходимо включить `pollmailboxes=yes`, в результате чего Asterisk будет проверять сервер на наличие изменений состояния сообщений. Например, когда кто-то оставляет нам голосовую почту, и мы слушаем ее, открывая сообщение через наше приложение электронной почты, сообщение будет помечено как прочитанное, но без опроса почтового ящика Asterisk не будет иметь возможности узнать об этом и включит индикатор MWI на связанном устройстве на неопределенный срок. Наконец, мы установили соответствующий параметр `pollfreq` равным `30` секундам. Этот параметр определяет как часто Asterisk будет запрашивать у сервера статус сообщений: установите его соответствующим образом, чтобы контролировать объем трафика, поступающего на сервер голосовой почты.

В Таблице 18-2 показаны некоторые другие доступные нам варианты.

Таблица 18-2. Дополнительные опции голосовой почты IMAP

Опция	Описание
`imapfolder`	Имя папки, в которой будут храниться сообщения голосовой почты IMAP-сервера. По умолчанию они хранятся в папке *INBOX*.[a]
`imapgreetings`	Определяет, хранятся ли приветствия голосовой почты на сервере IMAP или на сервере локально. Допустимые значения: `yes` или `no`.
`imapparentfolder`	Определяет родительскую папку на сервере IMAP. Обычно настроено как *INBOX*, но если она используется где-то еще, можно указать её здесь.
`greetingfolder`	Указывает папку для сохранения приветствий голосовой почты, если параметр `imapgreetings` включен значением `yes`. По умолчанию приветствия сохраняются в папке *INBOX*.
`authuser`	Задает главного пользователя для подключения к серверу IMAP если на сервере настроен один пользователь, имеющий доступ ко всем почтовым ящикам.
`authpassword`	Дополнение к директиве `authuser`. Дополнительные сведения см. в разделе `authuser`.
`opentimeout`	Указывает таймаут открытия TCP (в секундах).
`closetimeout`	Указывает таймаут закрытия TCP (в секундах).
`readtimeout`	Указывает таймаут чтения TCP (в секундах).
`writetimeout`	Указывает таймаут записи TCP (в секундах).

[a] Важно хранить сообщения голосовой почты в папке, отличной от папки INBOX, если количество сообщений, содержащихся в папке INBOX может быть довольно большим. Asterisk попытается собрать информацию обо всех сообщениях электронной почты, содержащихся в папке INBOX, и может либо истечь время ожидания, прежде чем получить всю информацию, либо просто занять очень много времени для сохранения или извлечения сообщений голосовой почты, что нежелательно.

Настроив раздел `[general]`, давайте определим почтовый ящик для подключения к серверу IMAP.

В Главе 8 мы определили некоторых пользователей в контексте голосовой почты `[shifteight]`. Вот исходная конфигурация, определенная в этой главе:

```
[shifteight]
100 => 0107,Leif Madsen,leif@shifteight.org
101 => 0523,Jim VanMeggelen,jim@shifteight.org,,attach=no|maxmsg=100
102 => 11042,Tilghman Lesher,,,attach=no|tz=central
```

Мы собираемся изменить почтовый ящик 100 таким образом, чтобы он подключался к серверу IMAP Gmail для хранения и извлечения сообщений голосовой почты:

```
[shifteight]
100 => 0107,Leif Madsen,,,|imapuser=leif@shifteight.org|imappassword=secret
```



Файл *voicemail.conf* использует как запятые, так и пайпы в качестве разделителей, в зависимости от используемого поля. Первые несколько полей имеют определенные настройки, а последнее поле может содержать дополнительную информацию о почтовом ящике, разделенную символом пайпа (|).

Мы удалили адрес электронной почты из третьего поля, потому что больше не будем использовать *sendmail* для отправки голосовых сообщений: теперь они будут храниться непосредственно на почтовом сервере. Мы настроили почтовый ящик для соединения с именем пользователя IMAP `leif@shifteight.org` (потому что мы включили Google Apps для домена, в котором размещается наша электронная почта) и подключаемся с помощью IMAP пароля `secret`.

После настройки Asterisk нам нужно перезагрузить модуль *app_voicemail.so*. Если вы включаете отладку консоли, то при подключении к серверу голосовой почты должны отображаться следующие выходные данные:

```
*CLI> core set debug 10
*CLI> module reload app_voicemail.so
DEBUG[3293]: app_voicemail.c:2734 mm_log: IMAP Info:
             Trying IP address [74.125.53.109]
DEBUG[3293]: app_voicemail.c:2734 mm_log: IMAP Info: Gimap ready for requests
             from 99.228.XXX.XXX 13if2973206wfc.0
DEBUG[3293]: app_voicemail.c:2757 mm_login: Entering callback mm_login
DEBUG[3293]: app_voicemail.c:2650 mm_exists:
             Entering EXISTS callback for message 7
DEBUG[3293]: app_voicemail.c:3074 set_update:
             User leif@shifteight.org mailbox set for update.
DEBUG[3293]: app_voicemail.c:2510 init_mailstream: Before mail_open, server:
             {imap.gmail.com:993/imap/ssl/user=leif@shifteight.org}INBOX, box:0
DEBUG[3293]: app_voicemail.c:2734 mm_log: IMAP Info: Reusing connection to
             gmail-imap.l.google.com/user="leif@shifteight.org"
```

При появлении `ERROR` ошибок проверьте конфигурацию и убедитесь, что библиотека IMAP скомпилирована с поддержкой SSL. Когда *app_voicemail.so* подключен, попробуйте оставить себе голосовую почту; затем проверить её через веб-интерфейс Gmail и убедитесь, что ваше сообщение хранится правильно. У вас также должен быть индикатор MWI на вашем устройстве, если оно поддерживает его, и если вы настроили `mailbox=100@shifteight` для устройства в *sip.conf*. Если вы загружаете сообщение голосовой почты и отмечаете его как прочитанное, индикатор MWI должен выключиться в течение 30 секунд (или другого значения, которое вы устанавливаете для `pollfreq` в *voicemail.conf*).

Использование XMPP (Jabber) с Asterisk

Расширяемый протокол обмена сообщениями о присутствия (XMPP, ранее называемый Jabber) используется для обмена мгновенными сообщениями и передачи информации о присутствии по сетям в режиме реального времени. В Asterisk он также используется для настройки вызова

(сигнализации). Мы можем делать различные интересные вещи при интеграции с XMPP, например получать сообщение, когда кто-то звонит нам. Мы даже можем отправлять сообщения обратно на Asterisk, перенаправляя наши звонки на голосовую почту или в другое место. Кроме того, с `chan_motif` мы можем принимать и совершать звонки через сеть Google Voice или принимать звонки от пользователей Google Talk через веб-клиент.

Компиляция поддержки XMPP в Asterisk

Модуль `res_xmpp` содержит различные приложения и функции диалплана, которые полезны из диалплана Asterisk. Он также является зависимостью модуля канала `chan_motif`. Начать с интеграции XMPP в Asterisk нужно скомпилировав `res_xmpp`.

Зависимости RHEL

Для установки `res_xmpp` нам нужна библиотека разработки iksemel. Если установлена библиотека разработки OpenSSL, `res_xmpp` также будет использовать ее для безопасных соединений (рекомендуется). Мы можем установить их обе на RHEL с помощью следующей команды:

```
$ sudo yum install iksemel-devel openssl-devel
```



Библиотека iksemel будет установлена через EPEL.

Зависимости Ubuntu

Для установки `res_xmpp` нам нужна библиотека разработки `iksemel`. Если установлена библиотека разработки OpenSSL, `res_xmpp` также будет использовать ее для безопасных соединений (рекомендуется). Мы можем установить их обе в Ubuntu с помощью следующей команды:

```
$ sudo apt-get install libiksemel-dev libssl-dev
```

Установка res_xmpp

После установки зависимостей необходимо выполнить ./configure в исходниках Asterisk и make menuselect. Затем перейдите в меню Resource Modules и убедитесь, что res_xmpp включен. После этого запустите make install, чтобы получить новые модули.

Команды диалплана Jabber

Для связи по XMPP через Asterisk можно использовать несколько приложений и функций диалплана. Мы рассмотрим, как подключить Asterisk к серверу XMPP, как отправлять сообщения клиенту из диалплана и как маршрутизировать вызовы на основе ответов на первоначально отправленные сообщения. Отправляя сообщение через XMPP мы, по сути, создаем простое приложение для всплывающих окон, чтобы пользователи знали, когда вызовы поступают в систему.

Подключение к XMPP серверу

Прежде чем мы сможем начать отправлять сообщения нашим участникам XMPP, нам нужно подключиться к серверу с поддержкой XMPP. Мы собираемся использовать сервер XMPP в Google, так как он открыт и доступен для всех. Для этого необходимо настроить *xmpp.conf* в каталоге конфигурации */etc/asterisk*. Следующий пример подключит нас к серверу XMPP в Google.



У вас уже должна быть учетная запись Gmail, которую вы можете получить на http://www.gmail.com.

Наш *xmpp.conf* должен выглядеть так:

```
[general]

[asterisk]
type=client
serverhost=talk.google.com
username=asterisk@shifteight.org
secret=<super_secret_password>
priority=1
port=5222
usetls=yes
usesasl=yes
status=available
statusmessage="Ohai from Asterisk"
timeout=5
```

Давайте взглянем на некоторые параметры, которые мы только что установили, для понимания что происходит. Параметры описаны в Таблице 18-3. Мы описали все доступные параметры в *xmpp.conf*, несмотря на то, что не установили все параметры в нашем примере файла.

Таблица 18-3. Опции xmpp.conf

Опция	Описание
`debug`	Включает/отключает отладку сообщений XMPP (которая может быть довольно подробной). Доступные варианты `yes` или `no`.
`autoprune`	Включает/отключает автоматическое удаление пользователей из списка участников каждый раз, когда *res_xmpp.so* подключается к вашему аккаунту. Не используйте это для учетных записей, которые вы можете использовать за пределами Asterisk (например, ваш персональный аккаунт). Доступные параметры: `yes` или `no`.
`autoregister`	Указывает, следует ли автоматически регистрировать пользователей из списка ваших друзей в памяти. Доступные параметры: `yes` или `no`.
`collection_nodes`	Включает поддержку расширения XEP-0248 XMPP. Оно может быть включено для использования с распределенными состояниями устройств. Доступные параметры: `yes` или `no`. По умолчанию `no`.
`pubsub_autocreate`	Если сервер PubSub автоматически создает узлы, то Asterisk должен явно предварительно создать узел перед публикацией. Доступные параметры: `yes` или `no`. По умолчанию `no`.
`auth_policy`	Определяет, следует ли автоматически принимать запросы на подписку. Доступные параметры: `accept` или `deny`.
`type`	Устанавливает тип клиента, с которым мы будем соединяться. Доступные параметры - `client` или `component`. (Вы почти всегда будете выбирать `client`.)
`serverhost`	Указывает к какому хосту должно подключаться это соединение (например, *talk.google.com*).
`pubsub_node`	Имя узла, используемого для публикации событий через PubSub. Значение является строкой.
`username`	Предоставляет имя пользователя, которое будет использоваться для подключения к `serverhost` (например *asterisk@gmail.com*).
`secret`	Задает пароль, который будет использоваться для подключения к `serverhost`.
`port`	Указывает на какой порт мы попытаемся установить соединение с `serverhost` (например 5222).
`usetls`	Указывает, следует ли использовать TLS при подключении к `serverhost`. Доступные параметры: `yes` или `no`.
`usesasl`	Указывает, следует ли использовать SASL при подключении к `serverhost`. Доступные параметры: `yes` или `no`.
`status`	Определяет состояние соединения по умолчанию при входе в учетную запись. Доступные варианты: `chat`, `available`, `away`, `xaway`, и `dnd`.

Опция	Описание
`statusmessage`	Задает пользовательское сообщение о состоянии, используемое при подключении к Asterisk, например "`Connected Via Asterisk`"; используйте двойные кавычки вокруг сообщения.
`buddy`	Вручную добавляет приятелей в список при подключении к серверу. Вы можете указать несколько друзей на нескольких линиях `buddy` (например, `buddy=jim@shifteight.org`).
`timeout`	Указывает время ожидания (в секундах), в течение которого сообщения хранятся в стеке сообщений. По умолчанию 5 секунд. Эта опция применяется только ко входящим сообщениям, которые предназначены для обработки функцией диалплана `JABBER_RECEIVE()`.
`priority`	Определяет приоритет этого ресурса по отношению к другим ресурсам. Чем меньше число, тем выше приоритет.
`sendtodialplan`	Если эта опция включена, входящие сообщения будут отправляться в диалплан. Доступные параметры: `yes` или `no`. По умолчанию `no`. (См. раздел "Внешние сообщения (инфраструктура обмена сообщениями)"
`context`	Если параметр `sendtodialplan` включен, то указывает контекст для отправки сообщений. Контекст по умолчанию - `default`.

После конфигурирования файла *xmpp.conf* мы можем загрузить (или перезагрузить) модуль *res_xmpp.so*.

Сделать это можно из консоли с помощью *module reload res_xmpp.so*:

```
*CLI> module reload res_xmpp.so
Reloaded res_xmpp.so
```

и проверим подключение с помощью команды *xmpp show connections*:

```
*CLI> xmpp show connections
Jabber Users and their status:
       User: asterisk@shifteight.org      - Connected
----
   Number of users: 1
```

Если у вас возникли проблемы с подключением вы можете попробовать выгрузить модуль, а затем загрузить его обратно в память. Если проблемы не устранены, то можно выполнить команду *xmpp purge nodes* чтобы удалить все существующие или поврежденные соединения из памяти. Кроме того, проверьте конфигурацию и убедитесь что у вас нет проблем с конфигурацией или опечаток. После того, как вы подключились, можете перейти к следующим разделам, где начинается самое интересное.

Отправка сообщений через JabberSend()

Приложение диалплана `JabberSend()` используется для отправки сообщений приятелям из диалплана Asterisk. Вы можете использовать это приложение в любом месте, где обычно используете диалплан, что делает его довольно гибким. Мы собираемся использовать его в качестве экранного приложения для отправки сообщения клиенту перед тем, как позвонить на телефон пользователя. В зависимости от используемого клиента сообщение может появиться на экране пользователя на панели задач.

Вот простой пример для начала:

```
[LocalSets]
exten => 104,1,Answer()

; *** Эта строка не должна содержать разрывов
   same => n,JabberSend(asterisk,jim@shifteight.org,Incoming call from
${CALLERID(all)})
   same => n,Dial(SIP/0000FFFF0002,30)
   same => n,Hangup()
```

Этот пример демонстрирует как использовать приложение `JabberSend()` для отправки сообщения кому-то до набора его номера. Давайте разберем значения, которые мы использовали. Первый аргумент - `asterisk` - это заголовок раздела, который мы определили в файле *jabber.conf* как `[asterisk]`. В нашем примере *xmpp.conf* мы задали пользователю с именем *asterisk@shifteight.org* отправлять сообщения через сервер Google XMPP, а `asterisk` - это название раздела, которое мы определили. Второй аргумент `jim@shifteight.org` - это приятель, кому мы посылаем сообщение. Мы можем определить любого приятеля здесь, либо как голый JID (как мы делали выше), либо как полный JID с ресурсом (например, *jim@shifteight.org/laptop*). Третий аргумент `JabberSend()` - это сообщение, которое мы хотим отправить приятелю. В этом случае мы отправляем `Incoming call from ${CALLERID(all)}`, а для ввода информации об CallerID абонента в сообщении используется функция диалплана `CALLERID()`.

Очевидно, что нам придется дополнительно построить наш диалплан, чтобы сделать это полезным: в частности, нам нужно будет связать имя приятеля (например, `jim@shifteight.org`) с устройством, которое мы вызываем (`SIP/0000FFFF0002`) для отправки сообщений правильному приятелю. Вы можете сохранить эти ассоциации в любых местах, например, в AstDB, в реляционной базе данных, извлекая с помощью `func_odbc` или даже в глобальной переменной.

Получение сообщений через JABBER_RECEIVE()

Функция диалплана `JABBER_RECEIVE()` позволяет нам получать ответы через сообщения XMPP, захватывать эти ответы и, предположительно, реагировать на них. Обычно мы используем функцию `JABBER_RECEIVE()` в сочетании с приложением `JabberSend()`, так как нам, скорее всего, потребуется отправить кому-то сообщение и предложить ему приемлемые значения, которые он может вернуть. Мы могли бы использовать функцию `JABBER_RECEIVE()`, либо для прямых вызовов на определенное устройство, такое как сотовый или стационарный телефон, или как текстовую версию автосекретаря, используемую для людей, которые плохо слышат подсказки (Например, для глухих пользователей или работающих в шумной рабочей среде). В последнем случае система должна быть предварительно настроена, чтобы знать, куда отправлять сообщения, возможно, на основе CallerID.

Вот простой пример, который отправляет сообщение кому-то, ждет ответа, а затем направляет вызов на основе ответа:

```
exten => 106,1,Answer()

   ; Весь текст должен размещаться в одну строку.
   same => n,JabberSend(asterisk,leif.madsen@gmail.com,Incoming call from
${CALLERID(all)}. Press 1 to route to desk. Press 2 to send to voicemail.)[6]

   same => n,Set(JabberResponse=${JABBER_RECEIVE(asterisk,leif@shifteight.org)})
   same => n,GotoIf($["${JabberResponse}" = "1"]?dial,1)
   same => n,GotoIf($["${JabberResponse}" = "2"]?voicemail,1)
   same => n,Goto(dial,1)

exten => dial,1,Verbose(2,Calling our desk)
   same => n,Dial(SIP/0000FFFF0002,6)
   same => n,Goto(voicemail,1)
   exten => voicemail,1,Verbose(2,VoiceMail)

; *** Эта строка не должна содержать разрывов
   same => n,Set(VoiceMailStatus=${IF($[${ISNULL(${DIALSTATUS})}
| "${DIALSTATUS}" = "BUSY"]?b:u)})
   same => n,Playback(silence/1)
   same => n,VoiceMail(100@lmentinc,${VoiceMailStatus})
   same => n,Hangup()
```

6 Входящий вызов для `${CALLERID(all)}`. Нажмите 1 для соединения. Нажмите 2 для отправки голосовой почты.



К сожалению, приложение JabberSend() требует чтобы весь текст сообщения был в одной строке. Если вы хотите разбить текст на несколько строк, вам нужно будет отправить его как несколько сообщений на отдельных строках с помощью JabberSend().

Наш простой диалплан сначала отправляет сообщение учетной записи Jabber (*leif@shifteight.org*) через учетную запись Jabber нашей системы (`asterisk`), настроенную в *xmpp.conf*. Затем мы используем функцию `JABBER_RECEIVE()` для ожидания ответа от *leif@shifteight.org*. Время ожидания по умолчанию составляет 5 секунд (как определено в *xmpp.conf*), но вы можете указать другой таймаут в третьем аргументе `JABBER_RECEIVE()`. Например, чтобы подождать ответа 10 секунд, мы должны были использовать такую строку:

```
Set(JabberResponse=${JABBER_RECEIVE(asterisk,leif@shifteight.org,10)})
```

Как только мы или получили ответ или таймаут истек, то переходим к следующей строке абонентской группы, которая начинает проверять ответ, сохраненный к переменной канала `${JabberResponse}`. Если значение равно `1`, мы продолжим наш диалплан на `dial,1` текущего контекста. Если ответ `2`, мы продолжаем наш диалплан в `voicemail,1`. Если ответ не получен (или неизвестен), то продолжаем в `dial,1`.

В диалплане `dial,1` и `voicemail,1` должны быть довольно очевидными. Это - не производственный пример; должна быть реализована некоторая дополнительная абонентская группа, чтобы сделать значения динамическими.

Однако, есть недостаток в том, как мы реализовали функцию `JABBER_RECEIVE()`. Наша функция блокирует или ожидает ответа от конечной точки. Если мы установим значение ответа низким, чтобы минимизировать задержку, мы не дадим получателю достаточно времени для ответа. Однако, если мы устанавливаем время для ответа достаточно большим, чтобы получатель мог комфортно ответить, мы указываем ненужную задержку в вызове устройства или отправке на голосовую почту.

Мы можем обойти эту проблему, используя локальный канал. Это позволяет нам выполнять два раздела диалплана одновременно, посылая вызов устройству и, в то же время, ожидая ответа от `JABBER_RECEIVE()`. Если мы получаем ответ от `JABBER_RECEIVE()` и нам нужно что-то сделать, мы можем выполнить `Answer()` на линию и заставить этот раздел диалплана продолжиться. Если устройство ответит на звонок, наш диалплан с `JABBER_RECEIVE()` будет просто отключен. Рассмотрим модифицированную абонентскую группу, реализующую локальный канал:

```
exten => 106,1,Verbose(2,Example using the Local channel)
   same => n,Dial(Local/xmpp@${CONTEXT}/n&Local/dial@${CONTEXT}/n)

exten => xmpp,1,Verbose(2,Send an XMPP message and expect a response)
; *** Эта строка не должна содержать разрывов
   same => n,JabberSend(asterisk,leif.madsen@gmail.com,Incoming call from
${CALLERID(all)}. Press 2 to send to voicemail.)

   same => n,Set(JabberResponse=${
JABBER_RECEIVE(asterisk,leif@shifteight.org,6)})
   same => n,GotoIf($["${JabberResponse}" = "2"]?voicemail,1)
   same => n,Hangup()

exten => dial,1,Verbose(2,Calling our desk)
   same => n,Dial(SIP/0000FFFF0002,15)
   same => n,Goto(voicemail,1)

exten => voicemail,1,Verbose(2,VoiceMail)
   same => n,Answer()

; *** Эта строка не должна содержать разрывов
```

```
   same => n,Set(VoiceMailStatus=${IF($[${ISNULL(${DIALSTATUS})}
| "${DIALSTATUS}" = "BUSY"]?b:u)})
   same => n,Playback(silence/1)
   same => n,VoiceMail(100@shifteight,${VoiceMailStatus})
   same => n,Hangup()
```

Добавляя оператор `Dial()` в начало и перемещая нашу функцию отправки и получения Jabber в новое расширение под названием `xmpp` мы гарантируем, что можем одновременно вызвать расширение `dial` и расширение `xmpp`.

Обратите внимание, что мы удалили приложение `Answer()` из первой строки примера. Причина этого в том, что мы хотим выполнять `Answer()` на строку только после ответа устройства (что приводит к завершению расширения `xmpp`); в противном случае мы хотим, чтобы расширение `voicemail` выполнило `Answer()` на линию. Если `voicemail` ответило на линию это означает, что либо расширение `xmpp` получит ответ и укажет сделать `Goto()` к расширению `voicemail,` либо `Dial()` для нашего телефона истекло, вызывающее расширение голосовой почты должно быть выполнено, сделав тем самым линию Answer().

Приведенные здесь примеры служат трамплином для разработки многофункциональных приложений, которые отправляют и получают сообщения через серверы XMPP. Некоторые другие приложения и функции дилплана могут помочь в разработке приложений, такие как `JABBER_STATUS()` (или приложение диалплана `JabberStatus()`), которые используются для проверки статуса приятеля; применяя `JabberJoin()` и `JabberLeave()`, которые используются для присоединения и отключения XMPP конференц-залов; и приложение `JabberSendGroup()`, которое позволяет отправлять сообщения по протоколу XMPP комнатам чата.

chan_motif

Модуль `chan_motif` можно использовать для подключения к сетям с поддержкой Jingle. Он поддерживает три различных транспортных и производных протокола (в порядке предпочтения):

- Jingle как определено в спецификации XEP-0166
- Google Jingle, который следует спецификации Jingle для сигнализации, но использует пользовательский транспорт Google для медиапотока
- Google-V1, оригинальная реализация Google предварительной спецификации Jingle

Модуль канала может автоматически определить какой транспорт использовать для входящих сессий. Для исходящих сеансов он может определить транспорт автоматически, если вызываемая конечная точка находится в списке. Если оконечная точка не находится в списке, то `chan_motif` попытается соединиться в предпочтительном порядке, как перечислено в пунктах маркированного списка выше, пока все возможности или не будут успешны или не потерпят неудачу.

В наших примерах мы будем использовать Google Talk (GTalk) для совершения и приема вызовов, так как он позволяет людям, использующим веб-приложения и другие инструменты Google, легко совершать звонки. GTalk - это веб-голосовая система, обычно используемая в веб-интерфейсе Gmail. Существуют разные клиенты и дополнения для внешних приложений, таких как Pidgin, но мы будем тестировать с веб-клиентом от Google.

Амортизация chan_gtalk и chan_jingle

При создании `chan_motif` использование `chan_gtalk` и `chan_jingle` следует считать устаревшим. В версиях, предшествующих Asterisk 11, для подключения к службам Gtalk и Jingle использовались драйверы `res_jabber` и `gtalk` и канал `jingle`. Однако эти драйверы каналов имеют довольно старые реализации и обслуживание этих модулей стало затруднительным. При

этом `chan_motif` был создан для поддержки подключения к сервисам Jingle и Google Talk. Более подробная информация о причинах создания `chan_motif` доступна в блоге Digium.

Прежде чем мы сможем подключиться к `chan_motif` мы должны убедиться, что мы связаны через `res_xmpp`, так что если вы ещё так не сделали, пересмотрите "Подключение к XMPP-серверу" для получения информации о том, как подключиться к XMPP-серверу Google.

motif.conf

После того, как мы подключились через `res_xmpp`, теперь можем настроить файл *motif.conf*, который определяет наше соединение для приема и совершения вызовов. Простой файл конфигурации для разрешения подключений по сети Google (например, вызовы через веб-интерфейс, встроенный в Gmail) выглядит следующим образом:

```
[google]
context=google_incoming
disallow=all
allow=ulaw
connection=asterisk
```

Кажется довольно простым, но давайте рассмотрим нашу конфигурацию более подробно.

Во-первых, мы определяем имя нашего соединения, которое мы назвали `google` (имя между квадратными скобками). Вы можете назвать его как угодно, но это имя мы будем использовать для ссылки на эту учетную запись в нашем диалплане при совершении исходящего вызова. Следующая строка - имя контекста в нашем диалплане, на который будут направляться входящие вызовы. Затем мы отключаем все ранее включенные кодеки, а после включаем `ulaw` для этого соединения. В последней строке мы ссылаемся на наш *xmpp.conf* с помощью XMPP-соединения, настроенного в предыдущем разделе.

После того, как мы определили наш раздел, нам нужно загрузить *chan_motif.so* (если он ранее не был загружен) или перезагрузить его, чтобы драйвер канала мог прочитать новую конфигурацию:

```
*CLI> module reload chan_motif.so
*CLI>    -- Reloading module 'chan_motif.so' (Motif Jingle Channel Driver)
```

На данный момент наш модуль имеет свою конфигурацию и мы можем перейти к настройке диалплана, чтобы разрешить входящие и исходящие вызовы.

Прием звонков из Google Talk

Чтобы принимать звонки из Google Talk нам нужно добавить некоторые функции в наш диалплан. Сначала определим имя контекста в *extensions.conf*, который мы настроили в *motif.conf*:

```
[google_incoming]
```

После определения нашего контекста мы можем добавить внутренний номер для маршрутизации вызовов. Входящие звонки из сети Google будут попадать на расширение `s`[7] так же, как аналоговый канал, поступающий через интерфейс DAHDI. Как только входящий вызов совпадает с внутр.номером, мы можем выполнить любую маршрутизацию, которую захотим, так же как с любым другим входящим вызовом. Чтобы сделать проще, мы направим вызов на одну из конечных точек SIP, которые мы настроили ранее:

```
[google_incoming]
exten => s,1,Verbose(2,Incoming call via Google from ${CALLERID(all)})
   same => n,Dial(SIP/0000FFFF0001,24)
   same => n,Voicemail(100@default,u)
   same => n,Hangup()
```

7 Дополнительную информацию смотрите в разделе "Расширение s" в Главе 7.

Конечно, мы можем проявить здесь немного больше фантазии и могли бы выполнить проверку вызовов, перенаправление на автосекретаря, переадресацию на мобильный телефон, или любые другие методы, которые захотели бы использовать.

После настройки нашего диалплана необходимо запустить *dialplan reload*, чтобы сделать его активным. После этого попробуйте позвонить из другого веб-интерфейса Gmail (не из той учетной записи, которую вы настроили для системы Asterisk), и ваш звонок должен быть перенаправлен на конечную точку SIP. Вы даже можете увидеть, что входящий абонент добавлен в список друзей вашей системы Asterisk:

```
*CLI> xmpp show buddies
XMPP buddy lists
Client: asterisk
    Buddy:    shifteight@gmail.com
        Resource: gmail.64F29731
        node: http://mail.google.com/xmpp/client/caps
            version: 1.1
            Google Talk capable: yes
            Jingle capable: no
```

Прием звонков из Google Voice

Конфигурация для приема звонков из Google Voice аналогична (если не идентична) конфигурации для Google Talk, которую мы создали в предыдущем разделе. Небольшой совет, однако, заключается в том, что иногда вы не сможете отключить функцию скрининга вызовов (по какой-то причине мы все еще получаем ее, даже когда мы отключили в панели управления Google Voice). Если вы столкнулись с этой проблемой, но не хотите, чтобы скрининг ваших вызовов был включен, вы можете автоматически отправить DTMF перед вызовом устройства, добавив три выделенные линии, показанные здесь перед выполнением `Dial()`:

```
[google_incoming]
exten => s,1,Verbose(2,Incoming call via Google from ${CALLERID(all)})
   same => n,Answer()
   same => n,Wait(2)
   same => n,SendDTMF(1)
   same => n,Dial(SIP/0000FFFF0001,24)
   same => n,Voicemail(100@default,u)
   same => n,Hangup()
```

Здесь мы используем приложения `Wait()` и `SendDTMF()`, чтобы сначала подождать 2 секунды после ответа на вызов (то есть, когда начнется скрининг вызова), а затем принять вызов автоматически (отправив DTMF-сигнал цифру 1). После этого мы отправляем вызов на наше устройство.

Исходящие вызовы через Google Talk

Чтобы позвонить пользователю Google Talk, настройте свой диалплан следующим образом:

```
[LocalSets]
exten => 123,1,Verbose(2,Extension 123 calling shifteight@gmail.com)
   same => n,Dial(Motif/google/shifteight@gmail.com,30)
   same => n,Hangup()
```

В `Motif/google/shifteight@gmail.com` часть строки `Dial()` может быть разбита на три части. Первая часть, `Motif` - это протокол, который мы используем для исходящего вызова. Вторая часть, `google` является именем учетной записи, как определено в *motif.conf*. Последняя часть, `shifteight@gmail.com` - это место, куда мы пытаемся звонить.



Человек, которому вы звоните через Google Talk, должен уже существовать в вашем списке приятелей. Если вы используете существующее соединение (например, соединение с

клиентами мгновенного обена сообщениями), возможно, у вас уже есть приятели в списке. Если нет, то вам нужно попросить своих приятелей сперва позвонить вам, чтобы Asterisk мог автоматически принять и добавить их в список друзей. Это требование сети Google, а не Asterisk. Контроль авторегистрации приятелей осуществляется через `autoregister` для вашего подключения в файле *xmpp.conf*.

Исходящие вызовы через Google Voice

Чтобы совершать звонки с помощью Google Voice на номера ТфОП, создайте следующий диалплан:

```
[LocalSets]
exten => _1NXXNXXXXXX,1,Verbose(2,Placing call to ${EXTEN} via Google Voice)
   same => n,Dial(Motif/google/${EXTEN}@voice.google.com)
   same => n,Hangup()
```

Давайте кратко обсудим строку `Dial()`, чтобы вы поняли что происходит. Начнем с `Motif` - технологии, с которой мы будем звонить. После этого мы определили пользователя `google` как соединение, которое мы будем использовать при совершении нашего исходящего вызова (это настроено в *motif.conf*). Далее идет номер, на который мы пытаемся позвонить, как определено переменной канала `${EXTEN}`. Мы добавили `@voice.google.com` чтобы серверы Google знали, что это вызов, который должен быть совершен через Google Voice[8] в отличие от другого пользователя Google Talk.

Внешние сообщения (инфраструктура обмена сообщениями)

Начиная с Asterisk 10 была добавлена возможность принимать сообщения вне диапазона (например, получать сообщение без существующего канала), чтобы разрешить запуск диалплана на основе полученного сообщения. Возможность принять сообщение означает что вы потенциально можете сделать вызов, просто отправив сообщение в диалплан с клиента XMPP. Как вы можете себе представить, существует множество возможностей для этого, в том числе возможность предоставлять опции меню до звонка, поиск людей в каталоге или получение информации о компании, никогда не делая звонок в систему. И как только кто-то будет готов принять звонок, он может попросить систему позвонить ему и соединить с агентом.

Инфраструктура обмена сообщениями позволяет как получать, так и отправлять сообщения через SIP или XMPP, используя комбинацию функции диалплана `MESSAGE()`, приложения `MessageSend()` и конфигурации любого (или обоих) *sip.conf* и *xmpp.conf* для направления входящих вызовов в соответствующее расположение в диалплане.

Конфигурация xmpp.conf

Предположим, что вы уже настроили *xmpp.conf* по разделу "Использование XMPP (Jabber) с Asterisk", поэтому мы пропустим начальную конфигурацию. Единственное, что вам нужно сделать, чтобы принимать сообщения через XMPP, это добавить два параметра конфигурации, описанные в таблице 18-4, в созданную вами учетную запись `[asterisk]`.

Table 18-4. Параметры, связанные с внешним обменом сообщениями с через XMPP

Опция	Описание
`sendtodialplan`	Отправлять ли входящие сообщения через XMPP в диалплан. Доступные параметры: `yes` или `no`. По умолчанию `no`.
`context`	Контекст, в котором должны обрабатываться входящие сообщения. Если не задано, то будет использован контекст `default`.

8 Для совершения звонков в определенные пункты назначения может потребоваться приобрести кредиты Google Voice в панели управления.

Установка `sendtodialplan` в `no` (по умолчанию) или в `yes` будет контролировать, будут ли отправляться входящие сообщения этому пользователю XMPP в диалплан. Какой контекст диалплана будет анализировать сообщения, управляется с помощью параметра `context`. Наша конфигурация в *xmpp.conf* будет выглядеть так:

```
[asterisk]
type=client
serverhost=talk.google.com
username=asterisk@shifteight.org
secret=<super_secret_password>
priority=1
port=5222
usetls=yes
usesasl=yes
status=available
statusmessage="Ohai from Asterisk"
timeout=5
sendtodialplan=yes
contexts=IncomingMessages
```

После обновления нашей конфигурации в *xmpp.conf*, вы можете перезагрузить модуль *res_xmpp.so* в Asterisk:

```
$ asterisk -rx "module reload res_xmpp.so"
```

Конфигурация sip.conf

Asterisk имеет начальную реализацию SIP-обмена сообщениями. Функциональность очень проста в том, что она поддерживает только метод SIP-сообщений `MESSAGE`. Сеансовый обмен сообщениями с использованием сеанса протокола MSRP не поддерживается. К сожалению, SIP-клиенты обычно поддерживают только сеансовые сообщения. Поскольку это не является частью Asterisk, обмен сообщениями через SIP наиболее полезен между самими боксами Asterisk.

Включение поддержки обмена сообщениями довольно просто. Существует три опции (описанные в Табл. 18-5), связанные с сообщениями, две из которых уже включены по умолчанию.

Table 18-5. Параметры, связанные с внешними сообщениями в SIP

Опция	Описание
`accept_outofcall_message`	Управляет включением обмена внешними сообщениями. Когда сообщения принимаются - они передаются в диалплан. Доступные настройки - `yes` и `no`. По умолчанию `yes`
`auth_message_requests`	Определяет, должны ли запросы `MESSAGE` проходить аутентификацию. Доступные варианты: `yes` или `no`. По умолчанию `yes`.
`outofcall_message_context`	Определяет, какой контекст должен обрабатывать внешние сообщения. Если параметр не задан, используется контекст, связанный с пиром во время сопоставления. Опция может быть установлена как на уровне `[general]`, так и на уровне пира.

Вместо того, чтобы полагаться на настройки по умолчанию в Asterisk, мы определим все три опции в разделе `[general]` нашего *sip.conf*. Таким образом, мы можем увидеть явно, что мы включили:

```
; Внешние сообщения
accept_outofcall_message=yes
outofcall_message_context=IncomingMessages
auth_message_requests=yes
```



Если вы хотите переопределить настройки по умолчанию, которые мы создали здесь, для определенных пиров, просто добавьте `outofcall_message_context=`*`Какой-тоКонтекст`* в определение пиров SIP.

После настройки перезагрузите драйвер SIP-канала:

```
$ asterisk -rx "sip reload"
```

Конфигурирование диалплана

В этом разделе мы покажем, как настроить дилплан для приема и отправки сообщений по протоколу XMPP.



В то время как отправка и получение сообщений через SIP возможны - это, как правило, не используется. Рассмотрим примеры в этом разделе, чтобы быть трамплином для вашей разработки, если вам нужна эта функциональность.

После того, как мы настроили нашу систему, как описано в "Конфигурация xmpp.conf", нам нужно настроить наш дилплан для обработки входящих сообщений. Во-первых, давайте настроим диалплан так, чтобы мы могли принять сообщение и отобразить его в консоли Asterisk:

```
[IncomingMessages]
exten => s,1,NoOp()
   same => n,Verbose(2,Totally getting a message yo)
   same => n,Verbose(2,Got message from ${MESSAGE(from)} destined for
           ${MESSAGE(to)} with body ${MESSAGE(body)}))
```

Как видите, мы определили наш контекст как `IncomingMessages`. Когда сообщение поступает в Asterisk, оно будет сопоставлено в контексте расширением `s`. В первую очередь мы просто выполняем `NoOp()`. Следующая строка выводит стандартный текст с помощью приложения `Verbose()`. В последней строке мы снова используем приложение `Verbose()` для отображения содержимого *body* сообщения, а также *from* - от кого пришло сообщение и *to* - кому оно было предназначено. Содержимое *body* сообщения и заголовков *from* и *to* отображаются с помощью функции `MESSAGE()`, которая используется как для отображения полученных данных (как мы показали), так и для отправки данных из нашей системы Asterisk.

Чтобы проверить этот пример диалплана, введите следующую команду:

```
$ asterisk -r
*CLI> dialplan reload
*CLI> core set verbose 5
```

Теперь мы отправляем XMPP-сообщение на сервер Asterisk после добавления УАТС в список приятелей. После отправки любого сообщения мы должны увидеть `Verbose()` в консоли. В противном случае проверяем соединение с *xmpp show connections* и удостоверяемся, что сообщения пользователя УАТС перечислены в *xmpp show buddies*.

Чтобы отправить ответ мы используем функцию `MESSAGE()` для определения тела сообщения и приложение `MessageSend()` для запуска ответа, отправляемого на другой конец. Мы просто добавим следующие две строки после тех, что уже были определены в *extensions.conf*:

```
same => n,Set(MESSAGE(body)=Thank you for messaging the Shift-Eight PBX)
same => n,MessageSend(${MESSAGE(from)},XMPP:asterisk)
```

Давайте посмотрим, что мы делаем. Первая строка определяет текст сообщения, которым мы будем отвечать на входящее сообщение. Здесь нельзя задать несколько строк или создать новые строки в теле сообщения. Вторая строка - это то, что передает тело сообщения обратно конечным точкам, от

которых мы получили сообщение. Аргументы в `MessageSend()` указывают кто ответил и какой аккаунт использовать для ответа.

Посмотрим на нашу вторую строку, наш первый аргумент использует функцию `MESSAGE()` с аргументом `from`, что приведет к отправке сообщения в ту же конечную точку, откуда мы получили сообщение. Второй аргумент представляет собой комбинацию протокола для ответа (в данном случае `XMPP`, другим параметром является SIP) и имени учетной записи, как определено в *xmpp.conf* (*asterisk*).

После внесения изменений перезагрузите диалплан с помощью *dialplan reload* и отправьте сообщение УАТС. Теперь вы должны получить ответ с благодарностью за обмен сообщениями Shift-Eight PBX.

Интеграция LDAP

Asterisk поддерживает возможность подключения к существующему серверу LDAP для загрузки информации на сервер Asterisk с помощью Архитектуры Realtime Asterisk (ARA). Преимущество интеграции Asterisk и LDAP сразу станет очевидным, когда вы начнете централизовывать свои механизмы аутентификации на сервере LDAP и использовать его для нескольких приложений: вы значительно сократите административные издержки управления пользователями, разместив всю их информацию в центральном расположении.

Существуют как коммерческие, так и открытые серверы LDAP. Наиболее популярным коммерческим решением, вероятно, является решение, реализованное серверами Microsoft Windows. Популярным сервером LDAP с открытым исходным кодом является OpenLDAP. Мы не будем углубляться в конфигурацию сервера LDAP здесь, но покажем схему, необходимую для подключения Asterisk к серверу и использование его для обеспечения SIP-соединений и службы голосовой почты для существующей пользовательской базы.

Конфигурирование OpenLDAP

Хотя обсуждение установки и конфигурации сервера LDAP выходит за рамки этой главы, оно, безусловно, применимо, чтобы показать вам, как мы расширили нашу первоначальную схему LDAP, чтобы добавить информацию, необходимую для интеграции Asterisk. Наша первоначальная установка следовала инструкциям со страницы документации Ubuntu. Нам нужно было только следовать инструкциям до и включить импорт *backend.example.com.ldif*; следующим шагом после импорта конфигурации серверной части является установка схем, связанных с Asterisk.

Если вы следуете вместе с импортированным бекэндом, перейдите в каталог исходников Asterisk. Затем скопируйте файл *asterisk.ldap-schema* в каталог */etc/ldap/schema*:

```
$ cd ~/src/asterisk-complete/asterisk/11/contrib/scripts/
$ sudo cp asterisk.ldap-schema /etc/ldap/schema/asterisk.schema
```

После копирования файла схемы перезапустите сервер OpenLDAP:

```
$ sudo /etc/init.d/slapd restart
```

Теперь мы готовы импортировать содержимое *asterisk.ldif* на наш сервер OpenLDAP. Файл *asterisk.ldif* находится в папке */contrib/scripts* каталога исходников Asterisk:

```
$ sudo ldapadd -Y EXTERNAL -H ldapi:/// -f asterisk.ldif
```

Теперь мы можем продолжить с онлайн-инструкцией и импортировать файл *frontend.example.com.ldif*. В этом файле находится начальный пользователь, которого мы можем пропустить на данный момент, поскольку собираемся изменить часть импорта пользователя, чтобы включить `objectClass` для Asterisk (т.е. в файле примера можно удалить раздел текста, начинающийся с `uid=john`).

Мы создадим пользователя и добавим значения конфигурации, что позволит пользователю зарегистрировать свой телефон (который, вероятно, софтфон, поскольку hardphone на рабочем столе пользователя, в большинстве случаев, настраивается централизовано) через SIP с использованием его логина и пароля, так же, как он обычно входит чтобы проверить почту и все остальное.

Файл конфигурации, который мы создадим далее, будет импортирован с помощью команды *ldapadd* и добавлен в объект `people` в пределах *shifteight.org*. Обязательно измените значения в соответствии с пользователем, которого вы хотите настроить в LDAP, и замените `dc=shifteight,dc=org` своим местоположением.

Однако, прежде чем мы создадим наш файл, нужно преобразовать пароль в MD5-хэш. Asterisk не будет аутентифицировать телефоны с помощью текстовых паролей при подключении через LDAP. Мы можем преобразовать пароль с помощью команды *md5sum*:

```
$ echo "my_secret_password" | md5sum
a7be810a28ca1fc0668effb4ea982e58  -
```

Мы вставим возвращаемое значение (без дефиса) в следующий файл в поле `userPassword` с префиксом `{md5}`:

```
$ cat > astuser.ldif

dn: uid=rbryant,ou=people,dc=shifteight,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: AsteriskSIPUser
uid: rbryant
sn: Bryant
givenName: Russell
cn: RussellBryant
displayName: Russell Bryant
uidNumber: 1001
gidNumber: 10001
userPassword: {md5}a7be810a28ca1fc0668effb4ea982e58
gecos: Russell Bryant
loginShell: /bin/bash
homeDirectory: /home/russell
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877
mail: russell.bryant@shifteight.org
postalCode: 31000
l: Huntsville
o: shifteight
title: Asterisk User
postalAddress:
initials: RB
AstAccountCallerID: Russell Bryant
AstAccountContext: LocalSets
AstAccountDTMFMode: rfc2833
AstAccountMailbox: 101@shifteight
AstAccountNAT: yes
AstAccountQualify: yes
AstAccountType: friend
AstAccountDisallowedCodec: all
```

```
AstAccountAllowedCodec: ulaw
AstAccountMusicOnHold: default

Ctrl+D
```



Единственное поле, которое мы должны здесь упомянуть - это поле `UserPassword`. Мы требуем, чтобы значение в этом поле было в формате MD5-хэш. В версиях Asterisk до 1.8.0, префикс `{MD5}` перед хэш требуется. Хотя в более новых он больше не требуется, но все еще рекомендуется.

С созданного файла мы можем добавить пользователя на наш сервер LDAP:

```
$ sudo ldapadd -x -D cn=admin,dc=shifteight,dc=org -f astusers.ldif -W
Enter LDAP Password:
adding new entry "uid=rbryant,ou=people,dc=shifteight,dc=org"
```

Теперь наш пользователь импортирован в LDAP. Следующий шаг - необходимо настроить Asterisk для подключения к серверу LDAP и позволить пользователям аутентифицировать и регистрировать свои телефоны.

Компиляция поддержки LDAP в Asterisk

После настройки сервера OpenLDAP и импорта схемы необходимо установить зависимости для Asterisk и скомпилировать модуль `res_config_ldap`. Этот модуль является ключом, который позволит нам настроить Asterisk realtime для доступа к нашим пирам через LDAP.

После установки зависимости, необходимо перезапустить скрипт *./configure* в каталоге исходников Asterisk, затем убедитесь, что выбран модуль `res_config_ldap`. Затем можете запустить *make install* для компиляции и установки нового модуля.

Зависимости Ubuntu

В Ubuntu нам нужно установить пакет *openldap-dev*, чтобы обеспечить зависимость для модуля `res_config_ldap`:

```
$ sudo apt-get install openldap-dev
```

Зависимости RHEL

В RHEL нам нужно установить пакет *openldap-devel*, чтобы обеспечить зависимость для модуля `res_config_ldap`:

```
$ sudo yum install openldap-devel
```

Конфигурирование Asterisk для поддержки LDAP

Теперь, когда мы настроили наш сервер LDAP и установили модуль `res_config_ldap`, нам нужно настроить Asterisk для поддержки загрузки пиров из LDAP. Для этого необходимо настроить файл *res_ldap.conf* для подключения к серверу LDAP и *extconfig.conf* для указания Asterisk, какую информацию следует получить от сервера LDAP и каким образом. Как только это будет сделано, мы можем настроить любые оставшиеся файлы конфигурации модуля, такие как *sip.conf*, *iax.conf*, *voicemail.conf* и так далее, где уместно. В нашем примере мы будем настраивать Asterisk для загрузки наших SIP-пиров из realtime, используя сервер LDAP в качестве нашей базы данных.

Конфигурирование res_ldap.conf

Файл *res_ldap.conf.sample* является хорошим местом для начала, потому что он содержит хороший набор шаблонов. Однако в верхней части файла, в разделе `[_general]`, нам нужно настроить как Asterisk будет подключаться к нашему серверу LDAP. Наш первый вариант - `url`, который определит, как подключиться к серверу. Мы определили соединение как `ldap://172.16.0.103:389`, которое соединится с сервером LDAP по IP-адресу `172.16.0.103` на порт `389`. Если у вас есть безопасное соединение с сервером LDAP, то можете заменить `ldap://` на `ldaps://`. Кроме того, мы установили `protocol=3`, чтобы указать, что мы подключаемся к 3 версии протокола, что в большинстве (если не во всех) случаев будет правильным.

Последние три опции `basedn`, `user` и `pass` используются для аутентификации на нашем сервере LDAP. Нам нужно уточнить:

- `basedn` (`dc=shifteight,dc=org`), которое является, по сути, доменным именем
- Имя `user`, под которым мы собираемся аутентифицироваться на сервере LDAP как (`admin`)
- Пароль пользователя для аутентификации (`canada`)

Если мы сложим все это вместе, мы получим что-то вроде следующего:

```
[_general]
url=ldap://172.16.0.103:389
protocol=3
basedn=dc=shifteight,dc=org
user=cn=admin,dc=shifteight,dc=org
pass=canada
```

Помимо этого, в остальной части примера файла конфигурации мы увидим множество шаблонов, которые можем использовать для сопоставления информации в Asterisk с нашей схемой LDAP. Давайте взглянем на первые строки шаблона `[sip]`, который мы будем использовать для сопоставления информации о наших узлах SIP с базой данных LDAP:

```
[sip]
name = cn
amaflags = AstAccountAMAFlags
callgroup = AstAccountCallGroup
callerid = AstAccountCallerID
...
lastms = AstAccountLastQualifyMilliseconds
useragent = AstAccountUserAgent
additionalFilter=(objectClass=AsteriskSIPUser)
```

С левой стороны у нас есть поле в Asterisk с именем которое будем искать, а справа - сопоставление со схемой LDAP для запроса. Наш первый набор полей сопоставляет поле `name` с полем `cn` на сервере LDAP. Если вы посмотрите на данные, которые мы импортировали в разделе “Конфигурирование OpenLDAP” то увидите, что мы создали пользователя и присвоили значение `RussellBryant` полю `cn`. Таким образом, в нашем случае мы сопоставляем имя аутентификации (поле `name`) пользователя SIP со значением поля `cn` на сервере LDAP (`RussellBryant`).

Это относится и к остальным значениям далее некоторых полей (т.е. `useragent`, `lastms`, `ipaddr`), которые просто должны существовать, поэтому Asterisk может записывать информацию (например, регистрационную) на сервер LDAP.

Конфигурирование extconfig.conf

Наш следующий шаг - рассказать Asterisk, какую информацию загружать в realtime и какую технологию использовать. Используя файл *extconfig.conf* у нас есть возможность загрузки нескольких

модулей динамически (но мы также можем загружать файлы статически). Дополнительные сведения об Asterisk realtime см. В разделе “Использование Realtime” в Главе 16.

В нашем примере мы настроим динамические объекты realtime `sipusers` и `sippeers` для загрузки наших SIP-пиров из LDAP. В следующем примере, у нас есть такой:

```
ldap,"ou=people,dc=shifteight,dc=org",sip
```

Мы указали три аргумента. Во-первых, `ldap` - это технология, которую мы будем использовать для подключения к нашему объекту в realtime. Существуют и другие технологии, такие как `odbc`, `pgsql`, `curl` и так далее. Наш второй аргумент, заключенный в двойные кавычки, указывает, к какой базе данных мы подключаемся. В случае LDAP мы подключаемся к объекту-модулю `people` в домене `shifteight.org`. Наконец, наш третий аргумент - `sip` определяет, какой шаблон мы используем (как определено в *res_ldap.conf*) для сопоставления данных realtime с базой данных LDAP.



Кроме того, можно указать четвертый аргумент, который является приоритетом. При определении нескольких объектов realtime, например при определении `queues` или `sippeers` можно использовать аргумент приоритета для управления отказоустойчивостью, если конкретный механизм хранения становится недоступным. Приоритеты должны начинаться с `1` и увеличиваться последовательно.

Чтобы определить использование `sipusers` и `sippeers` с сервера LDAP, мы включили бы эти строки в *extconfig.conf*:

```
sipusers => ldap,"ou=people,dc=shifteight,dc=org",sip
sippeers => ldap,"ou=people,dc=shifteight,dc=org",sip
```

Конфигурирование sip.conf для realtime

Эти шаги являются необязательными для настройки SIP в realtime, хотя вы, вероятно, ожидаете, что все будет работать так, как мы собираемся описать. В файле *sip.conf* мы включим несколько опций для realtime, которые будут кэшировать информацию в память по мере загрузки из базы данных. Таким образом, мы позволим Asterisk звонить на устройства, просто просматривая информацию, хранящуюся в памяти. Кэширование не только делает realtime потенциально более эффективным, но и такие вещи как обновление состояний устройства, просто не могут работать, если устройства не кэшируются в памяти.



Пир загружается в память только после регистрации устройства или его вызова. При выполнении команды *sip reload* из консоли пиры будут очищены из памяти, поэтому может потребоваться изменить время регистрации, если это будет вызывать проблемы в системе.

Чтобы включить кэширование пиров в Asterisk, используйте параметр `rtcachefriends` в *sip.conf*:

```
rtcachefriends=yes
```

Есть дополнительные опции в realtime, такие как `rtsavesysname`, `rtupdate`, `rtautoclean` и `ignoreregexpire`. Они все объяснены в файле *sip.conf.sample*, расположенном в исходниках Asterisk.

Утилиты Text-to-Speech

Утилиты преобразования текста-в-речь используются для преобразования строк слов в аудио, которое можно воспроизводить вызывающим абонентам. Преобразование текста-в-речь существует уже много лет и постоянно совершенствуется. Хотя мы не можем рекомендовать утилиты преобразования текста-в-речь вместо профессионально записанных приглашений, они могут быть полезны в приложениях, где динамические данные должны передаваться вызывающему абоненту.

Festival

Festival является одним из старейших приложений для преобразования текста-в-речь в Linux. Хотя качество Festival не является достаточным для того, чтобы рекомендовать его для использования в продакшене, это, безусловно, полезный способ тестирования приложения на основе текста-в-речь. Если для вашего применения необходим более отполированный звук, то мы рекомендуем вам посмотреть Cepstral (рассмотренный далее).

Установка Festival в RHEL

Установка Festival и его зависимостей для RHEL проста. Просто используйте *yum* для установки пакета *festival*:

```
$ sudo yum install festival
```

Установка Festival в Ubuntu

Установка Festival и его зависимостей для RHEL проста. Просто используйте *apt-get* для установки пакета *festival*:

```
$ sudo apt-get install festival
```

Использование Festival с Asterisk

С установленным фестивалем нам нужно изменить файл *festival.scm* для подключения Asterisk к серверу Festival. На RHEL и Ubuntu файл находится в */usr/share/festival*. Откройте файл и поместите следующий текст чуть выше последней строки (`provide 'festival'`):

```
(define (tts_textasterisk string mode)
"(tts_textasterisk STRING MODE)
Apply tts to STRING. This function is specifically designed for
use in server mode so a single function call may synthesize the string.
This function name may be added to the server safe functions."[9]
(let ((wholeutt (utt.synth (eval (list 'Utterance 'Text string)))))
(utt.wave.resample wholeutt 8000)
(utt.wave.rescale wholeutt 5)
(utt.send.wave.client wholeutt)))
```

После добавления этого необходимо запустить сервер фестиваля:

```
$ sudo festival_server 2>&1 > /dev/null &
```

С помощью *menuselect* из каталога исходников Asterisk убедитесь, что приложение `app_festival` выбрано в разделе *Applications*. Если оно еще не было выбрано не забудьте запустить *make install* после его выбора, чтобы установить приложение диалплана `Festival()`.

Прежде чем использовать приложение `Festival()`, необходимо сообщить Asterisk, как подключиться к серверу Festival. Файл *festival.conf* используется для управления подключением и взаимодействием Asterisk с сервером Festival. Файл *festival.conf.sample*, расположенный в каталоге с исходным кодом Asterisk, является хорошим местом для начала, поэтому скопируйте *festival.conf.sample* из подкаталога */configs* исходников Asterisk в каталог конфигурации */etc/asterisk*:

```
$ cp ~/asterisk-complete/asterisk/11/configs/festival.conf.sample \
/etc/asterisk/festival.conf
```

9 Применить tts к STRING. Эта функция специально разработана для использования в режиме сервера, поэтому один вызов функции может синтезировать строку. Это имя функции может быть добавлено к функциям безопасности сервера.

Конфигурации по умолчанию обычно достаточно чтобы подключиться к серверу Festival на локальной машине, но при необходимости вы можете настроить параметры, такие как `host`, где располагается сервер Festival (удаленный), `port` для подключения, нужно ли включать кэширование файлов (по умолчанию `no`), расположение директории кэша (по умолчанию */tmp*) и команда Asterisk передачи серверу Festival.

Чтобы убедиться, что приложение `Festival()` доступно, запустите *core show application festival* из консоли Asterisk:

```
*CLI> core show application festival
```

Если выходные данные не выводятся, может потребоваться загрузить модуль *app_festival.so*:

```
*CLI> module load app_festival.so
```

Убедитесь, что модуль *app_festival.so* существует в */usr/lib/asterisk/modules*, если у вас все еще есть проблемы с загрузкой модуля.

После загрузки приложения `Festival()` в Asterisk необходимо создать тестовое расширение диалплана, чтобы убедиться, что `Festival()` работает:

```
[LocalSets]
exten => 203,1,Verbose(2,This is a Festival test)
   same => n,Answer()
   same => n,Playback(silence/1)
   same => n,Festival(Hello World)
   same => n,Hangup()
```

Перезагрузите диалплан с помощью команды *dialplan reload* из консоли Asterisk и протестируйте соединение с Festival, набрав добавочный номер `203`.

Кроме того, если у вас возникли проблемы с сервером Festival, можно использовать следующий метод для создания файлов с помощью приложения *text2wave*, поставляемого с пакетом *festival*:

```
exten => 202,1,Verbose(2,Trying out Festival)
   same => n,Answer()
; *** Эта строка не должна содержать разрывов строк
   same => n,System(echo "This is a test of Festival"
| /usr/bin/text2wave -scale 1.5 -F 8000 -o /tmp/festival.wav)
   same => n,Playback(/tmp/festival)
   same => n,System(rm -f /tmp/festival.wav)
   same => n,Hangup()
```

Теперь вы имеете достаточно, чтобы начать работу с генерацией аудио текста-в-речь для вашей системы Asterisk. Качество звука не блестящее, и генерируемая речь недостаточно ясна, чтобы ее можно было легко понять по телефону, но для целей разработки и тестирования Festival может заполнить пробел, пока вы не будете готовы к более профессиональному звучанию генератора текста-в-речь, такому как Cepstral.

Cepstral

Cepstral - это механизм преобразования текста-в-речь, который работает аналогично приложению `Festival()` в диалплане, но он производит гораздо более качественный звук. Не только качество значительно лучше, Cepstral разработала движок текст-в-речь, который эмулирует голос Элисон, так что ваш текст-в-речь может выдавать те же английские звуковые файлы, которые поставляются с Астериск по умолчанию, чтобы дать совместимые фразы для вызывающего абонента.

Cepstral является коммерческим модулем, но примерно за 30 долларов США вы можете иметь механизм преобразования текста-в-речь, который более ясен, более согласован с другими звуковыми

подсказками в вашей системе и обеспечивает более приятный опыт для ваших абонентов. Инструкции программного обеспечения и установки Cepstral можно загрузить с интернет-магазина Digium.com.

Вывод

В этой главе мы сосредоточились на интеграции Asterisk с внешними службами, которые могут не иметь прямого отношения к созданию или обработке вызовов, но обеспечивают более тесную связь с существующими службами в сети, предоставляя сведения о маршрутизации вызовов или сведения о пользователях из существующей инфраструктуры.

ГЛАВА 19

Fax

Не бойтесь совершенства. Вы никогда не дотянитесь до него.
- Salvador Dali

Понятие факсимильной передачи появилось более 100 лет назад, но не использовалось до 1980-х годов, пока использование факсимильных аппаратов стало необходимым в бизнесе. Это продолжалось в течение двух десятилетий. Затем пришел Интернет и вскоре после этого, факс быстро стал отмирать.

Что такое Fax?

Факсимильный аппарат позволяет факсимилировать (копировать) документ, который будет передан через телефонную линию. В эпоху Интернета такого рода функциональность кажется бесполезной, однако, до распространения Интернета, это была очень полезная вещь. Факсимильные аппараты сканировали документ в цифровой формат, передавали цифровую информацию аналогично тому, как используется аналоговый модем, а затем преобразовывали и распечатывали полученную информацию на другом конце.

Способы обработки факсов в Asterisk

Asterisk предоставляет возможность отправлять и получать факсы, но следует отметить что Asterisk реализует основной транспорт факса. Это означает, что нужно использовать внешние программы и ресурсы для предоставления всех возможностей требуемых вашими пользователями сверх того, что умеет Asterisk.

Asterisk факс может:

- Распознавать входящее факс соединение, и переговорные сесии
- Сохранять (получать) входящие факс как Tagged Image File Format (TIFF) файл
- Принимать TIFF файлы в факс совместимых форматах
- Передавать TIFF файлы для других факс машин

Asterisk факс не может:

- Печатать факсы
- Получать документы для передачи в любых других форматах отличных от TIFF

Получение относительно простое, поскольку формат документа определяется на передающей стороне и, таким образом, всё, что нужно сделать Asterisk, это сохранить документ.

Передача несколько сложнее, так как передающая сторона несет ответственность за то, чтобы документ, который будет отправлен был в правильном формате для отправки по факсу. Как правило, это требует понимания от пользователя как правильно создать отформатированный документ или нужны сложные клиентские или серверные программы для обработки форматирования (например, через драйвер принтера на локальном компьютере) и помещение факса в место, где сервер может забрать и передать его.

spandsp

Первоначально единственным способом обработки факсов в Asterisk был через библиотеку `spandsp`. `spandsp` предоставляет множество Digital Signal Processing (DSP), но в этом контексте все мы заинтересованы в функциональности факса.

Asterisk имеет загвоздку с включением библиотеки `spandsp` в сборку. Из-за несовместимости лицензий, библиотека `spandsp` должна быть загружена и скомпилирована отдельно от Asterisk. Кроме того, поскольку `spandsp` была написана не только для Asterisk, она не будет считать, что установлена для Asterisk. Это означает, что нужно несколько дополнительных шагов, чтобы Asterisk мог использовать `spandsp`.

Получение spandsp

На момент написания, текущая версия `spandsp` - 0.0.6.
Скачать и распаковать исходный код `spandsp` можно так:

```
$ mkdir ~/src/asterisk-complete/thirdparty
$ cd ~/src/asterisk-complete/thirdparty
$ wget http://www.soft-switch.org/downloads/spandsp/spandsp-0.0.6pre21.tgz
$ tar zxvf spandsp-0.0.6pre21.tgz
$ cd spandsp-0.0.6
```

Компиляция и установка spandsp

`spandsp` опирается на библиотеки изображений - вам нужно установить несколько дополнительных пакетов. В Ubuntu это делается относительно легко, потому что вы можете установить один пакет, а установщик позаботиться о зависимостях, а вы отдыхайте:

```
$ sudo apt-get install libtiff-dev
```

Или, на RHEL:

```
$ sudo yum install -y libtiff-devel
```

Пакет `spandsp` должен компилироваться и устанавливаться следующими командами:

```
$ ./configure
$ make
$ sudo make install
```

Это позволит установить библиотеку в каталог */usr/local/lib/*. На многих системах Linux эта папка не выбирается автоматически как путь к библиотеке (*libpath*), так что путь нужно будет добавить вручную.

Добавление библиотеки spandsp в ваш libpath

Для того, чтобы библиотека `spandsp` была видима всеми приложениями в системе, папка, где она находится должна быть добавлена к *libpath* в системе. Обычно это делается путем редактирования файлов в каталоге */etc/ld.so.conf.d/*. Вы просто должны убедиться, что в одном из файлов есть этот каталог */usr/local/lib*. Если нет, то следующая команда создаст подходящий для вас файл:

```
$ sudo cat >> /etc/ld.so.conf.d/usrlocallib.conf
$ /usr/local/lib
```

Нажмите Ctrl+D для сохранения файла, затем запустите команду *ldconfig* для обновления путей библиотеки:

```
$ sudo ldconfig
```

Теперь вы готовы перекомпилировать Asterisk для поддержки `spandsp`.

Перекомпиляция Asterisk с поддержкой spandsp

Поскольку библиотека `spandsp`, вероятно, не была установлена в системе когда Asterisk был впервые собран, то вам нужно быстро перекомпилировать Asterisk, чтобы иметь поддержку `spandsp`:

```
$ cd ~/src/asterisk-complete/asterisk/11/
$ ./configure
$ make menuselect
```

Убедитесь, что под заголовком Resource Modules (модули ресурсов), раздел `spandsp` выглядит следующим образом:

```
[*] res_fax_spandsp
```

Если вы видите во это:

```
XXX res_fax_spandsp
```

Это значит, что Asterisk не смог найти библиотеку `spandsp`.

После того как вы убедились что Asterisk может видеть `spandsp`, вы готовы перекомпилировать его. Сохранитесь, выйдите из *menuselect* и выполните следующие команды:

```
$ make
$ make install
```

Вы можете убедиться, что `spandsp` работает с Asterisk, выполнив следующую команду из командной строки CLI Asterisk:

```
*CLI> module show like res_fax_spandsp.so
```

С этого момента приложения диалплана `SendFAX()` и `ReceiveFAX()` будут доступны для вас.

Отключение spandsp (Если вы хотите проверить Digium Fax)

Библиотека `spandsp` и библиотека Digium факс, обсуждаемая в следующем разделе, являются взаимоисключающими. Если вы хотите попробовать продукт Digium факс, то должны убедиться, что `spandsp` не загружен. Чтобы отключить `spandsp` в Asterisk, просто отредактируйте файл */etc/asterisk/modules.conf* следующим образом:

```
noload => res_fax_spandsp.so
;noload => res_fax_digium.so
```

Сохраните изменения и перезапустите Asterisk.

Digium факс для Asterisk

Digium Fax For Asterisk (FFA) был разработан на основе сильного желания сообщества Asterisk иметь поддерживаемый Digium факс в Asterisk. Бесплатно для одноканального использования, этот продукт также может быть лицензирован от Digium для использования факса с более чем одним каналом.

Получение Digium FFA

Digium FFA может быть получена с веб-сайта Digium. Процесс загрузки, установки и регистрации этой библиотеки тщательно задокументирован в документе *Fax For Asterisk Administrator Manual*,

который можно скачать с сайта Digium. Вам нужно будет зарегистрироваться на сайте компании Digium для того, чтобы получить бесплатный одноканальный лицензионный ключ факса и скачать руководство администратора. Существует также всеобъемлющий файл *README* в комплекте с программным обеспечением, в котором подробно описаны шаги, необходимые для получения Digium FFA и запуска на вашей системе.

Мы также рекомендуем периодически использовать программу `benchfax`, чтобы проверить какой процессор лучше всего на вашей системе. В одном случае, FFA оптимизированный для Core 2 Duo на самом деле показывают лучшие результаты на старых двухъядерных Pentium II системах, что явно неправильно. Повторное тестирование покажет какие ядра дают стабильно лучшие результаты.

Отключение Digium FFA (Если вы хотите проверить spandsp)

Библиотека Digium FFA и библиотека `spandsp` являются взаимоисключающими. Если вы хотите попробовать `spandsp`, то должны убедиться что Digium FFA не загружается. Чтобы отключить FFA, просто отредактируйте файл */etc/asterisk/modules.conf* следующим образом:

```
noload => res_fax_digium.so
;noload => res_fax_spandsp.so (1)
```

(1) Обычно мы помещаем инструкции `noload` в пары, а затем раскомментируем только одну из них, в зависимости от того, какую из них мы хотим проверить. Это позволяет переключаться между ними так же просто, как комментировать одну и раскомментировать другую.

Сохраните изменения и перезапустите Asterisk.

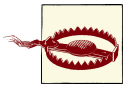
Обработка входящих факсов

Принятые факсы обычно закодированы в формат TIFF (Tagged Image File Format). Этот формат графических файлов хоть и не так известен как JPEG или GIF, но и не так неизвестен, как можно было бы подумать. В самом деле, мы подозреваем что ваш компьютер (не важно Windows, Linux, или MacOS) уже будет иметь встроенную возможность интерпретировать TIFF файлы. Было интересное предложение сделать PDF форматом для передачи факсов главным образом из-за того, что многие программы просмотра изображений ожидают от TIFF только одно изображение, содержащееся в нём. Если вы просматриваете полученное TIFF изображение, ожидая увидеть несколько страниц, и видите только первую страницу, виновником часто является программа просмотра изображений. Часто куда проще перевести ваши TIFF изображения в PDF и просматривать полученное, вместо того, чтобы выяснять, какая программа просмотра изображений позволит вам увидеть несколько страниц факса в TIFF формате.

Принятые факсы будут сохраняться Asterisk в виде файлов. Место хранения этих файлов будет зависеть от нескольких факторов, в том числе:

- Какую программу вы используете для эмуляции факс модема (т.е. IAXmodem, Digium ReceiveFAX и прочие.)
- Расположение в файловой системе, которое вы указали для хранения принятых факсов
- Любая постобработка, настроенная вами, для выполнения с полученными файлами

В диалплане вам необходимо построить логику присвоения имен факсам таким образом, что бы они отличались друг от друга. Есть много переменных, каналов и функций, которые могут быть использованы для этой цели, как например функция `STRFTIME()`. Asterisk может легко обрабатывать захват факса в файл, но вам нужно разобраться что происходит с этим файлом, когда он хранится в системе.



Использования некоторых переменных в имени может быть опасно для вашей системы безопасности, особенно при использовании переменных, которые могут быть установлены удаленно, например `${CALLERID(name)}` и даже `${CALLERID(num)}`. Если вы решили

попробовать использовать интеллектуальные имена, то должны быть осторожны чтобы гарантировать, что некоторые метасимволы оболочки будут либо удалены из названия (что проще) или должным образом экранированы (что более сложно) при работе с оболочкой. Мы рекомендуем при использовании значения `CALLERID()` переносить его с функцией диалплана `FILTER()`, как в `${FILTER(A-Za-z0-9\\x28-\\x2F,${CALLERID(name)})}`, чтобы удалить потенциально опасные символы. Кодирование диапазона `\\x28-\\x2F` разрешает символы типа запятая и круглые скобки без их интерпретации Asterisk.

Факс в TIFF

Подмножество формата TIFF уже в течение долгого времени было де-факто форматом файлов, используемым для факсов. Однако, поскольку просмотрщики изображений регулярно показывают только первую страницу факса, хранящегося в TIFF, прямой просмотр изображения часто приводит к разочарованию (или, что еще хуже, ложному диагнозу, что была получена только первая страница).

После завершения входящего факсимильного вызова полученный файл TIFF можно открыть непосредственно из папки, в которой он был сохранен,или преобразовать в PDF. Вы можете установить полезную утилиту командной строки `tiff2pdf` для выполнения перевода, которая может быть установлена на RHEL с командой:

```
$ sudo yum install libtiff
```

и на Ubuntu:

```
$ sudo apt-get install libtiff-tools
```

После установки программы будет немного сложнее, она выводит изначальный файл в PDF наэкран, если вы не перенаправите вывод в файл. Этот очевидный недостаток утилиты на самом деле является функцией, мы используем её с почтовыми вложениями, которые могут принимать команды в качестве аргумента, пока изображения TIFF хранятся на нашем сервере Asterisk поскольку они меньше, чем полученный PDF. Если это кажется слишком сложным, вы можете просто создать PDF файл:

```
$ tiff2pdf filename.tiff > filename.pdf
```

или, использовать команду для отправки вышеупомянутого вложения без сохранения промежуточных файлов:



Вам нужно будет установить пакет `MIME::Lite` для Perl, чтобы использовать этот скрипт. Для RHEL имя пакета - `perl-MIME-Lite`. Для Ubuntu имя пакета - `libmime-lite-perl`.

```
#!/usr/bin/perl

use MIME::Lite; (1)
my ($filename,$person,$email) = @ARGV;

my $msg = MIME::Lite->new(
    From    => 'fax-server@shifteight.org',
    To      => $email,
    Subject => "Fax received!",
    Type    => 'multipart/mixed'
);
$msg->attach(
    Type    => 'TEXT',
    Data    => "Dear $person,\n\n\tYou have received a fax. " .
               "It is attached.\n\n--The ShiftEight Fax Server\n"
);
$msg->attach(
    Type         => 'application/pdf',
```

```
    Path          => "/usr/bin/tiff2pdf -a \"$cidname\"
                     "/var/spool/asterisk/faxes/$filename |",
    Filename      => "fax.pdf",
    Disposition   => 'attachment'
);

$msg->send;
```

- Использование пакета `MIME :: Lite` выходит за рамки этой книги; однако, поскольку это очень просто иллюстрирует использование утилиты `tiff2pdf`, мы считаем, что оно стоит того, чтобы его включить.

Факс в Email

После того как Asterisk получил факс, полученный файл TIFF нужно передать в конечный пункт назначения: человеку.

Ключевым соображением является то, что если Asterisk не знает достаточно подробно о факсе, невозможно будет узнать предполагаемого получателя не прочитав сам факс (обычно для факса есть титульная страница с информацией получателя, написанной на ней, но даже самые способные программы распознавания текста будут иметь затруднения). Другими словами, если вы не выделите DID каждому пользователю, который может получить факс, Asterisk не сможет сделать больше, чем отправить все факсы на один адрес электронной почты. Вы можете написать обработку этого в диалплане, используя внешнее задание cron или другой дискриптор демона, распространяющий полученные факсы.

Простой диалплан для обработки факса по электронной почте может выглядеть примерно так (вам понадобится приведенный выше скрипт Perl, расположенный в */opt/asteriskbook/bin/sendfax.pl*):

```
exten => fax,1,Verbose(3,Incoming fax)
; где изначально будут храниться входящие факсы
    same => n,Set(FAXDEST=/tmp)

; вставляем метку времени в этот файл для уникальности этого файла
    same => n,Set(tempfax=${STRFTIME(,,%C%y%m%d%H%M)})
    same => n,ReceiveFax(${FAXDEST}/${tempfax}.tif)
    same => n,Verbose(3,- Fax receipt completed with status: ${FAXSTATUS})

; *** Эта строчка не должна содержать разрывов строчек
    same => n,System(/opt/asteriskbook/bin/sendfax.pl ${FAXDEST}/${tempfax}.tif
"Leif Madsen" lmadsen@shifteight.org)
```

Очевидно, что этот пример не подходит для продакшена (например, он не обрабатывает сбойные факсы), однако, этого достаточно, чтобы начать создавать более функциональный обработчик входящих факсов.

Fax в PDF

Популярная просьба отсылать пользователям факс в формате PDF. Настольные компьютеры на PC, Mac и Linux могут напрямую читать файлы TIFF без преобразования, это не является строго обязательным. Тем не менее, многие люди настаивают на этой функциональности.

Обнаружение факса

Вы можете иметь специальный телефонный номер для приема факсов. Однако, с Asterisk это не является обязательным требованием. Asterisk имеет возможность обнаружить, что входящий вызов является факсом и может обрабатывать его по-разному в диалплане. Обнаружение факса доступно

как для DAHDI так и для SIP-каналов. Чтобы включить его для DAHDI, установите параметр `faxdetect` в */etc/asterisk/chan_dahdi.conf*. В большинстве случаев, вы должны установить этот параметр для входящих. В Таблице 19-1 перечислены возможные значения для `faxdetect` в *chan_dahdi.conf*.

Таблица 19-1. Возможные значения параметра faxdetect в chan_dahdi.conf

Значение	Описание
`incoming`	Включает обнаружение факса на входящих вызовах. Когда факс обнаружен, применяется опция `faxbuffers` если она есть и вызов перенаправляется в расширение `fax` в диалплане. Для большей иформации о опции `faxbuffers`, смотри раздел «Использование буфера факса в chan_dahdi.conf».
`outgoing`	Включает обнаружение факса на исходящих вызовах. Если факс обнаружен, опция `faxbuffers` будет применена, канал будет переведен и запущено выполнение диалплана с расширением `fax`.
`both`	Включает определение факса для входящих и исходящих вызовов.
`no`	Отключает определение факса. Установленно по умолчанию.

Чтобы включить обнаружение факса для SIP вызовов, вы должны установить опцию `faxdetect` в */etc/asterisk/sip.conf*. Эта опция может быть установлена в разделе `[general]` или для конкретного пира. Таблица 19-2, описывает возможные значения опции `faxdetect` в *sip.conf*.

Таблица 19-2. Возможные значения faxdetect в sip.conf

Значение	Описание
`cng`	Включает обнаружение факса путем просмотра аудио для тона CNG. Если обнаружен тон CNG, то вызов перенаправляется в расширение `fax` в диалплане.
`t38`	Перенаправляет вызов в расширение `fax` в диалплане если получено приглашение T.38.
`yes`	Включает обнаружение факса на основе обоих - `cng` и `t38` систем.
`no`	Отключает обнаружение факса. Установлено по умолчанию.

Использование T.38

Одна из первых вещей, которую вы заметите при попытке получить факс от поставщика услуг VoIP - это выдающаяся частота сбоев. Как обсуждалось выше, факс особенно разборчив в задержке между конечными точками, и некоторые факсимильные аппараты просто быстро сдадутся, а не попытаются договориться, в то время как другие будут работать нормально. В сценарии на основе аудио факс-через-VoIP мы наблюдали показатель успешности менее 50 процентов при попытке получить факсы, а многие отправляющие факс машины просто не будут работать вообще.

T.38 особенно полезен в этом случае, поскольку протокол предназначен для согласования этой проблемы с задержкой. В самом простом смысле T.38 просто получает факс по частям с одной стороны, передает полученные фрагменты изображения по VoIP, затем повторно передает факс на другом конце. Избегая характерных особенностей сигнала факса, вероятность успеха может быть увеличена до более чем 90 процентов[1].



Мы обнаружили, что получение TIFF файлов, размер которых меньше, чем 2000 байт, как правило, неудачны и их не нужно отправлять на почту получателю.

К счастью, все что нужно сделать для того, чтобы включить сигнализацию T.38; просто включите `t38pt_udptl` в *sip.conf*.

```
t38pt_udptl = yes
```

1 90% может оказаться слишком низким значением, но большинство факсов попытается автоматически продолжить передачу при неудачной отправке; это случается гораздо чаще, чем вы можете себе представить, потому что повторное подключение происходит автоматически. На практике это означает, что успех ближе к 99%.

Это включает передачу факсов по T.38, если удаленная точка поддерживает его, а если нет, SendFax будет пытаться вернуться к передаче, основанной на аудио.

Какой VoIP-провайдер?

Стоит также отметить, что некоторые VoIP провайдеры, включая очень популярных, просто не поддерживают T.38 по какой-то причине. Есть много провайдеров в разных местах, которые не имеют с этим проблем, просто обеспечивающих поддержку T.38. Мы использовали Flowroute в нашем тестировании на производственных серверах, и были очень довольны той легкостью с которой они поддерживают T.38.

Обработка исходящих факсов

Передача факсов из Asterisk является задачей более сложной, чем их получение. Это связано с объемом работ по подготовке факса к передаче. Хотя нет ничего сложного в передаче факсов, но вы должны сделать некоторые проектные решения о таких вещах как:

- Как получить исходный файл факса, отформатированый для Asterisk
- Как получить исходный файл факса в системе Asterisk (указать в каком каталоге Asterisk может получить их)
- Что делать с переданными факсами (сохранять их? удалять их? перемещать их куда-то?)
- Как обрабатывать ошибки передачи

Передача факса из Asterisk

Для передачи факса из Asterisk вы должны иметь файл в формате TIFF. Важно как вы создаете этот TIFF и создание может включать множество шагов. Тем не менее, с точки зрения Asterisk, отправка факсов довольно проста. Вы просто запусткаете приложение `SendFAX()` в диалплане, передав ему путь к существующему файлу TIFF:

```
exten => faxthis,1,NoOp()
    same => n,SendFAX(/путь/к/файлу/факса,dz)
```

На практике, как правило, требуется установить некоторые параметры перед передачей, так что полное расширение для отправки факса с помощью Digium's Fax For Asterisk может выглядеть примерно так:

```
exten => faxthis,1,Verbose(2,Set options and transmit fax)

; некоторая папка, где будут найдены исходящие факсы
    same => n,Set(faxlocation=/tmp)
; В продакшене вы, вероятно, не захотите жестко указывать имя файла
    same => n,Set(faxfile=faxfile.tif)
    same => n,Set(FAXOPT(headerinfo)=Fax from ShiftEight.org)
    same => n,Set(FAXOPT(localstationid=4169671111)
    same => n,SendFax(${faxlocation}/${faxfile},z)
```

Формат файла для отправки факсов

Реальный трюк в отправке факсов - наличие исходного файла в формате, который факс может обработать. На базовом уровне эти файлы известны как файлы TIFF, однако, не все спецификации TIFF совместимы с факсом и не многие из них документированы должным образом. Кроме того, типы форматов TIFF, которые `spandsp` может обрабатывать отличаются от обрабатываемых Digium FFA.

В отсутствие одной четкой спецификации формата файла TIFF для отправки факсов из Asterisk, мы вместо этого задокументируем то, что мы знаем для работы, и предоставим читателю возможность поэкспериментировать, чтобы найти другие способы создания TIFF[2].

Документ Digium «*Fax For Asterisk Administration Manual*» описывает процесс преобразования PDF файлов в формат TIFF с использованием общедоступных инструментов командной строки Linux. Несмотря на то, что этот метод не очень полезен, он должен позволять вам создавать скрипты Linux для обработки преобразования файлов, и ваши пользователи смогут отправлять PDF-файлы в качестве заданий факса.

Вам понадобится интерпретатор PDF *ghostscript*, который можно установить в RHEL с помощью команды:

```
$ sudo yum -y install ghostscript
```

и в Ubuntu коммандой:

```
$ sudo apt-get install ghostscript
```

После установки, *ghostscript* может конвертировать PDF в совместимые с Asterisk TIFF файлы следующей коммандой:

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=tiffg4 -sPAPERSIZE=letter |
> -sOutputFile=<dest> <src>
```

Замените *<dest>* именем исходящего файла и укажите размещение исходного PDF в *<src>*.

Программа *ghostscript* создаст файл TIFF из вашего PDF, который будет передан используя `SendFax()`.

Эксперимент с отправкой в электронной почты по факсу

Многие пользователи хотели бы отправлять электронную почту как факсовый документ. Основная проблема - это обеспечение отправки пользователями документов в формате, подходящем для работы с факсами. В конечном итоге это требует разработки определенного приложения, что выходит за рамки этой книги.

Мы сделали простой пример, который можно рассматривать в качестве отправной точки, для передачи электронной почты через факс.

Одним из первых изменений, которые вы должны сделать - это отредактировать файл альясов */etc/aliases*, который будет перенаправлять входящие факсы на приложение, которое может с ними справиться. Нам неизвестно о каких-либо приложениях, которые могут это сделать, так что вам придется написать его. Изменения в файле */etc/aliases* будут выглядеть примерно так:

```
fax:  "| /путь/к/программе/которая/будет/направлять/входящий/факс/на email"
```

В нашем случае Russell создал небольшой скрипт на Python с именем *fax.py*, поэтому наш файл альясов будет содержать:

```
fax:  "| /asteriskpbx/fax.py"
```

Мы включили копию разработанного скрипта Python для ознакомления в Примере 19-1. Обратите внимание, что этот файл не подходит для продакшена, а лишь служит примером того, как можно отправлять электронную почту по факсу.

---

2 Мы попробовали один формат, была использована программа Microsoft Office Document Image Writer, которая предложила "монохромный TIFF-факс" в качестве выходного формата. Это казалось слишком хорошо, чтобы быть правдой и надежды не оправдались (ни `spandsp` ни Digium FFA не смогли обработать полученный файл). Было бы идеально, найти нечто общее для Windows-ПК, что можно использовать пользователям для "печати" Asterisk совместимых TIFF файлов.



Вам нужно будет включить учетные данные для интерфейса Asterisk Manager (AMI) в Главе 20, прежде чем использовать этот скрипт.

Пример 19-1. Шлюз доказательство концепции email-по-факсу, fax.py

```
#!/usr/bin/env python
"""Poor Man's Email to Fax Gateway.

This is a proof of concept email to fax gateway. There are multiple aspects
that would have to be improved for it to be used in a production environment.

Copyright (C) 2010 - Russell Bryant, Leif Madsen, Jim Van Meggelen Asterisk:
The Definitive Guide
"""


import sys
import os
import email
import base64
import shutil
import socket


AMI_HOST = "localhost"
AMI_PORT = 5038
AMI_USER = "hello"
AMI_PASS = "world"

# This script will pull a TIFF out of an email and save it off to disk to
# allow the SendFax() application in Asterisk to send it. This is the location
# on disk where the TIFF will be stored.

TIFF_LOCATION = "/tmp/loremipsum.tif"


# Read an email from stdin and parse it.
msg = email.message_from_file(sys.stdin)

# For testing purposes, if you wanted to read an email from a file, you could
# do this, instead.
#try:
#    f = open("email.txt", "r")
#    msg = email.message_from_file(f)
#    f.close()
#except IOError:
#    print "Failed to open email input file."
#    sys.exit(1)

# This next part pulls out a TIFF file attachment from the email and saves it
# off to disk in a format that can be used by the SendFax() application. This
# part of the script is incredibly non-flexible. It assumes that the TIFF file
# will be in a specific location in the structure of the message (the second
# part of the payload, after the main body). Further, it assumes that the
# encoding of the TIFF attachment is base64. This was the case for the test
# email that we were using that we generated with mutt. Emails sent by users'
# desktop email clients will vary in _many_ ways. To be used with user-
# generated emails, this section would have to be much more flexible.
```

```
try:
    f2 = open(TIFF_LOCATION, "w")
    f2.write(base64.b64decode(msg.get_payload()[1].get_payload().replace("\n", "")))
    f2.close()

except IOError:
    print "Failed to open file for saving off TIFF attachment."
    sys.exit(1)

# Now that we have a TIFF file to fax, connect to the Asterisk Manager Interface
# to originate a call.
ami_commands = """Action: Login\r
Username: %s\r
Secret: %s\r
\r
Action: Originate\r
Channel: Local/s@sendfax/n\r
Context: receivefax\r
Extension: s\r
Priority: 1\r
SetVar: SUBJECT=%s\r
\r
Action: Logoff\r
\r
""" % (AMI_USER, AMI_PASS, msg['subject'])

print ami_commands

def my_send(s, data):
    """Ensure that we send out the whole data buffer.
    """
    sent = 0
    while sent < len(data):
        res = s.send(data[sent:])
        if res == 0:
            break
        sent = sent + res

def my_recv(s):
    """Read input until there is nothing else to read.
    """
    while True:
        res = s.recv(4096)
        if len(res) == 0:
            break
        print res

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((AMI_HOST, AMI_PORT))
my_send(s, ami_commands)
my_recv(s)
s.shutdown(socket.SHUT_RDWR)
s.close()
```

Мы проверили его на элементарном уровне и доказали основную концепцию. Если вы хотите внедрить email-по-факсу в продакшен, то должны понимать, что придется еще поработать прежде чем у вас будет что-то достаточно надежное, чтобы передать это рядовой группе пользователей.

Сквозной факс

В теории должно быть возможно подключить традиционные факсимильные аппараты к FXS порту, а затем передавать входящие факсы на это устройство (см. Рисунок 19-1). Эта концепция привлекательна по нескольким причинам:

1. Это позволяет вам интегрировать существующую факс-машину в вашу систему Asterisk.
2. Это требует меньше настроек диалплана.

К сожалению, сквозной факс работает не так, как мы хотели бы. Аналоговый сигнал несущей, используемый двумя факсимильными аппаратами для связи, является деликатной вещью, и любое повреждение этого сигнала часто вызывает сбой передачи. В системе Asterisk, выполняющей сквозную передачу, внутренние проблемы синхронизации, в сочетании с ослаблением сигнала, могут создать нестабильную среду для использования факсов, особенно для больших (многостраничных) факсов.

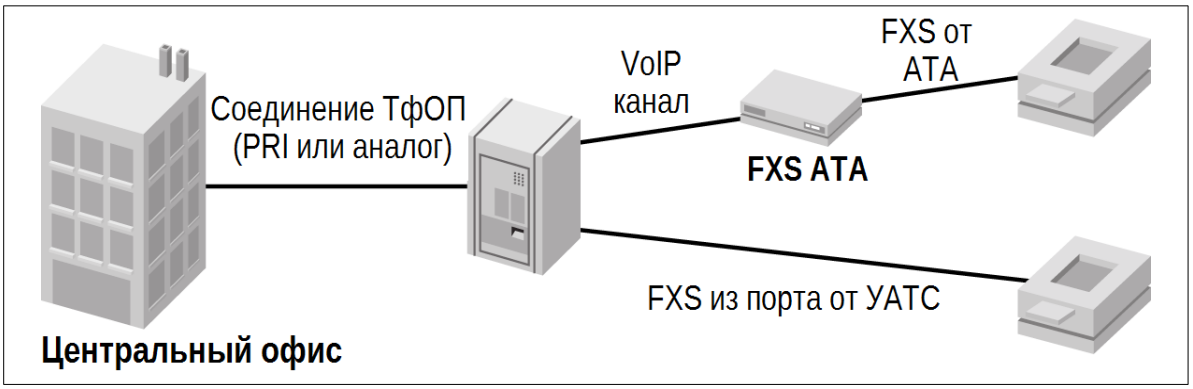


Рисунок 19-1. Сквозной факс

Если вы используете факс на нерегулярной основе (в основном некритические, одностраничные факсы), такая установка может работать хорошо. Если факс имеет решающее значение для вашего бизнеса или вы часто получаете многостраничные факсы, мы неохотно рекомендуем вам подключить факсы непосредственно к PSTN и убрать их из Asterisk.

Использование буфера факса в chan_dahdi.conf

Многие проблемы, связанные со сквозным факсом вызваны несогласованным временем. Поскольку факсы более терпимы к задержкам, чем голосовые вызовы (факс должен быть в состоянии обойти пол мира, что занимает несколько десятков миллисекунд), введение буфера в DAHDI (который будет использован исключительно для факсов), как сообщается, исправляет многие проблемы, которые мешали проходу факса.

В настоящее время предпочтительные настройки следующие:

```
faxbuffers => 12,half
```

Это нужно поместить в ваш файл */etc/asterisk/chan_ dahdi.conf* приведет к созданию *chan_dahdi* буфера для вызовов факса в 96 мс[3] и задержке начала передачи пока буфер не будет полон наполовину.

Вы также должны установить `faxdetect`, так как буфер факса является частью функциональности `faxdetect`:

```
faxdetect = both
```

Мы еще тщательно не тестирования эту возможность, но, по некоторым данным, это должно значительно повысить производительность передачи факсов в Asterisk.

3 96 мс происходит от числа 12 из-за того как работает DAHDI. В DAHDI кусок данных находится во фрагментах по 8 мс, поэтому 12 буферов данных - это 12 буферов * 8 мс = 96 мс буфера.

Шлюз T.38

Другой формой передачи факсов является поддержка шлюза T. 38, которая была добавлена в Asterisk 10. В этом сценарии Asterisk получает факс от конечной точки с поддержкой T. 38 с одной стороны и традиционной конечной точкой факса с другой. Одним из примеров может быть факсимильный аппарат, подключенный к порту FXS на компьютере Asterisk и перенаправление факсимильных вызовов поставщику SIP, поддерживающему T. 38.

Этот пример сценария довольно прост в реализации. Ключом включается поддержка протокола T.38 Gateway с помощью функции диалплана `FAXOPT`. В этом примере предположим, что все исходящие факсимильные вызовы с локального факса отправляются в контекст `outgoing-fax`. Следующий диалплан позволит поддерживать протокол T. 38 Gateway и отправит вызов к поддерживающему T.38 провайдеру SIP :

```
[outgoing-fax]

exten => _1NXXNXXXXXX,1,NoOp()
    same => n,Set(FAXOPT(gateway)=yes)
    same => n,Dial(SIP/t38provider)
```

Кроме этого, вы можете установить тайм-аут на режим шлюза T.38 так, чтобы он выключался, если нет активности факса в пределах указанного тайм-аута. Тайм-аут устанавливается как часть настройки функции `FAXOPT:`

```
; Отключить шлюз T. 38, если нет активности факса в течение 10 секунд
same => n,Set(FAXOPT(gateway)=yes,10)
```

Заключение

Факс - это технология, чьи дни уже позади. Тем не менее, он остается популярным. Asterisk имеет некоторые интересные встроенные технологии, предоставляющие вам некоторый уровень творчества в обработке факсов. Благодаря тщательному планированию и системному проектированию, а также этапу прототипирования и отладки вы можете использовать систему Asterisk для творческой обработки факсов.

Интерфейс управления Asterisk (AMI)

Джон Малкович: Я видел мир, который
НЕ должен видеть человек!
Крейг Шварц: Правда? Потому что для большинства людей
это довольно приятный опыт.
- Будучи Джоном Малковичем

Asterisk Manager Interface (AMI) - это интерфейс системного мониторинга и управления системой, предоставляемый Asterisk. Он позволяет осуществлять живой мониторинг событий, происходящих в системе, а также разрешать запросам к Asterisk выполнять некоторые действия. Доступные действия широко распространены и включают такие вещи, как возврат информации о статусе и появлении новых вызовов. Многие интересные приложения были разработаны поверх Asterisk и используют AMI в качестве основного интерфейса для Asterisk.

Эта глава также включает документацию по использованию файлов вызовов. Файлы вызовов Asterisk - это простой способ инициировать несколько вызовов. Как только объем вызовов увеличивается или ваши потребности становятся более серьезными, вы можете перейти к использованию AMI. Подробности можно найти в разделе “Файлы вызовов”.

Быстрый старт

Этот раздел предназначен для того, чтобы испачкать руки с помощью AMI как можно быстрее. Сначала поместите следующую конфигурацию в */etc/asterisk/manager.conf*:

```
;
; Включите AMI и попросите его принимать соединения только с localhost.
;
[general]
enabled = yes
webenabled = yes
bindaddr = 127.0.0.1
;
; Создайте учетную запись «hello» с паролем «world»
;
[hello]
secret = world
read = all  ; Получать все типы событий
write = all ; Разрешить этому пользователю выполнять все действия.
```



Эта типовая конфигурация настроена так, чтобы разрешать только локальные подключения к AMI. Если вы намерены сделать этот интерфейс доступным по сети, настоятельно рекомендуется использовать его только с помощью TLS. Использование TLS обсуждается более подробно далее в этой главе.

Как только конфигурирация AMI будет готова, включите встроенный HTTP-сервер, поместив следующее содержимое в */etc/asterisk/http.conf*:

```
;
; Включить встроенный HTTP-сервер и слушать только соединения на localhost.
;
[general]
enabled = yes
bindaddr = 127.0.0.1
```

AMI через TCP

Существует несколько способов подключения к AMI, но наиболее распространенным является TCP-сокет. Мы будем использовать *telnet* для демонстрации подключения AMI. В этом примере показаны следующие шаги:

- Подключитесь к AMI через сокет TCP на порт 5038.
- Войдите в систему, используя действие `Login`.
- Выполните действие `Ping`.
- Выйдите из системы, используя действие `Logoff`.

Вот как AMI отвечает на эти действия:

```
$ telnet localhost 5038
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Asterisk Call Manager/1.1
Action: Login
Username: hello
Secret: world

Response: Success
Message: Authentication accepted

Action: Ping

Response: Success
Ping: Pong
Timestamp: 1282739190.454046

Action: Logoff

Response: Goodbye
Message: Thanks for all the fish.

Connection closed by foreign host.
```

После того, как вы это сделаете, вы убедитесь, что AMI принимает соединения через TCP-соединение.

AMI через HTTP

Также возможно использовать AMI по HTTP. В этом разделе мы будем выполнять те же действия, что и раньше, но через HTTP вместо собственного TCP интерфейса к AMI. AMI по HTTP раскрыт более подробно в “AMI через HTTP”.



Учетные записи, используемые для подключения к AMI через HTTP - это те же учетные записи, которые настроены в файле */etc/asterisk/manager.conf*.

В этом примере показано как получить доступ к AMI через HTTP, войти в систему, выполнить действие `Ping` и выйти из системы:

```
$ wget "http://localhost:8088/rawman?action=login&username=hello&secret=world" \
> --save-cookies cookies.txt -O -

--2010-08-31 12:34:23--
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:8088... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55 [text/plain]
Saving to: `STDOUT'

Response: Success
Message: Authentication accepted

2010-08-31 12:34:23 (662 KB/s) - written to stdout [55/55]


$ wget "http://localhost:8088/rawman?action=ping" --load-cookies cookies.txt -O -

--2010-08-31 12:34:23--
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:8088... connected.
HTTP request sent, awaiting response... 200 OK
Length: 63 [text/plain]
Saving to: `STDOUT'

Response: Success
Ping: Pong
Timestamp: 1283258063.040293

2010-08-31 12:34:23 (775 KB/s) - written to stdout [63/63]


$ wget "http://localhost:8088/rawman?action=logoff" --load-cookies cookies.txt -O -

--2010-08-31 12:34:23--
Resolving localhost... 127.0.0.1
Connecting to localhost|127.0.0.1|:8088... connected.
HTTP request sent, awaiting response... 200 OK
Length: 56 [text/plain]
Saving to: `STDOUT'

Response: Goodbye
Message: Thanks for all the fish.

2010-08-31 12:34:23 (696 KB/s) - written to stdout [56/56]
```

Интерфейс HTTP для AMI позволяет интегрировать управление вызовами Asterisk в веб-службу.

Конфигурация

Раздел "Быстрый старт" показал очень простой набор конфигурационных файлов, чтобы вы начали. Однако для AMI доступно еще много вариантов.

manager.conf

Основным конфигурационным файлом для AMI является */etc/asterisk/manager.conf*. Раздел `[general]` содержит параметры (перечисленные в Таблице 20-1), которые контролируют общую работу AMI. Любые другие разделы в файле *manager.conf* будут определять учетные записи для входа в систему и использования AMI.

Таблица 20-1. Опции в разделе manager.conf [general]

Параметр	Значение/Пример	Описание
`enabled`	`yes`	Включает AMI. По умолчанию - `no`.
`webenabled`	`yes`	Позволяет получать доступ к AMI через встроенный HTTP-сервер. По умолчанию - `no`.[a]
`порт`	`5038`	Задает номер прослушиваемого порта для подключения AMI. Значение по умолчанию - `5038`.
`bindaddr`	`127.0.0.1`	Устанавливает прослушиваемый адрес для соединений AMI. По умолчанию используется прослушивание по всем адресам (`0.0.0.0`). Однако настоятельно рекомендуется установить значение `127.0.0.1`.
`tlsenable`	`yes`	Позволяет прослушивать AMI-соединения с использованием TLS. По умолчанию - `no`. За пределами локальной машины настоятельно рекомендуется открывать подключение только через TLS.[b]
`tlsbindport`	`5039`	Устанавливает прослушиваемый порт для TLS-соединений с AMI. Значение по умолчанию - `5039`.
`tlsbindaddr`	`0.0.0.0`	Задает прослушиваемый адрес для AMI на основе TLS. По умолчанию прослушиваются все адреса (`0.0.0.0`)
`tlscertfile`	`/var/lib/asterisk/keys/asterisk.pem`	Устанавливает путь к сертификату сервера для TLS. Это необходимо, если для параметра `tlsenable` установлено значение `yes`.
`tlsprivatekey`	`/var/lib/asterisk/keys/private.pem`	Устанавливает путь к закрытому ключу TLS. Если не указано, будет проверен `tlscertfile` чтобы узнать, содержит ли он также закрытый ключ.
`tlscipher`	`<cipher string>`	Задает список шифров для использования OpenSSL. Установка этого параметра необязательна. Чтобы просмотреть список доступных шифров запустите *openssl ciphers -v* в командной строке.
`allowmultiplelogin`	`no`	Позволяет одной учетной записи одновременно создавать более одного соединения. По умолчанию `yes`.
`displayconnects`	`yes`	Сообщает о подключении к AMI в виде подробных сообщений в консоли Asterisk. Это обычно полезно, но может мешать в системе с использованием сценариев, которые совершают много соединений с AMI. По умолчанию `yes`.
`timestampevents`	`no`	Добавляет временную метку на основе эпохи Unix для каждого события, сообщенного AMI. По умолчанию - `no`.
`brokeneventsaction`	`no`	Восстанавливает ранее нарушенное поведение для действия `Events` AMI, где ответ не будет отправлен при некоторых обстоятельствах. Эта опция существует ради обратной совместимости для приложений, которые работали над ошибкой; она не должна использоваться без крайней необходимости. По умолчанию `no`.
`channelvars`	`VAR1, VAR2, VAR3 [, VAR4 [...]]`	Задает список переменных канала для включения во все события AMI, ориентированные на канал. По умолчанию переменные канала не включаются.
`debug`	`no`	Включает дополнительную отладку в коде AMI. Это в первую очередь касается разработчиков кода C для Asterisk. По умолчанию `no`.
`httptimeout`	`60`	Задает таймаут HTTP в секундах. Эта задержка влияет на пользователей AMI через HTTP: устанавливает `Max - Age` на HTTP куки, задает как долго кэшируются события, чтобы позволить извлекать их через HTTP с помощью действий

	`WaitEvents` и отрезок времени когда HTTP-сервер сохраняет сессии в живых после завершения действия AMI. Значение по умолчанию-`60` секунд.

[a] Чтобы получить доступ к AMI через HTTP, встроенный HTTP-сервер также должен быть настроен в */etc/asterisk/http.conf*.
[b] Пакет разработки OpenSSL должен быть установлен для того, чтобы Asterisk использовал шифрование. На Ubuntu, пакет *libssl-dev*. На RHEL пакет называется *openssl-devel*.

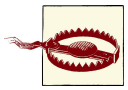
Файл конфигурации *manager.conf* также содержит конфигурацию учетных записей пользователей AMI. Учетная запись создается путем добавления раздела с именем пользователя в квадратных скобках. В каждом разделе [*username*] есть опции, которые можно установить и они будут применяться только к этой учетной записи. В Таблице 20-2 перечислены параметры, доступные в разделе [*username*].

Таблица 20-2. Параметры для разделов [username]

Параметр	Значение/Пример	Описание
`secret`	`password`	Устанавливает пароль, используемый для аутентификации. Он должен быть установлен.
`deny`	`0.0.0.0/0.0.0.0`	Задает список управления доступом IP-адресов (ACL), которым должно быть отказано в аутентификации от этого пользователя. По умолчанию этот параметр не установлен.
`permit`	`192.168.1.0/255.255.255.0`	Задает ACL для IP-адресов, которым разрешена аутентификация пользователя. Как и `deny`, по умолчанию не установлен. Без этих параметров любой IP-адрес, который может подключиться к AMI, сможет аутентифицироваться как этот пользователь.
`writetimeout`	`100`	Устанавливает тайм-аут, используемый Asterisk при записи данных в AMI-соединение для этого пользователя. Эта опция указывается в миллисекундах. Значение по умолчанию равно `100`.
`displayconnects`	`yes`	Также доступен в разделе `[general]` (см. Таблицу 20-1), но может определяться отдельно для каждого пользователя.
`read`	`system, call[, ...]`	Определяет какие события AMI этот пользователь будет получать. По умолчанию пользователь не получает никаких событий. Таблица 20-3 охватывает доступные типы разрешений для параметров `read` и `write`.
`write`	`system, call[, ...]`	Определяет какие события AMI этот пользователь сможет выполнять. По умолчанию пользователь не может выполнять никаких событий. Таблица 20-3 охватывает доступные типы разрешений для параметров `read` и `write`.
`eventfilter`	`!Channel: DAHDI*`	Используется для фильтрации событий AMI в белом или черном списке перед их доставкой в клиентское приложение AMI. Фильтры задаются с помощью регулярного выражения. Указанный фильтр является фильтром белого списка, если ему не предшествует восклицательный знак.[a]

[a] Если фильтры не заданы - будут доставлены все события, разрешенные на основе параметра `read`. Если указаны только фильтры белого списка - будут доставлены только события, соответствующие одному из фильтров. Если есть только фильтры в стиле черного списка, будут доставлены все события, которые не соответствуют ни одному из фильтров. Наконец, если существует сочетание фильтров в стиле белого и черного списка, сначала будут обработаны фильтры белого списка, а затем фильтры черного списка.

Как обсуждалось в Таблице 20-2, параметры `read` и `write` задают какие действия и события AMI доступны определенному пользователю. Таблица 20-3 показывает доступные значения разрешений, которые могут быть указаны для этих параметров.



Обратите особое внимание на разрешения `system`, `command` и `originate`. Эти разрешения предоставляют значительные возможности внешнему приложению. Предоставьте эти разрешения только тем приложениям, над которыми у вас есть полный контроль.

Таблица 20-3. Доступные значения для параметров read/write учетной записи пользователя AMI

Идентификатор разрешения	read	write
`all`	Сокращенный способ указания, что этот пользователь должен иметь доступ ко всем доступным параметрам привилегий.	Предоставляет пользователю все параметры привилегий.
`system`	Позволяет пользователю получать общие сведения о системе, такие как перезагрузка конфигурации.	Позволяет пользователю выполнять команды управления системой Asterisk, такие как *Restart*, *Reload* или *Shutdown*. Это разрешение дает пользователям возможность выполнять команды системы вне Asterisk. Предоставление этого разрешения эквивалентно предоставлению доступа к оболочке в качестве пользователя или группы, от имени которых выполняется процесс Asterisk. Это еще одна веская причина, почему Asterisk не должен запускаться от пользователя *root*.
`call`	Позволяет пользователю получать события о каналах в системе.	Позволяет пользователю устанавливать информацию о каналах.
`log`	Предоставляет пользователю доступ к информации журнала.[a]	Только `read`
`verbose`	Предоставляет пользователю доступ к подробному журналу информации.[b]	Только `read`
`agent`	Предоставляет пользователю доступ к событиям о состоянии агентов из модулей `app_queue` и `chan_agent`.	Позволяет пользователю выполнять действия по управлению и получению состояния очередей и агентов.
`user`	Предоставляет доступ к определяемым пользователем событиям, а также событиям о пользователях Jabber/XMPP.	Позволяет пользователю выполнить действие `UserEvent`, которое позволяет запросить Asterisk создать пользовательское событие.[c]
`config`	Только `write`	Позволяет пользователю извлекать, обновлять и перезагружать файлы конфигурации.
`command`	Только `write`	Позволяет пользователю выполнять команды CLI Asterisk через AMI.
`dtmf`	Позволяет пользователю получать события, генерируемые при прохождении DTMF через ядро Asterisk.[d]	Только `read`
`reporting`	Предоставляет пользователю доступ к событиям качества вызова, таким как статистика jitterbuffer или отчеты протокола управления RTP (RTCP).	Позволяет пользователю выполнить ряд действий для получения статистики и информации о состоянии всей системы.
`cdr`	Предоставляет пользователю доступ к записям CDR, сообщаемым модулем `cdr_manager`.	Только `read`
`dialplan`	Позволяет пользователю получать события, созданные при установке переменных или создании новых расширений.	Только `read`
`originate`	Только `write`	Позволяет пользователю выполнить действие `Originate`, которое позволяет клиенту AMI запросить создание нового вызова Asterisk.
`agi`	Позволяет пользователю получать события, генерируемые при обработке команд AGI.	Позволяет пользователю выполнять действия для управления каналами, на которых выполняется AGI в асинхронном режиме. Более подробно AGI рассматривается в Главе 21.
`cc`	Позволяет пользователю получать события, связанные с дополнительными службами завершения вызова (CCSS).	Только `read`
`aoc`	Позволяет пользователю просматривать события Advice of Charge , создаваемым при получении событий AOC.	Позволяет пользователю выполнять действие диспетчера `AOCMessage` для отправки сообщений AOC.

[a] Этот уровень определен, но в настоящее время не используется нигде в Asterisk.
[b] Этот уровень определен, но в настоящее время не используется нигде в Asterisk.
[c] Действие `UserEvent` является полезным механизмом для доставки сообщений другим клиентам AMI.

[d] События DTMF не будут генерироваться в мостовом вызове между двумя каналами, если не используется универсальное мостовое соединение в ядре Asterisk. Например, если DTMF передается с потоком мультимедиа, а поток мультимедиа течет непосредственно между двумя конечными точками, Asterisk не сможет сообщить о событиях DTMF.

http.conf

Как мы видели, AMI можно получить через HTTP, а также TCP. Для выполнения этой работы в Asterisk встроен очень простой HTTP-сервер. Все параметры, относящиеся к AMI, находятся в разделе `[general]` файла */etc/asterisk/http.conf*.



Для включения доступа к AMI через HTTP требуются оба параметра */etc/asterisk/manager.conf* и */etc/asterisk/http.conf*. AMI должен быть включен в *manager.conf* опцией `enabled` установленной в `yes`, и *manager.conf* опция `webenabled` должна иметь значение `yes`, чтобы разрешить доступ через HTTP. Наконец, опция `enabled` в *http.conf* должна быть установлена в `yes` для включения самого сервера HTTP.

Доступные параметры перечислены в Таблице 20-4.

Таблица 20-4. Параметры в разделе http.conf [general]

Идентификатор разрешения	read	write
`enabled`	`yes`	Включает встроенный HTTP-сервер. По умолчанию `no`.
`bindport`	`8088`	Устанавливает прослушиваемый порт для HTTP-соединений. Значение по умолчанию - `8088`.
`bindaddr`	`127.0.0.1`	Устанавливает адрес для прослушивания HTTP-соединений. По умолчанию используется прослушивание по всем адресам (`0.0.0.0`). Однако настоятельно рекомендуется установить это значение `127.0.0.1`.
`tlsenable`	`yes`	Позволяет прослушивать HTTPS-соединения. По умолчанию - `no`. Настоятельно рекомендуется использовать HTTPS только в том случае, если вы хотите открыть HTTP-соединение вне локальной машины.[a]
`tlsbindport`	`8089`	Устанавливает порт для прослушивания соединений HTTPS. Значение по умолчанию - `8089`
`tlsbindaddr`	`0.0.0.0`	Устанавливает адрес для прослушивания подключений AMI с поддержкой TLS. По умолчанию используется прослушивание по всем адресам (`0.0.0.0`).
`tlscertfile`	/var/lib/asterisk/keys/asterisk.pem	Устанавливает путь к сертификату сервера HTTPS. Это необходимо, если для параметра `tlsenable` установлено значение `yes`.
`tlsprivatekey`	/var/lib/asterisk/keys/private.pem	Устанавливает путь к приватному ключу HTTPS. Если не указано - будет проверен `tlscertfile` чтобы узнать, содержит ли он также приватный ключ.
`tlscipher`	`<cipher string>`	Определяет список шифров OpenSSL для использования. Это необязательно. Чтобы просмотреть список доступных шифров, запустите *openssl ciphers -v* в командной строке.

[a] Чтобы Asterisk мог использовать шифрование - необходимо установить пакет разработки OpenSSL. Для Ubuntu пакет *libssl-dev*. На RHEL пакет называется *openssl-devel*.

Обзор протокола

В AMI есть два основных типа сообщений: диспетчер событий и диспетчер действий.

Диспетчер событий - это односторонние сообщения, отправляемые от Asterisk к клиентам AMI для отчета о происходящем в системе. См. Рисунок 20-1 для графического представления передачи управляющих событий.

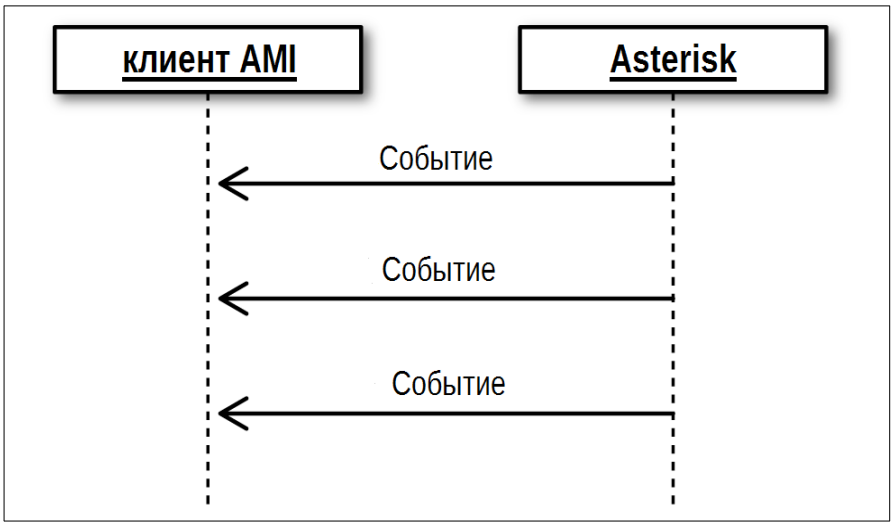


Рисунок 20-1. Диспетчер событий

Диспетчер действий - это запросы от клиента, у которого есть связанные ответы, приходящие обратно от Asterisk. То есть, действие диспетчера может быть запросом к Asterisk на выполнение какого-либо действия и возврат результата. Например, есть действие AMI для инициирования нового вызова. См. Рисунок 20-2. для графического представления клиента, отправляющего действия диспетчеру и получающего ответы.

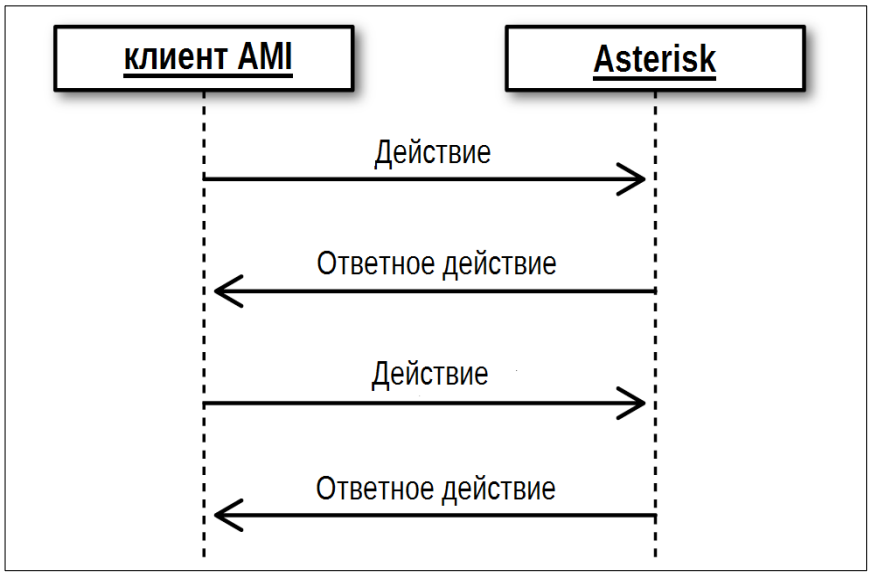


Рисунок 20-2. Диспетчер действий

Другие действия диспетчера - это запросы данных, о которых знает Asterisk. Например, существует действие диспетчера для получения списка всех активных каналов в системе: сведения о каждом канале доставляются как событие диспетчера. Когда список результатов будет завершен, будет отправлено сообщение о том, что конец достигнут. См. Рисунок 20-3 для графического представления клиента, отправляющего этот тип действия диспетчера и получающего список ответов.

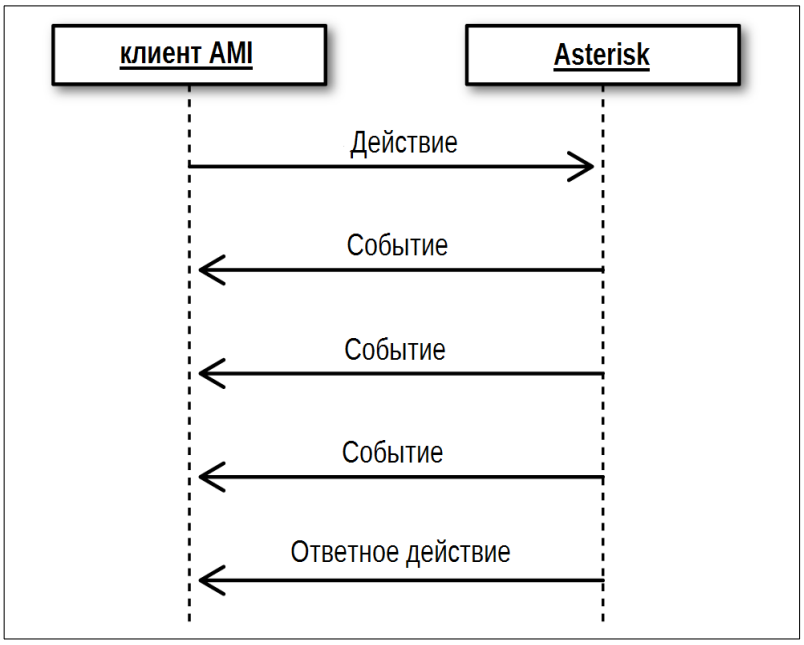


Рисунок 20-3. Действия диспетчера, возвращающие список данных.

Кодировка сообщений

Все сообщения AMI, включая события диспетчера, действия и ответы на действия кодируются одинаково. Сообщения основаны на тексте, а строки заканчиваются возвратом каретки и символом перевода строки. Сообщение заканчивается пустой строкой:

```
Header1: Это первый заголовок<CR><LF>
Header2: Это второй заголовок<CR><LF>
Header3: это последний заголовок этого сообщения<CR><LF>
<CR><LF>
```

События

События диспетчера всегда имеют заголовок `Event` и заголовок `Privilege`. Заголовок `Event` дает имя события, в то время как заголовок `Privilege` перечисляет уровни разрешений, связанные с событием. Любые другие заголовки, включенные в событие, зависят от типа события. Вот пример:

```
Event: Hangup
Privilege: call,all
Channel: SIP/0004F2060EB4-00000000
Uniqueid: 1283174108.0
CallerIDNum: 2565551212
CallerIDName: Russell Bryant
Cause: 16
Cause-txt: Normal Clearing
```

Asterisk 11 представила команды CLI *manager show events* и *manager show event* `<event>`. Выполните эти команды в CLI Asterisk, чтобы получить список событий или узнать подробности конкретного события.



Обратите внимание, что документация диспетчера событий доступна только из CLI Asterisk, если Asterisk был собран с помощью команды *make full*, а не просто *make*.

Действия

При выполнении действия диспетчера он должен включать заголовок `Action`. Заголовок `Action` определяет какой диспетчер действий выполняется. Остальные заголовки являются аргументами для действия и могут потребоваться или нет в зависимости от действия.

Чтобы получить список заголовков, связанных с определенным действием диспетчера, введите в командной строке Asterisk команду *manager show command <Action>*. Чтобы получить полный список действий диспетчера, поддерживаемых используемой версией Asterisk - введите команду *manager show commands* в CLI Asterisk.

Окончательным ответом на действие диспетчера обычно является сообщение, содержащее заголовок `Response`. Значение заголовка `Response` будет иметь значение `Success`, если действие диспетчера было выполнено успешно. Если действие диспетчера не было выполнено успешно, значением заголовка ответа будет `Error`. Например:

```
Action: Login
Username: russell
Secret: russell

Response: Success
Message: Authentication accepted
```

AMI через HTTP

В дополнение к собственному интерфейсу TCP можно также получить доступ к AMI по протоколу HTTP. Программисты, имеющие опыт написания приложений с использованием веб-API, вероятно, предпочтут родное подключение TCP. Хотя интерфейс TCP предлагает только один тип структуры сообщений, AMI через HTTP предлагает несколько вариантов кодирования. Вы можете получать ответы в том же формате, что и интерфейс TCP - в формате XML или в виде базовой HTML-страницы. Тип кодировки выбирается на основе поля в URL-адресе запроса. Параметры кодирования обсуждаются более подробно далее в этом разделе.

Аутентификация и обработка сеанса

Существует два метода выполнения аутентификации для AMI через HTTP. Во-первых, необходимо использовать действие `Login`, аналогичное аутентификации с собственным интерфейсом TCP. Этот метод, который использовался в примере быстрого старта, как замечено в "AMI через HTTP".

После успешной аутентификации Asterisk предоставит файл cookie, который идентифицирует аутентифицированный сеанс. Вот пример ответа на действие `Login`, который включает cookie сеанса от Asterisk:

```
$ curl -v "http://localhost:8088/rawman?action=login&username=hello&secret=world"

* About to connect() to localhost port 8088 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8088 (#0)
> GET /rawman?action=login&username=hello&secret=worlda HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-pc-linux-gnu) libcurl/7.19.7
OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15
> Host: localhost:8088
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Asterisk/SVN-branch-11-r378376
< Date: Tue, 07 Sep 2010 11:51:28 GMT
< Connection: close
< Cache-Control: no-cache, no-store
```

```
< Content-Length: 55
< Content-type: text/plain
< Cache-Control: no-cache;
< Set-Cookie: mansession_id="0e929e60"; Version=1; Max-Age=60
< Pragma: SuppressEvents
<

Response: Success
Message: Authentication accepted
* Closing connection #0
```

Второй вариант аутентификации - это дайджест аутентификации HTTP. В этом примере запрошенным типом кодировки на основе URL-адреса запроса является `rawman`. Чтобы указать, что следует использовать дайджест-проверку подлинности HTTP, префикс типа кодировки в URL-адресе запроса с `a`:

```
$ curl -v --digest -u hello:world http://127.0.0.1:8088/arawman?action=ping
* About to connect() to 127.0.0.1 port 8088 (#0)
*   Trying 127.0.0.1...
* connected
* Connected to 127.0.0.1 (127.0.0.1) port 8088 (#0)
* Server auth using Digest with user 'hello'
> GET /arawman?action=ping HTTP/1.1
> User-Agent: curl/7.27.0
> Host: 127.0.0.1:8088
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Server: Asterisk/SVN-branch-11-r378376
< Date: Thu, 17 Jan 2013 13:19:05 GMT
< Connection: close
< Cache-Control: no-cache, no-store
< Content-Length: 210
< WWW-authenticate: Digest algorithm=MD5, realm="asterisk", nonce="55b30b6d",
<                   qop="auth", opaque="55b30b6d"
< Content-type: text/html
<
* Closing connection #0
* Issue another request to this URL: 'http://127.0.0.1:8088/arawman?
action=ping'
* About to connect() to 127.0.0.1 port 8088 (#0)
*   Trying 127.0.0.1...
* connected
* Connected to 127.0.0.1 (127.0.0.1) port 8088 (#0)
* Server auth using Digest with user 'hello'
> GET /arawman?action=ping HTTP/1.1
> Authorization: Digest username="hello", realm="asterisk", nonce="55b30b6d",
>                uri="/arawman?action=ping", cnonce="MTQ0MTQ5", nc=00000001,
>                qop=auth, response="df8edf75f3571ad425cacb975d09280b",
>                opaque="55b30b6d", algorithm="MD5"
> User-Agent: curl/7.27.0
> Host: 127.0.0.1:8088
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Asterisk/SVN-branch-11-r378376
< Date: Thu, 17 Jan 2013 13:19:05 GMT
< Connection: close
< Cache-Control: no-cache, no-store
```

```
< Content-Length: 63
< Content-type: text/plain
<
Response: Success
Ping: Pong
Timestamp: 1358428745.481800
* Closing connection #0
```

Кодирование /rawman

Тип кодировки `rawman` - это то, что до сих пор использовалось во всех примерах AMI через HTTP в этой главе. Ответы, полученные от запросов с использованием `rawman`, форматируются точно так же, как если бы запросы отправлялись через прямое TCP-подключение к AMI.

Кодирование /manager

Тип кодировки `manager` предоставляет ответ в простой HTML-форме. Этот интерфейс в первую очередь полезен для экспериментов с AMI. Вот пример `Login` с использованием этого типа кодировки:

```
$ curl -v "http://localhost:8088/manager?
> "action=login&username=hello&secret=world"

* About to connect() to localhost port 8088 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8088 (#0)
> GET /manager?action=login&username=hello&secret=world HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-pc-linux-gnu) libcurl/7.19.7
OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15
> Host: localhost:8088
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Asterisk/SVN-branch-11-r378376
< Date: Tue, 07 Sep 2010 12:19:05 GMT
< Connection: close
< Cache-Control: no-cache, no-store
< Content-Length: 881
< Content-type: text/html
< Cache-Control: no-cache;
< Set-Cookie: mansession_id="139deda7"; Version=1; Max-Age=60
< Pragma: SuppressEvents
<

<title>Asterisk&trade; Manager Interface</title><body bgcolor="#ffffff">
<table align=center bgcolor="#f1f1f1" width="500">
<tr><td colspan="2" bgcolor="#f1f1ff"><h1>Manager Tester</h1></td></tr>
<tr><td colspan="2" bgcolor="#f1f1ff"><form action="manager" method="post">
Action: <select name="action">
<option value="">-----&gt;</option>
<option value="login">login</option>
<option value="command">Command</option>
<option value="waitevent">waitevent</option>
<option value="listcommands">listcommands</option>
</select>
or <input name="action"><br/>
CLI Command <input name="command"><br>
user <input name="username"> pass <input type="password" name="secret"><br>
<input type="submit">
```

```
</form>
</td></tr>
<tr><td>Response</td><td>Success</td></tr>
<tr><td>Message</td><td>Authentication accepted</td></tr>
<tr><td colspan="2"><hr></td></tr>
* Closing connection #0
</table></body>
```

Кодирование /mxml

Тип кодирования mxml предоставляет ответы на действия диспетчера, закодированные в XML. Вот пример Login с помощью типа кодирования mxml:

```
$ curl -v "http://localhost:8088/mxml?action=login&username=hello&secret=world"

* About to connect() to localhost port 8088 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8088 (#0)
> GET /mxml?action=login&username=hello&secret=world HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-pc-linux-gnu) libcurl/7.19.7
OpenSSL/0.9.8k zlib/1.2.3.3 libidn/1.15
> Host: localhost:8088
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Asterisk/SVN-branch-11-r378376
< Date: Tue, 07 Sep 2010 12:26:58 GMT
< Connection: close
< Cache-Control: no-cache, no-store
< Content-Length: 146
< Content-type: text/xml
< Cache-Control: no-cache;
< Set-Cookie: mansession_id="536d17a4"; Version=1; Max-Age=60
< Pragma: SuppressEvents
<

<ajax-response>
<response type='object' id='unknown'>
<generic response='Success' message='Authentication accepted' />
</response>
* Closing connection #0
</ajax-response>
```

События диспетчера

При подключении к собственному интерфейсу TCP для AMI события диспетчера доставляются асинхронно. При использовании AMI через HTTP события должны быть получены путем опроса. События загружаются через HTTP, выполнив `WaitEvent` диспетчера действий. В следующем примере показано как события могут быть получены с помощью `WaitEvent` диспетчера действий. Шаги:

1. Запустите сеанс AMI HTTP с помощью действия `Login`.
2. Зарегистрируйте SIP-телефон на Asterisk для создания события диспетчера.
3. Извлеките событие диспетчера с помощью действия `WaitEvent`.

Взаимодействие выглядит так:

```
$ wget --save-cookies cookies.txt \
> "http://localhost:8088/mxml?action=login&username=hello&secret=world" -O -
```

```
<ajax-response>
<response type='object' id='unknown'>
    <generic response='Success' message='Authentication accepted' />
</response>
</ajax-response>


$ wget --load-cookies cookies.txt \
< "http://localhost:8088/mxml?action=waitevent" -O -

<ajax-response>
<response type='object' id='unknown'>
    <generic response='Success' message='Waiting for Event completed.' />
</response>
<response type='object' id='unknown'>
    <generic event='PeerStatus' privilege='system,all'
             channeltype='SIP' peer='SIP/0000FFFF0004'
             peerstatus='Registered' address='172.16.0.160:5060' />
</response>
<response type='object' id='unknown'>
    <generic event='WaitEventComplete' />
</response>
</ajax-response>
```

Файлы вызовов

Перед приведением примеров использования AMI, стоит поговорить о файлах вызовов. Часто AMI рассматривают для исходящих вызовов, однако, во многих ситуациях проще использовать файлы вызовов. Файл вызова - это простой текстовый файл, описывающий вызов, который вы хотите совершить с Asterisk. После создания файла вызова вы перемещаете его в каталог */var/spool/asterisk/outgoing*. Asterisk обнаружит, что файл был помещен туда и будет обрабатывать вызов.

Чтобы использовать файлы вызовов в Asterisk должен быть загружен модуль `pbx_spool`. Выполните следующую команду в CLI Asterisk, чтобы убедиться, что он загружен. Если это не так, проверьте конфигурационный файл модулей */etc/asterisk/modules.conf*:

```
 *CLI> module show like spool
Module                  Description                       Use Count
pbx_spool.so            Outgoing Spool Support            0
1 modules loaded
```

Asterisk поставляется с образцом файла вызова. Его можно найти как *sample.call* в корневом каталоге исходников Asterisk.

У файлов вызовов простой синтаксис. Вы можете размещать комментарии в файле, помещая перед ними символ `#`. Параметры, указанные в файле, представляют собой пары ключ/значение, разделенные двоеточием. Например:

```
# Это комментарий
Option: Value
```

Таблица 20-5 описывает параметры, которые могут быть указаны в файле вызова.

Таблица 20-5. Параметры файла вызова

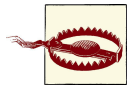
Параметр	Пример значения	Описание
`Channel`	`SIP/myphone`	Этот параметр является критическим и должен быть в каждом файле вызовов. Он описывает исходящий вызов, который будет инициирован. Это значение имеет тот же

Параметр	Пример значения	Описание
		синтаксис, который будет использоваться для аргумента канала приложения `Dial()` в диалплане.
`Context`	`default`	Этот параметр используется для указания местоположения в диалплане, где будет стартовать выполняться исходящий вызов. Параметры `Context`, `Extension` и `Priority` должны использоваться вместе. При использовании этих параметров не следует использовать параметры `Application` и `Data`.
`Extension`	`s`	См. документацию для параметра `Context`.
`Priority`	`1`	См. документацию для параметра `Context`. Если параметры `Context` и `Extension` заданы, а `Priority` - нет, по умолчанию используется значение `1`.
`Application`	`ConfBridge`	Параметры `Application` и `Data` можно использовать вместо параметров `Context`, `Extension` и `Priority`. В этом случае исходящий вызов напрямую подключается к одному приложению после ответа на вызов.
`Data`	`500`	См. документацию для параметра `Application`.
`MaxRetries`	`2`	Если исходящий вызов не отвечает, параметр `MaxRetries` указывает, сколько раз Asterisk повторит вызов перед отказом. Если не указано, значение по умолчанию равно `0`.
`RetryTime`	`60`	Если ответ на исходящий вызов не получен, этот параметр указывает время ожидания в секундах перед повторной попыткой. Значение по умолчанию - `300` секунд (5 минут).
`WaitTime`	`30`	Этот параметр указывает время ожидания ответа в секундах перед отказом от исходящего вызова. Значение по умолчанию - `45` секунд.
`CallerID`	`Jonathan Rose <(555) 555-1212>`	Этот параметр можно использовать для указания CallerID, используемого для исходящего вызова.
`Account`	`someaccount`	Этот параметр устанавливает код аккаунта CDR для исходящего вызова.
`Set`	`VARIABLE=VALUE or FUNCTION(arguments)=VALUE`	Параметр `Set` можно использовать для установки переменных канала или функций канала на исходящем канале. Его можно указать несколько раз в файле вызовов.
`Codecs`	`ulaw,alaw`	Эта опция может использоваться для ограничения разрешенных кодеков при исходящем вызове. Если этот параметр не указан - набор кодеков, настроенных в файле конфигурации драйвера канала, будет сохранен
`AlwaysDelete`	`yes`	По умолчанию файлы вызовов всегда удаляются после их обработки. Если этот параметр имеет значение `no` - Asterisk проверит время изменения файла. Если это время в будущем, файл не будет удален.
`Archive`	`no`	Если этот параметр имеет значение `yes`, то вместо удаления файла вызова он будет перемещен в каталог */var/spool/asterisk/outgoing_done*. Asterisk также добавит строку `Status` в файл вызова, которая будет отражать результат обработки файла вызова. Возможные значения `Status` - `Completed` (завершен), `Expired` (истек (максимальное количество попыток превышено)) или `Failed` (завершен с ошибкой (произошла ошибка)).

Ниже приведен пример создания файла вызова, а затем его размещение в соответствующем каталоге, чтобы Asterisk обработал его:

```
$ cat > demo-congrats.call << EOF
> Channel: SIP/someone@shifteight.org
> Application: Playback
> Data: demo-congrats
> CallerID: Asterisk <(555) 555-1212>
>
> EOF
```

```
$ mv demo-congrats.call /var/spool/asterisk/outgoing/
```



Использование *mv* вместо *cp* здесь важно. Asterisk следит за тем, чтобы содержимое отображалось в каталоге спула. Если вы используете копирование, Asterisk может попытаться прочитать новый файл до того, как содержимое будет полностью скопировано в него. Создание файла, а затем его перемещение позволяет избежать этой проблемы.

Использованием файлов вызовов для совершения исходящих вызовов легко и невероятно полезно. Оно отлично подходит для малого объема вызовов. Если вы хотели бы надежно инициировать много вызовов сразу, AMI является более подходящим. "Инициирование вызова с помощью Python и StarPy" дает пример инициирования вызова с помощью AMI.

Пример использования

Большинство из этой главы до сих пор обсуждали концепции и настройки, связанные с AMI. В этом разделе приведен пример использования.

Инициирование вызова

AMI имеет действие диспетчера `Originate`, которое может использоваться для инициирования вызова. Многие из принятых заголовков совпадают с параметрами, размещенными в файлах вызовов. Таблица 20-6 перечисляет заголовки, принятые действием `Originate`.

Таблица 20-6. Заголовки для действия Originate

Параметр	Пример значения	Описание
`ActionID`	`a3a58876-f7c9-4c28-aa97-50d8166f658d`	Этот заголовок принимается большинством действий AMI. Он используется для предоставления уникального идентификатора, который также будет включен во все ответы на действие. Это позволяет определить, с каким запросом связан ответ. Он важен, так как все действия, их ответы и события передаются по одному соединению (если только не используется AMI через HTTP).
`Channel`	`SIP/myphone`	Этот заголовок является критическим и должен быть указан. Он описывает исходящий вызов, который будет инициирован. Это значение имеет тот же синтаксис, который будет использоваться для аргумента канала приложения `Dial()` в диалплане.
`Context`	`default`	Этот параметр используется для указания местоположения в диалплане, где будет стартовать выполняться ответ на исходящий вызов. Заголовки `Context`, `Extension` и `Priority` должны использоваться вместе. При использовании этих параметров не следует использовать параметры `Application` и `Data`.
`Exten`	`s`	См. документацию для заголовка `Context`.
`Priority`	`1`	См. документацию для заголовка `Context`.
`Application`	`ConfBridge`	Вместо заголовков `Context`, `Exten` и `Priority` можно использовать заголовки `Application` и `Data`. В этом случае исходящий вызов напрямую подключается к одному приложению после ответа на вызов.
`Data`	`500`	См. документацию для заголовка `Application`.
`Timeout`	`30000`	Этот заголовок указывает время ожидания ответа в миллисекундах перед отказом от исходящего вызова. Значение по умолчанию - `30000` миллисекунд (30 секунд).
`CallerID`	`Matthew Jordan <(555) 867-5309>`	Этот заголовок может использоваться для указания CallerID, используемого для исходящего вызова.
`Account`	`someaccount`	Этот заголовок задает код аккаунта CDR для исходящего вызова.
`Variable`	`VARIABLE=VALUE or FUNCTION(arguments)`	Заголовок `Variable` может использоваться для установки переменных канала или функций канала на исходящем канале. Его можно указать

Параметр	Пример значения	Описание
	=VALUE	несколько раз.
Codecs	ulaw,alaw	Эта опция может использоваться для ограничения разрешенных кодеков при исходящем вызове. Если этот параметр не указан, набор кодеков, настроенных в файле конфигурации драйвера канала, по-прежнему будет учитываться.
EarlyMedia	true	Если этот заголовок указан и имеет значение `true`, исходящий вызов будет подключен к указанному добавочному номеру или приложению, как только появится какое-либо предответное состояние (Early Media).
Async	true	Если этот заголовок указан и имеет значение `true`, то вызов будет инициироваться асинхронно. Это позволит продолжить выполнение других действий с AMI-соединением во время обработки вызова.

Простейший пример использования действия `Originate` осуществляется через *telnet*:

```
$ telnet localhost 5038
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Asterisk Call Manager/1.3
Action: Login
Username: hello
Secret: world

Response: Success
Message: Authentication accepted

Action: Originate
Channel: Local/loop@test
Application: Playback
Data: demo-congrats

Response: Success
Message: Originate successfully queued

^]

telnet> quit
Connection closed
```

Перенаправление вызова

Перенаправление (или перевод) вызова из AMI - еще одна функция, заслуживающая упоминания. Действие AMI `Redirect` может использоваться для отправки одного или двух каналов на любой другой внутренний номер в диалплане Asterisk. Если вам нужно перенаправить два канала, которые соединены вместе, сделайте это для них обоих одновременно. В противном случае, как только один канал был перенаправлен, другой будет завершен.

Перенаправление одного канала:

```
Action: Redirect
Channel: SIP/myphone-0011223344
Exten: 1234
Context: default
Priority: 1
```

Для перенаправления двух каналов:

```
Action: Redirect
Channel: SIP/myphone-0011223344
Context: default
Exten: 1234
Priority: 1
ExtraChannel: SIP/otherphone-0011223344
ExtraContext: default
ExtraExten: 5678
ExtraPriority: 1
```

Одним довольно распространенным использованием действия перенаправления от внешнего приложения состоит в том, чтобы перенаправить обе стороны вызова в мост конференции. Поместите этот контекст в диалплан:

```
[confbridge]
exten => _X.,1,ConfBridge(${EXTEN})
```



Информацию о настройке поведения приложения ConfBridge см. в разделе «Расширенные возможности конференц-связи» в Главе 11.

Теперь можно перенаправить любых двух вызывающих абонентов в мост конференции, используя что-то вроде этого:

```
Action: Redirect
Channel: SIP/myphone-0011223344
Context: confbridge
Exten: 1234
Priority: 1
ExtraChannel: SIP/otherphone-0011223344
ExtraContext: confbridge
ExtraExten: 1234
ExtraPriority: 1
```

Теперь другие абоненты могут присоединиться к этой конференции, или внешнее приложение с помощью AMI может продолжить перенаправлять другие каналы сюда же.

Инициирование вызова с использованием Python и StarPy

Чтобы еще больше продемонстрировать использование действия `Originate`, мы предоставили скрипт, который его использует. Полный сценарий приведен в Примере 20-1. Его можно загрузить с https://github.com/russellb/amiutils. Чтобы получить представление об использовании примера приложения, вот вывод справки:

```
$ ./amioriginate.py --help
Usage: amioriginate.py [options] <channel>

Эта программа используется для инициирования вызова на сервере Asterisk с
помощью интерфейса управления Asterisk (AMI). Аргумент канала для этого
приложения сообщает Asterisk, какой исходящий вызов следует выполнить.
Существуют различные параметры, с помощью которых можно указать, к чему будет
подключен исходящий вызов после ответа.

Options:
  -h, --help Показать это сообщение справки и выйти
  -d, --debug Включить вывод отладочной информации
  -u USERNAME, --username=USERNAME
                    Имя пользователя AMI
  -p PASSWORD, --password=PASSWORD
```

```
                    пароль AMI
  -H HOST, --host=HOST Hostname или IP-адрес сервера Asterisk
  -t PORT, --port=PORT Номер порта для AMI
  -a APPLICATION, --application=APPLICATION
                    Приложение для подключения вызова. При использовании
                    этого параметра можно также указать аргументы приложения
                    с параметром-D/--data. Не используйте этот параметр и
                    параметры context, extension, и priority одновременно.
  -D DATA, --data=DATA  Аргументы для передачи приложению диалплана,
указанному
                    с помощью-a/--application.
  -c CONTEXT, --context=CONTEXT
                    Контекст в dialplan для отправки вызова, как только
                    получен ответ. При использовании этого параметра
                    необходимо также указать extension и priority. Не
                    указывайте параметры Application или Data при
                    использовании этого параметра.
  -e EXTEN, --extension=EXTEN
                    Добавочный номер для подключения вызова. Следует
                    использовать вместе с параметрами context и proirity.
  -P PRIORITY, --priority=PRIORITY
                    Приоритет добавочного номера для подключения вызова.
                    Должен использоваться вместе с параметрами context и
                    extension.
```

Если вы запустите утилиту в режиме отладки, то увидите сообщение об обмене сырыми AMI-сообщениями. Во-первых, вот пример использования скрипта для инициирования вызова и подключения его к приложению. `MSG OUT` и `Line In` в выводе определяют обмен AMI. Строки `MSG OUT` указывают приложению действие AMI во внутреннем формате вместо исходного формата AMI, но заголовки и их значения по-прежнему видны. Строки `Line In` показывают только ответ TCP от Asterisk. Вывод показывает:

1. Приложение запрашивает вход в AMI.
2. Asterisk отвечает, принимая запрос аутентификации.
3. Приложение отправляет запрос на инициирование вызова.
4. Asterisk отвечает, указывая, что запрос инициирования успешно обработан. Вызов будет выполняться асинхронно, так как запрос `originate` включает `Async: True`. В противном случае мы не получили бы ответа, пока исходящий вызов или не будет отвечен или не отклонен по какой-либо причине.

```
$ ./amioriginate.py -d -u hello -p world \
> -a playback -D demo-congrats SIP/myphone

DEBUG:AMI:MSG OUT: {'action': 'login', 'username': 'hello', 'secret': 'world',
              'actionid': 'rhel6.3-server-28064728-1'}

DEBUG:AMI:Line In: 'Asterisk Call Manager/1.3'
DEBUG:AMI:Line In: 'Response: Success'
DEBUG:AMI:Line In: 'ActionID: rhel6.3-server-28064728-1'
DEBUG:AMI:Line In: 'Message: Authentication accepted'
DEBUG:AMI:Line In: ''
DEBUG:AMI:MSG OUT: {'application': 'playback',
'actionid': 'rhel6.3-server-28064728-2',
'variable': '', 'async': 'True', 'data': 'demo-congrats',
'action': 'originate', 'channel': 'SIP/myphone'}

DEBUG:AMI:Line In: 'Response: Success'
DEBUG:AMI:Line In: 'ActionID: rhel6.3-server-28064728-2'
DEBUG:AMI:Line In: 'Message: Originate successfully queued'
```

```
DEBUG:AMI:Line In: ''
```



Этот скрипт использует два сторонних модуля Python: `twisted` и `starpy`. Модуль `twisted` широко используется и должен быть доступен в виде дистрибутива. Модуль `starpy` используется не так часто и его нужно установить вручную. См. “Разработка фреймворков”.

Пример 20-1. amioriginate.py

```
#!/usr/bin/env python
#
# Copyright (C) 2012, Russell Bryant
#
# Licensed under the Apache License, Version 2.0 (the "License"); you may
# not use this file except in compliance with the License. You may obtain
# a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
# License for the specific language governing permissions and limitations
# under the License.
# Если иное не требуется применимым законодательством или не согласовано в
# письменной форме, программное обеспечение, распространяемое по лицензии,
# распространяется на условиях "КАК ЕСТЬ", БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ ИЛИ
УСЛОВИЙ,
# явных или подразумеваемых. См. Лицензию для конкретного языка, регулирующего
# разрешения и ограничения в рамках Лицензии.
#
'''Originate вызов на сервере Asterisk

Разработан в качестве примера для "Asterisk: The Definitive Guide"
'''

import getpass
import logging
import optparse
import sys

import starpy.manager
from twisted.internet import reactor

LOG = logging.getLogger(__name__)


class OriginateCall(object):
    def __init__(self, options, channel):
        self.options = options
        self.channel = channel
        self.ami = starpy.manager.AMIFactory(self.options.username,
                                             self.options.password)


    def _on_login(self, ami):
        kwargs = dict(channel=self.channel, async=True)

        if self.options.application:
            kwargs['application'] = self.options.application
```

```
                kwargs['data'] = self.options.data
            else:
                kwargs['context'] = self.options.context
                kwargs['exten'] = self.options.exten
                kwargs['priority'] = self.options.priority

            def _on_success(ami):
                reactor.stop()

            def _on_error(reason):
                LOG.error('Originate failed: %s' % reason.getErrorMessage())
                reactor.stop()

            ami.originate(**kwargs).addCallbacks(_on_success, _on_error)

        def connect_and_call(self):
            def _on_login_error(reason):
                LOG.error('Failed to log in: %s' % reason.getErrorMessage())
                reactor.stop()

            df = self.ami.login(self.options.host, self.options.port)
            df.addCallbacks(self._on_login, _on_login_error)

    def main(argv=None):
        if argv is None:
            argv = sys.argv

        logging.basicConfig(level=logging.INFO)

        description = ('Эта программа используется для инициирования вызова на '
                 'сервере Asterisk с помощью AMI. Аргумент канала для этого'
                 'приложения сообщает Asterisk, какой исходящий вызов следует'
                 'выполнить. Существуют различные параметры, с помощью которых'
                 'можно указать, к чему будет подключен исходящий вызов после'
                 'ответа.')
        usage = '%prog [options] <channel>'
        parser = optparse.OptionParser(description=description, usage=usage)
        parser.add_option('-d', '--debug', action='store_true',
                          dest='debug', help='Enable debug output')
        parser.add_option('-u', '--username', action='store', type='string',
                          dest='username', default=None, help='AMI username')
        parser.add_option('-p', '--password', action='store', type='string',
                          dest='password', default=None, help='AMI password')
        parser.add_option('-H', '--host', action='store', type='string',
                          dest='host', default='localhost',
                          help='Hostname or IP address of the Asterisk server')
        parser.add_option('-t', '--port', action='store', type='int',
                          dest='port', default=5038,
                          help='Port number for the AMI')
        parser.add_option('-a', '--application', action='store', type='string',
                          dest='application', default='',
                          help='Приложение для подключения вызова. При'
                          'использовании' этого параметра можно также указать'
                          'аргументы приложения с параметром-D/--data. Не'
                          'используйте этот параметр и параметры context,'
                          'extension и priority одновременно.')

        parser.add_option('-D', '--data', action='store', type='string',
                          dest='data', default='',
```

```
                    help='Аргументы для передачи приложению диалплана,'
                    'указанному с помощью-a/--application.')

    parser.add_option('-c', '--context', action='store', type='string',
                    dest='context', default='',
                    help='Context in the dialplan to send the call to '
                    'once it answers. If using this option, you '
                    'must also specify an extension and priority. '
                    'Do not specify the application or data options '
                    'if using this option.')
    parser.add_option('-e', '--extension', action='store', type='string',
                    dest='exten', default='',
                    help='Extension to connect the call to. This should '
                    'be used along with the context and priority '
                    'options.')
    parser.add_option('-P', '--priority', action='store', type='string',
                    dest='priority', default='',
                    help='Priority of the extension to connect the call '
                    'to. This should used along with the context '
                    'and extension options.')

    (options, args) = parser.parse_args(argv)

    if options.debug:
        LOG.setLevel(logging.DEBUG)
        starpy.manager.log.setLevel(logging.DEBUG)
    if len(args) != 2 or not len(args[1]):
        LOG.error('Please specify a single outbound channel.')
        parser.print_usage()
        return 1
    channel = args[1]

    valid_application = len(options.application)
    valid_extension = (len(options.context) and len(options.exten) and
                       len(options.priority))

    if not valid_application and not valid_extension:
        LOG.error('Укажите допустимую точку для подключения вызова после ответа'
                  'Укажите приложение или контекст, расширение и приоритет.')
        parser.print_usage()
        return 1

    if valid_application and valid_extension:
        LOG.error('Пожалуйста, укажите только одно расширение или приложение.')
        parser.print_usage()
        return 1

    if not options.username:
        user = raw_input('Username [%s]: ' % getpass.getuser())

    if not user:
        user = getpass.getuser()
        options.username = user

    if not options.password:
        options.password = getpass.getpass()

    o = OriginateCall(options, channel)
    reactor.callWhenRunning(o.connect_and_call)
```

```
    reactor.run()

    return 0

if __name__ == '__main__':
    sys.exit(main())
```

Разработка фреймворков

Многие разработчики приложений пишут код, который напрямую взаимодействует с AMI. Тем не менее, существует ряд библиотек, целью которых является упрощение написания приложений AMI. Таблица 20-7 содержит некоторые списки из которых часть успешно используются. Если вы ищете библиотеки для Asterisk на любом популярном языке программирования по вашему выбору, вы, скорее всего, найдете их.

Таблица 20-7. Фреймворки разработки AMI

Фреймворк	Язык	URL
Adhearsion	Ruby	http://adhearsion.com/
StarPy	Python	https://github.com/asterisk-org/starpy https://github.com/gawel/panoramisk https://github.com/ettoreleandrotognoli/python-ami
Asterisk-Java	Java	http://asterisk-java.org/

CSTA

Телекоммуникационные приложения с компьютерной поддержкой (CSTA) - это стандарт интеграции компьютерной телефонии (CTI), используемый несколькими производителями. Некоторые из тех, что предоставлены CSTA, могут быть сопоставлены с операциями, доступными в AMI. Было нескольких усилий, чтобы обеспечить интерфейс CSTA для Asterisk, в том числе проект Open CSTA. Хотя ни один из авторов не имеет опыта работы с этим интерфейсом CSTA для Asterisk, его, безусловно, стоит рассмотреть, если у вас есть опыт CSTA или существующее приложение CSTA вы хотели бы интегрировать с Asterisk.

Интересные приложения

Было разработано много полезных приложений, которые используют преимущества AMI. Вот один из примеров.

Flash-панель оператора

Flash Operator Panel - это приложение, которое запускается в веб-браузере с помощью Flash. Оно используется в основном в качестве интерфейса, чтобы видеть какие внутренние номера в настоящее время звонят или используются. Оно также включает в себя возможность мониторинга состояния конференц-зала и очередей вызовов. Также могут быть выполнены некоторые действия вызова, такие как врывание (barging) и передача (transfer) вызовов. На Рисунке 20-4 показан интерфейс Flash-панели оператора.

Загрузить и получить более подробную информацию о Flash-панели оператора можно по адресу *http://www.asternic.org* и *http://www.fop2.com*.

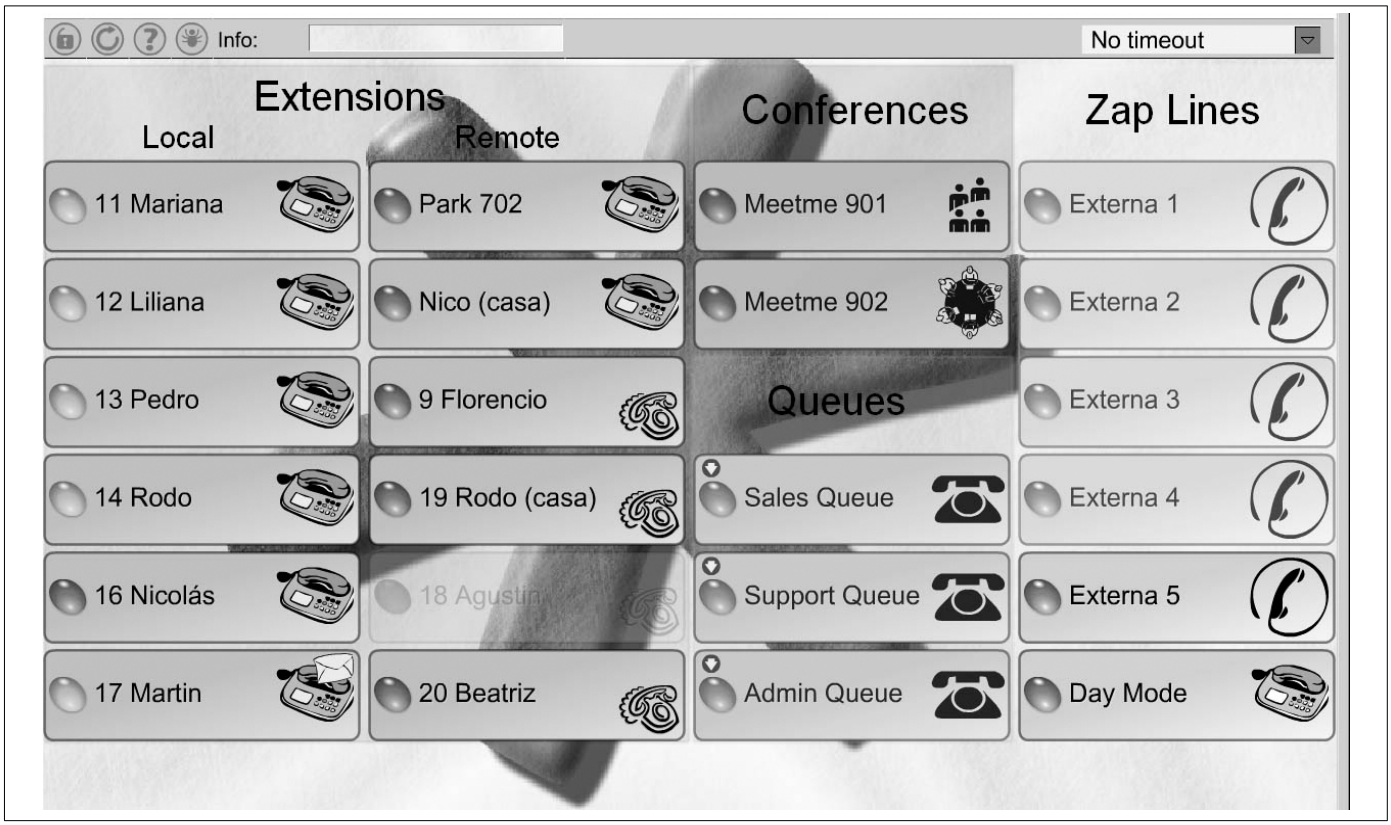


Рисунок 20-4. Flash-панель оператора.

Заключение

Интерфейс управления Asterisk (AMI) предоставляет API для мониторинга событий из системы Asterisk, а также запрашивает у Asterisk выполнение широкого спектра действий. Был представлен интерфейс HTTP и разработан ряд платформ, которые облегчают разработку приложений. Вся эта информация, а также примеры, которые мы рассмотрели в конце этой главы, должны заставить вас задуматься о том, какие новые приложения вы могли бы создать с помощью AMI.

Интерфейс шлюза Asterisk (AGI)

Кофеин. Шлюз наркотиков.
-Eddie Vedder

Asterisk превратилась в простой, но мощный программный интерфейс для обработки вызовов. Тем не менее, многие люди, особенно те, кто имеет малый опыт программирования, по-прежнему предпочитают реализовывать свою обработку пользовательских вызовов на другом языке программирования. Использование другого языка программирования также может позволить вам использовать существующий код для интеграции с другими системами. Интерфейс шлюза Asterisk (AGI) позволяет разрабатывать функции управления сторонними вызовами на языке программирования по вашему выбору. Если вы не заинтересованы в реализации управления вызовами вне диалплана Asterisk, вы можете спокойно пропустить эту главу.

Быстрый старт

В этом разделе приведен краткий пример использования AGI. Сначала добавьте следующую строку в */etc/asterisk/extensions.conf*:

```
exten => 500,1,AGI(hello-world.sh)
```

Затем создайте скрипт *hello-world.sh* в */var/lib/asterisk/agi-bin*, как показано в Примере 21-1

Пример 21-1. Пример сценария AGI, hello-world.sh

```
#!/bin/bash

# Использовать все переменные, отправленные Asterisk
while read VAR && [ -n ${VAR} ] ; do : ; done

# Ответить на вызов.
echo "ANSWER"
read RESPONSE

# Произнести "Hello World"
echo 'SAY ALPHA "Hello World" ""'
read RESPONSE
exit 0
```

Теперь вызовите внутренний номер 500 с включенной отладкой AGI и послушайте, как Эллисон произносит "Hello World”:

```
*CLI> agi set debug on
```

```
AGI Debugging Enabled
   -- Executing [500@phones:1] AGI("SIP/0004F2060EB4-00000009",
      "hello-world.sh") in new stack
   -- Launched AGI Script /var/lib/asterisk/agi-bin/hello-world.sh
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_request: hello-world.sh
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_channel: SIP/0004F2060EB4-00000009
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_language: en
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_type: SIP
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_uniqueid: 1284382003.9
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_version: SVN-branch-11-r378376
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_callerid: 2563619899
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_calleridname: Russell Bryant
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_callingpres: 0
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_callingani2: 0
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_callington: 0
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_callingtns: 0
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_dnid: 7010
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_rdnis: unknown
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_context: phones
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_extension: 500
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_priority: 1
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_enhanced: 0.0
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_accountcode:
<SIP/0004F2060EB4-00000009>AGI Tx >> agi_threadid: 140071216785168
<SIP/0004F2060EB4-00000009>AGI Tx >>
<SIP/0004F2060EB4-00000009>AGI Rx << ANSWER
<SIP/0004F2060EB4-00000009>AGI Tx >> 200 result=0
<SIP/0004F2060EB4-00000009>AGI Rx << SAY ALPHA "Hello World" ""
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/h.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/e.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/l.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/l.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/o.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/space.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/w.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/o.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/r.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/l.gsm' (language 'en')
   -- <SIP/0004F2060EB4-00000009> Playing 'letters/d.gsm' (language 'en')
<SIP/0004F2060EB4-00000009>AGI Tx >> 200 result=0
   --<SIP/0004F2060EB4-00000009>AGI Script hello-world.sh completed, returning
0
```

Варианты AGI

Существует несколько вариантов AGI, которые отличаются в основном тем, что используется для связи с Asterisk. Хорошо знать все варианты, чтобы вы сделать лучший выбор в зависимости от потребностей вашего приложения.

Process-Based AGI

Process-Based AGI (на основе процессов) - это самый простой вариант AGI. Пример быстрого запуска в начале этой главы является примером сценария AGI на основе процесса. Сценарий вызывается с использованием приложения `AGI()` из диалплана Asterisk. Приложение для запуска указано как первый аргумент `AGI()`. Если не указан полный путь, ожидается, что приложение будет находиться в каталоге */var/lib/asterisk/agi-bin*. Аргументы, которые необходимо передать вашему приложению AGI, могут быть указаны в качестве дополнительных аргументов для `AGI()` в диалплане Asterisk. Синтаксис:

```
AGI(command[,arg1[,arg2[,...]]])
```



Убедитесь, что у вашего приложения установлены соответствующие разрешения, чтобы пользователь процесса Asterisk имел разрешения на его выполнение. В противном случае `AGI()` завершится ошибкой.

После того, как Asterisk выполнит ваше приложение AGI, связь между Asterisk и вашим приложением будет проходить через *stdin* и *stdout*. Более подробно об этом сообщении будет освещено в "Обзор связи AGI". Дополнительные сведения о вызове `AGI()` из диалплана см. в документации, встроенной в Asterisk:

```
* CLI> core show application AGI
```

Плюсы AGI на основе процессов

Это простейшая форма AGI для реализации.

Недостатки AGI на основе процессов

Это наименее эффективная форма AGI в отношении потребления ресурсов. Системы с высокой нагрузкой должны учитывать FastAGI, обсуждаемые в "FastAGI-AGI через TCP" вместо этого.

EAGI

EAGI (Enhanced AGI) является небольшим вариантом для `AGI()`. Он вызывается в диалплане Asterisk как `EAGI()`. Разница заключается в том, что в дополнение к сообщениям *stdin* и *stdout* Asterisk также обеспечивает однонаправленный поток аудио, поступающий из канала в файловом дескрипторе 3. Для получения дополнительной информации о том, как вызвать `EAGI()` из диалплана Asterisk, проверьте встроенную документацию в Asterisk:

```
* CLI> core show application EAGI
```

Плюсы Enhanced AGI

Он имеет простоту процесса на основе AGI, с добавлением простого потока только для чтения аудио каналов. Это единственный вариант, который предлагает эту функцию.

Недостатки Enhanced AGI

Поскольку для запуска приложения для каждого вызова должен быть создан новый процесс, он имеет те же проблемы с эффективностью, что и обычный AGI на основе процесса.



Для альтернативного способа получения доступа к аудио вне Asterisk рассмотрите возможность использования JACK. Asterisk имеет модуль интеграции JACK, называемый `app_jack`. Он предоставляет приложение `JACK()` и функцию `JACK_HOOK()`.

DeadAGI Is Dead

В версиях Asterisk до 1.8, было приложение называемое `DeadAGI()`. Его цель была аналогична цели `AGI()`, за исключением того, что вы использовали ее на канале, который уже повесил трубку. Обычно это делается в специальном расширении `h`, когда хотите использовать приложение AGI, чтобы помочь в некоторых типах обработки после вызова. Вызов `DeadAGI()` из диалплана будет по-прежнему работать, но вы получите сообщение `WARNING` в логе Asterisk. Он устарел в пользу использования `AGI()` во всех случаях. Код для `AGI()` был обновлен, поэтому он знает как правильно настроить работу после завершения вызова канала.

Плюсы DeadAGI

Нет. Он мертв.

Недостатки DeadAGI

Он мертв. На самом деле, не используйте его. Если вы это сделаете, ваша конфигурация может сломаться, если `DeadAGI()` полностью удалят из Asterisk в будущей версии.

FastAGI-AGI через TCP

FastAGI - это термин, используемый для управления вызовами AGI через TCP-соединение. С AGI на основе процессов экземпляр приложения AGI выполняется в системе для каждого вызова, а связь с этим приложением выполняется посредством *stdin* и *stdout*. С FastAGI TCP-соединение выполняется на сервере FastAGI. Управление вызовами выполняется с использованием одного и того же протокола AGI, но связь осуществляется по TCP-соединению и не требует запуска нового процесса для каждого вызова. Протокол AGI обсуждается более подробно в "Обзор связи AGI".Использование FastAGI гораздо более масштабируемо, чем AGI на основе процессов, хотя его также сложнее реализовать.

FastAGI используется при вызове приложения `AGI()`, но вместо указания имени исполняемого приложения вы указываете `agi://` URL-адрес. Например:

```
exten => 1234,1,AGI(agi://127.0.0.1)
```

Номер порта по умолчанию для подключения FastAGI - `4573`. Другой номер порта может быть добавлен к URL после двоеточия. Например:

```
exten => 1234,1,AGI(agi://127.0.0.1:4574)
```

Так же как и с AGI на основе процессов, аргументы могут быть переданы в приложение FastAGI. Для этого добавьте их в качестве дополнительных аргументов в приложении `AGI()`, разделенных запятыми:

```
exten => 1234,1,AGI(agi://192.168.1.199,arg1,arg2,arg3)
```

FastAGI также поддерживает использование Service records записи (SRV-записи), если вы указываете URL-адрес в виде `hagi://`. Используя записи SRV, вы можете перечислить несколько хостов, к которым Asterisk может попытаться подключиться для высокой доступности и балансировки нагрузки. В следующем примере, чтобы найти сервер FastAGI для подключения, Asterisk выполнит поиск DNS для `_agi._tcp.shifteight.org`:

```
exten => 1234,1,AGI(hagi://shifteight.org)
```

Плюсы FastAGI

Он более эффективен, чем AGI на основе процессов. Вместо того, чтобы создавать процесс на вызов, сервер FastAGI может обрабатывать множество вызовов.

Может использоваться DNS для обеспечения высокой доступности и балансировки нагрузки между серверами FastAGI для дальнейшего повышения масштабируемости.

Недостатки FastAGI.

Сложнее реализовать сервер FastAGI, чем приложение AGI на основе процессов. Тем не менее, внедрение TCP-сервера было сделано намного раньше, поэтому есть множество примеров, доступных практически для любого языка программирования.

Async AGI-AMI-Контролируемый AGI

Async AGI - это новый метод использования AGI, который был впервые представлен в Asterisk 1.6.0. Цель async AGI - разрешить приложению, использующему AMI для асинхронной очереди команд, выполнение AGI на канале. Это может быть особенно полезно, если вы уже широко используете AMI и хотели бы воспользоваться одним и тем же приложением для обработки управления вызовами, в отличие от написания подробного диалплана Asterisk или разработки отдельного сервера FastAGI.



Более подробную информацию об AMI можно найти в Главе 20.

Async AGI вызывается приложением `AGI()`. Аргументом для `AGI()` должен быть `agi:async`, как показано в следующем примере:

```
exten => 1234,1,AGI(agi:async)
```

Дополнительную информацию о том, как использовать async AGI через AMI, можно найти в следующем разделе.

Плюсы async AGI

Существующее приложение AMI может использоваться для управления вызовами с использованием команд AGI.

Недостатки async AGI

Это самый сложный способ реализации AGI.

Настройка /etc/asterisk/manager.conf для Async AGI

"Конфигурация" в Главе 20 обсуждает параметры конфигурации *manager.conf* в подробностях. Чтобы использовать async AGI учетная запись AMI должна иметь `agi` разрешение `read` и `write`. Например, следующий определяемый пользователь в *manager.conf* будет определять возможность одновременно выполнять действия и получать события AGI диспетчера:

```
;
; Определите пользователя с именем 'hello' и паролем 'world'.
; Дайте этому пользователю разрешения read/write для AGI.
;
[hello]
secret = world
read = agi
write = agi
```

Обзор связи AGI

В предыдущем разделе обсуждались варианты AGI, которые можно использовать. В этом разделе более подробно описано как ваше пользовательское приложение AGI взаимодействует с Asterisk после вызова `AGI()`.

Настройка сеанса AGI

После вызова `AGI()` или `EAGI()` из диалплана Asterisk некоторая информация передается в приложение AGI для настройки сеанса. В этом разделе обсуждается, какие шаги предпринимаются в начале сеанса для разных вариантов AGI.

Process-Based AGI/FastAGI

Для приложения Process-Based AGI или подключения к серверу FastAGI переменные, перечисленные в Таблице 21-1, будут первыми данными, отправленными из Asterisk в ваше приложение. Каждая переменная будет в отдельной строке, в виде:

```
agi_variable: value
```

Таблица 21-1. Переменные среды AGI

Переменная	Значение/Пример	Описание
`agi_request`	`hello-world.sh`	Первый аргумент, переданный приложению `AGI()` или `EAGI()`. Для AGI на основе процесса это имя приложения AGI, которое было выполнено. Для FastAGI это был бы URL, который использовался для достижения сервера FastAGI.

Переменная	Значение/Пример	Описание
agi_channel	SIP/0004F2060EB4-00000009	Имя канала, который выполнил приложение AGI() или EAGI().
agi_language	en	Язык, установленный на agi_channel.
agi_type	SIP	Тип канала для agi_channel.
agi_uniqueid	1284382003.9	Уникальный идентификатор в agi_channel.
agi_version	1.8.0-beta4	Используемая версия Asterisk.
agi_callerid	12565551212	Полная строка CallerID, которая установлена на agi_channel.
agi_calleridname	Russell Bryant	CallerID name, заданное в agi_channel.
agi_callingpres	0	Представление абонента, связанное с CallerID, заданное на agi_channel. Дополнительные сведения см. в выходных данных *core show function CALLERPRES* в CLI Asterisk.
agi_callingani2	0	ANI2 абонента, связанное с agi_channel.
agi_callington	0	CallerID TON (тип номера), связанный с agi_channel.
agi_callingtns	0	Набранный номер TNS (Transit Network Select), связанный с agi_channel.
agi_dnid	7010	Набранный номер, связанный с agi_channel.
agi_rdnis	unknown	Номер перенаправления, связанный с agi_channel.
agi_context	phones	Контекст диалплана, в котором находился agi_channel при выполнении приложения AGI() или EAGI().
agi_extension	500	Внутр.номер в диалплане, который выполнял agi_channel при запуске приложения AGI() или EAGI().
agi_priority	1	Приоритет agi_extension в agi_context, который выполнил AGI() или EAGI().
agi_enhanced	0.0	Указание на то, использовался ли AGI() или EAGI() из диалплана. 0.0 указывает на использование метода AGI(), 1.0 - на EAGI().
agi_accountcode	myaccount	accountcode, связанный с agi_channel.
agi_threadid	140071216785168	threadid потока в Asterisk, в котором выполняется приложение AGI() или EAGI(). Это может быть полезно для связывания журналов, созданных приложением AGI, с журналами, созданными Asterisk, так как журналы Asterisk содержат идентификаторы потоков.
agi_arg_*<argument number>*	my argument	Эти переменные предоставляют содержимое дополнительных аргументов, предоставленных приложению AGI() или EAGI().

Пример переменных, которые могут быть отправлены в приложение AGI, см. в отчете об отладке связи AGI в "Быстрый старт". Конец списка переменных указывается пустой строкой. Пример 21-1 обрабатывает эти переменные, считывая строки ввода в цикле до тех пор, пока не будет получена пустая строка. В этот момент приложение продолжает работу и начинает выполнение команд AGI.

Async AGI

Когда вы используете async AGI, Asterisk отправляет событие диспетчера под названием AsyncAGI, чтобы инициировать сеанс async AGI. Это событие позволит приложениям, слушающим события диспетчера, взять на себя управление вызовом через действие диспетчера AGI. Вот пример события диспетчера, отправленного Asterisk:

```
Event: AsyncAGI
Privilege: agi,all
SubEvent: Start
Channel: SIP/0000FFFF0001-00000000
```

```
Env: agi_request%3A%20async%0Aagi_channel%3A%20SIP%2F0000FFFF0001-00000000%0A \
     agi_language%3A%20en%0Aagi_type%3A%20SIP%0A \
     agi_uniqueid%3A%201285219743.0%0A \
     agi_version%3A%201.8.0-beta5%0Aagi_callerid%3A%2012565551111%0A \
     agi_calleridname%3A%20Julie%20Bryant%0Aagi_callingpres%3A%200%0A \
     agi_callingani2%3A%200%0Aagi_callington%3A%200%0Aagi_callingtns%3A%200%0A \
     agi_dnid%3A%20111%0Aagi_rdnis%3A%20unknown%0Aagi_context%3A%20LocalSets%0A \
     agi_extension%3A%20111%0Aagi_priority%3A%201%0Aagi_enhanced%3A%200.0%0A \
     agi_accountcode%3A%20%0Aagi_threadid%3A%20-1339524208%0A%0A
```



Значение заголовка `Env` в этом событии диспетчера `AsyncAGI` находится на одной строке. Длинное значение заголовка `Env` кодируется в URI.

Команды и ответы

После настройки сеанса AGI Asterisk начинает обработку вызовов в ответ на команды, отправленные из приложения AGI. Как только команда AGI была выдана Asterisk, никакие другие команды не будут обработаны на том канале, пока текущая команда не будет завершена. По завершении обработки команды Asterisk выдаст результат.



AGI обрабатывает команды последовательно. После выполнения команды дальнейшие команды не могут быть выполнены до тех пор, пока Asterisk не вернет ответ. Выполнение некоторых команд может занять очень много времени. Например, команда AGI `EXEC` выполняет приложение Asterisk. Если команда является `EXEC Dial`, связь AGI заблокирована, пока не сделан вызов. Если на данном этапе приложению AGI требуется дальнейшее взаимодействие с Asterisk, оно может сделать это с помощью AMI, описанного в Главе 20.

Полный список доступных команд AGI можно получить из консоли Asterisk, выполнив команду *agi show commands*. Эти команды описаны в Таблице 21-2. Чтобы получить более подробную информацию о конкретной команде AGI, включая синтаксическую информацию для любых аргументов, ожидаемых командой, используйте команду *agi show commands topic <COMMAND>*. Например, чтобы увидеть встроенную документацию для команды AGI `ANSWER`, вы должны использовать команду *agi show commands topic ANSWER*.

Таблица 21-2. Команды AGI

Команда AGI	Описание
`ANSWER`	Ответить на входящий вызов.
`ASYNCAGY BREAK`	Завершить асинхронный сеанс AGI и вернуть канал в диалплан Asterisk.
`CHANNEL STATUS`	Получение статуса канала. Используется для получения текущего состояния канала, такого как up (отвечено), down (отклонено) или ringing.
`DATABASE DEL`	Удалить пару ключ/значение из встроенной базы данных AstDB.
`DATABASE DELTREE`	Удалить дерево пар ключ/значение из встроенной базы данных AstDB.
`DATABASE GET`	Получить значение для ключа в AstDB.
`DATABASE PUT`	Установите значение для ключа в AstDB.
`EXEC`	Выполнить приложение диалплана Asterisk в канале. Эта команда очень мощная в том, что между `EXEC` и `GET FULL VARIABLE` вы можете делать все, что угодно с помощью вызова, который вы можете сделать из диалплана Asterisk.
`GET DATA`	Считывание цифр вызывающего абонента.
`GET FULLVARIABLE`	Вычислить выражение диалплана Asterisk. Можно отправить строку, содержащую переменные и/или функции диалплана, а Asterisk вернет результат после выполнения соответствующих подстановок. Эта команда очень мощная в том, что между `EXEC` и

Команда AGI	Описание
	GET FULL VARIABLE вы можете делать все что угодно с помощью вызова, который вы можете сделать из диалплана Asterisk.
GET OPTION	Воспроизводить звуковой файл во время ожидания цифры от вызывающего абонента. Это похоже на приложение Background() диалплана.
GET VARIABLE	Получение значения переменной канала.
HANGUP	Завершить канал.[a]
NOOP	Ничего не делать. Вы получите в ответе результат выполнения этой команды, как и от любой другой. Её можно использовать как простой тест связи с Asterisk.
RECEIVE CHAR	Получить один символ. Работает только для типов каналов, которые поддерживают его, таких как IAX2 с помощью фреймов TEXT или SIP с помощью метода MESSAGE.
RECEIVE TEXT	Получить текстовое сообщение. Работает в тех же случаях, что и RECEIVE CHAR.
RECORD FILE	Записать звук от вызывающего абонента в файл. Эта блокирующая операция, аналогичная приложению диалплана Record(). Для записи вызова в фоновом режиме, во время выполнения других операций, используйте EXEC Monitor или EXEC MixMonitor.
SAY ALPHA	Произнести строку символов. Вы можете найти пример этого в разделе "Быстрый старт". Чтобы получить локализованную обработку этой и других команд SAY, установите язык канала либо в файле конфигурации устройства (например *sip.conf*), либо в диалплане посредством функции CHANNEL(language).
SAY DIGITS	Произнести строку цифр. Например, 100 будет звучать как "one zero zero", если язык канала установлен на английский.
SAY NUMBER	Произнести число. Например, 100 будет звучать как "one hundred", если язык канала установлен на английский.
SAY PHONETIC	Произнести строку символов, но использовать общее слово для каждой буквы (Альфа, Браво, Чарли...).
SAY DATE	Произнести данную дату.
SAY TIME	Произнести данное время.
SAY DATETIME	Произнести данную дату и время, используя указанный формат.
SEND IMAGE	Отправить изображение в канал. IAX2 это поддерживает, но нет активно развивающихся клиентов IAX2, о которых мы знаем, чтобы это поддерживали.
SEND TEXT	Отправить текст в канал, который его поддерживает. Может использоваться с каналами SIP и IAX2, по крайней мере.
SET AUTOHANGUP	Запланировать завершение канала в определенный момент времени в будущем.
SET CALLERID	Установить имя и номер вызывающего абонента для канала.
SET CONTEXT	Установить текущий контекст диалплана для канала.
SET EXTENSION	Установить текущий внутренний номер диалплана для канала.
SET MUSIC	Запуск или остановка музыки в режиме ожидания для канала.
SET PRIORITY	Установить текущий приоритет диалплана для канала.
SET VARIABLE	Задать для переменной канала данное значение.
STREAM FILE	Направить поток содержимого файла в канал.
CONTROL STREAM FILE	Направить поток содержимого файла в канал, но также позволяет каналу управлять потоком. Например, канал может приостановить, перемотать назад или перемотать поток вперед.
TDD MODE	Переключить режим TDD (телекоммуникационное устройство для глухих) для канала.
VERBOSE	Отправить подробное сообщение в логгер канала. Подробные сообщения отображаются в консоли Asterisk, если параметр verbose достаточно высок. Подробные сообщения также отправляются в любой файл журнала, настроенный на verbose для логгера канала в файле */etc/asterisk/logger.conf*.

Команда AGI	Описание
`WAIT FOR DIGIT`	Ожидать пока абонент нажмет цифру.
`SPEECH CREATE`	Инициализация распознавания речи. Это необходимо сделать перед использованием других речевых команд AGI.[b]
`SPEECH SET`	Настройка речевого движка. Доступные параметры относятся к используемому механизму распознавания речи.
`SPEECH DESTROY`	Уничтожить ресурсы, выделенные для распознавания речи. Эта команда должна быть последней выполненной речевой командой.
`SPEECH LOAD GRAMMAR`	Загрузить grammar.
`SPEECH UNLOAD GRAMMAR`	Выгрузить grammar.
`SPEECH ACTIVATE GRAMMAR`	Активировать загруженный grammar.
`SPEECH DEACTIVATE GRAMMAR`	Деактивировать grammar.
`SPEECH RECOGNIZE`	Воспроизводить подсказку и выполнять распознавание речи, а также ожидать цифры, которые будут нажаты.
`GOSUB`	Выполнить подпрограмму диалплана. Она будет выполняться так же, как и приложение диалплана `GoSub()`.

[a] Когда используется команда AGI `HANGUP`, канал завершается не сразу. Вместо этого канал помечается как требующий завершения. Ваше приложение AGI должно выйти первым, прежде чем Asterisk продолжит и выполнит фактический процесс завершения.

[b] Хотя Asterisk включает основной API для обработки распознавания речи, он не поставляется с модулем, обеспечивающим механизм распознавания речи. В настоящее время Digium предоставляет два коммерческих варианта распознавания речи: Lumenvox и Vestec.

Process-Based AGI/FastAGI

Команды AGI отправляются в Asterisk по одной строке. Строка должна заканчиваться одним символом новой строки. Как только команда была отправлена в Asterisk, дальнейшие команды не будут приниматься до тех пор, пока последняя команда не завершится, и ответ будет отправлен обратно в приложение AGI. Ниже приведен пример ответа на команду AGI:

```
200 result = 0
```



Консоль Asterisk позволяет отлаживать сообщения с помощью приложения AGI. Чтобы включить отладку связи AGI, запустите команду *agi set debug on*. Чтобы отключить отладку, используйте *agi set debug off*. Пока режим отладки включен, вся связь с и из приложения AGI будет выводиться в консоль Asterisk. Пример этого вывода можно найти в "Быстрый старт".

Async AGI

Когда вы используете async AGI, команды выдаются с помощью диспетчера действий AGI. Чтобы увидеть встроенную документацию для действия менеджера AGI, запустите *manager show AGI* в Asterisk CLI. Демонстрация поможет выяснить, как команды AGI выполняются с использованием метода async AGI. Сначала в диалплане создается расширение, которое запускает сеанс async AGI на канале:

```
exten => 7011,1,AGI(agi:async)
```

Когда приложение диалплана AGI выполняется, событие диспетчера, называемое AsyncAGI, будет отправлено во все переменные среды AGI. Подробная информация об этом событии находится в разделе "Async AGI". После этого действия диспетчера AGI могут начинаться через AMI.

Ниже показано примерное выполнение действия и события диспетчера, которые выдаются при обработке async AGI. После первоначального выполнения действия AGI диспетчера существует немедленный ответ, указывающий, что команда была поставлена в очередь на выполнение. Позже

появляется событие диспетчера, которое указывает, что очередь команд была выполнена. Заголовок `CommandID` может быть использован, чтобы связать первоначальный запрос с событием, которое указывает что команда была выполнена:

```
Action: AGI
Channel: SIP/0004F2060EB4-00000013
ActionID: my-action-id
CommandID: my-command-id
Command: VERBOSE "Puppies like cotton candy." 1

Response: Success
ActionID: my-action-id
Message: Added AGI command to queue

Event: AsyncAGI
Privilege: agi,all
SubEvent: Exec
Channel: SIP/0004F2060EB4-00000013
CommandID: my-command-id
Result: 200%20result%3D1%0A
```

Следующий результат - это то, что было видно на консоли Asterisk во время этого сеанса async AGI:

```
    -- Executing [7011@phones:1] AGI("SIP/0004F2060EB4-00000013",
       "agi:async") in new stack
 agi:async: Puppies like cotton candy.
  == Spawn extension (phones, 7011, 1)
exited non-zero on 'SIP/0004F2060EB4-00000013'
```

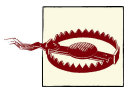
Завершение сеанса AGI

Сессия AGI завершается, когда ваше приложение AGI готово к завершению. Информация о том, как это происходит, зависит от того, использует ли ваше приложение Process-Based AGI, FastAGI или async AGI.

Process-Based AGI/FastAGI

Ваше приложение AGI может выйти или закрыть соединение в любое время. Пока канал не повесил трубку, прежде чем ваше приложение закончится, выполнение диалплана продолжится.

Если происходит завершение канала, пока ваш сеанс AGI по-прежнему активен, Asterisk будет уведомлять об этом, чтобы ваше приложение могло соответствующим образом настроить свою работу.



Это область, в которой изменилось поведение после Asterisk 1.4. В Asterisk 1.4 и более ранних версиях AGI автоматически выходил и останавливал работу, как только канал завершался. Теперь это дает вашему приложению возможность продолжить работу при необходимости.

Если канал завершается, пока приложение AGI все еще выполняется, произойдет несколько вещей. Если команда AGI находится в середине выполнения, вы можете получить код результата `-1`. Вы не должны зависеть от этого, так как не все команды AGI требуют взаимодействия с каналом. Если выполняемая команда не требует взаимодействия с каналом, результат не будет отражать завершение канала.

Следующее что происходит после завершения канала заключается в том, что явное уведомление о завершении отправляется в ваше приложение. Для Process-Based AGI сигнал `SIGHUP` будет отправлен процессу для уведомления об отключении. Для подключения FastAGI Asterisk отправит строку, содержащую слово `HANGUP`. Если вы хотите отключить отправку Asterisk сигнала `SIGHUP` к

приложению Process-Based AGI или строку HANGUP в ваш FastAGI сервер, вы можете сделать это, установив переменную канала AGISIGHUP как показано в следующем коротком примере:

```
;
; Не отправлять SIGHUP в процесс AGI
; или строку "HANGUP" на сервер FastAGI.
;
exten => 500,1,Set(AGISIGHUP=no)
    same => n,AGI(my-agi-application)
```

В этот момент Asterisk автоматически настраивает свою работу в режиме DeadAGI. Это означает, что приложение AGI может работать на канале, который был завершен. Единственными командами AGI, которые могут быть использованы на этом этапе, являются те, которые не требуют взаимодействия с каналом. Документация для команд AGI, встроенных в Asterisk, включает указание того, можно ли использовать каждую команду после того, как канал был завершен.

Async AGI

При использовании async AGI, интерфейс диспетчера предоставляет механизмы для уведомления о завершении каналов. Если вы хотите завершить сеанс async AGI для канала, то должны выполнить команду ASYNCAGI BREAK. Когда сеанс async AGI завершается, Asterisk отправит событие диспетчера AsyncAGI с SubEvent об End. Ниже приведен пример завершения сеанса async AGI:

```
Action: AGI
Channel: SIP/0004F2060EB4-0000001b
ActionID: my-action-id
CommandID: my-command-id
Command: ASYNCAGI BREAK

Response: Success
ActionID: my-action-id
Message: Added AGI command to queue

Event: AsyncAGI
Privilege: agi,all
SubEvent: End
Channel: SIP/0004F2060EB4-0000001b
```

В этот момент канал возвращается к диалплану Asterisk, если он еще не был завершен.

Пример: База данных учетных записей доступа

Пример 21-2 является примером сценария AGI. Чтобы запустить этот скрипт, вы должны сначала разместить его в каталоге */var/lib/asterisk/agi-bin*. Затем выполните его из диалплана Asterisk следующим образом:

```
exten => agiexample1,1,AGI(agiexample1.py)
```

Этот пример написан на Python и задокументирован повсеместно. Он показывает, как скрипт может предложить вызывающему абоненту ввести номер учетной записи, а затем воспроизвести некоторую информацию из этой учетной записи. Что еще более важно, он показывает, как скрипт AGI взаимодействует с Asterisk для запуска команд AGI и получения ответа от Asterisk.

Пример 21-2. agiexample1.py

```
#!/usr/bin/env python
#
# Пример для AGI (Asterisk Gateway Interface).
#
```

```
# Ответить на звонок. Получить номер аккаунта от вызывающего абонента и
# воспроизвести некоторую информацию об учетной записи вызывающего абонента.
#
# Разработано для "Asterisk: The Definitive Guide"
#
import sys
#
# Команды AGI записываются в stdout. После обработки команды
# ответ от Asterisk поступает в stdin.
#
def agi_command(cmd):
    '''Write out the command and return the response'''
    # печатает отправку команды в stdout.
    print cmd
    # Убедитесь, что он не буферизуется.
    sys.stdout.flush()
    # Прочитать строку ответа из stdin. Используйте strip(), чтобы удалить
    # любые пробелы, которые могут присутствовать (в первую очередь конец
    # строки).
    return sys.stdin.readline().strip()
#
# Прочитать переменные среды AGI, отправленные из Asterisk в скрипт при запуске.
# Ввод выглядит:
#     agi_request: example1.py
#     agi_channel: SIP/000FF2266EE4-00000009
# ...
#
# После этого куска кода Вы сможете получить эту информацию как:
#     asterisk_env['agi_request']
#     asterisk_env['agi_channel']
#
asterisk_env = {}
while True:
    line = sys.stdin.readline().strip()
    if not len(line):
        break
    var_name, var_value = line.split(':', 1)
    asterisk_env[var_name] = var_value
#
# Настроить поддельную "базу данных" учетных записей. В более реалистичном
# скрипте AGI, который выполняет поиск данных учетной записи, вы подключитесь к
# внешней базе данных или другому API для получения этой информации. Для целей
# данного примера мы пропустим это и сделаем поиск данных из этого локального
# набора примеров.
#
ACCOUNTS = {
    '12345678': {'balance': '50'},
    '11223344': {'balance': '10'},
    '87654321': {'balance': '100'},
}

#
# После вызова используем команду AGI - ANSWER.
#
# Смотри: *CLI> agi show commands topic ANSWER
#
response = agi_command('ANSWER')

#
```

```
# Попросить абонента ввести номер учетной записи с помощью команды GET DATA.
#     - Проиграть приглашение абоненту please-enter-account-number
#     - Установить время ожидания 30 секунд
#     - Прочитайть максимум 8 цифр (наши номера аккаунтов - 8 цифр)
#
# Смотри: *CLI> agi show commands topic GET DATA
#
response = agi_command('GET DATA please-enter-account-number 30000 8')
#
# Если таймаут истек, просто выйти обратно в диалплан.
#
# Ответ будет выглядеть так:
#     200 result=<digits> (timeout)
#
if 'timeout' in response:
    sys.exit(0)
#
# Ответ будет выглядеть так:
#     200 result=<digits>
# число будет -1 если произошла ошибка.
#
# Разделить ответ на '=', максимум 1 раз. В результате получается массив из двух
# элементов. Второй элемент в массиве (индекс 1) является частью, которая пришла
# после '=' и является номером аккаунта, который мы хотим.
#
account = response.split('=', 1)[1]

#
# Если произошла ошибка, просто повесить трубку.
#
# Смотри: *CLI> agi show commands topic HANGUP
#
if account == '-1':
    response = agi_command('HANGUP')
    sys.exit(0)

#
# Если аккаунт недействителен, сообщить об этом абоненту и выйти из диалплана.
#
# Смотри: *CLI> agi show commands topic STREAM FILE
#
if account not in ACCOUNTS:
    response = agi_command('STREAM FILE invalid-account ""')
    sys.exit(0)

balance = ACCOUNTS[account]['balance']

#
# Сообщить абоненту баланс, а затем выйти обратно в диалплан.
#
# Смотри: *CLI> agi show commands topic SAY NUMBER
#
response = agi_command('STREAM FILE your-balance-is ""')
response = agi_command('SAY NUMBER %s ""' % (balance))
sys.exit(0)
```

Разработка фреймворков

Был предпринят ряд усилий по созданию фреймворков или библиотек, облегчающих Программирование AGI. Таблица 21-3 перечисляет некоторые из них. Если вы не видите здесь библиотеки для предпочитаемого языка программирования, выполните быстрый поиск и вы, скорее всего, найдете ее как и другие.

Таблица 21-3. Разработка фреймворков AGI

Фреймворк	Язык	URL
Adhearsion	Ruby	http://adhearsion.com/
Asterisk-Java	Java	http://asterisk-java.org/
Asterisk-perl	Perl	http://asterisk.gnuinter.net/
PHPAGI	PHP	http://phpagi.sourceforge.net/
StarPy	Python	http://starpy.sourceforge.net/

Вывод

AGI предоставляет мощный интерфейс Asterisk, который позволяет реализовать управление вызовами от первого лица на выбранном языке программирования. Для реализации приложения AGI можно использовать несколько подходов. Некоторые подходы могут обеспечить более высокую производительность, но ценой большей сложности. AGI предоставляет среду программирования, которая может упростить интеграцию Asterisk с другими системами или просто обеспечить более удобную среду программирования для управления вызовами для опытного программиста.

ГЛАВА 22

Кластеризация

Вы не можете съесть гроздь винограда сразу,
но очень легко вы съедите их одну за другой.
—Жак Румен

Слово «кластеризация» может означать разные вещи для разных людей. Некоторые люди скажут, что кластеризация - это просто наличие реплицируемой системы в режиме ожидания, доступной для включения при сбое основной. Для других кластеризация предполагает наличие нескольких систем, работающих совместно друг с другом, с реплицированными данными, полностью избыточными и бесконечно расширяемыми. Для большинства людей истина, вероятно, где-то между этими двумя крайностями.

В этой главе мы рассмотрим возможности кластеризации, которые существуют в Asterisk на высоком уровне и дадим вам знания и направления для планирования вашей системы в будущем. В качестве примеров мы рассмотрим некоторые инструменты, которые мы использовали в наших собственных крупных развертываниях. Хотя нет единого способа создания кластера Asterisk, топологии, которые мы рассмотрим, оказались надежными и популярными с течением времени.

Наши примеры будут углубляться в создание распределенного центра обработки вызовов, одной из наиболее популярных причин построения распределенной системы. В некоторых случаях это необходимо просто потому, что у компании есть вспомогательные офисы, которые она хочет связать с первичной системой. Для других цель состоит в том, чтобы интегрировать удаленных сотрудников или иметь возможность обрабатывать большое количество мест. Мы начнем с простых, традиционных УАТС, и посмотрим, как эта система в конечном итоге может перерасти в нечто гораздо большее.

DevOps: автоматическое развертывание

В последнее время в моде облачные вычисления, это новые унифицированные коммуникации! Однако, в отличие от унифицированных коммуникаций, облачные вычисления, кажется, погружены в реальную разработку и уже произвели большое количество функциональных возможностей (см. OpenStack) и даже создали целые организации (см. Amazon Web Services и тому подобные).

Но нам нужен способ управлять всеми этими распределенными системами и огромной вычислительной мощностью. Обычно многие администраторы создают документацию, bash скрипты и другие пользовательские средства, которые помогают им развертывать системы и управлять ими. Конечно, это становится громоздким, когда вы начинаете масштабировать, и в конечном итоге вам нужно больше людей чтобы помочь поддерживать системы.

Ввод DevOps. Менталитет DevOps - это воспроизводимость до крайности. Используя фреймворк,

который позволяет описать как должны выглядеть ваши системы (или, скорее, что должно быть установлено на основе роли, которую система играет в вашей сети), вы сможете лучше масштабироваться и выходить за пределы без головной боли обслуживания несопоставимых систем. Это всегда было важно, но с появлением облачных вычислений потребность современных администраторов становится еще более насущной.

Есть несколько различных систем, которые вы можете использовать, наиболее популярными являются Puppet от PuppetLabs и Chef от Opscode. Обе делают по существу то же самое, но с использованием отдельного синтаксиса (Puppet имеет свой собственный доменный язык; Chef использует родной Ruby с дополнительными расширениями). Покрутите каждую, посмотрите несколько видео и решите для себя, какую из них вы предпочтете. Обе - удивительные подвиги системной инженерии.

На AstriCon 2012 пол Белангер из Polybeacon и Лейф Мэдсен говорили о DevOps, давая обзор DevOps в среде Asterisk.

Традиционные УАТС

Большинство систем УАТС, развернутых до 2000 года, выглядят очень похоже. Как правило, они включают в себя группу телефонных линий, поставляемых либо через PRI, либо через массив аналоговых линий, соединяющих внешний мир через УАТС, с группой фирменных телефонов и периферийных приложений. Эти системы предоставляют общий набор функций УАТС с дополнительными возможностями, такими как голосовая почта и конференц-связь, предоставляемые через внешние аппаратные модули (обычно это добавляет тысячи долларов к стоимости системы). Эта топология показана на Рисунке 22-1.

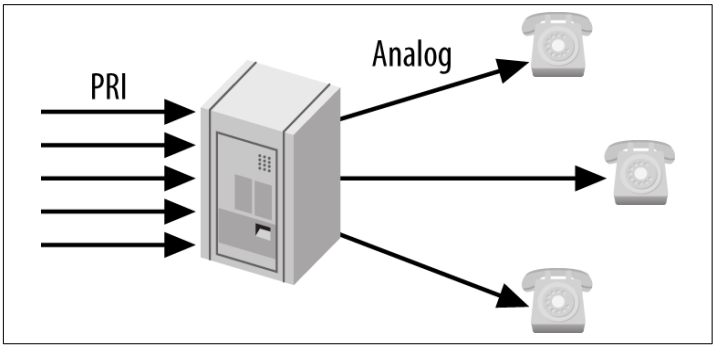


Рисунок 22-1. Традиционный call-центр.

Такие системы используют набор правил для доставки вызовов операторам через стандартные правила автоматического распределения вызовов (ACD) и имеют небольшую гибкость. Вероятно, либо невозможно, либо дорого добавить удаленных агентов так как вызовы должны приходить по ТфОП, которая использует две телефонные линии: одну для входящего звонящего в очередь, а другую для доставки удаленному агенту (в большинстве случаев агенты просто должны находиться в том же физическом местоположении, что и сама АТС).

Однако эти традиционные телефонные системы постепенно вытесняются, поскольку все больше людей начинают требовать использования функций VoIP. И даже для систем, которые не будут использовать VoIP, такие решения, как Asterisk, предлагают функции, которые когда-то стоили тысячи долларов как включенная часть программного обеспечения.

Конечно, с деньгами, вложенными в дорогостоящее оборудование в традиционные АТС, естественно, что организации захотят пользоваться этими системами как можно больше. Кроме того, простая замена существующей системы не только дорога (затраты на проводку для SIP-телефонов, затраты на замену проприетарных телефонных трубок и т.д.), но и может нарушить работу call-центра, особенно если он работает непрерывно.

Однако, возможно пришло время для расширения, и существующая система больше не в состоянии идти в ногу с количеством необходимых линий и количеством мест, необходимых для удовлетворения спроса. В этом случае, может быть выгодно взглянуть на гибридную систему, где существующее

оборудование продолжает использоваться, но новые места и функции добавляются в систему с использованием Asterisk.

Гибридные системы

Гибридная телефонная система (Рисунок 22-2) содержит те же функциональные возможности и аппаратное обеспечение, что и традиционная телефонная система, но к ней подключена другая система, например Asterisk, что обеспечивает дополнительную емкость и функциональность. Добавление Asterisk в традиционную систему обычно выполняется через соединение PRI. С точки зрения традиционной системы Asterisk будет выглядеть как телефонная компания (центральный офис или ЦО). В зависимости от того, как работает традиционная система, и услуг, доступных для или из ЦО, либо Asterisk будет доставлять вызовы от PRI через себя на существующую УАТС, либо существующая УАТС будет отправлять вызовы через соединение PRI в Asterisk, которая затем будет направлять звонки на новые конечные точки (телефоны).

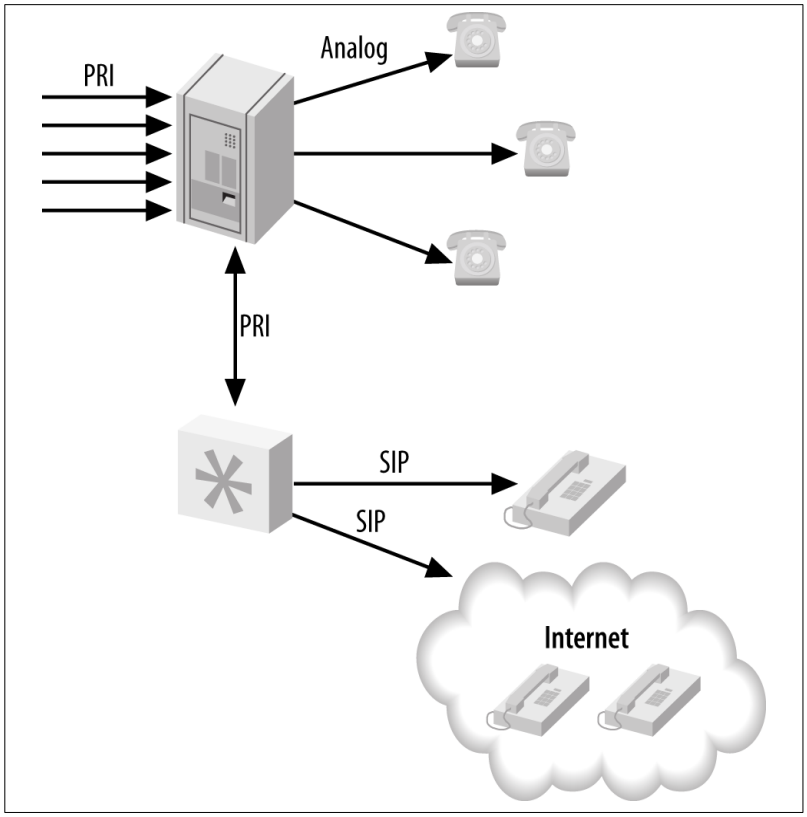


Рисунок 22-2. Удаленная гибридная система

На рисунке - с помощью Asterisk функциональность может быть перемещена по частям из существующей системы УАТС на Asterisk, который может взять на себя большую роль и управлять большей частью системы с течением времени. В конце концов, существующая УАТС может просто использоваться как метод для отправки вызовов на существующие телефоны на рабочих столах агентов, с тем расчетом, что со временем они могут быть заменены на SIP-телефоны, так как проводка и телефоны закуплены.

Добавив Asterisk в существующую систему, мы получаем новый набор функциональных возможностей и преимуществ, таких как:

- Поддержка удаленных сотрудников с осуществлением вызовов через существующее подключение к интернету
- Такие функции, как конференц-связь и голосовая почта (с возможностью уведомления пользователей по электронной почте о новых сообщениях)
- Расширенные телефонные линии с использованием VoIP, а также снижение затрат на междугороднюю связь

Такая система по-прежнему имеет ряд недостатков, поскольку все оборудование должно находиться в call-центре, и мы, по-прежнему, ограничены использованием дорогостоящего (относительно) оборудования в системе Asterisk для подключения к традиционной УАТС. Тем не менее, мы движемся в правильном направлении и с системой Asterisk мы можем начать миграцию с течением времени, ограничивая перерывы в бизнесе и применяя более постепенный подход к обучению пользователей.

Чистый Asterisk, нераспределенный

Следующим шагом в нашем путешествии является чистая система Asterisk. В этой системе мы успешно перешли от существующей системы УАТС и теперь обрабатываем все функции через Asterisk. Наш существующий PRI был присоединен к Asterisk, и мы расширили наши возможности за счет интеграции в систему поставщика услуг интернет-телефонии (ITSP). Все агенты теперь используют SIP-телефоны, и мы даже добавили несколько удаленных сотрудников. Эта топология показана на Рисунке 22-3.

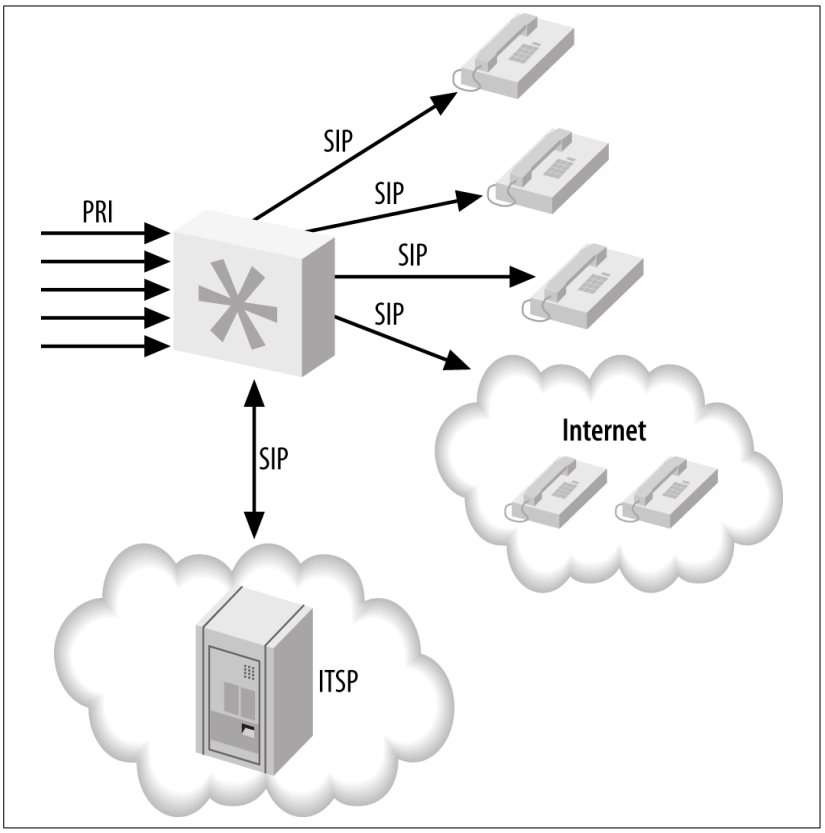


Рисунок 22-3. Нераспределенный Asterisk

Удаленные сотрудники могут быть большим преимуществом для компании. Мало того, что позволяя вашим сотрудникам работать из отдаленных мест вы повышаете их моральный дух, облегчая бремя потенциально длительных поездок на работу, так же это позволяет людям работать в среде, в которой они чувствуют себя комфортно, что может сделать их более продуктивными. Кроме того, менеджер колл-центра не имеет меньшего контроля над статистикой сотрудников: звонки удаленных работников по-прежнему могут контролироваться в учебных целях, а собранные статистические данные не отличаются от данных, полученных менеджером, чем для сотрудников, работающих на месте.

Измеримым преимуществом для компании является уменьшение количества оборудования, необходимого каждому сотруднику. Если агенты могут использовать свои существующие компьютерные системы, электрические сети и интернет-соединения, компания может сэкономить значительную сумму денег, поддерживая удаленных сотрудников. Кроме того, эти сотрудники могут быть расположены по всему миру, что увеличит количество часов, доступных вашим агентам и позволит вам обслуживать больше часовых поясов.

Использование этой системы является простым и эффективным, но по мере роста компании, система может достичь проблемы мощности. Мы рассмотрим как система может быть расширена в этой главе позже.

Asterisk и интеграция базы данных

Интеграция Asterisk с базой данных может добавить много функциональных возможностей в вашу систему. Кроме того, она позволяет создавать веб-утилиты настройки для упрощения обслуживания системы Asterisk. Более того, обеспечивает мгновенный доступ к информации из диалплана и других частей системы Asterisk.

Единая база данных

Добавление интеграции базы данных в Asterisk (Рисунок 22-4) - это мощный способ получить доступ к информации, которой можно манипулировать другими средствами. Например, мы можем считывать информацию о внутренних номерах и устройствах в системе из базы данных, используя архитектуру Asterisk Realtime (обсуждается в Главе 16), и можем изменять информацию, хранящуюся в базе данных, через внешнюю систему, такую как веб-страница.

Интеграция с базой данных добавляет слой между Asterisk и веб-интерфейсом, с которым знаком веб-дизайнер, и позволяет манипулировать данными таким образом, который не требует новых навыков. Знание самого Asterisk остается за администратором Asterisk, и веб-разработчик может с радостью работать с инструментами, с которыми он знаком.

Конечно, это делает систему Asterisk несколько более сложной для построения, но интеграция с базой данных через ODBC добавляет новые возможности (такие как hot-desking, обсуждаемый в разделе "Веселимся с func_odbc: горячий стол" в Главе 16). `func_odbc` - это мощный инструмент для администратора Asterisk, предоставляющий возможность строить статический диалплан с использованием данных, которые по своей природе являются динамическими. См Главу 16 для получения дополнительной информации о том, как интегрировать Asterisk с базой данных и предоставляемыми ею функциональными возможностями.

Нам также очень нравится модуль `func_curl`, который обеспечивает интеграцию с веб-сервисами по HTTP напрямую с диалпланом.

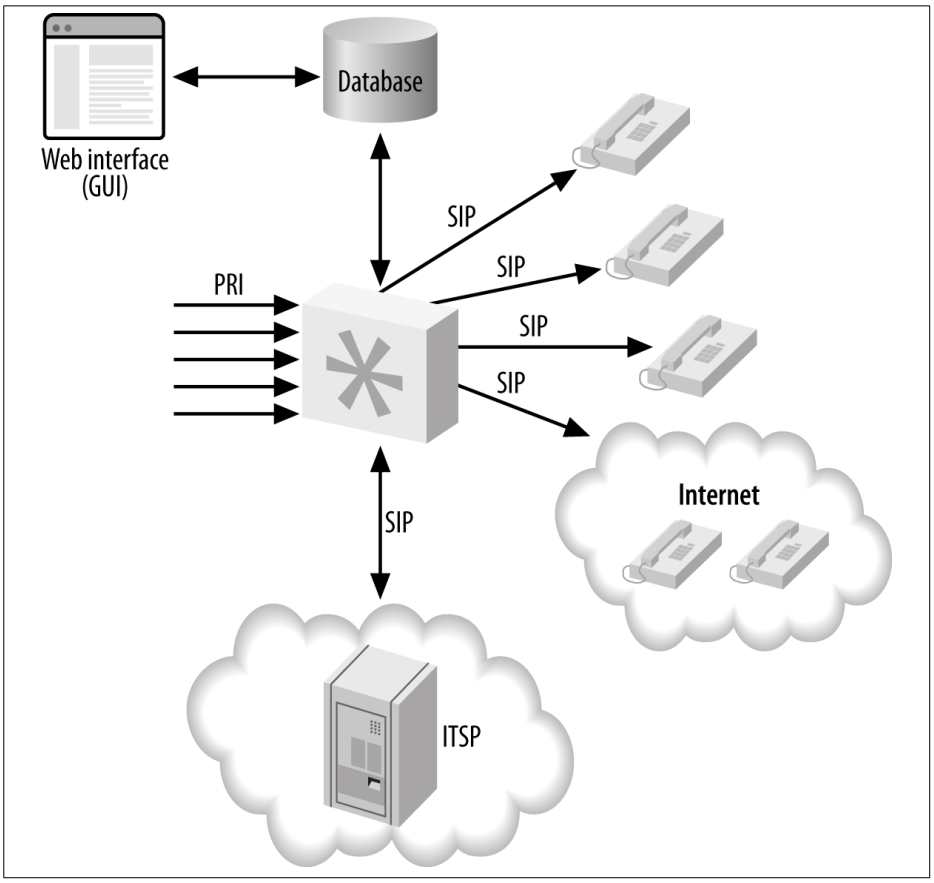


Рисунок 22-4. Интеграция базы данных Asterisk, один сервер.

С данными, абстрагированными непосредственно от Asterisk, нам будет проще перейти к системе, которая готовится к кластеризации. Мы можем использовать что-то вроде LinuxHA для обеспечения автоматической отработки отказа между системами. В то время, как в случае сбоя вызовы в системе, которая отказала, будут потеряны, отработка отказа займет несколько мгновений (меньше секунды), чтобы система снова стала немедленно доступна своим пользователям. В этой конфигурации, поскольку наши данные абстрагируются за пределами Asterisk, мы можем использовать такие приложения, как *unison* или *rsync*, чтобы синхронизировать файлы конфигурации между основной и резервной системами. Мы также можем использовать *subversion* или *git* для отслеживания изменений в файлах конфигурации, что упрощает откат изменений, которые не работают.

Конечно, если наша база данных исчезнет из-за сбоя оборудования или программного обеспечения, наша система будет недоступна, если не будет запрограммирована таким образом, чтобы иметь возможность работать без подключения к базе данных. Это может быть достигнуто либо с помощью локальной базы данных, которая периодически обновляется из основной, либо с помощью информации, запрограммированной непосредственно в диалплане. В большинстве случаев функциональность системы в этом режиме будет ниже, чем когда база данных была доступна, но, по крайней мере, система не будет полностью непригодна для использования.

Лучшим решением было бы использовать реплицированную базу данных, которая позволяет одновременно писать данные, записанные на сервер базы данных, на другой сервер. После этого Asterisk может автоматически переключиться на другую базу данных, если основной сервер станет недоступен.

Реплицирование базы данных

Использование реплицированной базы данных обеспечивает некоторую избыточность в бэкэнде, чтобы ограничить время простоя вызывающих абонентов и агентов в случае сбоя базы данных. Конфигурация базы данных master-master необходима, чтобы данные могли быть записаны в любую базу данных и автоматически реплицированы в другую систему, гарантируя существования точной копии данных на двух физических машинах. Еще одним преимуществом такого подхода является то,

что одной системе больше не нужно обрабатывать все транзакции в базу данных; нагрузку можно разделить между серверами. Рисунок 22-5 иллюстрирует эту распределенную структуру.

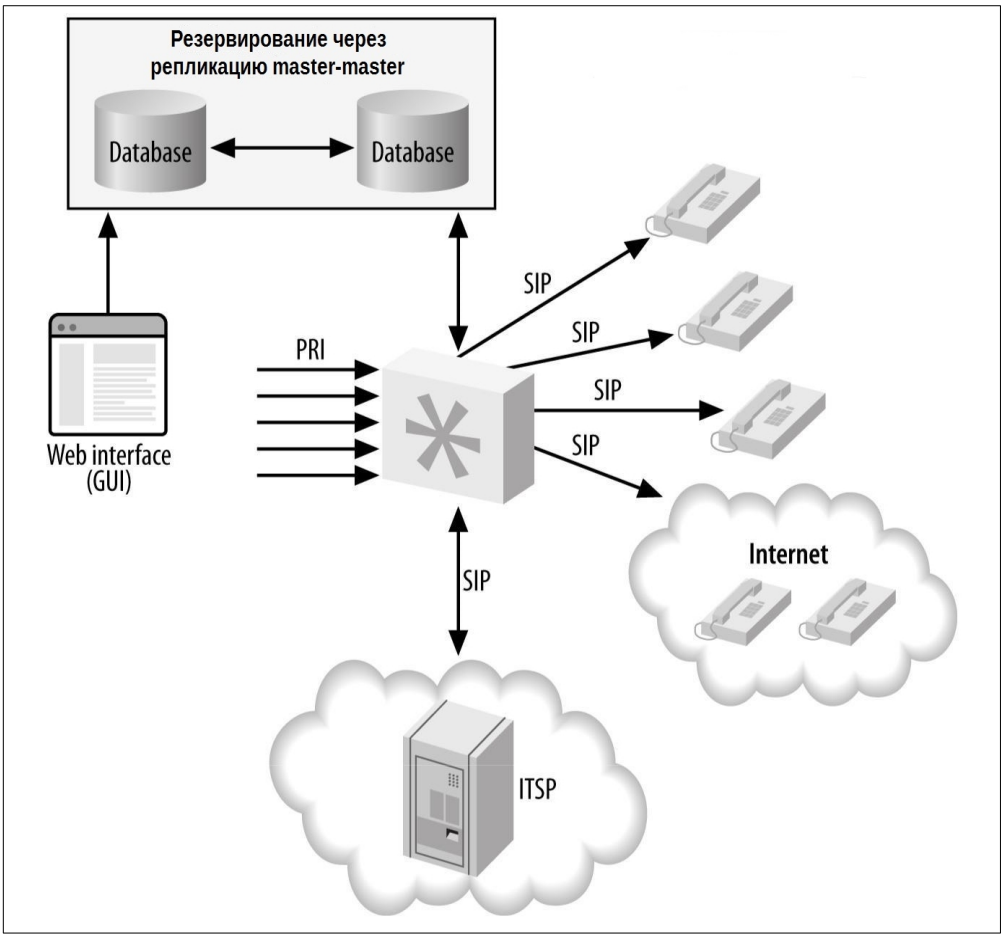


Рисунок 22-5. Интеграция базы данных Asterisk, распределенная база данных



Мы уже использовали репликацию MySQL master-master раньше, и она работает довольно хорошо. Это также не так трудно настроить, и несколько учебников доступны в Интернете. Также стоит отметить, что репликация master-master - это обходной путь для двух репликаций master-slave, например:

```
host1.master --> host2.slave
host1.slave <-- host2.slave
```

Это работает довольно хорошо, когда у вас есть только два сервера, но по мере того, как вы начинаете расти и расширяться и требовать еще больше серверов баз данных, вам нужны более тяжелые технологии, такие как MySQL Cluster или Galera Cluster for MySQL.
Другие системы баз данных, вероятно, также будут содержать эту функциональность, особенно если вы используете коммерческую систему, такую как Oracle или MS SQL.

Отработку отказа можно выполнить в Asterisk, так как `res_odbc` и `func_odbc`[1] содержат параметры конфигурации, позволяющие указать несколько баз данных. В `res_odbc` можно указать предпочтительный порядок подключения к базе данных в случае сбоя. В `func_odbc` можно даже указать различные серверы для чтения и записи данных с помощью создаваемых функций диалплана. Вся эта гибкость позволяет вам обеспечить систему, которая работает хорошо для вашего бизнеса.

Внешние программы также можно использовать для управления отработкой отказа между системами. Приложение *pen* - это балансировщик нагрузки для простых TCP-приложений, таких как HTTP или SMTP, который позволяет нескольким серверам отображаться как один. Это означает, что Asterisk необходимо настроить для подключения только к одному IP-адресу (или имени хоста); приложение *pen* будет контролировать, какой сервер будет использоваться для каждого запроса.

1 К сожалению, `cdr_odbc` в настоящее время не поддерживает отработку отказа.

Asterisk и предоставление статуса устройства

Состояния устройств в Asterisk являются важными как с точки зрения программного обеспечения (Asterisk нужно знать состояние устройства или линии на устройстве для того, чтобы узнать, возможно ли совершить вызов через нее) и с точки зрения пользователей (например, индикатор может быть включен или выключен, чтобы показать, является ли конкретная линия занятой или же агент доступен для любых звонков). С точки зрения очереди чрезвычайно важно знать статус устройства, которое использует агент, чтобы определить, может ли следующий абонент в очереди быть распределен этому агенту. Без знания состояния устройства очередь просто выполняла бы несколько вызовов к одной и той же конечной точке.

Как только вы начнете расширять свою единую систему до нескольких блоков (потенциально в нескольких физических местоположениях, таких как удаленные или вспомогательные офисы), вам нужно будет распределить состояния устройств конечных точек между системами. Тип реализации, которая требуется, будет зависеть от того, распределяете ли вы их между системами в той же локальной сети (каналы с низкой задержкой) или по глобальной сети (каналы с более высокой задержкой). Мы обсудим два метода распределения состояний устройств в данном разделе: Corosync для низких задержек связи и XMPP при высоких задержках.

Распространение состояний устройств по локальной сети

Реализация Corosync (ранее OpenAIS) была впервые добавлена к Asterisk в ветви 1.6.1, чтобы обеспечить распространение информации о состоянии устройств между серверами. Добавление Corosync обеспечило большие возможности для распределенных систем, так как осведомленность о состоянии устройств является важным аспектом таких систем. Предыдущие методы требовали использования `GROUP()` и `GROUP_COUNT()` для каждого канала, с этой информацией, запрашиваемой для распределенного универсального распознавания номеров (DUNDi). В то время как этот подход полезен в некоторых сценариях (мы могли бы использовать эту функциональность для поиска количества вызовов, которые наши системы обрабатывают и направляют вызовы разумно системам, обрабатывающим меньшее количество вызовов), в качестве механизма определения информации о состоянии устройства, которого ему крайне не хватает.

Corosync действительно дал нам первую реализацию системы, которая позволяет распределять состояния устройств и индикаторы сообщений ожидания между несколькими системами Asterisk (см. Рисунок 22-6). Недостатком реализации Corosync является то, что она требует чтобы все системы работали по каналам с низкой задержкой, что, как правило, означает, что все они должны находиться в одном физическом месте и быть подключены к одному коммутатору. Тем не менее, хотя библиотека Corosync не работает в физически раздельных сетях, она позволяет `Queue()` находиться в одной системе, а участникам очереди - в другой (или нескольких системах). Он делает это, не требуя от нас использования локальных каналов и проверки их доступности другими способами, тем самым ограничивая (или исключая) количество попыток подключения по сети и звонки с нескольких устройств.

Использование Corosync имеет преимущество в том, что его относительно легко настроить и начать работать. Недостатком является то, что он не распространяется по физическим локациям, хотя мы можем использовать XMPP для распределения состояния устройств по глобальной сети, как вы увидите в следующем разделе.

Дополнительную информацию о настройке распределенных состояний устройств с помощью Corosync можно найти в разделе "Использование Corosync" в Главе 14.

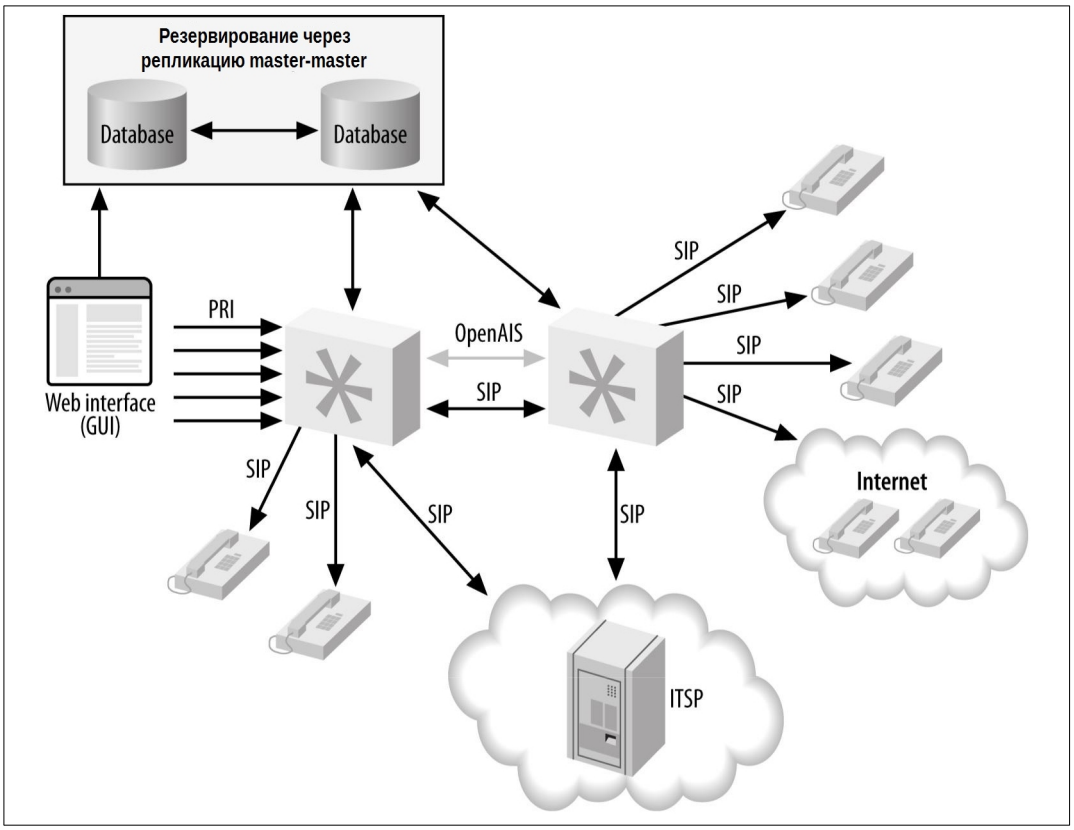


Рисунок 22-6. Распределение состояний устройств с помощью Corosync

Распространение состояний устройств по глобальной сети

Поскольку протокол XMPP предназначен (или, по крайней мере, позволяет) использовать его в глобальных сетях, мы можем иметь системы Asterisk в разных физических местоположениях, которые передают информацию о состоянии устройств друг другу (см. Рисунок 22-7). При реализации Corosync библиотека будет использоваться в каждой системе, что позволит им распространять информацию о состоянии устройств. В сценарии XMPP центральный сервер (или кластер серверов) используется для распределения состояний между всеми блоками Asterisk в кластере. В настоящее время лучшим приложением для этого является сервер Tigase XMPP, поскольку он поддерживает события PubSub. Хотя другие серверы XMPP могут так же поддерживать их в будущем, известно, что в настоящее время работает только Tigase.

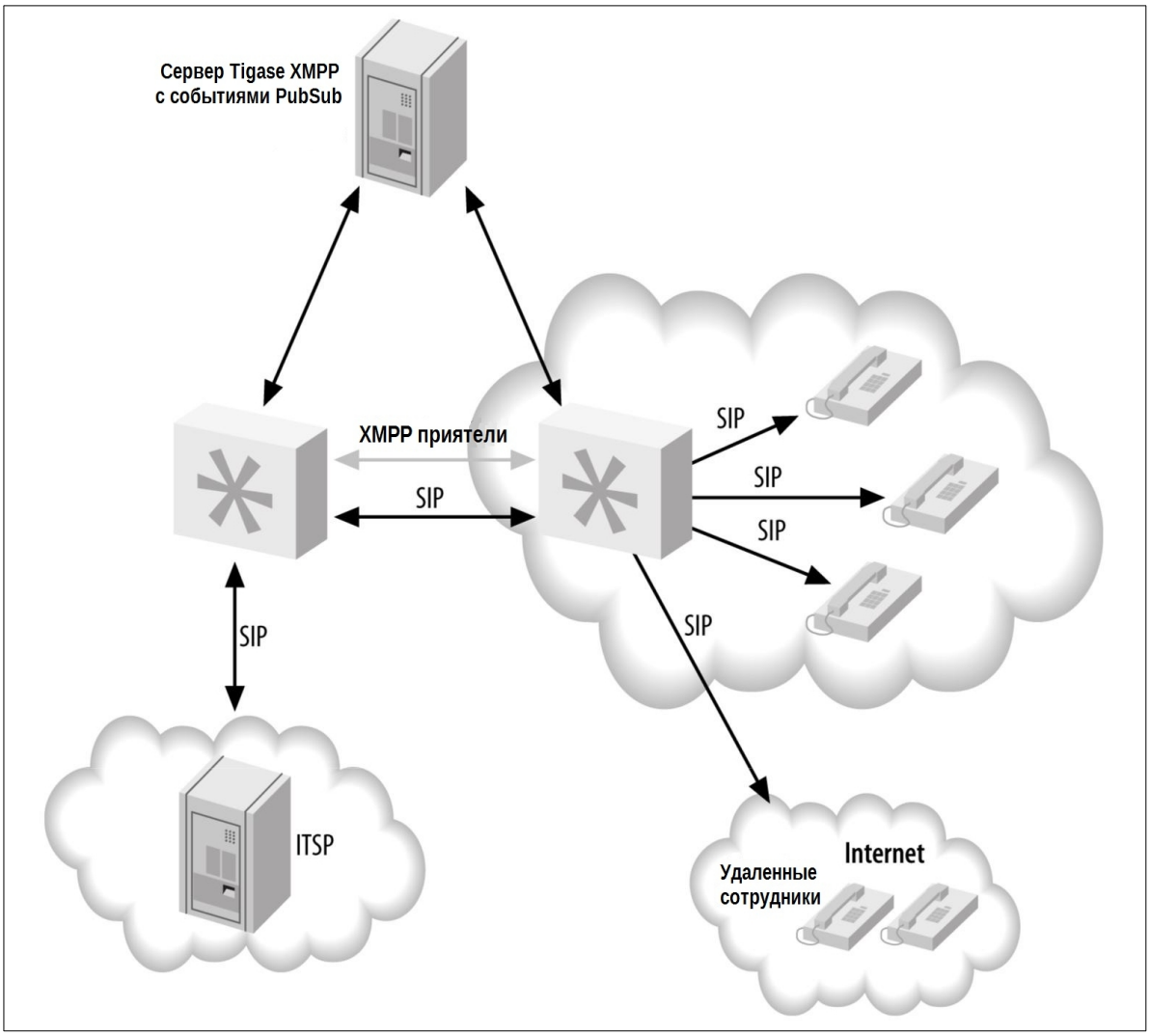


Рисунок 22-7. Распределение состояний устройств с помощью XMPP

В XMPP очереди могут находиться в разных физических местах, а удаленные офисы могут принимать вызовы из основного офиса или наоборот. Это обеспечивает еще один уровень избыточности, поскольку если основной офис отключается и ITSP настроен таким образом, чтобы переключаться на другой офис, вызовы могут распределяться между этими офисами до тех пор, пока основной не перейдет в рабочий режим. Это очень увлекательно для многих, поскольку добавляет слой функциональности, который ранее не был доступен, и большинство из них можно сделать с относительно минимальной конфигурацией.

Преимущество XMPP в распределении состояний устройств состоит в том, что можно распределять состояние по нескольким физическим местоположениям, что невозможно с Corosync. Недостатком является то, что его сложнее настроить (так как вам нужен внешний сервис, на котором работает сервер Tigase XMPP), чем реализацию Corosync.

Дополнительную информацию о настройке распределенных состояний устройств с помощью XMPP можно найти в разделе «Использование XMPP» в Главе 14.

Несколько очередей, несколько местоположений

Теперь давайте проявим творческий подход и используем различные инструменты, которые мы обсуждали в предыдущих разделах, для построения инфраструктуры распределенных очередей. Рисунок 22-8 иллюстрирует пример установки, где у нас есть пять серверов Asterisk, запущенных другим кластером, используемым для распределения/маршрутизации вызовов в различные очереди, которые мы настроили. Наш ITSP отправляет вызовы в кластер маршрутизации (который может быть чем-то вроде Kamailio, или даже на несколько серверов Asterisk, реализующих DUNDi или какой-либо другой метод для маршрутизации и распределения вызовов), который затем отправляет вызовы

в соответствии с одной из трех систем Asterisk, в которых наши очереди настроены. Каждый сервер обрабатывает свою очередь, такую как sales, technical support и returns. Эти серверы, в свою очередь, используют агентов, расположенных в двух разных физических местах. Устройства агентов регистрируются на их собственных локальных серверах регистрации (которые также могут выполнять другие функции).



Мы не показываем все аспекты системы, чтобы сохранить диаграмму простой, но в этом случае мы будем использовать систему распространения состояний устройств XMPP, поскольку подразумеваем, что агенты распределены по нескольким физическим местоположениям.

Все агенты в разных местах могут быть загружены в одну или несколько очередей и, поскольку мы распространяем информацию о состоянии устройств, каждая очередь будет знать текущее состояние агентов в очереди и будет только распределять вызывающих абонентов агентам, в зависимости от ситуации. Кроме того, мы можем настроить штрафы для очередей и/или для агентов, чтобы доставлять вызывающих абонентов лучшим агентам, если они доступны, и использовать других агентов только когда все лучшие агенты заняты (для получения дополнительной информации о штрафах и приоритетах, см. «Расширенные очереди» в Главе 13).

Мы можем добавить больше агентов в систему, добавив больше серверов в кластер либо в том же местоположении, либо в дополнительных физических местах. Мы также можем увеличить количество поддерживаемых очередей, добавив больше серверов, каждый из которых обрабатывает свою очередь или очереди.

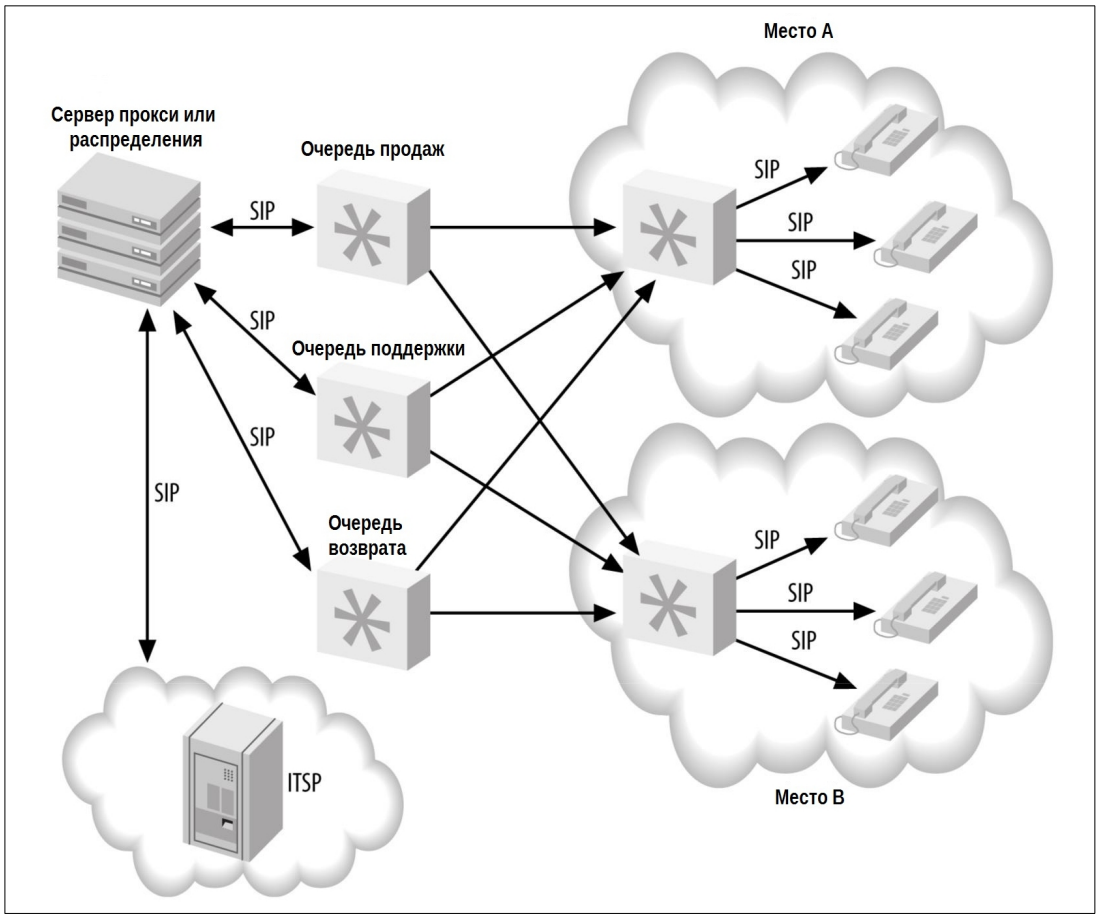


Рисунок 22-8. Инфраструктура распределенных очередей

Недостатком использования этой системы является способ разработки приложения `Queue()`. `Queue()` является одним из старых приложений в Asterisk, и, к сожалению, оно не поспевает за темпами развития в области распределения состояний устройств, поэтому нет способа распределить одну и ту же `Queue()` по нескольким блокам. Например, предположим что у вас есть очереди продаж в двух системах. Если вызывающий абонент входит в очередь продаж в первой системе Asterisk, а затем другой вызывающий абонент входит в очередь продаж во втором блоке, между этими

очередями не будет распределяться информация о том, кто первый, а кто второй в очереди. Две очереди эффективно разделены. Возможно, будущие версии Asterisk добавят эту возможность, но в настоящее время она не поддерживается. Мы упоминаем об этом, чтобы вы могли планировать свою систему соответственно.

Поскольку очереди в некоторых реализациях (таких как call-центры) могут потребоваться для обработки нескольких вызовов одновременно, требования к обработке и загрузке для одной системы могут быть довольно высокими. Имея возможность использовать одни и те же ресурсы агента в нескольких системах, мы можем распределять наших абонентов между несколькими блоками, значительно снижая требования к обработке, предъявляемые к любой отдельной системе. Больше не нужно чтобы это все делала одна система — мы можем распределять различные компоненты системы на разные сервера.

Заключение

В этой главе мы рассмотрели, как можно преобразовать традиционную (не Asterisk) систему телефонии в распределенный call-центр. Попутно мы рассмотрели как колл-центр с несколькими местоположениями может превратиться в систему с сотнями мест в разных физических локациях.

В то время как способность развивать свой бизнес и планировать будущее имеет решающее значение, также важно не создавать систему, которая является более сложной, чем должна быть. Чем дальше вы идете, и чем более распределенную систему строите, тем больше времени потребуется, чтобы оторваться от Земли и тем труднее будет выполнить все то, что важно, при возникновении изменений — таких как тестирование, внедрение изменений, и поддержка синхронизации вещей. Если ваша система никогда не выйдет за пределы 40-местного колл-центра, не стройте ее на 500 мест. В этом случае все что вы делаете добавляет затраты и сложность для размещения системы в масштабе, который никогда не будет полностью реализован.

Построение простой системы сейчас и планирование на будущее того, как вы собираетесь туда попасть (особенно если вы можете сделать это в итерациях, без необходимости разрывать всю инфраструктуру на части или начинать с нуля) поможет вам запуститься и заработать быстрее. По мере того как вы растете, вы можете добавить больше частей и определить, правильный ли подход принимаете. Если неправильный, то сможете вернуться и переделать проблемную часть. Такой подход может избавить вас от многих головных болей в будущем, когда вы понимаете, что не нужно переделывать всю сложную систему из-за какого-то требования, которое вы не предвидели вначале.

Мы также упомянули некоторые преимущества распределенной системы с удаленными сотрудниками, такие как улучшение морального духа сотрудников и экономия средств. Вы можете использовать существующие у ваших сотрудников интернет-соединения, аппаратное обеспечение и электричество, что может сэкономить деньги компании, и ваши сотрудники выиграют, избегая обострения и затрат на поездку в офис каждый день. Хотя не все ситуации допускают такой сценарий, стоит изучить будет ли поддержка удаленных сотрудников полезна для вашего бизнеса.

Наконец, распространение состояний устройств может открыть для вашей компании целый мир возможностей, позволив ей выйти за рамки единой системы Asterisk, которая делает все. Выход из функциональности в несколько блоков теперь является реальностью, и к ней можно подойти с уверенностью, которую раньше не видели.

Распределённое универсальное распознавание номеров (DUNDi)

Общество подобно кораблю; каждый должен быть готов принять штурвал.
—Генрик Ибсен

Distributed Universal Number Discovery, или DUNDi (Распределенное универсальное распознавание номеров) - это протокол обнаружения служб, который можно использовать для поиска ресурсов в удаленных местоположениях. Первоначальное намерение DUNDi состояло в том, чтобы разрешить децентрализованную маршрутизацию между многими узлами с использованием General Peering Agreement (GPA - общее соглашение о пиринге). GPA призван взять на себя роль централизованного органа управления документацией для создания доверительных отношений между пирами в облаке. Хотя идея интересная и здравая, GPA еще не взлетел. Это не означает, что сам протокол DUNDi не нашел применения: первоначальное намерение DUNDi было расширено, так что теперь он действует не только как служба определения местоположения, а может использоваться для запроса и передачи информации среди пиров.

Как работает DUNDi?

Думайте о DUNDi как о большой телефонной книге, которая позволяет вам спрашивать коллег, знают ли они об альтернативном маршруте VoIP к добавочному номеру или номеру телефона PSTN.

Например, предположим что вы подключены к другому набору из блоков Asterisk, слушаете и отвечаете на запросы DUNDi, и те блоки, которые в свою очередь подключены к другим блокам Asterisk слушают и отвечают на запросы DUNDi. Предположим также, что ваша система не имеет прямого доступа для запроса чего-либо с удаленных серверов.

Рисунок 23-1 иллюстрирует, как работает DUNDi. Вы спрашиваете своего друга Боба, знает ли он, как достичь внутреннего номера `4001`, к которому у вас нет прямого доступа. Боб отвечает: “Я не знаю, как добраться до этого номера, но позвольте мне спросить моего пира Салли.”

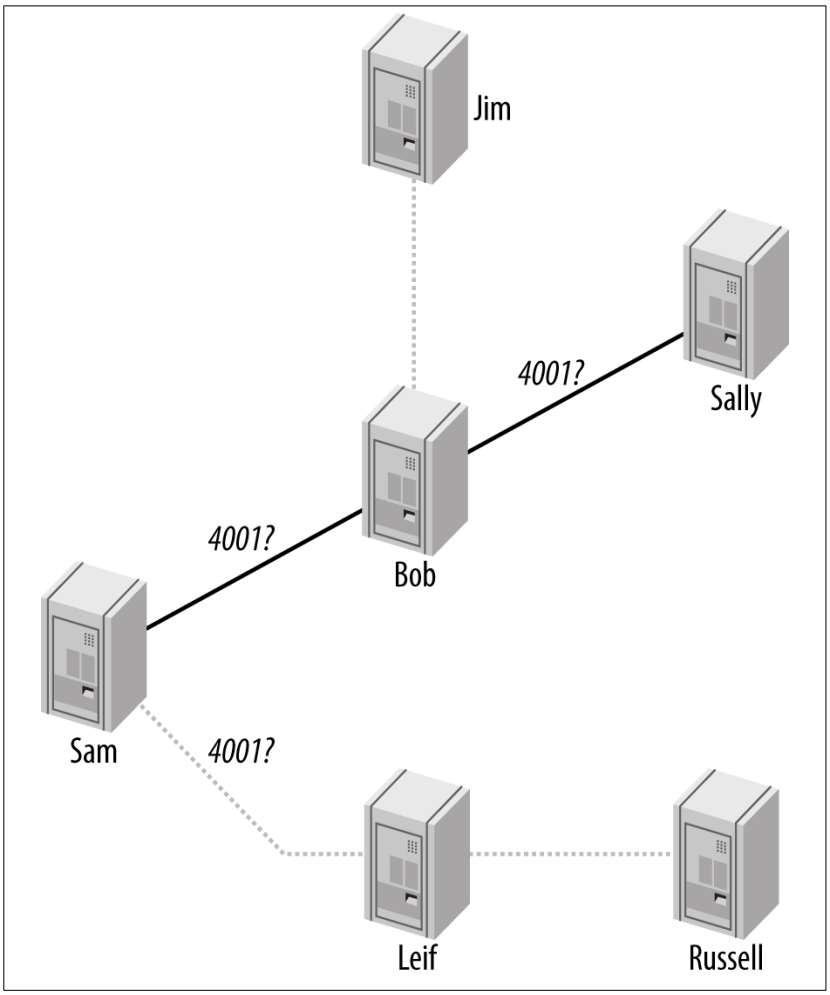


Рисунок 23-1. Система запросов DUNDi peer-to-peer

Боб спрашивает Салли, знает ли она, как достичь запрошенного внутр.номера, и она отвечает: “Вы можете достичь этого номера в `IAX2/dundi:`*`very_long_password@hostname/extension`*.” Затем Боб сохраняет адрес в своей базе данных и передает вам информацию о том, как добраться до `4001`. С новообретенной информации, вы можете сделать отдельный запрос, чтобы на самом деле разместить вызов к боксу Салли, чтобы достичь номера `4001`. (DUNDi только помогает вам найти информацию, в которой вы нуждаетесь для соединения; но фактически не совершает вызов.)

Поскольку Боб сохранил найденную информацию, он сможет предоставить ее всем коллегам, которые позже запросят у него тот же номер, так что поиск не будет идти дальше. Это помогает уменьшить нагрузку на сеть и уменьшает время отклика для номеров, которые часто просматриваются. (Однако следует отметить, что DUNDi создает сменяющийся ключ, и таким образом сохраненная информация действительна в течение ограниченного периода времени.)

DUNDi выполняет поиск динамически, либо с помощью оператора `switch =>` в вашем файле *extensions.conf* или с использованием функции диалплана `DUNDILOOKUP()`.

В то время как DUNDi был первоначально разработан и предназначен для использования в качестве пиринговой матрицы для ТфОП, он чаще всего используется в частных сетях.[1] Если вы являетесь администратором Asterisk на крупном предприятии (или даже установки с парой блоков Asterisk в разных физических расположениях), то можете упростить администрирование добавочных номеров. DUNDi - это фантастический инструмент для этого, потому что он позволяет вам просто делиться внутренними номерами, которые были настроены в каждом местоположении динамически, запрашивая номера из удаленного местоположения, когда ваш локальный блок не знает, как их достичь.

1 Для публичных сетей и сетей, не контролируемых вашей организацией, используйте ISN (абонентский номер ITAD) это наш предпочтительный метод. Мы говорили об ITAD и ISN в “ISN, ITAD, и freenum.org” в Главе 12.

Кроме того, если в одной локации есть более дешевый маршрут к номеру ТфОП, который вы хотели набрать, то можете запросить маршрут в DUNDi облаке. Например, если один блок находится в Ванкувере, а другой - в Торонто, то Ванкуверский офис может отправлять звонки, предназначенные для района Торонто, по сети с помощью VoIP и из PRI в Торонто, чтобы они могли быть размещены локально в ТфОП. Аналогичным образом, отделение в Торонто могло бы осуществлять звонки, предназначенные для Ванкувера, из PRI отделения в Ванкувере.

Файл dundi.conf

Часто бывает полезно ознакомиться с опциями, доступными нам, до того, как углубиться в файл конфигурации, но не стесняйтесь пропустить этот раздел сейчас и вернуться по ссылкам на конкретные параметры после того, как вы получили свою первоначальную конфигурацию и работаете.

В *dundi.conf* есть три секции: раздел `[general]`, раздел `[mappings]` и определения пиров, такие как `[FF:FF:FF:FF:FF:FF]`. Мы покажем варианты, доступные для каждого раздела В отдельных таблицах.

В Таблице 23-1 перечислены параметры, доступные в разделе `[general]` *dundi.conf*.

Таблица 23-1. Параметры, доступные в разделе [general]

Опция	Описание
`department`	Используется при запросе контактных данных удаленной системы. Примером может служить `Communications`.
`organization`	Используется при запросе контактных данных удаленной системы. Примером может быть `ShiftEight.org`.
`locality`	Используется при запросе контактных данных удаленной системы. Примером может быть `Toronto`.
`stateprov`	Используется при запросе контактных данных удаленной системы. Примером может быть `Ontario`.
`country`	Используется при запросе контактных данных удаленной системы. Примером может быть `Canada`.
`email`	Используется при запросе контактных данных удаленной системы. Примером может быть `support@shifteight.org`
`phone`	Используется при запросе контактных данных удаленной системы. Примером может быть `+1-416-555-1212`
`binaddr`	Используется для управления IP-адресом, к которому будет привязываться система. Это может быть только IPv4-адрес. Значение по умолчанию - `0.0.0.0`, что означает, что система будет слушать (и отвечать) на всех доступных интерфейсах.
`port`	Порт для прослушивания запросов. Значение по умолчанию - `4520`.
`tos`	Значение условия обслуживания или качества обслуживания (ToS/QoS), которое будет использоваться для запросов. Дополнительные сведения о доступных значениях и способах их использования см. в wiki.
`entityid`	ID объекта системы. Должен быть внешний (сетевой) MAC-адрес. Формат `00:00:00:00:00:00`.
`cachetime`	Как долго пиры должны кэшировать наши ответы, в секундах. По умолчанию `3600`.
`ttl`	Время жизни или максимальная глубина поиска ответа в сети. Максимальное время ожидания ответа рассчитывается с использованием (2000 + 200 * ttl) мс.
`autokill`	Используется для контроля того, как долго мы ждем ACK для нашего `DPDISCOVER`. Установка этого тайм-аута предотвращает остановку поиска из-за скрытого пира. Он может быть `yes`, `no` или числовое значение, представляющее количество миллисекунд ожидания. Вы можете использовать параметр `qualify`, чтобы включить его для каждого пира.
`secretpath`	Сменяющийся ключ создается и хранится в AstDB. Значение хранится в ключе `'secret'` под семейством, определенным `secretpath`. По умолчанию `secretpath` - `dundi`, в результате чего ключ будет храниться по умолчанию в `dundi/secret`.
`storehistory`	Используется для указания, следует ли хранить в памяти историю последних нескольких запросов, а также сколько времени заняли запросы. Допустимые значения-yes и no (также доступны с помощью команд cli dundi store history и dundi no store history). Это средство отладки, которое по умолчанию отключено из-за возможных воздействий на производительность.

В Таблице 23-2 перечислены параметры, которые можно настроить в разделе `[mappings]` *dundi.conf*.

Таблица 23-2. Параметры, доступные в разделе [mappings]

Опция	Описание
`nounsolicited`	Используется для рекламы без нежелательных звонков на возвращаемый результат. Применяется в публичных сетях.
`nocomunsolicit`	Используется для рекламы, некоммерческих нежелательных звонков на возвращаемый результат. Применяется в публичных сетях.
`residential`	Используется для определения маршрута, возвращаемого в качестве жилого местоположения. Применяется в общественных сетях.
`commercial`	Используется для определения маршрута, возвращаемого как коммерческое местоположение. Применяется в общественных сетях.
`mobile`	Используется для определения маршрута, возвращаемого как мобильный телефон. Применяется в общественных сетях.
`nopartial`	Используется для предотвращения частичного поиска номеров, выполняемого для этого сопоставления.
`${NUMBER}`	Переменная, содержащая значение просматриваемого запроса.
`${IPADDR}`	Переменная, содержащая IP-адрес локальной системы. Может использоваться для динамического построения ответов сопоставления. Не рекомендуется.
`${SECRET}`	Переменная, содержащая значение сменяющегося секретного ключа, как определено в `secretpath` в AstDB.

Наконец, в Таблице 23-3 перечислены параметры, доступные в разделах пиров dundi.conf.

Таблица 23-3. Параметры, доступные для определений пиров в dundi.conf

Опция	Описание
`inkey`	Входящий ключ аутентификации.
`outkey`	Ключ, используемый для проверки подлинности удаленного узла.
`host`	Имя хоста или IP-адрес удаленного узла.
`port`	Порт, по которому можно связаться с удаленным узлом.
`order`	Порядок поиска, связанный с этим узлом. Значения включают `primary`, `secondary`, `tertiary` и `quartiary`. Будет искать только первичные пиры, если они недоступны - в этом случае будут искать вторичные и т.д.
`include`	Используется для управления включением этого пира в поиск определенного сопоставления. Может быть установлено значение `all`, если используется для всех сопоставлений.
`noinclude`	Используется для управления исключениями этого пира из поиска определенного сопоставления. Может быть установлено `all`, если этот узел должен быть исключен из поиска.
`permit`	Используется для управления возможностью выполнения пиром поиска по определенным сопоставлениям. Если значение равно `all`, этот пир может выполнять поиск по всем определенным сопоставлениям.
`deny`	Используется для контроля того, какие сопоставления этому узлу запрещено искать. Значение может быть установлено на `all`, чтобы ограничить этому узлу возможность выполнять любые поиски по определенным сопоставлениям.
`model`	Используется для управления тем, может ли этот пир получать запросы (`inbound`), передавать запросы (`outbound`) или выполнять и то и другое (`symmetric`).
`precache`	Обычно используется, когда у нас есть узел с несколькими маршрутами, который хочет передать эти значения на другой узел, предоставляющий больше ответов (это известно как предварительное кэширование, предоставление ответа, когда запрос не был получен). Значения включают `outgoing`, `incoming` и `symmetric`. Если установлено в `outgoing` - мы прокладываем маршруты к этому узлу. При значении `incoming` мы получаем маршруты от этого узла. Если задано `symmetric` мы делаем и то и другое.

Настройка Asterisk для использования с DUNDi

Есть три файла, которые необходимо настроить для DUNDi: *dundi.conf*, *extensions.conf* и *sip.conf*.[2] Файл *dundi.conf* контролирует аутентификацию пиров, которым мы разрешаем выполнять поиск через нашу систему. Этот файл также управляет списком пиров, которым мы можем отправить наши собственные запросы поиска. Поскольку можно запускать несколько разных сетей на одном и том же блоке, необходимо определить разные секции для каждого узла, а затем настроить сети, в которых этим пирам разрешено выполнять поиск. Кроме того, нам нужно определить, какие пиры мы хотим использовать для выполнения поиска.

Общая конфигурация

В разделе `[general]` в *dundi.conf* содержатся параметры, относящиеся к общей работе клиента и сервера DUNDi:

```
;Файл конфигурации DUNDi для Toronto
;
[general]
;
department=IT
organization=toronto.shifteight.org
locality=Toronto
stateprov=ON
country=CA
email=support@toronto.shifteight.org
phone=+14165551212
;
; Укажите адрес привязки и номер порта. По умолнчанию порт 4520.
;bindaddr=0.0.0.0
port=4520
entityid=FF:FF:FF:FF:FF:FF
ttl=32
autokill=yes
;secretpath=dundi
```

Идентификатор объекта, определенный в `entityid`, обычно является MAC-адресом интерфейса на компьютере. ID объекта по умолчанию является первым Ethernet-адресом сервера, но вы можете переопределить его с помощью `entityid`, если он установлен на MAC-адрес *некоторого* вашего устройства. Рекомендуется использовать MAC-адрес основного внешнего интерфейса. Этот адрес будет использоваться другими узлами для идентификации.

Поле time-to-live (`ttl`) определяет, на сколько прыжков удалены пиры, от которых мы получаем ответы, и используется для прерывания цикла. Каждый раз, когда запрос передается по линии, которой запрошенный номер не известен, значение в поле TTL уменьшается на единицу, подобно полю TTL пакета ICMP. Поле TTL определяет максимальное количество секунд, в течение которого мы готовы ждать ответа.

Когда вы запрашиваете поиск номера, исходный запрос (называемый `DPDISCOVER`) отправляется вашим пирам, запрашивающим этот номер. Если вы не получили подтверждение (`ACK`) вашего запроса (`DPDISCOVER`) в течение 2000 мс (достаточное время только для одной передачи) и `autokill` установлен в `yes`, Asterisk отправит пирам `CANCEL`. (Обратите внимание, что подтверждение не обязательно является ответом на запрос; это просто подтверждение того, что пир получил запрос.)

2 Файлы *dundi.conf* и *extensions.conf* должны быть настроены. Мы решили настроить *sip.conf* для целей адресной рекламы в нашей сети, но DUNDi является протоколом-агностиком, поэтому вместо этого можно использовать *iax.conf*, *h323.conf* или *mgcp.conf*. DUNDi просто выполняет поиск; стандартные методы совершения вызовов по-прежнему необходимы.

Цель `autokill` - предотвратить остановку поиска из-за хостов с высокой задержкой. В дополнение к параметрам `yes` и `no`, вы также можете указать количество миллисекунд ожидания.

Модуль `pbx_dundi` создает сменяющийся ключ и сохраняет его в локальной базе данных Asterisk (AstDB). `secret` - имя ключа хранится в семействе `dundi`. Значение ключа можно просмотреть с помощью команды *database show* в консоли Asterisk. Семейство баз данных можно переопределить с помощью параметра `secretpath`.

Нам нужен другой пир для взаимодействия, поэтому вот конфигурация для другого узла:

```
; Файл конфигурации DUNDi для Vancouver
;
[general]
;
department=IT
organization=vancouver.shifteight.org
locality=Vancouver
stateprov=BC
country=CA
email=support@vancouver.shifteight.org
phone=+16135551212
;
; Укажите адрес привязки и номер порта. По умолнчанию порт 4520.
;bindaddr=0.0.0.0
port=4520
entityid=00:00:00:00:00:00
ttl=32
autokill=yes
;secretpath=dundi
```

В следующем разделе мы создадим наши начальные узлы DUNDi.

Первоначальное определение пиров DUNDi

Пир (узел) DUNDi идентифицируется уникальным MAC-адресом 2 уровня интерфейса удаленной системы. Файл *dundi.conf* - это место, где мы определяем, в каком контексте искать пиры, запрашивающие поиск, и какие пиры мы хотим использовать при поиске для конкретной сети. Следующая конфигурация определена в *dundi.conf* в нашей системе Торонто:

```
[00:00:00:00:00:00] ; Удаленный офис Vancouver
model = symmetric
host = vancouver.shifteight.org
inkey = vancouver
outkey = toronto
qualify = yes
dynamic=yes
```

Идентификатор удаленного узла (MAC-адрес) заключен в квадратные скобки (`[]`). `inkey` и `outkey` - это пары открытого и закрытого ключей, которые мы используем для аутентификации. Пары ключей генерируются с помощью скрипта *astgenkey*, расположенного в каталоге исходников *~/src/asterisk-complete/asterisk/11/contrib/scripts*. Мы используем флаг *-n*, чтобы не инициализировать пароли при каждом запуске Asterisk:

```
$ cd /var/lib/asterisk/keys
$ sh ~/src/asterisk-complete/asterisk/11/contrib/scripts/astgenkey -n toronto
```

Мы разместим полученные ключи - *toronto.pub* и *toronto.key* в нашем каталоге */var/lib/asterisk/keys*. Файл *toronto.pub* - это открытый ключ, который мы разместим на веб-сервере, чтобы он был легко доступен для всех, с кем мы хотим соединяться. Когда мы сверяем, то можем дать нашим пирам

открытый ключ, доступный по HTTP, который они могут поместить в свои каталоги */var/lib/asterisk/keys* (используя что-либо наподобие *wget*).

На блоке Vancouver мы будем использовать следующую конфигурацию пира в *dundi.conf*:

```
[FF:FF:FF:FF:FF:FF] ; Удаленный офис Toronto
model = symmetric
host = toronto.shifteight.org
inkey = toronto
outkey = vancouver
qualify = yes
dynamic=yes
```

Затем мы выполним тот же скрипт *astgenkey* на блоке Vancouver для создания открытых и закрытых ключей *vancouver*. Наконец, мы разместим ключ *toronto.pub* на сервере Ванкувера в */var/lib/asterisk/keys* и поместим *vancouver.pub* на сервере Toronto в том же месте.

После загрузки ключей мы должны перезагрузить модули *res_crypto.so* и *pbx_dundi.so* в Asterisk:

```
toronto*CLI> module reload res_crypto.so
    -- Reloading module 'res_crypto.so' (Cryptographic Digital Signatures)
    -- Loaded PUBLIC key 'vancouver'
    -- Loaded PUBLIC key 'toronto'
    -- Loaded PRIVATE key 'toronto'

vancouver*CLI> module reload res_crypto.so
    -- Reloading module 'res_crypto.so' (Cryptographic Digital Signatures)
    -- Loaded PUBLIC key 'toronto'
    -- Loaded PUBLIC key 'vancouver'
    -- Loaded PRIVATE key 'vancouver'
```

Мы можем проверить ключи, поэтому мы знаем, что они готовы к загрузке в любое время с помощью команды CLI *keys show*:

```
*CLI> keys show
Key Name           Type     Status           Sum
------------------ -------- ---------------- --------------------------------
vancouver          PRIVATE  [Loaded]         c02efb448c37f5386a546f03479f7d5e
vancouver          PUBLIC   [Loaded]         0a5e53420ede5c88de95e5d908274fb1
toronto            PUBLIC   [Loaded]         5f806860e0c8219f597f876caa6f2aff

3 known RSA keys.
```

С ключами, загруженными в память, мы можем перезагрузить модуль pbx_dundi.so на обеих системах, чтобы сверить их вместе:

```
*CLI> module reload pbx_dundi.so
    -- Reloading module 'pbx_dundi.so' (Distributed Universal Number
       Discovery (DUNDi))
  == Parsing '/etc/asterisk/dundi.conf': Found
```



Иногда перезагрузка вызывает синхронизацию не так быстро, как мы хотели бы. Если вы столкнетесь с этим, то можете просто *module unload pbx_dundi.so* с обеих сторон, а затем на обеих системах нужно сделать новый запуск *module load pbx_dundi.so*.

Наконец, мы можем проверить, что системы успешно соединились с помощью *dundi show peers*:

```
toronto*CLI> dundi show peers
EID                 Host                 Port  Model       AvgTime     Status
00:00:00:00:00:00   172.16.0.104     (S) 4520  Symmetric   Unavail     OK (3 ms)
1 dundi peers [1 online, 0 offline, 0 unmonitored]
```

Теперь, когда наши пиры настроены и доступны, нам нужно создать контексты сопоставления, которые будут контролировать, какая информация будет возвращена в поиске.

Создание контекстов сопоставления

Файл *dundi.conf* определяет контексты DUNDi, которые сопоставляются с контекстами диалплана в файле *extensions.conf*. Контексты DUNDi - это способ определения различных и отдельных групп служб каталогов. Контексты в разделе `[mapping]` указывают на контексты в *extensions.conf*, который управляет номерами, которые вы рекламируете.

Когда вы создаете пир, то нужно определить, какие контексты сопоставления вы позволите этому пиру искать. Это делается с помощью инструкции `permit` (каждый узел может содержать несколько инструкций `permit`). Контексты сопоставления связаны с контекстами диалплана в том смысле, что они являются границей безопасности для ваших пиров. Мы включим наше отображение в следующем разделе.

Все контексты отображения DUNDi принимают форму:

```
dundi_context => local_context,weight,technology,destination[,options]]
```

Следующая конфигурация создает контекст сопоставления DUNDi, который мы будем использовать для объявления наших локальных добавочных номеров группе. Мы добавим эту конфигурацию к *dundi.conf* в системе Toronto под заголовком `[mappings]`. Обратите внимание, что все это должно отображаться в одной строке:

```
[mappings]
; Все в одной строке
extensions => RegisteredDevices,0,SIP,dundi:very_secret_secret@
toronto.shifteight.org/
${NUMBER},nopartial
Конфигурация системы Vancouver будет выглядеть следующим образом:
[mappings]
; Всё в одной строке
extensions => RegisteredDevices,0,SIP,dundi:very_secret_secret@
vancouver.shifteight.org/
${NUMBER},nopartial
```

В этом примере контекст сопоставления - это `extensions`, указывающийе на контекст `Registered Devices` в *extensions.conf* (предоставление списка добавочных номеров для ответа: наша телефонная книга). Номера, которые разрешаются для УАТС, должны быть объявлены с нулевым *`weight`* (подключены напрямую). Числа выше нуля указывают на увеличение числа переходов или путей для достижения конечного пункта назначения. Это полезно когда несколько ответов для одного и того же поиска получены в конце, который первоначально запросил номер; путь с меньшим весом будет предпочтительным. Мы рассмотрим, как контролировать ответы в разделе “Контроль ответов”.

Если мы можем ответить на поиск, наш ответ будет содержать метод, с помощью которого другой конец можно подключиться к системе. Он включает в себя технологию для использования (например, IAX2, SIP, H.323 и т.д.), имя пользователя и пароль для аутентификации, на какой хост отправить аутентификацию, и, наконец, добавочный номер.

Asterisk предоставляет некоторые ярлыки, позволяющие нам создать ”шаблон", с помощью которого мы можем создавать наши ответы. Для построения шаблона можно использовать следующие переменные канала:

`${SECRET}`

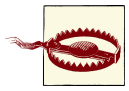
Заменяется паролем, хранящимся в локальной AstDB. Используется только с *iax.conf*.

`${NUMBER}`

Запрашиваемый номер.

`${IPADDR}`

IP-адрес для подключения.



Обычно безопаснее всего статически настраивать имя хоста, а не использовать переменную `${IPADDR}`. Переменная `${IPADDR}` иногда отвечает адресом в частном IP-пространстве, который недоступен из интернета.

Настроив наше сопоставление, давайте создадим простой контекст диалплана, в котором мы можем выполнять поиск для тестирования. Мы сделаем его более динамичным в разделе "Контроль ответов".

В *extensions.conf*, мы можем добавить следующее в обе системы:

```
[RegisteredDevices]
exten => 1000,1,NoOp()
```

С нашими настроенными диалпланом и сопоставлениями нам нужно загрузить их в память из CLI:

```
*CLI> dialplan reload

*CLI> module reload pbx_dundi.so
    -- Reloading module 'pbx_dundi.so' (Distributed Universal Number
       Discovery (DUNDi))
  == Parsing '/etc/asterisk/dundi.conf':   == Found
```

Мы можем проверить, что отображение было загружено в память с помощью команды *dundi show mappings*:

```
toronto*CLI> dundi show mappings
DUNDi Cntxt  Weight     Local Cntxt  Options    Tech  Destination
extensions   0          RegisteredDe NONE       SIP   dundi:${SECRET}@172.16.0.
```

Наш простой диалплан и сопоставления настроены, нам нужно определить какие сопоставления каждому из наших пиров разрешено использовать. Мы сделаем это в следующем разделе.

Использование контекстов сопоставления с пирами

В наших определениях сопоставлений в файле *dundi.conf* мы должны дать нашим пирам разрешение на их использование. Управление различными сопоставлениями осуществляется через парметры `permit`, `deny`, `include` и `noinclude` в определении пиров. Мы используем `permit` и `deny` для управления разрешено ли удаленному узлу искать определенное сопоставление в нашей локальной системе. Мы используем `include` и `noinclude` для управления, какие узлы мы будем использовать для выполнения поиска в определенном сопоставлении.

Поскольку у нас есть только одно определенное отображение (`extensions`), мы собираем `permit` и `include extensions` в наши определения пиров как в системе Торонто, так и в Ванкувере.

В Торонто мы разрешим Ванкуверу искать сопоставление `extensions` и использовать Ванкувер всякий раз, когда мы выполняем поиск в сопоставлении `extensions`:

```
[00:00:00:00:00:00] ; Удаленный офис Vancouver
model = symmetric
host = vancouver.shifteight.org
inkey = vancouver
outkey = toronto
qualify = yes
dynamic=yes
permit=extensions
include=extensions
```

Аналогично, мы включим `permit` и `include` сопоставление для `extensions` для офиса Торонто в системе Ванкувера:

```
[FF:FF:FF:FF:FF:FF] ; Удаленный офис Toronto
model = symmetric
host = toronto.shifteight.org
inkey = toronto
outkey = vancouver
qualify = yes
dynamic=yes
permit=extensions
include=extensions
```

После изменения пиров мы перезагружаем модуль pbx_dundi.so, чтобы изменения вступили в силу:

```
*CLI> module reload pbx_dundi.so
```

Конфигурацию `include` и `permit` можно проверить с помощью команды *dundi show peer* в CLI Asterisk:

```
*CLI> dundi show peer 00:00:00:00:00:00
Peer:       00:00:00:00:00:00
Model:      Symmetric
Host:       172.16.0.104
Port:       4520
Dynamic:    no
Reg:        No
In Key:     vancouver
Out Key:    toronto
Include logic:
-- include extensions
Query logic:
-- permit extensions
```

Теперь мы можем проверить наши поиски. Мы можем сделать это легко из CLI Asterisk, используя команду *dundi lookup*. Если выполним поиск из системы Ванкувера, то получим ответ от системы Торонто с адресом, который можем использовать для вызова. Мы добавили ключевое слово `bypass` в конец поиска, чтобы обойти кэш (в случае, если хотим выполнить несколько тестов):

```
vancouver*CLI> dundi lookup 1000@extensions bypass
  1.  0 SIP/dundi:very_secret_secret@172.16.0.161/1000 (EXISTS)
from ff:ff:ff:ff:ff:ff, expires in 3600 s
DUNDi lookup completed in 12 ms
```

Ответ `SIP/dundi:very_secret_secret@172.16.0.161/1000` дает нам адрес, который мы можем использовать для вызова внутреннего номера `1000`. (Конечно, мы не можем использовать этот адрес в данный момент, потому что мы не настроили никаких пиров в системе Торонто или Ванкувер для фактического приема вызова, но, по крайней мере, у нас есть часть поиска DUNDi, работающая сейчас!) В следующем разделе мы рассмотрим, как принимать вызовы в нашу систему после того, как нам ответили на запрос DUNDi.

Разрешение удаленных подключений

В файле *sip.conf* нам нужно включить пира, от которого мы можем принимать вызовы, и соответствующим образом обрабатывать вызовы этого пира в диалплане. Аутентификация выполняется с использованием пароля, как определено в сопоставлении в *dundi.conf*.



Если вы используете *iax.conf*, то можете использовать переменную `${SECRET}` в сопоставлении вместо пароля, который динамически заменяется сменяющимся ключом и

обновляется каждые 3600 секунд (1 час). Значение секретного ключа хранится в базе данных Asterisk и доступно с помощью параметра `dbsecret` в определении пира в *iax.conf*.

Вот определение пользователя для пользователя *dundi*, как определено в *sip.conf*:

```
[dundi]
type=user
secret=very_secret_secret
context=DUNDi_Incoming
disallow=all
allow=ulaw
allow=alaw
```

Запись `context - DUNDi_Incoming` - это место, где авторизованные абоненты входят в *extensions.conf*. Оттуда мы можем контролировать вызов так же, как и в диалплане любого другого входящего соединения.



Мы также можем использовать опции `permit` и `deny` для пира в *sip.conf* для управления IP-адресами, с которых будем принимать вызовы. Управление IP-адресами даст нам дополнительный уровень безопасности, если мы ожидаем вызовов только от известных конечных точек, например внутри нашей организации.

Обязательно перезагрузите *chan_sip.so*, чтобы включить вновь созданного пользователя в *sip.conf*:

```
toronto*CLI> sip reload
```

Чтобы принимать входящие вызовы определите контекст `[DUNDi_Incoming]` в *extensions.conf* и добавьте следующее в диалплан системы Toronto:

```
[DUNDi_Incoming]
exten => 1000,1,Verbose(2,Incoming call from the DUNDi peer)
   same => n,Answer()
   same => n,Playback(silence/1)
   same => n,Playback(tt-weasels)
   same => n,Hangup()
```

Перезагрузите диалплан с помощью *dialplan reload* после сохранения изменений в *extensions.conf*. Для нашего первого теста мы создадим номер в контексте `LocalSets` и попробуем позвонить на номер `1000`, используя информацию, предоставленную через DUNDi:

```
[LocalSets]
exten => 1000,1,Verbose(2,Test extension to place call to remote server)
   same => n,Dial(SIP/dundi:very_secret_secret@172.16.0.161/1000,30)
   same => n,Hangup()
```

Если мы перезагрузим диалплан и попробуем протестировать номер, набрав `1000`, мы должны быть подключены к подсказке *tt-weasels* на удаленной машине. С нашим пользователем, настроенным правильно для приема входящих вызовов, давайте сделаем наш диалплан и ответы более динамичными с некоторыми дополнительными инструментами.

Использование dbsecret с iax.conf

Если вы используете драйвер канала *iax.conf*, то можете аутентифицировать входящие вызовы, используя директиву `dbsecret` в *iax.conf* вместе с переменной `${SECRET}` в вашем сопоставлении. Использование переменной `${SECRET}` в сопоставлении вызывает отправку сменяющегося пароля в ответ, который затем может использоваться для аутентификации через IAX2. Вот пример определения аутентификации в файле *iax.conf*:

```
[dundi]
type=friend
context=DUNDi_Incoming
dbsecret=dundi/secret
disallow=all
allow=ulaw
allow=alaw
```

Пароль хранится в AstDB и сменяется каждые 3600 секунд (1 час). Чтобы использовать пароль в сопоставлениях, измените сопоставления в *dundi.conf* на использование `${SECRET}` вместо *`very_secret_secret`*. Это сопоставление, которое мы настроили в системе Ванкувер:

```
[mappings]
; Всё в одной строке
extensions => RegisteredDevices,0,SIP,dundi:${SECRET}@
vancouver.shifteight.org/
${NUMBER},nopartial
```

Контроль ответов

Ответы контролируются с помощью диалплана. Всякий раз, когда входящий запрос совпадает с дилпланом, настроенным для сопоставления (независимо от того, является ли запрос определенным номером или шаблоном), будет отправлен ответ. Если запрос не совпадает с диалпаном, ответ не отправляется. В примере, который мы строим, номер `1000` является единственным, который может быть сопоставлен и, таким образом, генерировать ответ.

В следующих разделах мы рассмотрим некоторые из методов, которые можно использовать для управления ответами на запросы.

Добавление ответов вручную

Файл *extensions.conf* обрабатывает, какие номера вы рекламируете и что вы делаете с вызовами, которые подключаются к ним.

Простой способ управления ответами - добавить их вручную в контекст `[RegisteredDevices]`. Если бы у нас было несколько номеров в одном из наших местоположений, мы могли бы добавить их все в этот контекст:

```
[RegisteredDevices]
exten => 1000,1,NoOp()
exten => 1001,1,NoOp()
exten => 1002,1,NoOp()
```

Здесь используется приложение `NoOp()`, поскольку сопоставление и ответ выполняются только по добавочному номеру, а диалплан не выполняется. Хотя мы можем перегрузить этот контекст и заставить его также быть местом назначения для наших вызовов, но это не рекомендуется. Другие причины использования приложения `NoOp()` должны стать ясными по мере продвижения.

Использование шаблонов совпадений

Конечно, добавление всех, кому мы хотим ответить вручную, было бы глупо, особенно если бы мы хотели рекламировать больший набор номеров, например все номера с кодом города. Как упоминалось ранее, в нашем примере мы могли бы позволить нашим офисам в Торонто и Ванкувере звонить друг от друга при совершении бесплатных или дешевых звонков из другого места.

Мы можем ответить всем по коду города, используя совпадения шаблонов, как и в других частях диалплана:

```
[RegisteredDevices]
exten => _416NXXXXXX,1,NoOp()
exten => _647NXXXXXX,1,NoOp()
exten => _905NXXXXXX,1,NoOp()
```

Мы также можем рекламировать полный или частичный диапазон номеров, используя совпадения шаблонов:

```
[RegisteredDevices]
exten => _1[1-3]XX,1,NoOp()   ; extensions 1100->1399

exten => _1[7-9]XX,1,NoOp()   ; extensions 1700->1999
```

Совпадения шаблонов - хороший способ добавления диапазонов чисел, но они все еще статичны. В следующем разделе мы рассмотрим, как можно добавить некоторую текучесть в контекст `RegisteredDevices`.

Динамическое добавление добавочных номеров

В некоторых случаях мы можем рекламировать только номера в нашем местоположении, которые в настоящее время зарегистрированы в системе. Возможно, у нас есть продавец, который летает между офисами в Торонто и Ванкувере, подключает свой ноутбук к сети и регистрируется в любом месте, где он сейчас находится. В этом случае мы хотели бы удостовериться в том, что звонки этому человеку направляются в соответствующий офис, с тем чтобы избежать ненужной отправки звонков по всей стране.

Параметры `regcontext` и `regexten` в *iax.conf* и *sip.conf* полезны в этом деле. При регистрации пира значение, определенное `regexten` для зарегистрированного пира, приведет к заполнению того же значения внутри контекста, как определено атрибутом `regcontext`. Так, например, если мы определяем `regcontext` в разделе `[general]` *sip.conf*, чтобы он содержал `RegisteredDevices`, и определяем `regexten` для каждого пира, чтобы он содержал добавочный номер этого пира, когда пиры регистрируются, контекст `RegisteredDevices` будет заполнен автоматически для нас. Мы изменим наш *sip.conf*, чтобы он выглядел так:

```
[general]
regcontext=RegisteredDevices

[0000FFFF0001](office-phone)
regexten=1001
```

а теперь перезагрузим *chan_sip.so*.
Теперь мы зарегистрируем наше устройство в системе и посмотрим на контекст `RegisteredDevices`:

```
*CLI> dialplan show RegisteredDevices
[ Context 'RegisteredDevices' created by 'SIP' ]
  '1001' => 1. Noop(0000FFFF0001)   [SIP]
  '1002' => 1. Noop(0000FFFF0002)   [SIP]
```

С нашими зарегистрированными устройствами и контекстом, используемым для определения когда заполнять ответ, единственная оставшаяся задача - включить контекст `LocalSets` в контекст `DUNDi_Incoming` чтобы разрешить маршрутизацию вызовов к конечным точкам.

Использование функций диалплана в сопоставлениях

Иногда полезно использовать функцию диалплана в сопоставлениях для управления ответами пира. На протяжении всей этой книги мы расхваливали преимущества отделения добавочного номера пользователя от устройства, чтобы разрешить hot-desking. Поскольку другой конец просто собирается запросить добавочный номер и не обязательно будет знать местоположение устройства в нашей

системе, мы можем использовать функции DB() и DB_EXISTS() в сопоставлении для выполнения поиска из нашей AstDB для вызова устройства.[3]



В предыдущих версиях Asterisk максимальная длина поля destination (см. "Создание контекстов сопоставления") составляла 80 символов, что делало использование вложенных функций диалплана практически невозможным. В настоящее время максимальная длина - 512 знаков.

Во-первых, мы должны убедиться, что наша база данных заполнена информацией, которой мы могли бы ответить. Хотя это обычно делается диалпланом, написанным для реализации hot-desking, мы просто добавим содержимое непосредственно из консоли Asterisk для демонстрационных целей:

```
*CLI> database put phones 1001/device 0000FFFF0001
Updated database successfully
```

При заполнении базы данных нам нужно изменить наше сопоставление, чтобы использовать некоторые функции диалплана, которые будут принимать запрошенное значение, выполнять поиск в нашей базе данных для этого значения и возвращать значение. Если в базе данных нет значения, мы вернем значение None.

Наше существующее сопоставление выглядит следующим образом:

```
[mappings]
; Сопоставление должно быть в одну строку
extensions => RegisteredDevices,0,SIP,
   dundi:very_secret_secret@toronto.shifteight.org/${NUMBER},nopartial
```

Наш текущий пример просто отражает тот же добавочный номер, который был запрошен, вместе с некоторой информацией аутентификации. Запрашиваемое число - это номер, который ищет пир. Однако, поскольку мы используем hot-desking, добавочный номер телефона может быть в разных местах, поэтому мы можем вернуть идентификатор устройства напрямую.[4] Мы можем сделать это с использованием функций диалплана в нашем ответе. Хотя у нас может не быть полной мощности диалплана (нескольких линий, сложной логики и т.д.) в нашем распоряжении мы можем, по крайней мере, использовать некоторые из более простых функций диалплана, таких как DB(), DB_EXISTS() и IF().

Мы собираемся заменить ${NUMBER} следующим битом логики диалплана:

```
; все это должно быть в одной строке
${IF($[${DB_EXISTS(phones/${NUMBER}/device)}]?
${DB(phones/${NUMBER}/device)}:None)}
```

Если мы сломаем это, то получим оператор IF(), который вернет либо истину, либо ложь. Если ложь, то возвращается значение None. Если истина, мы возвращаем значение, расположенное в базе данных в phones/${NUMBER}/device (где ${NUMBER} содержит значение 1001 для нашего примера), используя функцию DB(). Чтобы определить какое значение будет возвращать функция IF(), мы используем функцию DB_EXISTS(). Эта функция проверяет, существует ли значение в phones/${NUMBER}/device в AstDB и возвращает 1 или 0 (истина или ложь).



Функция DB_EXISTS() не только возвращает 1 или 0, но и задает переменную канала ${DB_RESULT}, содержащую значение внутри базы данных, если возвращаемое значение равно 1. Однако мы не можем использовать это значение, потому что функция IF() оценивается до поля условия, что означает, что ${DB_RESULT} будет пустым. Таким образом, нам нужно использовать функцию DB() для поиска значения до оцениваемого поля условия.

3 ...или func_odbc, или func_curl, или res_ldap (используя функцию REALTIME_FIELD()).

4 Мы также рассмотрели использование функций GROUP() и GROUP_COUNT() для поиска текущего использования канала в удаленной системе, чтобы определить, в какое место маршрутизировать вызовы (с самой низкой занятостью канала) в качестве простого балансировщика нагрузки.

После перезагрузки *pbx_dundi.so* из консоли (*module reload pbx_dundi.so*), мы можем выполнить поиск с другого сервера и проверить результат:

```
vancouver*CLI> dundi lookup 1001@extensions bypass
  1.  0 SIP/dundi:very_awesome_password/0000FFFF0001 (EXISTS)
from ff:ff:ff:ff:ff:ff, expires in 3600 s
DUNDi lookup completed in 77 ms
```

С помощью функций диалплана вы можете сделать ответы в своих диалпланах намного более динамичными. В следующем разделе мы рассмотрим, как вы можете выполнять эти поиски из диалплана с помощью функций `DUNDILOOKUP()`, `DUNDIQUERY()` и `DUNDIRESULT()`.



При выполнении поиска с помощью примера в этой главе, поскольку все пиры в вашей сети будут возвращать результат (`None` или значение которое вы хотите), вам нужно будет использовать функции `DUNDIQUERY()` и `DUNDIRESULT()` для анализа списка возвращаемых результатов. Альтернативой было бы попробовать позвонить `SIP/dundi:very_long_pass@remote_server/None` но это будет не очень эффективно. Возможно, вы даже захотите обработать расширение `None` элегантно, если оно будет вызвано.

Выполнение поиска из диалплана

Выполнение поиска из диалплана - это действительно хлеб и масло всего этого, потому что это позволяет использовать более динамическую маршрутизацию изнутри диалплана. С помощью DUNDi вы можете выполнять поиск и маршрутизацию вызовов в кластере с помощью функций `DUNDILOOKUP()` или `DUNDIQUERY()` и `DUNDIRESULT()`.

Функция `DUNDILOOKUP()` заменяет старое приложение диалплана `DUNDiLookup()`, выполняя почти те же функции. С помощью `DUNDILOOKUP()` вы выполняете поиск как в консоли Asterisk, и результат может быть сохранен в переменной канала или использоваться везде, где вы можете использовать функцию диалплана. Вот пример:

```
[TestContext]
exten => 1001,1,Verbose(2,Look up extension 1001)
   same => n,Set(DUNDi_Result=${DUNDILOOKUP(1001,extensions,b)})
   same => n,Verbose(2,The result of the lookup was ${DUNDi_Result})
   same => n,Hangup()
```

Аргументы, переданные `DUNDILOOKUP()`: *`extension,context,options`*. Для функции `DUNDILOOKUP()` доступен только один параметр `b`, который используется для обхода локального кэша. Преимущество использования функции `DUNDILOOKUP()` заключается в простоте и удобстве в использовании. Недостатком является то, что она будет устанавливать только первое возвращаемое значение; если возвращается несколько значений, они будут отброшены.



Вы не всегда захотите использовать опцию обхода при выполнении поиска, потому что использование кэша - это то, что снизит количество запросов по вашей сети и ограничит количество необходимых ресурсов. Мы используем его в наших примерах просто потому, что он полезен для целей тестирования, поэтому знаем что каждый раз возвращаем результат, а не просто кэшированное значение из предыдущего поиска.

Чтобы проанализировать несколько возвращаемых значений, нам нужно использовать функции `DUNDIQUERY()` и `DUNDIRESULT()`. Каждый из них играет важную роль в просеивании нескольких возвращаемых значений из поиска. Функция `DUNDIQUERY()` выполняет начальный поиск и сохраняет полученный хэш в память. Затем возвращается значение ID, которое можно сохранить в переменной канала. Затем значение ID, возвращаемое функцией `DUNDIQUERY()`, может быть передано функции `DUNDIRESULT()` для анализа возвращаемых значений из запроса.

Давайте посмотрим на некоторый диалплан, который использует эти функции:

```
[TestContext]
exten => _1XXX,1,Verbose(2,Looking up results for extension ${EXTEN})

; Выполнить поиск и сохранить полученный ID в DUNDI_ID
   same => n,Set(DUNDI_ID=${DUNDIQUERY(${EXTEN},extensions,b)})
   same => n,Verbose(2,Showing all results returned from the DUNDi Query)

; Функция DUNDIRESULT() может возвращать количество результатов с помощью
'getnum'
   same => n,Set(NumberOfResults=${DUNDIRESULT(${DUNDI_ID},getnum)})
   same => n,Set(ResultCounter=1)

; Если результат меньше 1, результаты не возвращаются
   same => n,GotoIf($[0${NumberOfResults} < 1]?NoResults,1)

; Начало цикла, показывающего возвращаемые значения
   same => n,While($[0${ResultCounter} <= ${NumberOfResults}])

; Сохраните возвращаемый результат в позиции ${ResultCounter} в thisResult
   same => n,Set(thisResult=${DUNDIRESULT(${DUNDI_ID},${ResultCounter})})

; Показать текущий результат в консоли
   same => n,Verbose(2,One of the results returned was: ${thisResult})

; Увеличить счетчик на единицу
   same => n,Set(ResultCounter=${INC(ResultCounter)})

; Конец нашего цикла
   same => n,EndWhile()
   same => n,Playback(silence/1)
   same => n,Playback(vm-goodbye)
   same => n,Hangup()

; Если результат не найден - выполнить этот диалплан
exten => NoResults,1,Verbose(2,No results were found)
   same => n,Playback(silence/1)
   same => n,Playback(invalid)
   same => n,Hangup()
```

В нашем примере диалплан выполняет поиск с помощью функции `DUNDIQUERY()` и сохраняет результирующее значение ID в переменной канала `DUNDI_ID`. Используя функцию `DUNDIRESULT()` и параметр `getnum` мы храним общее количество возвращаемых результатов в переменной канала `NumberOfResults`. Затем мы устанавливаем переменную канала `ResultCounter` в `1` в качестве нашей начальной позиции в цикле.

Используя `GotoIf()` проверяем, меньше ли возвращаемого `${NumberOfResults}` и, если да, переходим к расширению `NoResults`, где мы делаем `Playback()` "Invalid Extension". Если хотя бы один номер расширение будет найден, мы продолжим в диалплане.

Используя приложение `While()`, мы проверяем, меньше или равно `${ResultCounter}` значения `${NumberOfResults}`. Если это правда - мы продолжим в диалплане, а в противном случае перейдем к приложению `EndWhile()`.

Для каждой итерации нашего цикла функция `DUNDIRESULT()` используется для сохранения значения в позиции `${ResultCounter}` в переменную канала `thisResult`. После сохранения значения мы выводим его в консоль Asterisk с помощью приложения `Verbose()`. После этого увеличиваем значение `ResultCounter` на единицу, используя функцию `INC()`. Затем наш цикл теста выполняется

снова в цикле While() и будет продолжаться, пока значение ${ResultCounter} меньше или равно значения ${NumberOfResults}.

Используя тот же тип логики, мы могли бы проверить значения, отличные от None и, если такое значение найдено, выполнить ExitWhile() и продолжить в диалплане для выполнения вызова конечной точки. Логика диалплана может выглядеть примерно так:

```
[subLookupExtension]
exten => _1XXX,1,Verbose(2,Looking up results for extension ${EXTEN})

; Выполнить поиск и сохранить полученный ID в DUNDI_ID
   same => n,Set(DUNDI_ID=${DUNDIQUERY(${EXTEN},extensions,b)})
   same => n,Set(NumberOfResults=${DUNDIRESULT(${DUNDI_ID},getnum)})
   same => n,Set(ResultCounter=1)

; Если результат не найден, вернуть 'None'
   same => n,GotoIf($[0${NumberOfResults} < 1]?NoResults,1)

; Выполнить нашу петлю
   same => n,While($[0${ResultCounter} <= ${NumberOfResults}])

; Получить текущее значение
   same => n,Set(thisResult=${DUNDIRESULT(${DUNDI_ID},${ResultCounter})})

; Если текущее возвращаемое значение не равно None, у нас есть
; результирующее местоположение для вызова, и мы можем выйти из цикла
   same => n,ExecIf($["${thisResult}" != "None"]?ExitWhile())
; Если мы зашли так далеко, но еще не возвращено значение, которое мы хотим
; использовать, увеличте счетчик и попробуйте следующее значение.
   same => n,Set(ResultCounter=${INC(ResultCounter)})

; Конец нашей петли
   same => n,EndWhile()

; Мы находимся здесь, потому что дошли до конца цикла или нашли
; значение, которое хотим вернуть. Проверьте, что это оно. Если у нас просто
; закончились значения, вернуть "None".
;
   same => n,GotoIf($["${thisResult}" = "None"]?NoResults,1)

; Если мы находимся здесь, у нас есть значение, которое хотим вернуть.
   same => n,Return(${thisResult})

; Если не было приемлемых результатов, вернуть значение "None"
exten => NoResults,1,Verbose(2,No results were found)
   same => n,Return(None)
```

С функциями DUNDIQUERY() и DUNDIRESULT() у вас есть много возможностей контролировать как обрабатывать возвращаемые результаты и выполнять логику маршрутизации с этими значениями.

Вывод

В этой главе мы рассмотрели, как протокол DUNDi помогает выполнять поиск других систем Asterisk в кластере, выполнять динамическую маршрутизацию. С помощью DUNDi вы можете взять несколько систем и контролировать когда и где вызовы совершаются в них, предоставляя возможности обхода дороговизны оплаты вызовов и даже предоставляя своим сотрудникам возможность перемещаться между физическими местоположениями, а также ограничивать количество внесистемных переходов, которые должен совершить вызов, чтобы найти их.

В то время как первоначальное намерение DUNDi состояло в том, чтобы помочь нам мигрировать из централизованных служб каталогов — намерение, которое еще не принесло результатов — DUNDi является чрезвычайно эффективным и полезным инструментом, который можно использовать в организациях для рекламы и маршрутизации вызовов динамически между системами в облачной среде. DUNDi - это инструмент, который дает большую власть администраторам Asterisk, желающим создать распределенную сеть.

ГЛАВА 24

Системный мониторинг и журналирование

Хаос присущ всем сложным вещам.
Стремитесь с усердием.
- Будда

Asterisk поставляется с несколькими подсистемами, которые позволяют получить детальную информацию о работе вашей системы. Для устранения неполадок или для ведения отчетов или в целях укомплектации, разные модули мониторинга Asterisk могут помочь вам следить за внутренней работой вашей системы.

logger.conf

При устранении неполадок в системе Asterisk, вы найдете очень полезной возможность обратиться к записям истории о том, что происходило в системе в момент появления проблем. Параметры для хранения этой информации определяются в */etc/asterisk/logger.conf*.

В идеале, хотелось бы чтобы система сохраняла запись всего, что она делает. Однако, это очень затратно. На загруженной системе, с полностью включенным протоколированием, можно полностью заполнить жесткий диск данными в течение одного дня или около того. Таким образом, нужен баланс между детализацией и местом хранения.

Файл */etc/asterisk/logger.conf* позволяет вам определить различные уровни журналирования для многих файлов, если нужно. Он обладает великолепной гибкостью, но это может привести к запутанности.

Формат записей в файле *logger.conf* следующий:

```
filename => type[,type[,type[,...]]]
```

Это простой файл *logger.conf*, который идет с исходниками Asterisk, но не стоит просто копировать файл примера, мы рекомендуем использовать следующее в вашем начальном файле *logger.conf*:

```
[general]

[logfiles]
console => notice,warning,error,dtmf
messages => notice,warning,error
;verbose => notice,warning,error,verbose
```

После сохранения файла, вам нужно перезагрузить logger используя следующую команду из оболочки:

```
$ asterisk -rx 'logger reload'
```

или из Asterisk CLI:

```
*CLI> logger reload
```

Детализация журналирования: полезна, но опасна

Мы боролись с желанием рекомендовать добавление следующей строки в ваш *logger.conf*:

```
verbose => notice,warning,error,verbose
```

Вполне возможно, это является одним из самых полезных средств отладки при сборке и устранении неисправностей диалплана и, поэтому, настоятельно рекомендуется. Опасность в том, что если вы забудете отключить эту директиву после отладки, то вы заложите бомбу замедленного действия в вашу систему Asterisk, которая будет медленно заполнять жесткий диск и убьет вашу систему в один прекрасный день, когда вы меньше всего этого ожидаете.

Используйте это. Это фантастика. Но помните, что это нужно отключить после завершения!

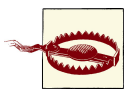
Вы можете указать любое имя файла, но специальное имя файла *console* фактически будет производить вывод в Asterisk CLI, а не в любой файл на жестком диске. Все остальные имена будут храниться в файловой системе в каталоге */var/log/asterisk*. Типы *logger.conf* описаны в Таблице 24-1.

Таблица 24-1. Типы logger.conf

Тип	Описание
`notice`	Вы увидите много сообщений во время перезагрузки, а также они будут приходить во время обычных вызовов. `notice` просто любое событие, о котором Asterisk хотел бы проинформировать вас.
`warning`	`Warning` (предупреждение) представляет собой проблему, которая может быть достаточно серьезной чтобы повлиять на вызов (в том числе отключение вызова, потому что поток вызова не может продолжаться). Предупреждения нужно устранять.
`error`	`Errors` (ошибки) представляют собой значительные проблемы в системе, которые должны быть решены немедленно.
`debug`	Отладка полезна только если вы пытаетесь устранить проблемы с кодом Asterisk. Вы не будете использовать отладку для устранения проблем в диалплане, но вы должны использовать её если разработчики Asterisk попросили вас предоставить журнал для отчетности о проблеме. Не используйте отладку в продакшене, поскольку количество сохраненных записей может заполнить жесткий диск в течение нескольких дней.[a]
`verbose`	Это один из самых полезных типов протоколирования, но он также является одним из наиболее рискованных; если его оставить без присмотра, то велика вероятность заполнения жесткого диска.[b]
`dtmf`	Журналирование DTMF может быть полезно если вы получаете жалобы, что вызовы неправильно маршрутизируются от автосекретаря.
`fax`	Этот тип протоколирования обрабатывает сообщения связанные с технологией факса (`res_fax_spandsp` или `res_fax_digium`) сохраняемые факс-регистратором.
`*`	Будет протоколировать ВСЁ (EVERYTHING) (и мы подразумеваем всё). Не используйте это, если вы не представляете последствия хранения такого объема данных. Ничем хорошим это не закончится.

[a] Это не теория. Это произошло с нами и было невесело.

[b] Это не так рискованно как при отладке, так как займет месяцы пока жесткий диск заполнится, но опасность в том, что это произойдет, скажем через год, когда вы находитесь в отпуске летом и сразу не будет очевидно в чем проблема.



Существует особенность у системы логирования Asterisk, которая может вызвать у вас некоторое замешательство, если вы не знаете о ней. Уровень журналирования для типов `verbose` и `debug` связан с детализацией, установленной в консоли. Это означает, что если

вы пишете лог в файл с типом `verbose` или `debug`, а кто-то зашел в консоль CLI и дает команду *core set verbose 0*, или *core set debug 0* журналирование в ваш файл журнала будет остановлено.

Просмотр журналов Asterisk

Поиск в лог-файлах может стать проблемой. Хитрость заключается в том, чтобы иметь возможность фильтровать то, что вы видите так, что бы показывалась вам только информация, которая имеет отношение к тому, что вы ищете.

Для начала, вам нужно иметь приблизительное представление о времени, когда возникла проблема. После того как вы узнали приблизительное время, нужно найти улики, которые помогут определить вызов в запросе. Очевидно, чем больше у вас информации о вызове, тем быстрее вы сможете его найти.

В Asterisk 11 введена функция ведения журнала, которая помогает с отладкой конкретного вызова. Протоколирование записей связанных с вызовом в настоящее время включает идентификатор вызова (call ID) в записи журнала. Этот call ID может использоваться с *grep*, чтобы найти все записи, связанные с этим вызов. В следующем примере журнал с идентификатором вызова `C-00000004`.

```
[Dec 4 08:22:32] WARNING[14199][C-00000004]: app_voicemail.c:6286
leave_voicemail: No entry in voicemail config file for '234123452'
```

В более ранних версиях Asterisk, есть еще одна уловка, которую вы можете использовать. Если, например, вы делаете детальное ведение журнала следует отметить, что каждый отдельный вызов имеет идентификатор потока, который при использовании *grep* часто может помочь вам отфильтровать все, что не относится к вызову. Например, в следующем подробном журнале у нас есть несколько вызовов, и поскольку вызовы происходят одновременно, трассировка одного вызова может быть очень запутанной:

```
$ tail -1000 verbose
[Mar 11 …] VERBOSE[31362] logger.c: -- IAX2/shifteight-4 answered Zap/1-1
[Mar 11 …] VERBOSE[2973] logger.c: -- Starting simple switch on 'Zap/1-1'
[Mar 11 …] VERBOSE[31362] logger.c: == Spawn extension (shifteight, s, 1)
exited non-zero on 'Zap/1-1'
[Mar 11 …] VERBOSE[2973] logger.c: -- Hungup 'Zap/1-1'
[Mar 11 …] VERBOSE[3680] logger.c: -- Starting simple switch on 'Zap/1-1'
[Mar 11 …] VERBOSE[31362] logger.c: -- Hungup 'Zap/1-1'
```

Для фильтрации одного вызова, мы можем произвести *grep* в потоке ID. Например:

```
$ grep 31362 verbose
```

который дал бы нам следующее:

```
[Mar 11 …] VERBOSE[31362] logger.c: -- IAX2/shifteight-4 answered Zap/1-1
[Mar 11 …] VERBOSE[31362] logger.c: == Spawn extension (shifteight, s, 1)
exited non-zero on 'Zap/1-1'
[Mar 11 …] VERBOSE[31362] logger.c: -- Hungup 'Zap/1-1'
```

Этот метод не гарантирует, что вы будете видеть все, что касается одного звонка, так как теоретически вызов может порождать дополнительные потоки, но для отладки базового диалплана мы находим такой подход весьма полезным.

Логирование демоном Linux syslog

Linux содержит очень мощный движок логирования, которым Asterisk способен воспользоваться. Сейчас обсуждаются различные варианты *syslog* и описание всех возможных способов логирования

Asterisk выходит за рамки этой книги, достаточно сказать, что если вы хотите вести протокол Asterisk демоном *syslog*, вам просто необходимо указать следующую строку в файле */etc/asterisk/logger.conf*:

```
syslog.local0 => notice,warning,error     ; или любые типы, которые хотите
                                          ; регистрировать
```

Вы должны иметь запись в вашем конфигурационном файле[1] системного журнала (*syslog*) с именем `local0`, которая должна выглядеть примерно так:

```
local0.*          /var/log/asterisk/syslog
```



Вы можете для этого использовать `local0` до `local7`, но проверьте ваш *syslog.conf* чтобы убедиться что ничего другого не использует один из этих каналов *syslog*.

syslog[2] это гораздо более мощный инструмент регистрации, но он требует больше знаний, чем логирование Asterisk напрямую в файлы.

Проверка логирования

Вы можете просмотреть статус всех ваших настроек *logger.conf* через консоль Asterisk CLI используя команду:

```
*CLI> logger show channels
```

Вы должны увидеть примерно следующее:

```
Channel                    Type      Status      Configuration
-------                    ----      ------      -------------
syslog.local0              Syslog    Enabled     - NOTICE WARNING ERROR VERBOSE
/var/log/asterisk/verbose  File      Enabled     - NOTICE WARNING ERROR VERBOSE
/var/log/asterisk/messages File      Enabled     - NOTICE WARNING ERROR
                           Console   Enabled     - NOTICE WARNING ERROR DTMF=
```

Ротация логов

Существует некоторая поддержка ротации, встроенная в Asterisk, когда Asterisk протоколирует в файлы. Ротация журналов будет выполнена в следующих случаях:

- Если вы запустили ротацию журналов в Asterisk CLI командой.

  ```
  *CLI> logger rotate
  ```

- Во время перезагрузки конфигурации, если существующие файлы журнала больше 1 Гб.
- Если Asterisk получает сигнал `SIGXFSZ`, указывающий что файл для записи слишком велик.

Call Detail Records (CDR) — запись деталей вызовов

Система CDR в Asterisk используется для логирования истории вызовов в системе. В некоторых решениях эти записи используются для биллинга. В других — запись вызовов используется для анализа количества вызовов во времени. Её можно также использовать как инструмент отладки администратора Asterisk.

Содержание CDR

CDR имеет несколько полей, которые включены по умолчанию. Таблица 24-2 приводит их список.

1 Который обычно находится в */etc/syslog.conf*.
2 И *rsyslog*, *syslog-ng* и все остальное.

Таблица 24-2. Поля CDR по умолчанию

Опция	Значение/Пример	Примечание
accountcode	12345	Идентификатор учетной записи. Это поле определяется пользователем и пустое по умолчанию.
src	12565551212	ID вызывающего абонента. Он устанавливается автоматически и доступен только для чтения.
dst	102	Добавочный номер назначения для вызова. Это поле устанавливается автоматически и доступно только для чтения.
dcontext	PublicExtensions	Контекст назначения для вызова. Это поле устанавливается автоматически и доступно только для чтения.
clid	"Big Bird" <12565551212>	Полный идентификатор вызывающего абонента, включая имя вызывающего абонента. Это поле устанавливается автоматически и доступно только для чтения.
channel	SIP/0004F2040808-a1bc23ef	Канал вызывающей стороны. Это поле устанавливается автоматически и доступно только для чтения.
dstchannel	SIP/0004F2046969-9786b0b0	Канал вызываемой стороны. Это поле устанавливается автоматически и доступно только для чтения.
lastapp	Dial	Последнее приложение диалплана, которое было выполнено. Это поле устанавливается автоматически и доступно только для чтения.
lastdata	SIP/0004F2046969,30,tT	Аргументы переданные в lastapp. Это поле устанавливается автоматически и доступно только для чтения.
start	2010-10-26 12:00:00	Время начала вызова. Это поле устанавливается автоматически и доступно только для чтения.
answer	2010-10-26 12:00:15	Время ответа на вызов. Это поле устанавливается автоматически и доступно только для чтения.
end	2010-10-26 12:03:15	Время окончания вызова. Это поле устанавливается автоматически и доступно только для чтения.
duration	195	Количество секунд между началом вызова и окончанием. Это поле устанавливается автоматически и доступно только для чтения.
billsec	180	Количество секунд между ответом на вызов и окончанием вызова. Это поле устанавливается автоматически и доступно только для чтения.
disposition	ANSWERED	Индикатор того, что случилось с вызовом. Это может быть NO ANSWER, FAILED, BUSY, ANSWERED или UNKNOWN.
amaflags	DOCUMENTATION	Automatic Message Accounting (AMA) флаг связанный с этим вызовом. Он может быть одним из следующего: OMIT, BILLING, DOCUMENTATION или Unknown.
userfield	PerMinuteCharge:0.02	Поле пользователя общего назначения. По умолчанию это поле пусто и может быть задано в виде пользовательской строки.[a]
uniqueid	1288112400.1	Уникальный ID канала src. Это поле устанавливается автоматически и доступно только для чтения.

[a] userfield не так актуально в настоящее время, как это было раньше. Пользовательские переменные CDR являются более гибким способом получить пользовательские данные в CDR.

Все поля записи CDR можно получить в диалплане Asterisk с помощью функции CDR(). Функция CDR() также используется для установки полей CDR, которые определяются для пользователя.

```
exten => 115,1,Verbose(Call start time: ${CDR(start)})
   same => n,Set(CDR(userfield)=zombie pancakes)
```

В дополнение к полям, которые всегда включены в CDR, можно добавлять пользовательские поля. Это делается в диалплане с помощью приложения Set() функцией CDR():

```
exten => 115,1,NoOp()
```

```
same => n,Set(CDR(mycustomfield)=coffee)
same => n,Verbose(I need some more ${CDR(mycustomfield)})
```



Если вы решите использовать пользовательские переменные CDR убедитесь что реализация CDR способна их регистрировать.

Для просмотра встроенной документации по функции `CDR()`, выполните следующую команду в консоли Asterisk:

```
*CLI> core show function CDR
```

В дополнение к функции `CDR()`, есть приложения в диалплане, которые могут использоватся для обработки CDR записей. Мы рассмотрим эти приложения далее.

Приложения диалплана

Есть несколько приложений диалплана, которые могут обрабатывать CDR для текущего вызова. Для получения списка приложений CDR, которые загружаются в текущую версию Asterisk, мы можем использовать следующую команду CLI:

```
*CLI> core show applications like CDR
    -= Matching Asterisk Applications =-
                ForkCDR: Forks the Call Data Record.
                  NoCDR: Tell Asterisk to not maintain a CDR for the current call
              ResetCDR: Resets the Call Data Record.
    -= 3 Applications Matching =-
```

Каждое приложение имеет встроенную в Asterisk документацию, которую можно просмотреть с помощью следующей команды:

```
*CLI> core show application <application name>
```

cdr.conf

Файл *cdr.conf* имеет раздел `[general]`, который содержит параметры, применяемые ко всей системе CDR. Дополнительные необязательные разделы могут существовать в этом файле, они применяются к конкретному логированию CDR модулей. Таблица 24-3 приводит список доступных опций раздела `[general]`.

Таблица 24-3. Раздел cdr.conf [general]

Опция	Значение/Пример	Примечание
`enable`	`yes`	Включает логирование CDR. По умолчанию — `yes`.
`unanswered`	`no`	Протоколирует неотвеченные вызовы. Обычно только отвеченные вызовы попадают в CDR. Протоколирование всех попыток вызова может привести к большому числу дополнительных записей о вызовах, а в большинстве своем они не нужны. По умолчанию - `no`.
`end before hexten`	`no`	Закрывает вывод CDR до запуска расширения `h` в диалплане Asterisk. Обычно CDR не прекращается пока диалплан полностью не завершит работу. По умолчанию - `no`.
`initiated seconds`	`no`	При вычислении поля `billsec` всегда округляется вверх. Например, если разница между ответом и завершением составляет 1 секунду и 1 микросекунду, `billsec` будет установлен на 2 секунды. Это помогает гарантировать поведение CDR Asterisk аналогичному поведению телекоммуникационных компаний. По умолчанию - `no`.

batch	no	Очередь записей CDR будет протоколирована в пакетах, а не синхронизирована по завершению каждого вызова. Это предотвращает протоколирование CDR блокированных завершенных вызовов с разрушенным процессом Asterisk. Использование режима batch может быть невероятно полезно при работе с базой данных, которая может быть медленной для обработки запросов. Значение по умолчанию - no, но мы рекомендуем его включить.[a]
size	100	Устанавливает число записей CDR, находящихся в очереди прежде чем они запишутся в пакетном режиме. По умолчанию значение - 100.
time	300	Устанавливает максимальное количество секунд, которое CDR записи будут ждать в очереди, прежде чем пакет будет сохранен. Процесс регистрации пакета CDR будет выполнен по завершению этого периода времени, даже если размер не был достигнут. Значение по умолчанию составляет 300 секунд.
scheduler only	no	Устанавливает когда процесс пакетной обработки CDR должен порождать новый поток или запланировано новое содержимое пакета CDR. Значение по умолчанию - no и мы рекомендуем не менять его.
safe shutdown	yes	Блокирует выключение Asterisk, чтобы убедиться, что все в очереди записей CDR сохранены в журнал. По умолчанию - yes и мы рекомендуем оставить его таким, так как эта опция предотвращает потери важных данных.

[a] Недостатком включения этой опции является тот момент, когда Asterisk слетает или умирает по какой-либо причине, записи CDR будут потеряны, так как они хранятся только в памяти, а процесса Asterisk уже не существует. См. safeshutdown для получения дополнительной информации.

Конечные решения (Backends)

Модули конечных решений Asterisk CDR реализуют различные пути логирования CDR. Большинство конечных решений CDR требуют определенной конфигурации для своего запуска.

cdr_adaptive_odbc

Как следует из названия модуль cdr_adaptive_odbc позволяет сохранять CDR в базе данных через ODBC. Часть имени "adaptive" указывает на то, что он приспосабливается к структуре таблицы: нет статической структуры таблицы, которая должна быть использована с этим модулем. При загрузке модуля (или перезагрузке) он читает структуру таблицы. При логировании CDR, он ищет переменную CDR, которая соответствует каждому имени столбца. Это относится как к встроенным CDR переменным, так и пользовательским. Если вы хотите логировать CDR переменные канала, просто создайте столбец с названием channel.

Добавление содержания пользовательских CDR это так просто, как установить их в диалплане. Например, если мы хотим логировать User-Agent предоставляемый устройством SIP, мы можем добавит пользовательскую переменную CDR:

```
exten => 105,n,Set(CDR(useragent)=${CHANNEL(useragent)})
```

Для того, чтобы эта переменная CDR записывалась в базу данных cdr_adaptive_odbc, все, что нужно сделать - это создать столбец с именем useragent.

Несколько таблиц могут быть настроены в файле конфигурации cdr_adaptive_ odbc. Каждая имеет в свой собственный раздел конфигурации. Название раздела может быть каким угодно, модуль не использует его. Вот пример простой таблицы конфигурации:

```
[mytable]

connection = asterisk
table = asterisk_cdr
```

Более подробный пример установок базы данных для логирования CDR может быть найден в «Хранение записей деталей вызовов (CDR)».

Таблица 24-4, содержит список опций, которые можно определить для таблицы в разделе конфигурационного файла *cdr_adaptive_odbc.conf*.

Таблица 24-4. Таблица конфигурационных опций cdr_adaptive_odbc.conf

Опция	Значение/Пример	Примечание
`connection`	`pgsql1`	Соединение с базой данных, которая будет использоваться. Это ссылка на настроеное соединение в *res_odbc.conf*. Поле является обязательным.
`table`	`asterisk_cdr`	Имя таблицы. Это поле является обязательным.
`usegmtime`	`no`	Указывает, следует ли временные метки логирования использовать по GMT, а не местному времени. По умолчанию значение этой опции - `no`.

В дополнение к паре полей ключ/значение (key/value), которые показаны в предыдущей таблице, *cdr_adaptive_odbc.conf* позволяет использовать несколько других элементов конфигурации. Во-первых, это псевдоним столбца. Обычно, переменные CDR записываются в столбцах с одноименным названием. Псевдоним позволяет изменять имя, которое будет отображаться в колонке с другим именем. Синтаксис такой:

```
alias <CDR variable> => <column name>
```

Вот пример отображение колонки, используя параметр `alias`:

```
alias src => source
```

Кроме того, можно задать фильтр контента. Это позволяет определить критерии, которым должны соответствовать записи, вставленные в таблицу. Синтаксис:

```
filter <CDR variable> => <content>
```

Вот пример фильтра содержания:

```
filter accountcode => 123
```

Наконец, *cdr_adaptive_odbc.conf* позволяет определять статическое содержание для столбцов. Это может быть полезно когда используется вместе с набором `filters`. Статический контент может помочь дифференцировать записи, которые были вставлены в одну таблицу по различным разделам конфигурации. Синтаксис для статического содержимого:

```
static <"Static Content Goes Here"> => <column name>
```

Это пример определения статического содержания для вставки с CDR:

```
static "My Content" => my_identifier
```

cdr_csv

`cdr_csv` - это очень простой модуль CDR, который протоколирует CDR в CSV файл (значения разделены запятыми). Файл */var/log/asterisk/cdr-csv/Master.csv*. Пока логирование CDR включено в *cdr.conf* и этот модуль загружен, CDR будет логироваться в файл *Master.csv*.

Таблица 24-5. Параметры раздела cdr.conf [csv]

Опция	Значение/Пример	Примечание
`usegmtime`	`no`	Записывает временные метки по GMT, а не по локальному времени. По умолчанию - `no`.
`loguniqueid`	`no`	Записывает переменную CDR `uniqueid`. По умолчанию - `no`.

`loguserfield no`	Записывает переменную CDR `userfield`. По умолчанию - `no`.
`accountlogs yes`	Создает отдельный файл CSV для каждого значения переменной CDR `accountcode`. По умолчанию - `yes`.

Порядок переменных CDR в CSV-файлах создаваемых модулем `cdr_csv`:

```
<accountcode>,<src>,<dst>,<dcontext>,<clid>,<channel>,<dstchannel>,<lastapp>, \
    <lastadata>,<start>,<answer>,<end>,<duration>,<billsec>,<disposition>, \
    <amaflags>[,<uniqueid>][,<userfield>]
```

cdr_custom

Этот модуль CDR позволяет задавать пользовательский формат CDR записей в файле протоколирования. Этот модуль наиболее часто используется для индивидуальных выводов CSV. Файл конфигурации для этого модуля */etc/asterisk/cdr_custom.conf*. В этом файле должен быть один раздел под названием `[mappings]`. Этот раздел содержит соответствия между именем файла и шаблоном для CDR. Шаблон задается с помощью функций диалплана Asterisk.

В следующем примере показана конфигурация для `cdr_custom`, она включает логирование в файл CDR - *Master.csv*. Этот файл будет создан в */var/log/asterisk/cdr-custom/Master.csv*. Шаблон, который был определен, использует две функции диалплана `CDR()` и `CSV_QUOTE()`. `CDR()` извлекает значения из сохраненого CDR. Функция `CSV_QUOTE()` гарантирует, что значения экранируются для формата файла CSV:

```
[mappings]

Master.csv => ${CSV_QUOTE(${CDR(clid)})},${CSV_QUOTE(${CDR(src)})},
   ${CSV_QUOTE(${CDR(dst)})},${CSV_QUOTE(${CDR(dcontext)})},
   ${CSV_QUOTE(${CDR(channel)})},${CSV_QUOTE(${CDR(dstchannel)})},
   ${CSV_QUOTE(${CDR(lastapp)})},${CSV_QUOTE(${CDR(lastdata)})},
   ${CSV_QUOTE(${CDR(start)})},${CSV_QUOTE(${CDR(answer)})},
   ${CSV_QUOTE(${CDR(end)})},${CSV_QUOTE(${CDR(duration)})},
   ${CSV_QUOTE(${CDR(billsec)})},${CSV_QUOTE(${CDR(disposition)})},
   ${CSV_QUOTE(${CDR(amaflags)})},${CSV_QUOTE(${CDR(accountcode)})},
   ${CSV_QUOTE(${CDR(uniqueid)})},${CSV_QUOTE(${CDR(userfield)})}
```



В реальном файле конфигурации значение в *Master.csv* должно быть в одной строке.

cdr_manager

Модуль cdr_manager записывает в CDR события Asterisk Manager Interface (AMI), которые мы детально обсуждали в Главе 20. Этот модуль настраивается в файле */etc/asterisk/cdr_manager.conf*. Первый раздел в этом файле `[general]`, который содержит простой параметр для включения этого модуля (по умолчанию значение — `no`):

```
[general]

enabled = yes
```

Другой раздел в *cdr_manager.conf* - это раздел `[mappings]`. Он позволяет добавлять пользовательские переменные CDR для событий менеджера. Синтаксис такой:

```
<CDR variable> => <Header name>
```

Это пример добавления двух пользовательских переменных CDR:

```
[mappings]
```

```
rate => Rate
carrier => Carrier
```

При такой конфигурации, записи CDR будут отображаться как события в интерфейсе менеджера. Чтобы создать пример события менеджера, мы используем следующий пример диалплана:

```
exten => 110,1,Answer()
    same => n,Set(CDR(rate)=0.02)
    same => n,Set(CDR(carrier)=BS&S)
    same => n,Hangup()
```

Эта команда используется для выполнения этого расширения и генерирует образец события менеджера:

```
*CLI> console dial 110@testing
```

В итоге, вот пример события менеджера, образующегося в результате этого тестового вызова:

```
Event: Cdr
Privilege: cdr,all
AccountCode:
Source:
Destination: 110
DestinationContext: testing
CallerID:
Channel: Console/dsp
DestinationChannel:
LastApplication: Hangup
LastData:
StartTime: 2010-08-23 08:27:21
AnswerTime: 2010-08-23 08:27:21
EndTime: 2010-08-23 08:27:21
Duration: 0
BillableSeconds: 0
Disposition: ANSWERED
AMAFlags: DOCUMENTATION
UniqueID: 1282570041.3
UserField:
Rate: 0.02
Carrier: BS&S
```

cdr_mysql

Этот модуль позволяет размещать CDR в базе данных MySQL. Мы рекомендуем использовать в новой инсталляции `cdr_adaptive_odbc` вместо него.

cdr_odbc

Этот модуль обеспечивает старый интерфейс ODBC для логирования CDR. Новые установки должны использовать вместо него `cdr_adaptive_odbc`.

cdr_pgsql

Этот модуль позволяет размещать CDR в базе данных PostgreSQL. Мы рекомендуем использовать в новой инсталляции `cdr_adaptive_odbc` вместо него.

cdr_radius

Модуль `cdr_radius` позволяет размещать CDR на сервере RADIUS. Когда используется это модуль, каждая CDR сообщается RADIUS серверу как одно событие останова. Этот модуль настраивается в файле */etc/asterisk/cdr.conf*. Опции для этого модуля размещены в разделе с именем `[radius]`. Доступные параметры перечислены в Таблице 24-6.

Таблица 24-6. Параметры раздела cdr.conf [radius]

Опция	Значение/Пример	Примечание
`usegmtime`	`no`	Включает логирование с временными метками по GMT, а не по локальному времени. По умолчанию - `yes`.
`log unique id`	`no`	Включает логирование переменной CDR `uniqueid`. По умолчанию - `yes`.
`loguserfield`	`no`	Включает логирование переменной CDR `userfield`. По умолчанию - `yes`.
`radiuscfg`	`/etc/radiusclient-ng/radiusclient.conf`	Задает расположение файла конфигурации *radiusclient-ng*. По умолчанию */etc/radiusclient-ng/radiusclient.conf*

cdr_sqlite

Этот модуль позволяет размещать CDR в базе данных SQLite используя SQLite версии 2. Если у вас нет особой необходимости в SQLite версии 2 вместо 3, мы рекомендуем, чтобы все новые установки использовали `cdr_sqlite3_custom`.

Этот модуль не требует настройки работы. Если модуль был скомпилирован и загружен в Asterisk, он будет вставлять CDR в таблицу называемую *cdr* в базе данных размещенной в */var/log/asterisk/cdr.db*.

cdr_sqlite3_custom

Этот модуль CDR вставляет CDR в базу данных SQLite используя SQLite версии 3. База данных, созданная этим модулем находится в */var/log/asterisk/master.db*. Этот модуль требует наличия конфигурационного файла */etc/asterisk/cdr_sqlite3_custom.conf*. Конфигурационный файл определяет имя таблицы, а также настраивает какие переменные CDR будут вставлены в базу данных:

```
[master]

table = cdr

;
; Список имен столбцов, используемых при вставке CDRs.
;
columns => calldate, clid, dcontext, channel, dstchannel, lastapp, lastdata,
    duration, billsec, disposition, amaflags, accountcode, uniqueid, userfield,
    test

;
; Сопоставьте содержимое CDR с ранее указанными столбцами.

;

values => '${CDR(start)}','${CDR(clid)}','${CDR(dcontext)}','${CDR(channel)}',
    '${CDR(dstchannel)}', '${CDR(lastapp)}','${CDR(lastdata)}','${CDR(duration)}',
    '${CDR(billsec)}','${CDR(disposition)}', '${CDR(amaflags)}',
    '${CDR(accountcode)}','${CDR(uniqueid)}','${CDR(userfield)}','$
{CDR(test)}'
```



В файле *cdr_ sqlite3_custom.conf* содержимое столбцов (`columns`) и значения каждой опции (`values`) должны быть в одной строке.

cdr_syslog

Этот модуль позволяет логировать CDR используя *syslog*. Для его включения сперва добавьте конфигурационный файл системы *syslog - /etc/syslog.conf*. Например:

```
local4.*    /var/log/asterisk/asterisk-cdr.log
```

Asterisk модуль также имеет конфигурационный файл. Добавьте следующий раздел в */etc/asterisk/cdr_syslog.conf*:

```
[cdr]

facility = local4
priority = info
template = "We received a call from ${CDR(src)}"
```

Вот пример вывода *syslog* с помощью этой конфигурации:

```
$ cat /var/log/asterisk/asterisk-cdr.log

Aug 12 19:17:36 pbx cdr: "We received a call from 2565551212"
```

cdr_tds

Модуль `cdr_tds` использует библиотеку FreeTDS чтобы получить возможность отправлять CDR в Microsoft SQL Server и Sybase базу данных. Можно использовать FreeTDS с *unixODBC* поэтому мы рекомендуем использовать `cdr_adaptive_odbc` вместо этого модуля.

Пример Call Detail Records

Мы будем использовать модуль `cdr_custom` для иллюстрации нескольких примеров CDR записей для различных сценариев вызова. Конфигурация для */etc/asterisk/cdr_custom.conf* показана в разделе «cdr_custom».

Односторонний вызов

В этом примере мы покажем как CDR выглядит для простого одностороннего вызова. В частности, мы будем использовать пример пользовательского вызова и проверим свою голосовую почту. Вот расширение из */etc/asterisk/extensions.conf*:

```
exten => *98,1,VoiceMailMain(@${GLOBAL(VOICEMAIL_CONTEXT)})
```

Эта запись CDR из */var/log/asterisk/cdr-custom/Master.csv* была создана как результат вызова этого номера:

```
"""Console"" <2565551212>","2565551212","*98","UserServices",
    "Console/dsp","", "VoiceMailMain","@shifteight.org","2010-08-16 01:08:44",
    "2010-08-16 01:08:44","2010-08-16 01:08:53","9","9","ANSWERED",
    "DOCUMENTATION","","1281935324.0","",0
```

Двусторонний вызов

В следующем примере мы покажем как выглядит CDR для простого двухстороннего вызова. Мы имеем один SIP телефон, вызывающий другой SIP телефон. Ответ на вызов, а затем повешенную трубку после короткого периода времени. Вот расширение, которое было вызвано:

```
exten => 101,1,Dial(SIP/0000FFFF0002)
```

Вот CDR, который был запротоколирован в *Master.csv* в результате этого вызова:

```
"""Console"" <2565551212>","2565551212","101","LocalSets","Console/dsp",
    "SIP/0000FFFF0002-00000000","Dial","SIP/0000FFFF0002","2010-08-16
01:16:10",
    "2010-08-16 01:16:16","2010-08-16 01:16:29","19","13","ANSWERED",
    "DOCUMENTATION","","1281935770.2","",2
```

Предостережения

Система CDR в Asterisk работает очень хорошо для достаточно простых сценариев вызова. Однако, когда сценарии вызова становятся более сложными, включая звонки на нескольких сторон, трансфер, парковку и другие подобные функции, система CDR начинает быстро падать. Многие пользователи сообщают, что записи не показывают всей информации, которую они ожидали. Многие исправления ошибок были сделаны для решения некоторых вопросов, но стоимость ошибки или изменения в поведении при внесении изменений в этой области очень высока, так как эти записи используются при выставлении счета.

В результате команда разработчиков Asterisk все меньше вносит дополнительные изменения в систему CDR. Вместо этого, была разработана новая система регистрации событий канала (CEL), которая предназначена помочь в адресном логировании более сложных сценариев обработки вызовов. Имейте в виду, что CDR проще и легче использовать и мы по-прежнему рекомендуем использовать CDR если они удовлетворяют вашим потребностям.

CEL (Channel Event Logging)

CEL является новой системой, которая была создана чтобы обеспечить более гибкое средство регистрации деталей сложных сценариев вызова. Вместо одной записи на вызов в журнале, вызовы регистрируются серией событий. Это обеспечивает более точную картину того, что произошло при вызове за счет более сложного журнала.

Типы событий канала

Каждая запись CEL представляет событие, которое произошло для канала в системе Asterisk. Таблица 24-7 содержит список событий, которые генерирует Asterisk когда происходит вызов.

Таблица 24-7. Типы событий CEL

Типы событий CEL	Описание
CHAN_START	Канал был создан.
CHAN_END	Канал был разрушен.
LINKEDID_END	Последний канал с заданным `linkedid` был уничтожен.
ANSWER	Канал ответил. Канал создан для исходящего вызова. Это событие будет создано, когда ответит удаленный конец.
HANGUP	Канал повесил трубку. Как правило, это событие будет следовать очень быстро за событием CHAN_END. Разница в том, что это событие происходит, как только получен запрос hangup, в то время как CHAN_END происходит после завершения Asterisk процедуры очистки вызова и все ресурсы, связанные с этим каналом были освобождены.
APP_START	Отслеживаемое приложение начало выполняться на канале. Отслеживаемые приложения задаются в файле конфигурации основной ячейки, который описан в разделе "cel.conf".
APP_END	Отслеживаемое приложения прекратило выполнение на канале.
PARK_START	Канал был припаркован.
PARK_END	Канал снял парковку.

BRIDGE_START	Запущен соединения каналов. Это событие наступает, когда два канала соединяются с помощью приложений Dial() или Queue().
BRIDGE_END	Соединение каналов был закрыто.
BRIDGE_UPDATE	Произошло обновление соединения. Это событие быдет происходить если имя канала или другая информация были изменены во время соединения.
BLINDTRANSFER	Канал выполнил слепой трансфер.
ATTENDEDTRANSFER	Канал выполнил трансфер с уведомлением.
USER_DEFINED	Произошло пользовательское событие канала. Это событие создано с использованием приложения CELGenUserEvent().

Есть еще несколько событий, которые были определены, но еще не используются в любом месте кода Asterisk. Предположительно, некоторые будущие версии будут генерировать эти события в нужном месте. Они перечислены в Таблице 24-8.[3]

Таблица 24-8. Определённые, но не используемые типы событий CEL

Тип события CEL	Описание
CONF_ENTER	Канал, который подключается к комнате конференции.
CONF_EXIT	Канал, который отключился от комнаты конференции.
CONF_START	Конференция была начата. Это событие происходит, когда первый канал входит в комнату конференции.
CONF_END	Конференция была закончена. Это событие происходит, когда последний канал покидает комнату конференции.
3WAY_START	Старт трехстороннего вызова.
3WAY_END	Завершение трехстороннего вызова.
TRANSFER	Основной трансфер был выполнен.
HOOKFLASH	Канал сообщил о событии сигнал отбоя (hookflash).

Содержание событий канала

Каждое событие CEL содержит поля перечисленные в таблице Таблица 24-9:

Таблица 24-9. Поля событий CEL:

Имя поля	Значение/Пример	Примечания
eventtype	CHAN_START	Имя события. Список событий, которые могут произойти можно найти в Таблице 24-7.
eventtime	2010-08-19 07:27:19	Время когда произошло событие.
cidname	Julie Bryant	Имя caller ID установленное на канале связанным с этим событием .
cidnum	18435551212	Номер caller ID установленный на канале связанным с этим событием.
cidani	18435551212	Номер Automatic Number Identification (ANI) установленный на канале связанным с этим событием.
cidrdnis	18435551234	Перенаправление номера установленного на канале связанным с этим событием.
ciddnid	18435550987	Вызываемый номер установленный на канале связанным с этим событием.
exten	101	Расширение в диалплане, которое сейчас начало выполняться.

3 Если вы представите патч для добавления любого из этих событий в код и опишите эту сноску, Рассел вышлет вам бесплатно футболку Asterisk. Сноска за взятку!

context	LocalSets	Контекст для расширения в диалплане, которое сейчас начало выполняться.
channame	SIP/0004F2060EB4-00000010	Имя канала связанное с этим событием.
appname	Dial	Имя приложения диалплана начавшее сейчас выполняться.
appdata	SIP/0004F2060E55	Аргументы, передаваемые приложению диалплана, которое сейчас начало выполняться.
amaflags	DOCUMENTATION	Флаг Automatic Message Accounting (AMA) асоциируется с этим вызовом. Может принимать одно из следующих значений: OMIT, BILLING, DOCUMENTATION, или Unknown.
accountcode	1234	account ID. Это поле определяется пользователем и по умолчанию пустое.
uniqueid	1282218999.18	Уникальный ID для канала, который связан с этим событием.
userfield	I like waffles!	Содержание пользовательского события.
linkedid	1282218999.18	ID каждого вызова. Этот ID позволяет связать воедино несколько событий из нескольких каналов, которые являются частью одного и того же логического вызова. ID исходит от uniqueid первого канала в вызове.
peer	SIP/0004F2060E55-00000020	Название канала соединенного с каналом, определенным в channame.

Некоторые содержания событий CEL определяются пользователем. Например, userfield определяется пользователем и будет пустым по умолчанию. Чтобы установить в него какое-то значение, используйте функцию диалплана CHANNEL(). Вот пример определения userfield для канала:

```
exten => 101,1,Set(CHANNEL(userfield)=I like waffles!)
```

Приложения диалплана

Система CEL включает простые приложения диалплана, которые находятся в модуле *app_celgenuserevent.so*. Эти приложения используют созданные пользовательские события типа EV_USER_EVENT. Практический пример использования этого может быть логирование выбора звонящих в меню:

```
exten => 7,1,CELGenUserEvent(MENU_CHOICE,Caller chose option 7)
```

Для полной текущей информации о синтаксисе приложения CELGenUserEvent() используйте встроенную документацию в Asterisk CLI:

```
*CLI> core show application CELGenUserEvent
```

cel.conf

Система CEL имеет простой конфигурационный файл */etc/asterisk/cel.conf*. Все параметры, заданные здесь влияют на обработку CEL независимо от модуля логирования, находящегося в работе. Таблица 24-10 показывает параметры, которые существуют в этом файле. Все они должны быть установлены в разделе [general] конфигурационного файла.

Таблица 24-10. cel.conf раздел параметров [general]

Опция	Значение/Пример	Примечание
enable	yes	Включает/Отключает CEL. По умолчанию — no.
apps	Dial,queue	Устанавливает приложение диалплана для отслеживания. По умолчанию для отслеживания приложений нет. События EV_APP_START и EV_APP_END

		будут генерироваться когда каналы запустят и остановят выполнение любых отслеживаемых приложений.
`events`	`CHAN_START,CHAN_END, ANSWER,HANGUP`	Списки генерируемых событий. Это полезно, если вы заинтересованы только в подмножестве событий, генерируемых CEL. Если же хотите увидеть все события, установите эту опцию в `ALL`. Значение по умолчанию не генерирует события.
`dateformat`	`%F %T`	Указывает формат даты когда событие CEL содержит отметку времени. Сведения о синтаксисе см. в руководстве для `strftime`, запустив *man strftime* в командной строке. Формат по умолчанию для метки времени CEL `second.microseconds` с эпохи.



Как минимум для начала использования CEL, вы должны установить параметры `enable` и `events` в */etc/asterisk/cel.conf*.

Конечные решения (Backends)

Как и в системе CDR есть ряд модулей, доступных для регистрации CEL событий. На самом деле, все CEL модули были получены из модулей CDR, поэтому их конфигурация очень похожа. В дополнение к параметрам конфигурации для *cel.conf*, которые были описаны в предыдущем разделе, эти модули требуют настройки, чтобы заставить их работать.

cel_odbc

Модуль *cel_odbc.so* предоставляет возможность протоколировать события CEL в базу данных с помощью ODBC. Этот модуль является не совсем таким как CDR для ODBC. Для событий CEL нет пользовательских переменных. Тем не менее, этот модуль будет по-прежнему приспосабливаться к структуре базы данных в том, что он будет логировать поля событий CEL, для которых имеются соответствующие столбцы и не будет выдавать ошибки если нет столбца для каждого поля. Конфигурация для этого модуля находится в */etc/asterisk/cel_odbc.conf*.

Несколько таблиц могут быть настроены в файле конфигурации `cel_odbc`. Каждая имеет свой раздел конфигурации. Название раздела может быть каким угодно, модуль не использует его. Вот пример простой таблицы конфигурации:

```
[mytable]

connection = asterisk
table = asterisk_cel
```

Модуль `cel_odbc` будет использовать следующие столбцы, если они существуют (см. таблицу после этого списка для установки сопоставлений между типами событий и их целочисленным значением, которое будет вставлено в базу данных):

- `eventtype`
- `eventtime`
- `userdeftype`
- `cid_name`
- `cid_num`
- `cid_ani`
- `cid_rdnis`
- `cid_dnid`
- `exten`

- context
- channame
- appname
- appdata
- accountcode
- peeraccount
- uniqueid
- linkedid
- amaflags
- userfield
- peer

Таблица 24-11 показывает сопоставление типов событий и их целых значения, которые будут вставлены в столбец `eventtype` базы данных.

Таблица 24-11. Тип события отображаемое в целом значении для столбца eventtype

Тип события	Целочисленное значение
CHANNEL_START	1
CHANNEL_END	2
HANGUP	3
ANSWER	4
APP_START	5
APP_END	6
BRIDGE_START	7
BRIDGE_END	8
CONF_START	9
CONF_END	10
PARK_START	11
PARK_END	12
BLINDTRANSFER	13
ATTENDEDTRANSFER	14
TRANSFER	15
HOOKFLASH	16
3WAY_START	17
3WAY_END	18
CONF_ENTER	19
CONF_EXIT	20
USER_DEFINED	21
LINKEDID_END	22
BRIDGE_UPDATE	23
PICKUP	24
FORWARD	25

Таблица 24-12 показывает параметры, которые могут быть определены в разделе конфигурации таблицы в файле *cel_odbc.conf*.

Таблица 24-12. Таблица конфигурации cel_odbc.conf

Опция	Значение/Пример	Примечание
`connection`	`pgsql1`	Определяет, какая база данных будет использоваться. Это ссылка на настроенное соединение в *res_odbc.conf*. Это поле является обязательным.
`table`	`asterisk_cdr`	Определяет имя таблицы. Это поле обязательно.
`usegmtime`	`no`	Включает/выключает при логировании использование временных меток GMT, а не локального времени. По умолчанию значение этой опции - `no`.

В дополнение к паре полей ключ/значение, которые показаны в предыдущей таблице, *cel_odbc.conf* позволяет несколько других элементов конфигурации. Во-первых, это псевдоним столбца. Как правило, поля CEL записываются в столбцах с одноименным названием. `alias` позволяет отображать переменную с другим именем. Синтаксис:

```
alias <CEL field> => <column name>
```

Вот пример отображение колонки, используя параметр `alias`:

```
alias exten => extension
```

Кроме того, можно задать фильтр контента. Это позволяет определить критерии, которым должны соответствовать записи, вставляемые в таблицу. Синтаксис такой:

```
filter <CEL field> => <content>
```

Вот пример фильтра содержания:

```
filter appname => Dial
```

Наконец, *cel_odbc.conf* позволяет указать статический контент для столбца. Это может быть полезно когда используется совместно с фильтром. Статический контент может помочь различить записи, которые были вставлены в одну таблицу по различным разделам конфигурации. Синтаксис для статического контента такой:

```
static <"Static Content Goes Here"> => <column name>
```

Вот пример определяющий статический контент для записи с CEL событием:

```
static "My Content" => my_identifier
```

cel_custom

Этот модуль CEL позволяет логировать события CEL в пользовательском формате. Чаще всего он используется для настройки данных CSV. Файл конфигурации для этого модуля */etc/asterisk/cel_custom.conf*. В файле должен быть один раздел с названием `[mappings]`. Этот раздел содержит сопоставления между именами файлов и пользовательскими шаблонами для событий CEL. Шаблоны задаются с помощью функций диалплана Asterisk и нескольких специальных переменных CEL.

Следующий пример - это простая конфигурация для `cel_custom`, которая включает один файл протокола CEL - *Master.csv*. Этот файл будет создан в */var/log/asterisk/cel-custom/Master.csv*. Шаблон, который будет определен, использует функции диалплана `CHANNEL()`, `CALLERID()` и `CSV_QUOTE()`. Функция `CSV_QUOTE()` проверяет что значения правильно подготовлены для формата CSV-файла. Этот пример также рассматривает некоторые специальные переменные CEL, которые перечислены в Таблице 24-13.

Таблица 24-13. Переменные CEL, доступные для использования в [mappings]

Переменная CEL	Значение	Описание
`${eventtype}`	`CHAN_START`	Имя события CEL.
`${eventtime}`	`1281980238.660403`	Временной штамп для событий CEL. В этом примере штамп времени приводится в формате по умолчанию.
`${eventextra}`	`Whiskey Tango Foxtrot`	Пользовательские данные, включаемые с событием CEL. Дополнительные данные включаются когда используется `CELGenUserEvent()`.

Это пример файла */etc/asterisk/cel_custom.conf*:

```
[mappings]

Master.csv => ${CSV_QUOTE(${eventtype})},${CSV_QUOTE(${eventtime})},
${CSV_QUOTE(${CALLERID(name)})},${CSV_QUOTE(${CALLERID(num)})},
${CSV_QUOTE(${CALLERID(ANI)})},${CSV_QUOTE(${CALLERID(RDNIS)})},
${CSV_QUOTE(${CALLERID(DNID)})},${CSV_QUOTE(${CHANNEL(exten)})},
${CSV_QUOTE(${CHANNEL(context)})},${CSV_QUOTE(${CHANNEL(channame)})},
${CSV_QUOTE(${CHANNEL(appname)})},${CSV_QUOTE(${CHANNEL(appdata)})},
${CSV_QUOTE(${CHANNEL(amaflags)})},${CSV_QUOTE(${CHANNEL(accountcode)})},
${CSV_QUOTE(${CHANNEL(uniqueid)})},${CSV_QUOTE(${CHANNEL(linkedid)})},
${CSV_QUOTE(${CHANNEL(peer)})},${CSV_QUOTE(${CHANNEL(userfield)})},
${CSV_QUOTE(${eventextra})}
```



В реальном файле конфигурации значения в *Master.csv* должны быть в одной строке.

cel_manager

Модуль `cel_manager` создает события CEL на Asterisk Manager Interface (мы подробно обсуждали AMI в Главе 20). Этот модуль настраивается в файле */etc/asterisk/cel.conf*. Он должен содержать одну секцию, называемую `[manager]`, которая содержит один параметр включения этого модуля. Значение по умолчанию - `no`, но вы вожете включить его вот так:

```
[manager]

enabled = yes
```

При такой конфигурации события CEL будут выглядеть как события AMI. Для примера генерации событий менеджера, мы будем использовать следующий диалплан:

```
exten => 111,1,Answer()
    same => n,CELGenUserEvent(Custom Event,Whiskey Tango Foxtrot)
    same => n,Hangup()
```

Эта команда используется для выполнения этого расширения и генерации некоторых событий CEL:

```
*CLI> console dial 111@testing
```

И наконец, это один из примеров менеджера событий созданный в результате этого тестового вызова:

```
Event: CEL
Privilege: call,all
EventName: CHAN_START
AccountCode:
CallerIDnum:
CallerIDname:
CallerIDani:
```

```
CallerIDrdnis:
CallerIDdnid:
Exten: 111
Context: testing
Channel: Console/dsp
Application:
AppData:
EventTime: 2010-08-23 08:14:51
AMAFlags: NONE
UniqueID: 1282569291.1
LinkedID: 1282569291.1
Userfield:
Peer:
```

cel_pgsql

Этот модуль позволяет сохранять события CEL в базе данных PostgreSQL. Мы рекомендуем использовать в новых установках `cel_odbc`.

cel_radius

Модуль `cel_radius` позволяет сохранять события CEL на сервере RADIUS. Когда используется этот модуль, каждое событие CEL заносится на сервер RADIUS как отдельное событие. Этот модуль настраивается в файле */etc/asterisk/cel.conf*. Параметры для этого модуля перечислены в Таблице 24-14 и размещаются в разделе `[radius]`.

Таблица 24-14. Разрешенные параметры в разделе [radius] файла cel.conf

Опция	Значение/Пример	Примечание
`usegmtime`	`no`	Использует штамп времени GMT (гринвич меридиан тайм) вместо локального времени. По умолчанию - `yes`.
`radiuscfg`	`/etc/radiusclient-ng/radiusclient.conf`	Устанавливает размещение конфигурационного файла *radiusclient-ng*. По умолчанию */etc/radiusclient-ng/radiusclient.conf.*

cel_sqlite3_custom

Этот модуль CEL записывает события CEL в базу данных SQLite используя SQLite вирсии 3. База данных, созданная этим модулем, находится в */var/log/asterisk/master.db*. Файл конфигурации для этого модуля */etc/asterisk/cel_sqlite3_custom.conf* определяет имя таблицы, а также настраивает какие переменные CEL будут вставлены в базу данных. Выглядит это примерно так:

```
[master]

table = cel

;
; Список имен столбцов, используемых при вставке событий CEL.
;
columns => eventtype, eventtime, cidname, cidnum, cidani, cidrdnis, ciddnid,
context, exten, channame, appname, appdata, amaflags, accountcode, uniqueid,
userfield, peer

;
; Сопоставьте содержимое события CEL с ранее указанными столбцами.
;
values => '${eventtype}','${eventtime}','${CALLERID(name)}','$
{CALLERID(num)}',
'${CALLERID(ANI)}','${CALLERID(RDNIS)}','${CALLERID(DNID)}',
'${CHANNEL(context)}','${CHANNEL(exten)}','${CHANNEL(channame)}',
```

```
'${CHANNEL(appname)}','${CHANNEL(appdata)}', '$CHANNEL(amaflags)}',
'${CHANNEL(accountcode)}','${CHANNEL(uniqueid)}', '${CHANNEL(userfield)}',
'${CHANNEL(peer)}'
```



В файле *cel_ sqlite3_custom.conf* содержимое столбцов и значения параметров должны находиться в одной строке.

cel_tds

Модуль `cel_tds` использует библиотеку FreeTDS для записи событий CEL в базу данных Microsoft SQL Server или Sybase. Это можно сделать используя FreeTDS с *unixODBC*, поэтому мы рекомендуем использовать `cel_odbc` вместо этого модуля.

Примеры событий канала

Сейчас мы покажем вам несколько примеров создания событий вызова из системы CEL. Модуль `cel_custom` будет использован из-за своей простоты. Используемая конфигурация */etc/asterisk/cel_custom.conf* приведена в разделе «cel_custom». Кроме того, следующая конфигурация была использована для */etc/asterisk/cel.conf*:

```
[general]

enable = yes
apps = Dial,Playback
events = ALL
```

Односторонний вызов

В этом примере показан одиночный телефонный вызов в расширение, которое проигрывает сообщение “Hello World.” Вот этот диалплан:

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

Вот события CEL, которые регистрируются в результате принятия этого вызова:

```
"CHAN_START","1282062437.436130","Julie Bryant","12565553333","","","","200",
   "LocalSets", "SIP/0000FFFF0003-00000010","","","3","","1282062437.17",
   "1282062437.17","",""

"ANSWER","1282062437.436513","Julie Bryant","12565553333","12565553333","",
   "200","200", "LocalSets","SIP/0000FFFF0003-00000010","Answer","","3","",
   "1282062437.17","1282062437.17","",""

"APP_START","1282062437.501868","Julie Bryant","12565553333","12565553333",
   "","200","200", "LocalSets","SIP/0000FFFF0003-00000010","Playback",
   "hello-world","3","","1282062437.17", "1282062437.17","",""

"APP_END","1282062439.008997","Julie Bryant","12565553333","12565553333","",
   "200","200", "LocalSets","SIP/0000FFFF0003-00000010","Playback",
   "hello-world","3","","1282062437.17", "1282062437.17","",""

"HANGUP","1282062439.009127","Julie Bryant","12565553333","12565553333","",
   "200","200", "LocalSets","SIP/0000FFFF0003-00000010","","","3","",
   "1282062437.17","1282062437.17","",""

"CHAN_END","1282062439.009666","Julie Bryant","12565553333","12565553333",
```

```
   "","200","200", "LocalSets","SIP/0000FFFF0003-00000010","","","3","",
   "1282062437.17","1282062437.17","",""

"LINKEDID_END","1282062439.009707","Julie Bryant","12565553333",
   "12565553333","","200","200", "LocalSets","SIP/0000FFFF0003-00000010","",
   "","3","","1282062437.17","1282062437.17","",""
```

Двусторонний вызов

Для второго примера один телефон буде вызывать другой через расширение `101`. В результате вызов имеет два канала, которые объединяются в мост. Вот расширение, которое будет вызываться в диалплане:

```
exten => 101,1,Dial(SIP/0000FFFF0001)
```

А это события CEL, которые генерируются в результате создания этого вызова:

```
"CHAN_START","1282062455.574611","Julie Bryant","12565553333","","","","101",
   "LocalSets", "SIP/0000FFFF0003-00000011","","","3","","1282062455.18",
   "1282062455.18","",""

"APP_START","1282062455.574872","Julie Bryant","12565553333","12565553333","",
   "101","101", "LocalSets","SIP/0000FFFF0003-00000011","Dial",
   "SIP/0000FFFF0001","3","","1282062455.18","1282062455.18","",""

"CHAN_START","1282062455.575044","Candice Yant","12565551111","","","","s",
   "LocalSets", "SIP/0000FFFF0001-00000012","","","3","","1282062455.19",
   "1282062455.18","",""

"ANSWER","1282062458.068134","","101","12565551111","","","101","LocalSets",
   "SIP/0000FFFF0001-00000012","AppDial","(Outgoing Line)","3","",
   "1282062455.19","1282062455.18","",""

"ANSWER","1282062458.068361","Julie Bryant","12565553333","12565553333","",
   "101","101", "LocalSets","SIP/0000FFFF0003-00000011","Dial",
   "SIP/0000FFFF0001","3","","1282062455.18", "1282062455.18","",""

"BRIDGE_START","1282062458.068388","Julie Bryant","12565553333",
   "12565553333","","101","101", "LocalSets","SIP/0000FFFF0003-00000011",
   "Dial","SIP/0000FFFF0001","3","","1282062455.18", "1282062455.18","",""

"BRIDGE_END","1282062462.965704","Julie Bryant","12565553333","12565553333",
   "","101","101", "LocalSets","SIP/0000FFFF0003-00000011","Dial",
   "SIP/0000FFFF0001","3","","1282062455.18", "1282062455.18","",""

"HANGUP","1282062462.966097","","101","12565551111","","","","LocalSets",
   "SIP/0000FFFF0001-00000012","AppDial","(Outgoing Line)","3","",
   "1282062455.19","1282062455.18","",""

"CHAN_END","1282062462.966119","","101","12565551111","","","","LocalSets",
   "SIP/0000FFFF0001-00000012","AppDial","(Outgoing Line)","3","",
   "1282062455.19","1282062455.18","",""

"APP_END","1282062462.966156","Julie Bryant","12565553333","12565553333","",
   "101","101","LocalSets","SIP/0000FFFF0003-00000011","Dial",
   "SIP/0000FFFF0001","3","","1282062455.18","1282062455.18","",""

"HANGUP","1282062462.966215","Julie Bryant","12565553333","12565553333",
   "","101","101","LocalSets","SIP/0000FFFF0003-00000011","","","3","",
```

```
   "1282062455.18","1282062455.18","",""

"CHAN_END","1282062462.966418","Julie Bryant","12565553333","12565553333",
   "","101","101","LocalSets","SIP/0000FFFF0003-00000011","","","3","",
   "1282062455.18","1282062455.18","",""

"LINKEDID_END","1282062462.966441","Julie Bryant","12565553333",
   "12565553333","","101","101","LocalSets","SIP/0000FFFF0003-00000011",
   "","","3","","1282062455.18","1282062455.18","",""
```

Слепой трансфер

В этом последнем примере будет выполнен трансфер. Вызов начнется вызовом телефона по добавочному номеру `102`. Затем вызов будет переведен на другой телефон с номером `101`. Вот соответствующий диалплан:

```
exten => 101,1,Dial(SIP/0000FFFF0001)
exten => 102,1,Dial(SIP/0000FFFF0002)
```

Это запротоколированные события CEL в результате этого сценария вызова:

```
"CHAN_START","1282062488.028200","Julie Bryant","12565553333","","","",
   "102","LocalSets", "SIP/0000FFFF0003-00000013","","","3","",
   "1282062488.20","1282062488.20","",""

"APP_START","1282062488.028464","Julie Bryant","12565553333","12565553333",
   "","102","102", "LocalSets","SIP/0000FFFF0003-00000013","Dial",
   "SIP/0000FFFF0002","3","","1282062488.20", "1282062488.20","",""

"CHAN_START","1282062488.028762","Brooke Brown","12565552222","","","",
   "s","LocalSets", "SIP/0000FFFF0002-00000014","","","3","","1282062488.21",
   "1282062488.20","",""

"ANSWER","1282062492.565759","","102","12565552222","","","102","LocalSets",
   "SIP/0000FFFF0002-00000014","AppDial","(Outgoing Line)","3","",
   "1282062488.21","1282062488.20","",""

"ANSWER","1282062492.565973","Julie Bryant","12565553333","12565553333","",
   "102","102","LocalSets","SIP/0000FFFF0003-00000013","Dial",
   "SIP/0000FFFF0002","3","","1282062488.20","1282062488.20","",""

"BRIDGE_START","1282062492.566001","Julie Bryant","12565553333",
   "12565553333","","102","102","LocalSets","SIP/0000FFFF0003-00000013",
   "Dial","SIP/0000FFFF0002","3","","1282062488.20","1282062488.20","",""

"CHAN_START","1282062497.940687","","","","","","","s","LocalSets",
   "AsyncGoto/SIP/0000FFFF0002-00000014","","","3","","1282062497.22",
   "1282062488.20","",""

"BLINDTRANSFER","1282062497.940925","Julie
Bryant","12565553333","12565553333","",

"102","102","LocalSets","SIP/0000FFFF0003-00000013","Dial","SIP/0000FFFF0002",
   "3","","1282062488.20","1282062488.20",
   "AsyncGoto/SIP/0000FFFF0002-00000014<ZOMBIE>",""

"BRIDGE_END","1282062497.940961","Julie
Bryant","12565553333","12565553333","",
   "102","102","LocalSets","SIP/0000FFFF0003-00000013","Dial",
```

```
   "SIP/0000FFFF0002","3","","1282062488.20","1282062488.20","",""

"APP_START","1282062497.941021","","102","12565552222","","","101","LocalSets"
,
   "SIP/0000FFFF0002-00000014","Dial","SIP/0000FFFF0001","3","",
   "1282062497.22","1282062488.20","",""

"CHAN_START","1282062497.941207","Candice Yant","12565551111","","","","s",
   "LocalSets","SIP/0000FFFF0001-00000015","","","3","","1282062497.23",
   "1282062488.20","",""

"HANGUP","1282062497.941361","","","","","","","LocalSets",
   "AsyncGoto/SIP/0000FFFF0002-00000014<ZOMBIE>","AppDial",
   "(Outgoing Line)","3","","1282062488.21","1282062488.20","",""

"CHAN_END","1282062497.941380","","","","","","","LocalSets",
   "AsyncGoto/SIP/0000FFFF0002-00000014<ZOMBIE>","AppDial","(Outgoing Line)",
   "3","","1282062488.21","1282062488.20","",""

"APP_END","1282062497.941415","Julie Bryant","12565553333","12565553333","",
   "102","102","LocalSets","SIP/0000FFFF0003-00000013","Dial",
   "SIP/0000FFFF0002","3","","1282062488.20","1282062488.20","",""

"HANGUP","1282062497.941453","Julie Bryant","12565553333","12565553333",
   "","102","102","LocalSets","SIP/0000FFFF0003-00000013","","","3","",
   "1282062488.20","1282062488.20","",""

"CHAN_END","1282062497.941474","Julie Bryant","12565553333","12565553333",
   "","102","102","LocalSets","SIP/0000FFFF0003-00000013","","","3","",
   "1282062488.20","1282062488.20","",""

"ANSWER","1282062500.559578","","101","12565551111","","","101","LocalSets",
   "SIP/0000FFFF0001-00000015","AppDial","(Outgoing Line)","3","",
   "1282062497.23","1282062488.20","",""

"BRIDGE_START","1282062500.559720","","102","12565552222","","","101","LocalSe
ts",

"SIP/0000FFFF0002-00000014","Dial","SIP/0000FFFF0001","3","","1282062497.22",
   "1282062488.20","",""

"BRIDGE_END","1282062512.742600","","102","12565552222","","","101","LocalSets
",

"SIP/0000FFFF0002-00000014","Dial","SIP/0000FFFF0001","3","","1282062497.22",
   "1282062488.20","",""

"HANGUP","1282062512.743006","","101","12565551111","","","","LocalSets",
   "SIP/0000FFFF0001-00000015","AppDial","(Outgoing
Line)","3","","1282062497.23",
   "1282062488.20","",""

"CHAN_END","1282062512.743211","","101","12565551111","","","","LocalSets",
   "SIP/0000FFFF0001-00000015","AppDial","(Outgoing
Line)","3","","1282062497.23",
   "1282062488.20","",""

"APP_END","1282062512.743286","","102","12565552222","","","101","LocalSets",
```

```
"SIP/0000FFFF0002-00000014","Dial","SIP/0000FFFF0001","3","","1282062497.22",
   "1282062488.20","",""

"HANGUP","1282062512.743346","","102","12565552222","","","101","LocalSets",
   "SIP/0000FFFF0002-00000014","","","3","","1282062497.22","1282062488.20",
   "",""

"CHAN_END","1282062512.743371","","102","12565552222","","","101","LocalSets",
   "SIP/0000FFFF0002-00000014","","","3","","1282062497.22","1282062488.20",
   "",""

"LINKEDID_END","1282062512.743391","","102","12565552222","","","101",
   "LocalSets","SIP/0000FFFF0002-00000014","","","3","","1282062497.22",
   "1282062488.20","",""
```

SNMP

Simple Network Management Protocol (SNMP) - это стандартизированный протокол для управления сетью. Он очень широко используется и реализован во многих приложениях и сетевых устройствах. Такая платформа как OpenNMS[4] - это платформа управления сетью с открытым исходным кодом, использующая SNMP (а также другие вещи). Asterisk поддерживает SNMP через модуль `res_snmp`. Этот раздел описывает установку и настройку `res_snmp` и как мы можем использовать платформу подобную OpenNMS.

Установка модуля SNMP для Asterisk

По умолчанию Asterisk не компилирует модуль разработки SNMP, поскольку сначала нужно установить все необходимые (зависимые) пакеты.

Зависимости в RHEL

В RHEL вам просто нужно установить пакет *net-snmp-devel*:

```
$ sudo yum install net-snmp-devel
```

Смотри далее раздел под названием “Перекомпиляция Asterisk с модулем res_snmp” для обзора, как перекомпилировать Asterisk с поддержкой SNMP.

Зависимости в Ubuntu

Под Ubuntu, нужно установить следующий пакет:

```
$ sudo apt-get install snmp libsnmp-dev snmpd
```



На Ubuntu необходимо устанавливать оба пакета *snmp* и *snmpd*, поскольку они не разрешают зависимостей библиотек разработки SNMP наподобие RHEL. Пакет *snmp* устанавливает инструменты SNMP подобно *snmpwalk*, который нам нужен, а пакет *snmpd* устанавливает демон SNMP.

Смотри следующий раздел для описания, как пере компилировать Asterisk с поддержкой SNMP.

4 OpenNMS, конечно не единственная платформа, которая может быть использована с модулем `res_snmp`. Тем не менее, мы решили обсудить её здесь по ряду причин. Во-первых, OpenNMS очень хорошая платформа управления сетью, которая имеет специальную интеграцию с Asterisk. Во-вторых, это открытый исходный код и 100% бесплатный. Наконец, Джефф Гельбах из OpenNMS внесший свой вклад в развитие Asterisk, прилагает значительные усилия в поддержке SNMP. OpenNMS также был достаточно хорош, чтобы помочь нам получить описание как все это работает, что мы могли задокументировать это.

Перекомпиляция Asterisk с модулем res_snmp

Как только вы установите все нужные пакеты для SNMP, то можете перекомпилировать Asterisk с поддержкой SNMP:

```
$ cd ~/src/asterisk-complete/asterisk/11/
$ ./configure
$ make menuselect # verify that res_snmp is selected under Resource Modules
$ make
$ sudo make install
```

Затем вам нужно скопировать пример конфигурационного файла в каталог */etc/asterisk*:

```
$ sudo cp ~/src/asterisk-complete/asterisk/11/configs/res_snmp.conf.sample \
/etc/asterisk/res_snmp.conf
```

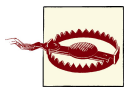
Мы раскажем о настройке этого файла, для использования с OpenNMS, в следующем разделе.

Настройка SNMP для использования OpenNMS в Asterisk

Проект OpenNMS - это платформа с открытым исходным кодом управления сетью, поддержка которой встроена в Asterisk. Однако, нужно сделать несколько шагов, чтобы включить эту поддержку. В этом разделе мы рассмотрим необходимые действия, чтобы подружить ваш Asterisk сервер с OpenNMS.

Установка OpenNMS

Вики OpenNMS имеет детальную иструкцию по установке OpenNMS.



OpenNMS можно не устанавливать на ваш сервер Asterisk. Вы можете выделить отдельную машину для сервера OpenNMS.

Поскольку Вики OpenNMS предоставляет все необходимые инструкции, мы позволим экспертам провести вас по первой части установки. После установки OpenNMS, вернитесь сюда и мы покажем как настроить его для работы с Asterisk.

Инструкции по установке OpenNMS в вики используют SNMPv2c, который не является безопасным методом абстрагирования данных протокола SNMP. Так как мы хотим построить безопасную систему, наши инструкции покажут вам как включить поддержку SNMPv3.[5]

Однако, поскольку SNMPv3 может быть немного громоздким зверем или потому что вы не желаете включить SNMPv3 по каким-то причинам (например, если ваша версия SNMP не была собрана с поддержкой OpenSSL), мы предоставим вам инструкции по настройке демона SNMP для SNMPv2c и SNMPv3.

Как правило, проще настроить систему для SNMPv2c, а затем обновить ее до SNMPv3, правильная настройка SNMPv3 более сложная.

Редактирование /etc/asterisk/res_snmp.conf для работы с вашим сервером OpenNMS

В файле */etc/asterisk/res_snmp.conf*, который вы скопировали из каталога с исходными кодами, нужно раскоментировать две строки:

```
[general]
;subagent=yes
;enabled=yes
```

5 Дополнительно вы можете найти пост в блоге о включении SNMPv3 для OpenNMS.

Изменим файл *res_snmp.conf* для обоих клиентов SNMP и включим `subagent`:

```
[general]
subagent=yes
enabled=yes
```

После изменения этого файла, вам нужно перезагрузить модуль *res_snmp.so*, чтобы изменения вступили в силу:

```
*CLI> module unload res_snmp.so

Unloaded res_snmp.so
 Unloading [Sub]Agent Module
  == Terminating SubAgent

*CLI> module load res_snmp.so

Loaded res_snmp.so
  == Parsing '/etc/asterisk/res_snmp.conf':     == Found
 Loading [Sub]Agent Module
 Loaded res_snmp.so => (SNMP [Sub]Agent for Asterisk)
  == Starting SubAgent
```

Редактирование /etc/snmp/snmpd.conf для работы с вашим сервером OpenNMS

Теперь мы можем изменить файл */etc/snmp/snmpd.conf* для SNMP на хосте. Переименуем текущий пример конфигурационного файла и создадим новый файл *snmpd.conf*:

```
$ cd /etc/snmp
$ sudo mv snmpd.conf snmpd.sample
```

Первое что нужно сделать - это добавить права доступа этому файлу. Мы предлагаем вам прочитать файл */etc/snmpd/snmp.sample*, который вы переименовали для лучшего представления об установке прав доступа. Затем добавим следующее в ваш файл *snmpd.conf*:

```
$ sudo sh -c cat > snmpd.conf

com2sec     notConfigUser     default     public

group       notConfigGroup    v1          notConfigUser
group       notConfigGroup    v2c         notConfigUser

view all    included    .1
view system included    .iso.org.dod.internet.mgmt.mib-2.system

access  notConfigGroup  ""  any  noauth  exact  all   none  none

syslocation Caledon, ON
syscontact Leif Madsen lmadsen@shifteight.org
```

Ctrl+D



Строки `syslocation` и `syscontact` необязательны, но они могут облегчить идентификацию сервера, если вы мониторите несколько узлов.

Теперь нам нужно включить поддержку субагента AgentX что бы мы могли найти информацию о нашей системе Asterisk:

```
$ sudo cat >> snmpd.conf
```

```
master agentx
agentXSocket /var/agentx/master
agentXPerms 0660 0775 nobody root

sysObjectID .1.3.6.1.4.1.22736.1
```

Ctrl+D

В дополнении к строке `master agentx` и опции `agentX`, мы включаем на Asterisk демон SNMP для коммуникации. Опция `agentXPerms` говорит, что Asterisk запущен от *root*. Если ваш Asterisk запущен от другой группы замените *root* на группу от которой запущен Asterisk.

Чуть ниже конфигурации AgentX, мы добавили параметр `sysObjectID`. Цель добавления строки `sysObjectID` указать OpenNMS на хост-систему Asterisk, что позволяет динамически захватить дополнительную информацию для графиков.

После того, как вы выполните эти шаги по настройке, вам нужно перезапустить демон SNMP:

```
$ sudo /etc/init.d/snmpd restart
```

Чтобы убедиться что информация будет передаваться правильно, используйте приложение *snmpwalk*:

```
$ snmpwalk -On -v2c -c public 127.0.0.1 .1.3.6.1.4.1.22736
```

Если вы правильно все настроили, то должны получить несколько строк с информацией на вашем экране, подобно этим:

```
.1.3.6.1.4.1.22736.1.5.4.1.4.3 = INTEGER: 2
.1.3.6.1.4.1.22736.1.5.4.1.4.4 = INTEGER: 2
.1.3.6.1.4.1.22736.1.5.4.1.4.5 = INTEGER: 1
.1.3.6.1.4.1.22736.1.5.4.1.4.6 = INTEGER: 1
.1.3.6.1.4.1.22736.1.5.4.1.5.1 = INTEGER: 1
...etc
```

На этом этапе ваша хост-система должна быть готова для подключения к OpenNMS и сбора необходимой информации. Далее добавьте узел в систему и заполните соответствующую информацию. Через некоторое время OpenNMS опросит хост-систему и получит доступ к статистике Asterisk. Вы должны иметь возможность кликнуть графики ресурсов (Resource Graphs) после выбора созданного узла и просмотреть выбор доступных графиков, таких как SIP, DAHDI, Local и т.д.

Мониторинг Asterisk с помощью OpenNMS

После того, как вы установили OpenNMS и настроили модуль `res_snmp` в Asterisk, вы можете использовать OpenNMS для наблюдения за вашим сервером Asterisk. Вы можете настроить какие статистические данные отслеживать, а также какие уведомления хотели бы получать на основе этих статистических данных. Изучение возможностей OpenNMS остается в качестве упражнения для читателя. Тем не менее, мы приложили несколько графиков для демонстрации основной информации, которую вы можете получить от сервера Asterisk. Эти графики от сервера Asterisk, который не очень сильно загружен, но они дают хороший пример того, что вы можете увидеть.

Рисунок 24-1 содержит график, показывающий количество активных каналов в Asterisk в различное время.

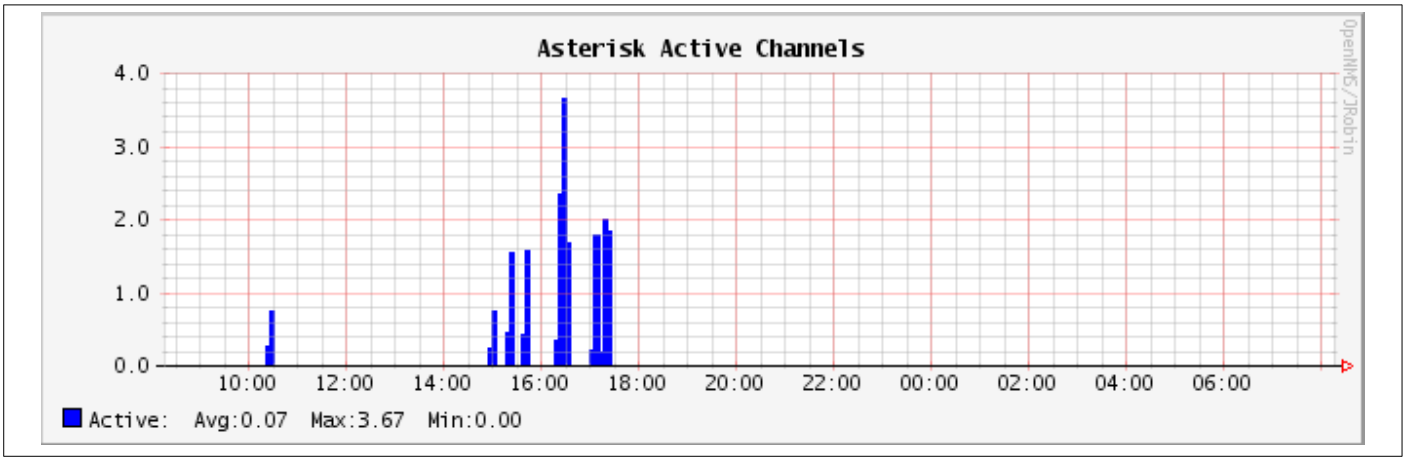


Рисунок 24-1. График активности каналов Asterisk

Рисунок 24-2 показывает график активности каналов определенного типа. В данном случае мы просматриваем количество активных каналов DAHDI в системе. Мониторинг каналов DAHDI интересен с практической стороны, поскольку каналы DAHDI взаимосвязаны с физическими ресурсами и доступное количество каналов предопределено. Это будет очень полезно для управления использования каналами DAHDI и получения уведомлений о занятости всех каналов, это может послужить сигналом для добавления дополнительных каналов.

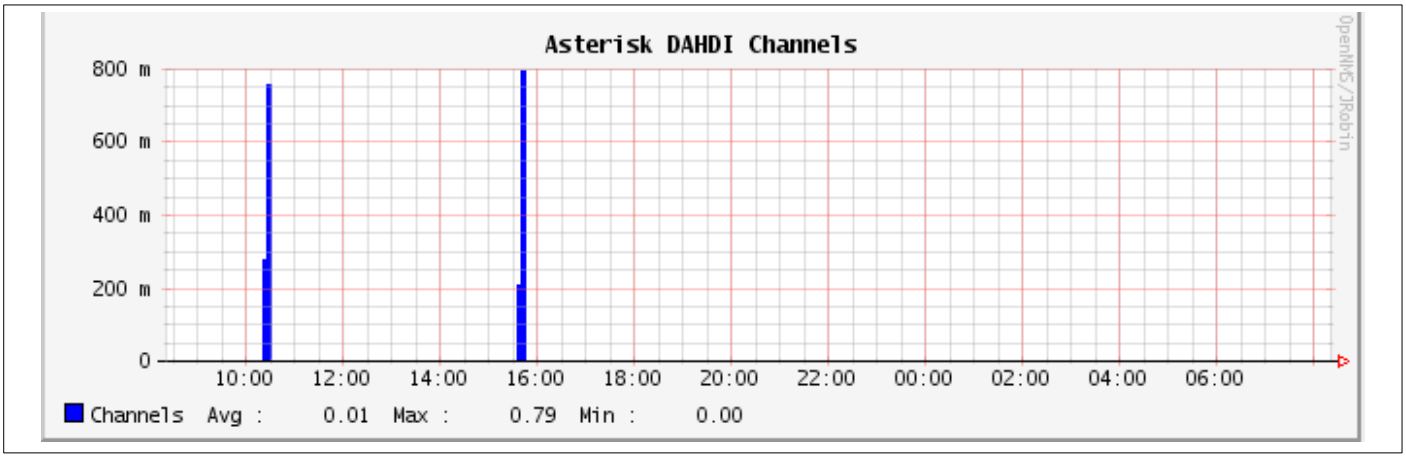


Рисунок 24-2. График активности каналов DAHDI

В завершение, Рисунок 24-3 показывает использование сетевого интерфейса. Как вы можете видеть, были всплески трафика, следующего «в» и «из» системы, когда происходили SIP вызовы.

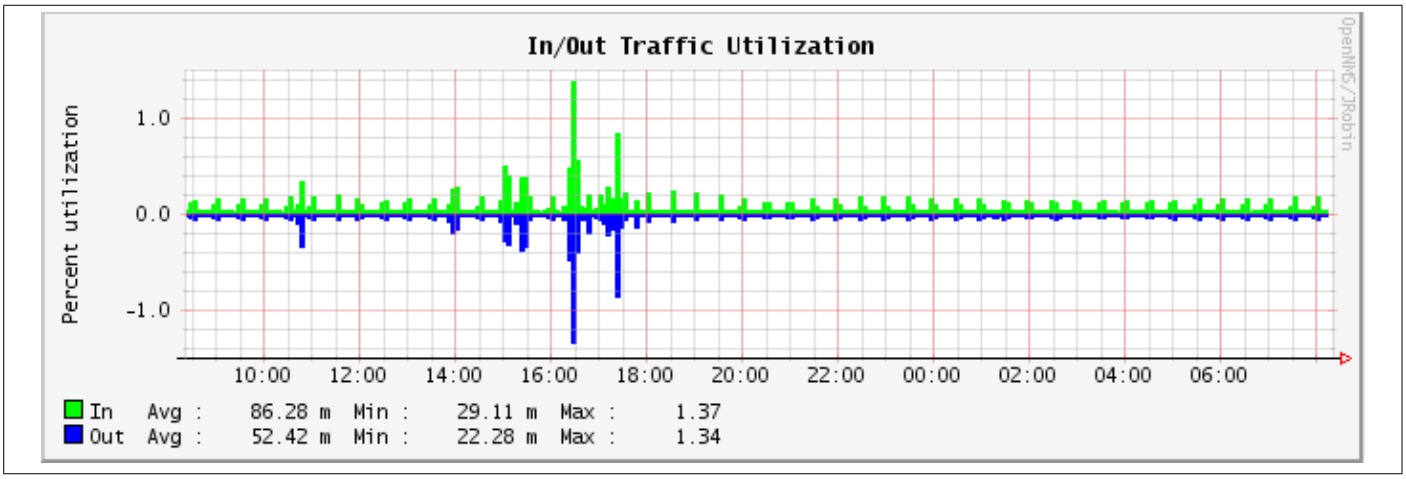


Рисунок 24-3. График трафика на сетевом интерфейсе

Заключение

Asterisk очень хорошо позволяет отслеживать различные аспекты своей деятельности, начиная от CDR до полной отладки выполнения кода. Эти различные механизмы помогут вам в управлении

УАТС Asterisk и представляют собой один из моментов, в которых Asterisk значительно превосходит большинство (если не все) традиционные АТС.

ГЛАВА 25

Web-интерфейсы

Точка зрения может быть опасной роскошью,
если ее заменить проницательностью и пониманием.
—Маршал МакЛухан

Прежде чем вы будете слишком взволнованы, эта глава не будет говорить о GUI конфигурации абонентской группы, таком как FreePBX. Мы признаем, что большая часть успеха Asterisk связана с успехом FreePBX-проектов, таких как AsteriskNOW и PBX in a Flash, но в этой книге мы сосредоточимся на Asterisk. Таким образом, мы не будем обсуждать каких-либо парней, которые по существу удаляют ваши отношения с диалпланом. Дело не в том что мы против этих вещей, а просто в том, что у нас есть так много места в этой книге, и наша цель - смотреть на Asterisk снизу вверх. Большинство проектов графических интерфейсов Asterisk скрывают внутреннюю работу Asterisk за интерфейсом и по этой причине они несовместимы с целями этой книги. Поэтому наше обсуждение веб-интерфейсов Asterisk будет сосредоточено на интерфейсах и компонентах, отличных от диалплана.

FreePBX GUI диалплана

Теперь, когда мы пообещали не говорить об интерфейсах диалплана, мы считаем что было бы неправильно вообще ничего не сказать о FreePBX, гиганте сообщества Asterisk. Этот интерфейс (который лежит в основе многих самых популярных дистрибутивов Asterisk, таких как AsteriskNOW, Elastix, дистрибутивов FreePBX и PBX in a Flash), несомненно, является очень большой частью того, почему Asterisk был столь успешным, какой он есть. С помощью интерфейса FreePBX вы можете настраивать и управлять многими аспектами системы Asterisk, не касаясь ни одного файла конфигурации. В то время как мы, пуристы, любим, чтобы все работали только с конфигурационными файлами, мы понимаем, что для многих изучение Linux и редактирование этих файлов вручную просто не произойдет. Для этих людей существует FreePBX, и мы уважаем важный вклад, который он внес в успех Asterisk.

В этой главе мы представим несколько проектов, обеспечивающих веб-интерфейсы в других частях системы, а также выборку веб-приложений, которые являются важными, полезными или рекомендуемыми. В целом, мы сосредоточились на бесплатных приложениях с открытым исходным кодом, но упомянем некоторые коммерческие продукты, где, по нашему мнению, это оправдано.

Многие сторонние приложения были разработаны для Asterisk. Описанные здесь, являются одними из лучших, на момент написания этой статьи.

Flash Operator Panel

Флеш-панель оператора (или FOP, как оно более широко известно) является интерфейсом в первую очередь для использования операторов коммутатора. FOP использует Adobe Flash для представления

интерфейса через веб-браузер и подключается к Asterisk через AMI (см. Главу 20 для обсуждения AMI).

Существует две версии Флеш-панели оператора: оригинальная версия (версия 0.30, теперь только версия для обслуживания и, вероятно, несовместимая с Asterisk 11) и версия FOP2 (показана на Рис. 25-1), которая является значительно улучшенной по сравнению с оригинальной версией, но требует покупки лицензии для любой системы с более чем 15 внутренними номерами.

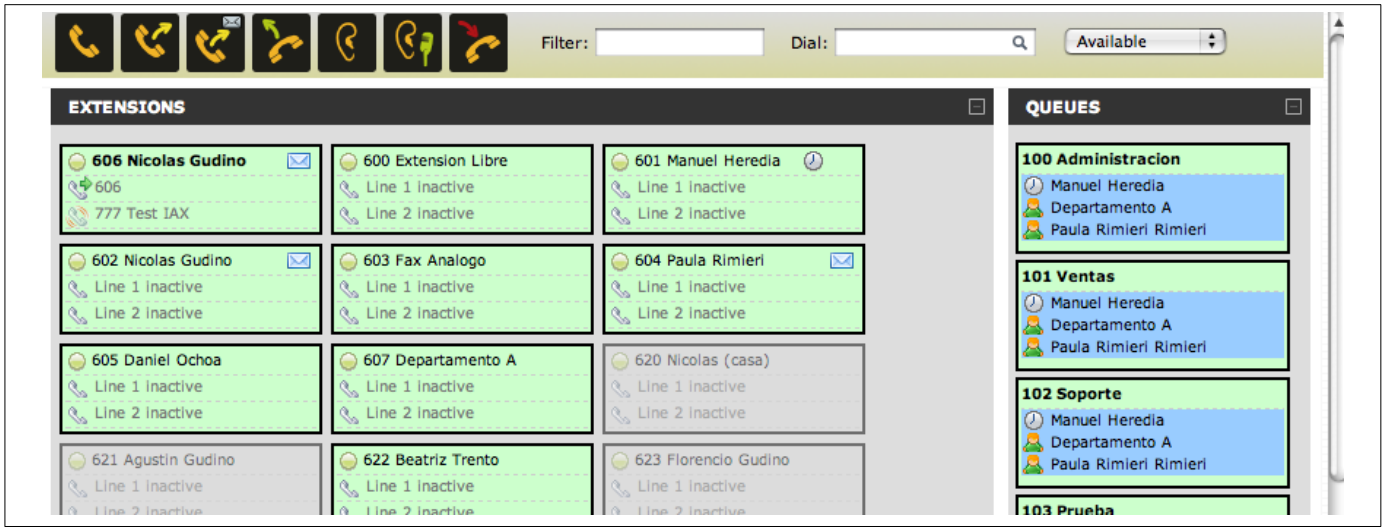


Рисунок 25-1. FOP2

Вы можете найти FOP на http://www.asternic.org, и FOP2 на http://www.fop2.com.

Состояние очереди и отчеты

В большинстве call-центров недостаточно просто правильно маршрутизировать вызовы. Равное значение для большинства колл-центров имеет способность руководящего и управленческого персонала определять как работают очередь и агенты. Для этого будут полезны две вещи: информация о состоянии живой очереди и некоторый пакет отчетов.

Отображение состояния очереди

Состояние очереди часто отображается на большой настенной панели или доске для чтения. Вот некоторые из видов информации, которые могут быть включены:

- Количество агентов, вошедших в систему
- Количество удерживаемых абонентов
- Количество вызовов
- Текущее время удержания
- Среднее время
- Процент отказов
- Уровень обслуживания

Может потребоваться и другая информация; целью отображения состояния очереди является представление как руководящему персоналу, так и агентам очереди быстрой визуальной индикации состояния очереди в определенный момент времени.

Кроме того, в качестве информационного средства могут отображаться показатели производительности группы или агента.

Программное обеспечение Asternic Call Center Stats обеспечивает базовое отображение состояния и доступно в версии lite с открытым исходным кодом. Есть также несколько коммерческих продуктов, которые предлагают эту функцию.

Отчеты очереди

Отчеты об очередях состоят из отчетов и графиков, которые могут использоваться наблюдающим персоналом для анализа производительности очередей и агентов с исторической перспективой. Многие показатели будут аналогичны индикации состояния; однако, цель отчетности — обеспечивать мониторинг укомплектованности, выявления проблем и анализа тенденций.

Мы обсудили несколько интерфейсов отчетов очереди в Главе 13.

Детальная запись вызовов (CDR)

В то время, как Asterisk выполняет достаточно хорошую работу по созданию и хранению CDR, записи находятся в очень сыром формате, что затрудняет их анализ.

Вступление в пакет отчетов CDR. В 1990-х годах, когда тарифы на междугородную связь были сложными и дорогостоящими, целая субиндустрия была порождена компаниями, стремящимися помочь другим компаниям понять смысл сложных тарифов на междугородние вызовы. В настоящее время, когда междугородняя связь является гораздо менее дорогостоящей, а также, как правило, более простой с точки зрения моделей ценообразования, существует меньше необходимости в детальном анализе записей вызовов. Тем не менее, многие из этих опытных компаний добавили поддержку анализа CDR Asterisk; таким образом, если вы хотите отличные возможности отчетности, вы найдете огромную отрасль с большим количеством опытных участников.

Для простого интерфейса записей вызовов популярной программой является CDR-Stats, которая является преемницей чрезвычайно популярного пакета Asterisk-Stat. Этот интерфейс отчетов с открытым исходным кодом предоставляет простой способ изучения записей сведений о вызовах, а также некоторых базовых метрик для шаблонов вызовов.

A2Billing

Проект A2Billing — это не просто биллинговый интерфейс для Asterisk: это, по сути, полноценный VoIP-оператор из коробки. Этот сложный и всеобъемлющий продукт предоставляет большую часть технологий, которые вам понадобятся для предоставления услуг реселлера VoIP.[1]

Платформа A2Billing была щедро выпущена под GNU Affero General Public License (APGL) как open source. Спонсор проекта A2Billing — Star2Billing, предлагает консультационные услуги чтобы помочь вам быстрее освоиться.

Вывод

В этой короткой главе мы предоставили некоторые указатели на популярные графические приложения, которые могут быть использованы в сочетании с Asterisk. Хотя мы не рассматривали их подробно, мы признаем важность FreePBX, который предоставляет интерфейс конфигурации УАТС поверх Asterisk. Если вас интересует полное графическое решение для простой конфигурации УАТС, мы рекомендуем вам взглянуть на него. Чтобы попробовать мы рекомендуем использовать дистрибутив AsteriskNOW, который предоставляет FreePBX в качестве графического интерфейса.

1 Он не может предоставить вам бизнес-смекалку, опыт работы в телефонной компании или автоматическую безопасность, поэтому, пожалуйста, не думайте, что все, что вам нужно сделать, это загрузить A2Billing, прежде чем вы сможете взять на себя AT&T!

ГЛАВА 26

Безопасность

Мы тратим свое время на поиски безопасности и ненавидим ее, когда получаем.
-John Steinbeck

Безопасность для вашей системы Asterisk имеет решающее значение, особенно если система открыта для доступа из Интернета. Для заработка денег злоумышленники могут использовать чужие системы с целью совершения бесплатных телефонных звонков. В этой главе приведены рекомендации по обеспечению большей безопасности для развертывания VoIP.

Сканирование действительных учетных записей

Если вы публикуете свою систему Asterisk в общедоступном Интернете, одна из вещей, которую вы почти наверняка увидите — это проверка действительных учетных записей. В Примере 26-1 содержатся записи журнала из одной из систем Asterisk, выпускаемых авторами.[1] Это сканирование началось с проверки различных общих имен пользователей, а затем продолжилось сканирование нумерованных учетных записей. Люди обычно называют SIP-аккаунты так же, как внутренние номера на АТС. Это сканирование использует этот факт. И всё это приводит к нашему первому вопросу безопасности Asterisk:

> *Совет № 1:* Используйте нечисловые имена пользователей для своих учетных записей VoIP чтобы сделать сложнее их угадывание. Например, в некоторых частях этой книги мы используем MAC-адрес телефона SIP в качестве имени учетной записи в Asterisk.

Пример 26-1. Выдержка лога сканирования аккаунта

```
[Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"123"<sip:123@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found[2]
[Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"1234"<sip:1234@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"12345"<sip:12345@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"123456"<sip:123456@127.0.0.1>' failed for '203.86.167.220:5061' - No
matching peer found
```

1 Реальный IP-адрес в записях журнала был заменен на 127.0.0.1.

2 [22 авг. 15:17:15] УВЕДОМЛЕНИЕ [25690] chan_sip.c: Регистрация от ‘“123” <sip: 123@127.0.0.1>' не удалась для '203.86.167.220:5061' - не найдено совпадения пира

```
[Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"test"<sip:test@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"sip"<sip:sip@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found [Aug 22 15:17:15] NOTICE[25690] chan_sip.c: Registration from
'"user"<sip:user@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"admin"<sip:admin@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"pass"<sip:pass@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"password"<sip:password@127.0.0.1>' failed for '203.86.167.220:5061' - No
matching peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"testing"<sip:testing@127.0.0.1>' failed for '203.86.167.220:5061' - No
matching peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"guest"<sip:guest@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"voip"<sip:voip@127.0.0.1>' failed for '203.86.167.220:5061' - No matching
peer found
[Aug 22 15:17:16] NOTICE[25690] chan_sip.c: Registration from
'"account"<sip:account@127.0.0.1>' failed for '203.86.167.220:5061' - No
matching peer found
...
[Aug 22 15:17:17] NOTICE[25690] chan_sip.c: Registration from
'"100"<sip:100@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found [Aug 22 15:17:17] NOTICE[25690] chan_sip.c: Registration from
'"101"<sip:101@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found [Aug 22 15:17:17] NOTICE[25690] chan_sip.c: Registration from
'"102"<sip:102@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found [Aug 22 15:17:17] NOTICE[25690] chan_sip.c: Registration from
'"103"<sip:103@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found [Aug 22 15:17:17] NOTICE[25690] chan_sip.c: Registration from
'"104"<sip:104@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found [Aug 22 15:17:17] NOTICE[25690] chan_sip.c: Registration from
'"105"<sip:105@127.0.0.1>' failed for '203.86.167.220:5061' - No matching peer
found
```

Это сканирование учетных записей использует тот факт, что ответ, возвращаемый от сервера при попытке регистрации, будет отличаться в зависимости от того, существует или нет учетная запись. Если учетная запись существует, сервер запрашивает аутентификацию. Если учетная запись не существует, сервер немедленно отклонит попытку регистрации. Такое поведение определяется только тем, как определяется протокол. Это приводит нас к нашему второму вопросу по безопасности Asterisk:

> *Совет № 2:* Установите `alwaysauthreject` в `yes` в разделе `[general]` файла */etc/asterisk/sip.conf*. Этот параметр указывает Asterisk отвечать так, как если бы каждая учетная запись была действительной, что делает сканирование на действительные имена пользователей бесполезным. К счастью, это параметр по умолчанию для этой опции. Не меняйте его.

Уязвимости аутентификации

В первом разделе этой главы обсуждалось сканирование для имен пользователей. Даже если у вас есть имена пользователей, которые трудно угадать, очень важно, чтобы у вас были и сильные пароли. Если злоумышленник может получить действительное имя пользователя, он попытается подобрать пароль. Сильные пароли делают перебор намного затруднительней.

По-умолчанию схема аутентификации для протоколов SIP и IAX2 слаба. Аутентификацию проводят с использованием механизма MD5 для запроса-и-ответа. Если злоумышленник способен фиксировать любой трафик вызова, например, SIP-вызов, сделанный с ноутбука в открытой беспроводной сети, гораздо проще работать с подбором пароля перебором, так как для этого не потребуются запросы проверки подлинности на сервере.

> *Совет № 3:* используйте надежные пароли. В Интернете имеется бесчисленное количество ресурсов, которые помогут определить, что является надежным паролем. Есть также много надежных генераторов паролей. Используй их!

IAX2 предоставляет возможность использования проверки подлинности на основе ключа, а также полного шифрования вызова. Поддержка SIP в Asterisk включает поддержку TLS, которая обеспечивает шифрование для сигнализации SIP.

> *Совет № 4:* Если вы используете IAX2, используйте аутентификацию на основе ключа. Это более сложный метод проверки подлинности, чем метод ответа на запрос по умолчанию на основе MD5. Для дальнейшего повышения безопасности с помощью IAX2 используйте параметр для шифрования всего вызова. Если вы пользуетесь SIP, используйте TLS для шифрования SIP-сигнализации. Это позволит предотвратить успешный захват запроса аутентификации у сервера.

Дополнительные сведения о настройке шифрования IAX2 или SIP см. в Главе 7.

Fail2ban

В последних двух разделах обсуждались атаки, включающие сканирование для действительных имен пользователей и подбора паролей перебором. Fail2ban — это приложение, которое может просматривать ваши журналы Asterisk и обновлять правила брандмауэра, чтобы блокировать источник атаки в ответ на большое количество неудачных попыток аутентификации.

> *Совет № 5:* Используйте Fail2ban при размещении услуг Voice over IP в ненадежных сетях, чтобы автоматически обновлять правила брандмауэра для блокировки источников атак.

Установка

Fail2ban доступен как пакет во многих дистрибутивах. Кроме того, вы можете установить его из источника, загрузив с веб-сайта Fail2ban. Чтобы установить его на Ubuntu, используйте следующую команду:

```
$ sudo apt-get install fail2ban
```

Чтобы установить Fail2ban на RHEL, вы должны включить репозиторий EPEL. Дополнительные сведения о репозитории EPEL см. в разделе «Сторонние репозитории» в Главе 3. После того, как репозиторий включен, Fail2ban можно установить, выполнив следующую команду:

```
$ sudo yum install fail2ban
```



Установка Fail2ban из пакета будет включать скрипт инициализации, чтобы гарантировать запуск при загрузке машины. Если вы устанавливаете из исходных кодов, убедитесь, что вы предпринимаете необходимые шаги для обеспечения того, чтобы Fail2ban всегда работал.

iptables

Чтобы Fail2ban мог делать что-нибудь полезное после обнаружения атаки, вы также должны установить *iptables*. Чтобы убедиться что он установлен в Ubuntu, используйте следующую команду:

```
$ sudo apt-get install iptables
```

Чтобы убедиться, что *iptables* установлен на RHEL, используйте следующую команду:

```
$ sudo yum install iptables
```

Вы можете убедиться в том, что *iptables* установлен, запустив команду *iptables*. Опция *-L* запрашивает отображение текущих правил брандмауэра. В этом случае не отображаются сконфигурированные правила:

```
$ sudo iptables -L

Chain INPUT (policy ACCEPT)
targetprot opt source               destination

Chain FORWARD (policy ACCEPT)
targetprot opt source               destination

Chain OUTPUT (policy ACCEPT)
targetprot opt source               destination
```

Отправка электронной почты

Это интересно и полезно разрешить Fail2ban отправлять по электронной почте системному администратору уведомления когда он блокирует IP-адрес. Для этого необходимо установить MTA. Если вы не знаете, какой из них использовать, во время тестирования для написания этой главы использовался Postfix. Чтобы установить Postfix на Ubuntu, используйте следующую команду. Вам может быть предложено ответить на пару вопросов установщиком:

```
$ sudo apt-get install postfix.
```

Чтобы установить Postfix на RHEL, используйте следующую команду:

```
$ sudo yum install postfix.
```

Чтобы протестировать установку вашего MTA, вы можете отправить быстрое письмо используя *mutt*. Чтобы установить его — используйте те же команды установки, которые указаны для установки Postfix, но замените имя пакета на *mutt*. Затем запустите следующие команды для проверки MTA:

```
$ echo «Просто тестирование.» > email.txt
$ mutt -s "Тестирование" youraddress@shifteight.org < mail.txt
```

Конфигурация

Первым файлом, который необходимо настроить, является файл конфигурации ведения журнала Asterisk. Его содержимое в рабочей системе /etc/asterisk/logger.conf. Убедитесь, что у вас установлены хотя бы `dateformat` и `messages`, поскольку они необходимы для Fail2ban:

```
[general]

dateformat = %F %T

[logfiles]

console => notice,warning,error,debug
```

```
messages => notice,warning,error
```

Следующий файл конфигурации, который должен быть создан — это тот, который учит Fail2ban, что нужно отслеживать в файлах Asterisk. Поместите следующее содержимое в новый файл с именем */etc/fail2ban/filter.d/asterisk.conf*:

```
[INCLUDES]

# Прочитайте префиксы common. Если какие-либо настройки доступны -
# прочитайте про них из common.local
# before = common.conf

[Definition]

#_daemon = asterisk

# Опция:          failregex
# Примечание:     regex соответствует сообщениям о сбоях паролей в файле
#                 журнала.
#                 Хост должен быть согласован группой названием «host».
#                 Тег «<HOST>» может использоваться для стандартного
#                 сопоставления IP/хостов и является только альясом для
#                 (?:::f{4,6}:)?(?P<host>\S+)
# Значения:       TEXT
#

# *** Все строки ниже должны начинаться с NOTICE
#     Некоторые строки должны быть обернуты двумя пробелами из-за
#     требований книги. Все новые строки должны начинаться с NOTICE.
#
failregex = NOTICE.* .*: Registration from '.*' failed for '<HOST>'
 - Wrong password
            NOTICE.* .*: Registration from '.*' failed for '<HOST>'
- No matching peer found
            NOTICE.* .*: Registration from '.*' failed for '<HOST>'
- Username/auth name mismatch
            NOTICE.* .*: Registration from '.*' failed for '<HOST>'
- Device does not match ACL
            NOTICE.* <HOST> failed to authenticate as '.*'$
            NOTICE.* .*: No registration for peer '.*' \(from <HOST>\)
            NOTICE.* .*: Host <HOST> failed MD5 authentication for '.*' (.*)
            NOTICE.* .*: Failed to authenticate device .*@<HOST>.*

# Параметр:       ignoreregex
# Примечание:     регулярное выражение (regex) игнорируется. Если это
# выражение (regex) совпадает — строка игнорируется.
# Значение:       TEXT
#
ignoreregex =
```

Затем вы должны включить новый фильтр Asterisk, который вы только что создали. Для этого добавьте следующее содержимое в */etc/fail2ban/jail.conf*. Вам нужно будет изменить параметры `dest` и `sender`, чтобы указать соответствующие адреса электронной почты для заголовков `To` и `From`:

```
[asterisk-iptables]

enabled     = true
filter      = asterisk
action      = iptables-allports [name = ASTERISK, protocol = all]
```

```
            sendmail- whois [name = ASTERISK, dest=me@shifteight.org,
            sender=fail2ban@shifteight.org]
logpath     = /var/log/asterisk/messages
maxretry    = 5
bantime = 259200
```

Наконец, есть несколько параметров раздела `[DEFAULT]` файла */etc/fail2ban/jail.conf*, который должен быть обновлен. Опция `ignoreip` указывает список IP-адресов, которые никогда не должны блокироваться. Это хорошая идея, чтобы перечислить ваш(и) IP-адрес(а) здесь, чтобы вы никогда не блокировали себя если ошибетесь, например, при попытке настроить телефон.[3] Вы также должны рассмотреть возможность добавления других IP-адресов, например вашего SIP-провайдера. Белый список хороших IP-адресов защищает вас от злоупотребления конфигурацией Fail2ban. Умный атакующий может вызвать отказ в обслуживании, создав серию пакетов, которые приведут к тому, что Fail2ban блокирует IP-адрес по своему выбору.

Параметр `destemail` также должен быть установлен. Этот адрес будет использоваться для сообщений электронной почты, не относящихся к фильтру Asterisk, например, по электронной почте Fail2ban уведомляет о первом запуске. Вот как вы настраиваете эти параметры:

```
[DEFAULT]

# Можно указать несколько адресов, разделенных пробелом.
ignoreip = 127.0.0.1 10.1.1.1

destemail = youraddress@shifteight.org
```

Безопасность файлов лога Asterisk

Формат записи журнала, используемый в нашем примере конфигурации для fail2ban, предназначен для чтения человеком. Анализ и автоматизация действий на основе результата несколько безобразны. В настоящее время предпринимаются усилия по созданию файла журнала событий, связанных с безопасностью, предназначенного для автоматической обработки. К сожалению, он не использовался нами активно и мы не видели примеров инструментов, которые обрабатывают записи из этого файла журнала.

Чтобы включить файл журнала безопасности, добавьте следующую строку в раздел `[logfiles]` файла */etc/asterisk/logger.conflogfiles*:

```
[logfiles]

security => security
```

Asterisk 11 помещает записи в этот файл для событий, которые происходят в AMI, а также `chan_sip` (обозначается как `SIP` в записях журнала событий безопасности). Тело сообщения журнала представляет собой список пар ключ/значение, разделенный запятыми. Каждая пара ключ/значение представляет собой немного метаданных о событии безопасности. Вот пример записи журнала безопасности о попытки войти в AMI с недопустимым пользователем:

```
[13.12.08:46:09] SECURITY [995] res_security_log.c:
SecurityEvent = "InvalidAccountID", EventTV = "1355406369-716573" ,
Severity = "Error", Service = "AMI", EventVersion = "1",
AccountID = "foo", SessionID = "0x7f793ac32220",
LocalAddress = «IPV4/TCP/0.0.0.0/5038»,
RemoteAddress = «IPV4/TCP/127.0.0.1/58491», SessionTV=«0-0»
```

Некоторые события считаются ошибками, а другие просто информационными. Значение

3 Лейф усвоил этот трудный путь. Он думал, что его АТС не работает, в то время как у Рассела и Джима не было проблем с подключением к конференц-мосту. Оказалось, что Fail2ban запретил ему пользоваться собственной АТС.

связанное с ключом `Severity` будет указать либо `Error` или `Informational`.

Значение, связанное с ключом `SecurityEvent`, определяет тип события, произошедшего в системе. Asterisk 11 (по крайней мере, от Asterisk 11.1.0) способен передавать следующие типы событий безопасности. Некоторые из них еще не используются ни в одной из частей Asterisk. Если вы работаете над инструментом, который использует эти события, стоит знать что они существуют если будут использоваться в будущем.

`FailedACL`

ACL, определенный параметрами конфигурации `permit` и `deny`, вызвал отказ в запросе. Это передается как для AMI, так и для SIP.

`InvalidAccountID`

Запрос был получен с недопустимым идентификатором учетной записи. Обычно это плохое имя пользователя. Это передается для AMI.

`SessionLimit`

Количество сеансов, разрешенных для данного пользователя, было превышено. Это передается как для AMI, так и для SIP.

`MemoryLimit`

Предел памяти превышен. Это событие определено, но не используется в настоящее время.

`LoadAverageLimit`

Лимит нагрузки превышен. Это событие определено, но не используется в настоящее время.

`RequestNotSupported`

Получен запрос, который не поддерживается. Это событие определено, но не используется в настоящее время.

`RequestNotAllowed`

Был получен запрос, который был административно запрещен. Это передается для AMI.

`AuthMethodNotAllowed`

Был предпринят метод аутентификации, который не разрешен. Это событие определено, но не используется в настоящее время.

`RequestBadFormat`

Получен плохо отформатированный запрос. Это событие передается для AMI.

`SuccessfulAuth`

Аутентификация прошла успешно. Это событие передается как для SIP, так и для AMI.

`UnexpectedAddress`

Сообщение было получено от неожиданного исходного адреса для сеанса уже выполняющегося. Это событие определено, но не используется в настоящее время.

`ChallengeResponseFailed`

Ошибка аутентификации на основе запроса-ответа. Это событие выбрано как для SIP, так и для AMI.

`InvalidPassword`

Для попытки аутентификации был предоставлен неверный пароль. Это событие передается как для SIP, так и для AMI.

`ChallengeSent`

Запрос был отправлен для проверки подлинности на основе запроса-ответа. Это событие передается для SIP.

`InvalidTransport`

Попытка использовать административно запрещенный транспорт (например, с использованием UDP, когда для этого пира разрешен только TCP). Это событие передается для SIP.

Зашифрованные медиаданные

Помните, что звук для VoIP обычно передается в незашифрованном формате. Любой, кто может перехватить трафик — может прослушивать телефонный звонок. К счастью, Asterisk поддерживает шифрование медиаданных вызовов VoIP. Если вы используете SIP, то можете зашифровать медиа с помощью SRTP. IAX2 также полностью поддерживает шифрование вызовов. Подробную информацию о шифровании медиаданных можно найти в Главе 7.

Совет № 6: Шифруйте медиаданные для вызовов в ненадежных сетях с использованием шифрования SRTP или IAX2.

Уязвимости диалплана

Диалплан Asterisk — еще одна область, в которой важно учитывать безопасность. Диалплан можно разбить на несколько контекстов, чтобы обеспечить контроль доступа к внутренним номерам. Например, вы можете позволить своим офисным телефонам совершать звонки через своего провайдера. Тем не менее, вы не хотите разрешать анонимным абонентам, которые попадают в основное меню вашей компании, иметь возможность позвонить через вашего провайдера. Используйте контексты, чтобы гарантировать только известным вам абонентам доступ к услугам, которые стоят вам денег.

Совет № 7: Стройте контексты диалплана с большой осторожностью. Кроме того, избегайте размещения каких-либо номеров, которые могут стоить вам денег, в контексте `[default]`.

Одна из последних уязвимостей диалплана Asterisk, которые были обнаружены и опубликованы — это идея инъекции диалплана. Уязвимость диалплан-инъекций начинается с номера, у которого есть шаблон, который заканчивается символом match-all - точкой. Возьмите это расширение в качестве примера:

```
exten => _X.,1,Dial(IAX2/otherserver/${EXTEN},30)
```

Шаблон для этого расширения соответствует всем номерам (любой длины), начинающимся с цифры. Подобные шаблоны довольно распространены и удобны. Номер затем посылает этот вызов на другой сервер, используя протокол IAX2, с таймаутом набора 30 секкунд. Обратите внимание на использование здесь переменной `${EXTEN}`. Вот где существует уязвимость.

В мире VoIP нет причин, чтобы набираемое расширение было числовым. На самом деле, довольно часто использование SIP позволяет набирать кого-то по имени. Так как нечисловые символы могут быть частью набираемого номера, что произойдет, если кто-то отправит вызов этому расширению?

```
1234&DAHDI/g1/12565551212
```

Подобный вызов является попыткой использовать уязвимость, называемую диалплан-инъекцией. В предыдущем определении расширения, когда `${EXTEN}` будет оценено, оператор `Dial()` выполнит:

```
exten => _X,1,Dial(IAX2/otherserver/1234&DAHDI/g1/12565551212,30)
```

Если в системе настроен PRI — этот вызов пойдет на номер, выбранный злоумышленником, даже если вы явно не предоставили доступ к PRI этому вызывающему абоненту. Эта проблема может стоить вам больших денег.

Для избежания подобной проблемы существуют (по крайней мере) два подхода. Первый и самый простой подход — всегда использовать строгое сопоставление шаблонов. Если вы знаете длину внутренних номеров, которые ожидаете и ожидаете только числовые номера, используйте строгое сопоставление с числовым шаблоном. Например, этот пример будет работать, если вы ожидаете только четырехзначных числовых внутренних номеров:

```
exten => _XXXX,1,Dial(IAX2/otherserver/${EXTEN},30)
```

Другой подход к смягчению уязвимостей, связанных с использованием диалплана, заключается в использовании `FILTER()` диалплана. Возможно, вы хотите разрешить числовые номера любой длины. `FILTER()` делает это безопасным:

```
exten => _X.,1,Set(SAFE_EXTEN=${FILTER(0-9,${EXTEN})})
    same => n,Dial(IAX2/otherserver/${SAFE_EXTEN},30).
```

Для получения дополнительной информации о синтаксисе функции диалплана `FILTER()` смотрите вывод *core show function FILTER* в командой строке Asterisk.

> *Совет № 8:* Будьте осторожны с уязвимостями инъекцими диалплана. Используйте строгое сопоставление или шаблоны функции диалплана FILTER() чтобы избежать этих проблем.

Обеспечение безопасности сети Asterisk API

FastAGI и AMI - это два сетевых API, обычно используемых в развертываниях Asterisk. Более подробную информацию об AGI см. в Главе 21. Дополнительную информацию об AMI см. в Главе 20.

В случае с FastAGI нет доступа к шифрованию и аутентификации. Администратор должен решить, что единственное сообщение, разрешенное для сервера FastAGI - от Asterisk.

Протокол AMI включает аутентификацию, но она очень слаба. Кроме того, данные, которыми обмениваются через AMI часто являются конфиденциальными, с точки зрения приватности. Крайне важно обеспечить безопасность AMI. Лучше всего выставлять AMI только доверенным сетям. Если необходимо предоставить доступ в ненадежной сети, мы рекомендуем разрешать подключения только с использованием SSL.

Очень важно понять какую силу дает AMI. Если пользователю AMI предоставлены все доступные разрешения, он сможет выполнять произвольные команды в вашей системе. Если учетная запись имеет возможность обновлять файлы конфигурации, она сможет добавить расширение в диалплан, которое запускает приложение `System()`, позволяющее ему запускать любую команду, которую он захочет. Если он также имеет доступ к инициированию вызовов, то сможет инициировать вызов этого расширения, что приведет к выполнению этой команды. Будьте осторожны при открытии доступа администратора в вашей системе и ограничте разрешения, предоставляемые каждой учетной записи в */etc/asterisk/manager.conf*.

> *Совет № 9:* Безопасные сетевые API Asterisk. Используйте правила брандмауэра, для ограничения доступа к серверу FastAGI. Используйте шифрование в AMI. Максимально ограничте доступ к учетным записям AMI.

IAX2 отказ в обслуживании

Хотя протокол SIP является текстовым протоколом, IAX2 является бинарным протоколом. Стандарт IAX2 - это RFC 5456. Каждый пакет IAX2 содержит номер вызова, который используется для связывания пакета с активным вызовом. Это аналогично заголовоку `Call-ID` в SIP. Номер вызова IAX2 - это 15-битное поле. Оно достаточно велико, чтобы иметь дело с количеством вызовов, которые будут реальными в одной системе. К сожалению, оно также достаточно мало, что злоумышленнику достаточно легко отправить несколько маленьких пакетов, чтобы в течение короткого периода времени использовать все доступные номера вызовов в системе, что привело бы к атаке отказа в обслуживании.

Поддержка IAX2 в Asterisk была изменена для автоматической защиты от этого типа атаки. Эта защита называется поддержкой токена вызова и требует трехстороннего установления связи до того, как будет назначен номер вызова. Однако, более старые версии Asterisk и некоторых не-Asterisk IAX2-реализации могут не поддерживать этого, поэтому есть несколько вариантов, которые позволяют вам настраивать это поведение.

По умолчанию, механизмы безопасности включены и никаких изменений конфигурации не требуется. Если по какой-либо причине вы хотите полностью отключить поддержку токена вызова, вы можете сделать это, используя следующую конфигурацию в */etc/asterisk/iax.conf*:

```
[general]
calltokenoptional = 0.0.0.0/0.0.0.0
maxcallnumbers = 16382
```

При конфигурации по умолчанию хост, который может передавать обмен токенами вызовов, все еще может использовать таблицу номеров вызовов. Обмен токенами вызовов гарантирует, что номера вызовов будут назначены только после того, как мы узнаем, что не получили запрос с поддельным IP-адресом источника. Как только мы узнаем, что запрос является законным, принудительное ограничение ресурсов на хост становится достижимо. Рассмотрим следующие варианты в *iax.conf*:

```
[general]

; Установите ограничение числа вызовов по умолчанию на хост
maxcallnumbers = 16

[callnumberlimits]

; Установите другой номер вызова для всех хостов в указанном диапазоне.
192.168.1.0/255.255.255.0 = 1024

[some_peer]

; Адрес динамического пира неизвестен до тех пор, пока этот пир не
; будет зарегистрирован. Ограничение номера вызова может быть указано в
; разделе peer, а не в секции callnumberlimits.

type = peer
host = dynamic
maxcallnumbers = 512
```

Если пир еще не поддерживает проверку токенов вызовов, но вы хотели бы включить её, как только обнаружите что пир был обновлен для его поддержки, есть опция, которая позволяет это сделать:

```
[some_other_peer]

requirecalltoken = auto
```

Если вы хотите разрешить гостевой доступ через IAX2, скорее всего, вы захотите отключить проверку токена для неавторизованных вызовов. Это гарантирует, что наибольшее количество людей может позвонить в вашу систему через IAX2. Однако, если вы это сделаете, то также должны установить параметр, предоставляющий глобальное ограничение тому, сколько номеров вызова может быть использовано хостами, которые не прошли проверку на токен:

```
[general]

maxcallnumbers_nonvalidated = 2048

[guest]
```

```
type = user
requirecalltoken = no
```

Если в любой момент вы захотите увидеть статистику использования номера вызова в своей системе, выполните команду *iax2 show callnumber usage* в Asterisk CLI.

> *Совет № 10:* Будьте спокойны, зная, что IAX2 обновлен, чтобы обезопасить себя от отказа из-за исчерпания номера вызова. Если вы в некоторых случаях должны отключить эти функции безопасности, используйте предоставленные параметры, чтобы ограничить вашу атаку.

Другие меры по снижению риска

В Asterisk есть еще несколько полезных функций, которые могут быть использованы для снижения риска атак. Во-первых, использование опций `permit` (разрешить) и `deny` (запретить) для создания списков контроля доступа (ACL) для привилегированных учетных записей. Рассмотрим АТС с SIP-телефонами в локальной сети, но которая также принимает SIP-вызовы из общедоступного Интернета. Звонки, поступающие через Интернет, получают доступ только к главному меню компании, а местные SIP-телефоны имеют возможность делать исходящие звонки, которые стоят вам денег. В этом случае очень хорошая идея - установить ACL для обеспечения того, чтобы только устройства в вашей локальной сети могли использовать учетные записи для телефонов. Вот пример этого в */etc/asterisk/sip.conf*:

```
[phoneA] ; Используйте лучшее имя учетной записи, чем это.

type = friend

; Начните с запрета всех.
deny = 0.0.0.0/0.0.0.0

; Разрешить соединения, созданные из 192.168.X.X, для попытки
; аутентификации от этой учетной записи.
permit = 192.168.0.0/255.255.0.0
```

Параметры `permit` и `deny` принимаются почти везде, где настроены подключения к IP-службам. Еще одно полезное место для ACL - в */etc/asterisk/manager.conf*, чтобы ограничить использование учетной записи AMI на одном хосте, который должен использовать менеджерский интерфейс.

В Asterisk 11 был введен второй метод конфигурирования списков ACL. Если вы применяете одни и те же правила ACL в нескольких местах, вы найдете это очень полезным. Именованные ACL могут быть определены в */etc/asterisk/acl.conf*.

```
[named_acl_1]
deny=0.0.0.0/0.0.0.0
permit = 10.1.1.50
permit = 10.1.1.55

[named_acl_2] ; Именованные ACL поддерживают IPv6.
deny=::
permit=::1/128

[local_phones]
deny=0.0.0.0/0.0.0.0
permit=192.168.0.0/255.255.0.0
```

Как только имена ACL определены в *acl.conf* Asterisk загрузит их, используя *reload acl*. После загрузки они должны быть доступны через CLI Asterisk:

```
* CLI> reload acl

* CLI> acl show
```

```
acl
---
named_acl_1
named_acl_2
local_phones

* CLI> acl show named_acl_1

ACL: named_acl_1
------------ --------------------------------
  0:  deny - 0.0.0.0/0.0.0.0
  1: allow - 10.1.1.50/255.255.255.255
  2: allow - 10.1.1.55/255.255.255.255
```

Теперь вместо того, чтобы потенциально повторять одни и те же записи `permit` и `deny` в нескольких местах, вы можете применить ACL по его имени. Обновленная версия первого примера использования ACL теперь будет выглядеть следующим образом:

```
[phoneA]
type = friend
acl = local_phones
```

Совет № 11: Используйте ACL, если это возможно, во всех привилегированных учетных записях для сетевых служб.

Другим способом снижения риска безопасности является настройка лимитов вызовов. Рекомендованный метод реализации лимитов вызовов является использование функций диалплана `GROUP()` и `GROUP_COUNT()`. Вот пример, который ограничивает количество вызовов от каждого SIP-пира до двух раз:

```
exten => _X.,1,Set(GROUP(users)=${CHANNEL(peername)})

; *** Эта строка не должна иметь разрывов строки
    same => n,NoOp(Существующий ${GROUP_COUNT(${CHANNEL((peername)})}
вызывает аккаунт ${CHANNEL(peername)}.)
    same => n,GotoIf($[${GROUP_COUNT(${CHANNEL(peername)})} > 2]?denied:continue)
    same => n(denied),NoOp(Слишком много звонков. Кладу трубку.)
    same => n,HangUp()
    same => n(continue),NoOp(продолжить обработку вызова как обычно).
```

Совет № 12: Используйте лимиты вызовов чтобы гарантировать, что в случае если учетная запись скомпрометирована, ее нельзя использовать для выполнения сотни телефонных звонков одновременно.

Разрешения CLI

У CLI Asterisk много мощи. Есть команды для изменения диалплана, инициирования вызовов и их отбоя, среди многих других. Asterisk имеет возможность предоставить локальному пользователю доступ к подмножеству доступных команд CLI. Это делается с помощью файла конфигурации */etc/asterisk/cli_permissions.conf*.

По умолчанию любой локальный пользователь, имеющий доступ к CLI Asterisk, может запускать все команды CLI. Если вы хотите изменить это, начните с добавления опции `default_perm` в раздел `[general]` файла *cli_permissions.conf*:

```
[general]
;
; Установка default_perm=permit разрешает пользователям доступ ко всем
; командам по умолчанию, если в этом файле не указано иное.
```

```
;
default_perm = deny
```

Остальная часть файла используется для установки разрешений для локальных пользователей или групп. Например, если вы хотите предоставить полный доступ пользователю с именем `admin`, вы должны добавить его в файл:

```
[admin]
allow = all ; Предоставление полного доступа определенному пользователю.
```

Вы также можете установить разрешения для локальной группы. Например, вы можете создать локальную группу под названием `asterisksupport`. Вы можете предоставить доступ только к некоторым базовым диагностическим командам для всех пользователей этой группы. Префикс `@` в названии раздела указывает Asterisk, что вы устанавливаете разрешения для локальной группы:

```
[@asterisksupport]
deny = all
allow = sip show        ; Все команды, начинающиеся с «sip show»
allow = core show       ; Все команды, начинающиеся с «core show»
```

Совет № 13: Установите разрешения CLI, если вы хотите предоставить доступ к CLI кому-то кроме администраторов системы.

Ресурсы

Некоторые уязвимости требуют модификации исходного кода Asterisk. Когда эти проблемы обнаружены — команда разработчиков Asterisk выпускает новые версии, которые содержат только исправления для проблем безопасности, что позволяет быстро и легко обновлять их. Когда это происходит, команда разработчиков Asterisk также публикует консультативный документ безопасности в котором обсуждаются детали уязвимости. Рекомендуем вам подписаться на список рассылки asterisk-announce чтобы вы езнали об этих проблемах, когда они появляются.

Совет № 14: Подпишитесь на список рассылки *asterisk-announce*, чтобы оставаться в курсе уязвимостей безопасности Asterisk.

Одним из самых популярных инструментов для сканирования SIP-аккаунтов и взлома пароля является SIPVicious. Мы настоятельно рекомендуем вам взглянуть на него и использовать для аудита ваших собственных систем. Если ваша система находится в Интернете, найдутся те, кто, скорее всего, будут работать против вашей системы, поэтому убедитесь, что вы сделали это в первую очередь.

Другим ресурсом для всех вещей, связанных с безопасностью VoIP, является список рассылки VOIPSEC на VOIPSA.org. На сайте также есть дополнительные ресурсы.

Наконец, в проекте VoIP Blacklist Project есть полезная информация. Автор предоставляет список адресов, известных как источники атак VoIP, а также инструкции о том, как заблокировать все адреса в этом списке. Автор также предоставляет образец сценария под названием AntiToll, который блокирует все адреса за пределами США.

Заключение—лучший идиот

В технологической отрасли есть принцип: «Как только что-то сделается идиотским, природа будет изобретать лучшего идиота». Дело в том, что никакие усилия по развитию не могут считаться завершенными. Всегда есть место для улучшения.

Когда дело доходит до безопасности, вы всегда должны помнить что люди, которые хотят использовать вашу систему, очень мотивированы. Независимо от того, насколько безопасна ваша система, кто-то всегда будет искать брешь в безопасности.

Мы не выступаем за паранойю, но предполагаем кое-что, о чем мы здесь написали, отнюдь не последнее слово в безопасности VoIP. Хотя и старались быть настолько всеобъемлющими, как можем быть в этой книге, вы должны нести ответственность за безопасность своей системы.

Как бесплатная Интернет-телефония становится все более распространенной, преступники будут упорно работать, чтобы найти недостатки и использовать их.

Asterisk: Будущее телефонии

Эй, я только что встретил тебя.,
И это безумие,
Но вот мой номер,
Может, позвонишь мне?

- Карли Рей Джепсен

Мы подошли к заключительной главе этой книги. Мы много писали (и эта книга с годами расширилась), но надеемся ясно дали понять, что эта книга просто поцарапала поверхность явления, называемого Asterisk. В завершение, мы хотим потратить некоторое время на изучение того, что мы можем увидеть в Asterisk и телефонии с открытым исходным кодом в ближайшем будущем.

Когда мы писали первое издание *Asterisk: Будущее телефонии*, мы уверенно утверждали что коммуникационные двигатели с открытым исходным кодом, такие как Asterisk, вызовут сдвиг в мышлении, который преобразует телекоммуникационную индустрию. Во многих отношениях наше убеждение оказалась верным. В то время как телекоммуникационная отрасль все еще находится в процессе развития, Asterisk сыграла ключевую роль в стимулировании сдвига в мышлении, который затронул всю отрасль.

Проблемы традиционной телефонии

Хотя Александр Грэхем Белл наиболее известен как отец телефона,[1] реальность такова, что во второй половине 1800-х годов десятки умов работали над целью передачи голоса по телеграфным линиям. Это были в основном деловые люди, стремящимися создать продукт, с помощью которого они могли бы сделать себе состояние.

Мы привыкли думать о традиционных телефонных компаниях как о монополиях, но это было не так в их первые дни. Ранняя история телефонных услуг проходила в очень конкурентной среде, с появлением новых компаний по всему миру, часто без особого уважения к патентам, которые они могли нарушить. Многие известные монополии получили свое начало через ведение (и победу) патентных войн.

Интересно сравнить историю телефона с историей GNU Linux и Интернета. В то время как телефон был создан как коммерческое мероприятие, а телекоммуникационная индустрия была создана через судебные процессы и корпоративные поглощения, Linux и Интернет возникли из академического сообщества, которое всегда ценило обмен знаниями выше прибыли.

Культурные различия очевидны. Телекоммуникационные технологии, как правило, являются закрытыми, запутанными и дорогостоящими, в то время как сетевые технологии сравнительно открыты, хорошо документированы и конкурентоспособны.

1 Вы когда-нибудь слышали об Элише Грее или Антонио Меуччи?

Закрытое мышление

Если сравнивать культуру телекоммуникационной отрасли с культурой интернета, то иногда трудно поверить, что они связаны между собой. Технология Интернета была разработана в значительной степени учеными и энтузиастами, в то время как вклад в развитие ТфОП невозможен для любого человека. Это эксклюзивный клуб и членство в нем открыто не для всех.[2]

Хотя МСЭ является санкционированным органом Организации Объединенных Наций, ответственным за международные Телекоммуникации, многие протоколы VoIP (SIP, MGCP, RTP, STUN) поступают не от МСЭ, а от IETF (который публикует все свои стандарты бесплатно для всех и позволяет любому представить проект интернета для рассмотрения).

Открытые протоколы, такие как SIP, могут иметь тактическое преимущество перед протоколами МСЭ, такими как H.323, благодаря легкости их получения.[3] Хотя H.323 был широко используем в качестве протокола VoIP в магистрали, гораздо сложнее найти конечные точки на основе H.323; новые продукты с гораздо большей вероятностью будут поддерживать SIP.

Успех открытого подхода IETF не остался незамеченным МСЭ. Со времени первого издания этой книги МСЭ сделал все рекомендации МСЭ-Т и МСЭ-Р доступными для бесплатного скачивания в формате PDF со своего веб-сайта.

Что касается Asterisk, он охватывает как прошлое, так и будущее—поддержка H.323 доступна, хотя сообщество по большей части избегает H.323 в пользу протокола IETF SIP (и собственного протокола VoIP Asterisk — IAX, который несколько упал из-за доминирования SIP в отрасли).

Ограниченное соблюдение стандартов

Одна из самых странных вещей во всех стандартах в мире устаревших телекоммуникаций — кажущаяся неспособность различных производителей последовательно внедрять их. Каждый производитель стремится к полной монополии, поэтому концепция интероперабельности имеет тенденцию занимать заднее сиденье, чтобы быть первым на рынке с новой творческой идеей.

Классическим примером этого являются протоколы ISDN. Развертывание ISDN было (и во многом до сих пор является) болезненным и дорогостоящим предложением, поскольку каждый производитель решил реализовать его немного по-своему. ISDN вполне мог бы помочь в создании массовой публичной сети передачи данных за 10 лет до Интернета. К сожалению, из-за своей стоимости, сложности и совместимости ISDN никогда не доставлял намного большего, чем голос, со случайным подключением к видео или данным для тех, кто готов платить. ISDN довольно распространен (особенно в Европе и в Северной Америке в более крупных реализациях УАТС), но он не обеспечивает все с возможности, которые были предусмотрены для него.

По мере того, как VoIP становится все более и более вездесущим, потребность в ISDN исчезнет. Это происходит прямо сейчас и очень быстро, так как все больше и больше новых систем используют только IP-телефонию.

Медленный цикл выпусков

На то, чтобы признать тенденции, а тем более выпустить совместимый с ними продукт, могут уйти месяцы, а иногда и годы. Кажется, что прежде чем новая технология может быть принята, она должна быть проанализирована до смерти, а затем должна успешно пройти через различные слои бюрократии, прежде чем она даже запланирует цикл развития. Должны пройти месяцы или даже годы, прежде чем можно ожидать какого-либо полезного продукта. Когда эти продукты, наконец,

2 Сравните это со страницей членства IETF, которая гласит: "IETF не является членской организацией (без карточек, без сборов, без секретных рукопожатий :-) ...она открыта для любого заинтересованного лица...Добро пожаловать в IETF.- Поговорим об общине!

3 Многие люди, знакомые с обоими протоколами, предполагают, что H.323 на самом деле технически лучше. Бетамакс, серьезно?

выпущены, они часто основаны на устаревшем оборудовании; они также, как правило, дороги и предлагают не более минимального набора функций.

Эти медленные циклы выпуска просто не работают в современном мире деловых коммуникаций. В интернете новые идеи могут укорениться в течении нескольких недель и стать жизнеспособными за очень короткий период времени. Поскольку любая другая технология должна адаптироваться к этим изменениям, то и телекоммуникации должны.

Разработки с открытым исходным кодом по своей сути лучше адаптируются к быстрым технологическим изменениям, что дает им огромное конкурентное преимущество.

Впечатляющий крах телекоммуникационной отрасли, возможно, был вызван в значительной степени неспособностью изменяться. Теперь выбора нет: измениться или перестать существовать. Об этом заботятся сообщества разработчиков, такие как Asterisk.

Отказ отпустить прошлое и принять будущее

Традиционные телекоммуникационные компании потеряли связь со своими клиентами. Хотя концепция добавления функциональности за пределы базовой телефонии хорошо понятна, идея о том, что пользователь должен быть тем, кто определяет эту функциональность, не укоренилась.

В наши дни люди обладают почти безграничной гибкостью во всех других формах общения. Они просто не могут понять, почему Телекоммуникации не могут быть поставлены так гибко, как индустрия обещала в течение многих лет. Концепция гибкости не знакома телекоммуникационной отрасли, и вполне может быть не будет знакома до тех пор, пока продукты с открытым исходным кодом, такие как Asterisk, не начнут трансформировать фундаментальную природу отрасли. Это революция, подобная той, которую Linux и Интернет охотно начали более 20 лет назад (а IBM невольно начала с ПК, за 15 лет до этого). Что это за революция? Коммодитизация[4] оборудования, телефонии и программного обеспечения, обеспечивающая распространение индивидуальных телекоммуникационных систем.

Сдвиг парадигмы

В своей статье “Paradigm Shift (Сдвиг парадигмы)” Тим О'Рейли говорит о сдвиге, происходящем в способе доставки технологий (как аппаратных, так и программных).[5] О'Рейли выделяет три тенденции: коммерциализация программного обеспечения, сетевое сотрудничество и настраиваемость программного обеспечения (программное обеспечение как услуга). Эти три концепции предполагают, что телефония с открытым исходным кодом — это идея, время которой пришло.

Обещание телефонии с открытым исходным кодом

> Каждая хорошая работа программного обеспечения начинается с царапин личного зуда разработчика.
>
> - Eric S. Raymond, *The Cathedral & The Bazaar*

В своей книге The Cathedral & The Bazaar (O'Reilly, 2001) Эрик С. Раймонд объясняет, что “При достаточном количестве глазных яблок все ошибки поверхностны." Причина, по которой разработка программного обеспечения с открытым исходным кодом производит такое стабильное качество, проста: дерьмо невозможно скрыть.

4 Превращение продукта в сравнительно дешёвый товар массового потребления.

5 Большая часть следующего раздела-это просто наша интерпретация статьи О'Рейли. Чтобы получить полную суть этих идей, настоятельно рекомендуется прочтение оригинала.

Зуд, который вызывает Asterisk

В эту эпоху разработки пользовательских баз данных и веб-сайтов люди не только устали слышать, что их телефонная система “не может этого сделать”, но, откровенно говоря, они просто не верят в это. Творческие потребности клиентов, в сочетании с ограничениями технологии, породили тип творчества, рожденный необходимостью: инженеры телекоммуникаций подобны участникам эпизода Войны на свалке,[6] пытающимся создать функциональные устройства из кучи несовпадающих компонентов.

Методология разработки проприетарной телефонной системы диктует что она будет иметь огромное количество функций и количество функций будет в значительной степени определять цену. Производители скажут вам что их продукция дает вам сотни функций, но если вам нужно только пять из них, кого это волнует? Хуже того, если есть одна недостающая функция, без которой вы действительно не можете обойтись, ценность этой системы будет разбавлена тем фактом, что она не может полностью удовлетворить ваши потребности.

Тот факт, что клиенту может понадобиться только пять из пятисот функций, игнорируется, а желание клиента иметь пять недоступных функций, которые отвечают потребностям его бизнеса, отклоняется как необоснованное.[7] До тех пор, пока гибкость не станет стандартной, телеком останется застрявшим в прошлом веке — несмотря на все VoIP в мире.

Asterisk решает эту проблему напрямую и решает ее таким образом, каким немногие другие телекоммуникационные системы не могут похвастаться. Это чрезвычайно разрушительная технология, в значительной степени потому, что она основана на концепциях, которые были доказаны снова и снова: “мир с закрытым исходным кодом не может выиграть эволюционную гонку вооружений с сообществами с открытым исходным кодом, которые могут потратить на порядок меньше времени для решения проблемы.”[8]

Открытая архитектура

Одним из камней преткновения традиционной телекоммуникационной отрасли является ее явный отказ сотрудничать с самой собой. Крупные телекоммуникационные гиганты существуют уже более ста лет. Концепция закрытых, проприетарных систем настолько укоренилась в их культуре, что даже попытки соблюдения стандартов запятнаны их желанием получить скачок на конкуренции, добавив одну особенность, которую никто другой не поддерживает. Для примера этого мышления нужно просто посмотреть на продукты VoIP, предлагаемые телекоммуникационной отраслью сегодня. Хотя они утверждают, что стандарты соответствуют, мысль о том, что вы действительно ожидаете увидеть возможность подключить телефон Cisco к коммутатору Nortel или способность системы голосовой почты Avaya интегрироваться через IP к УАТС Siemens, не является тем, что следует обсуждать.

В компьютерной индустрии все по-другому. В начале 80-х годов, если вы купили сервер IBM, вам нужна была сеть IBM и терминалы IBM, чтобы общаться с ним. Теперь этот сервер IBM, вероятно, соединится с терминалами Dell через сеть Cisco (и запустит Linux для этих вещей). Каждый может легко придумать тысячи вариаций на эту тему. Если бы какая-нибудь из этих компаний предложила нам использовать их продукцию только с тем, что они нам скажут, их бы высмеяли.

Телекоммуникационная отрасль сталкивается с теми же изменениями, но не спешит их принимать. Asterisk, с другой стороны, очень спешит не только принять изменения, но и использовать их.

---

6 "Война на свалке" - реалити-шоу, которое было популярно в 2005 году, когда мы написали первое издание этой книги, где команды соревновались, чтобы построить машину из металлолома для решения конкретной задачи.

7 С точки зрения индустрии закрытых источников их отношение понятно. В своей книге *The Mythical Man-Month: Essays on Software Engineering* (Addison-Wesley, 1995) Фред Брукс высказал мнение, что "сложность и коммуникационные затраты проекта растут с квадратом числа разработчиков, в то время как выполненная работа растет только линейно."Без методологии разработки на базе сообщества очень сложно поставлять продукты, которые в лучшем случае являются лишь постепенными улучшениями по сравнению с их предшественниками, а в худшем-просто коллекциями патчей.

8 Eric S. Raymond, The Cathedral and the Bazaar.

IP-телефоны Cisco, Nortel, Avaya и Polycom (приведем лишь несколько) успешно подключены к системам Asterisk. Сегодня в мире нет другой АТС, которая может заявить об этом. Никто. Открытость – это сила Asterisk.

Соблюдение стандартов

В последние несколько лет стало ясно, что стандарты развиваются такими быстрыми темпами, что для того, чтобы идти в ногу с ними, требуется способность быстро реагировать на новые технологические тенденции. Asterisk, в силу того что является открытым ПО, ориентированным на сообщество, уникально подходит для быстрого развития, которое требует соблюдение стандартов.

Asterisk не фокусируется на анализе затрат и выгод или исследовании рынка. Он развивается в ответ на все, что сообщество находит захватывающим или необходимым.

Молниеносная реакция на новые технологии

После того, как Марк Спенсер посетил свое первое мероприятие по тестированию совместимости SIP (SIPIT) у него был рудиментарный, но рабочий стек SIP для Asterisk, написанный в течение нескольких дней. Это было до того, как SIP стал предпочтительным протоколом в мире VoIP, но он видел его ценность и импульс и гарантировал готовность Asterisk.

Такая дальновидность и гибкость характерны для сообщества разработчиков с открытым исходным кодом (и очень необычны для крупной корпорации).

Страстное сообщество

Список *Asterisk-Users* получает более трехсот e-mail-сообщений в день. На него подписано более десяти тысяч человек. Такого рода поддержка сообщества неслыханна в мире проприетарных телекоммуникаций, в то время как в мире с открытым исходным кодом — это обычное дело.

Ожидалось что самое первое мероприятие AstriCon привлечет сто участников. Пришли почти пятьсот человек (гораздо больше хотели, но не смогли). Такая поддержка сообщества практически гарантирует успех с открытым исходным кодом.

Некоторые вещи, которые уже возможны

Итак, какие вещи можно построить с помощью Asterisk? Давайте посмотрим на некоторые вещи, которые мы придумали.

Шлюз миграции традиционной УАТС

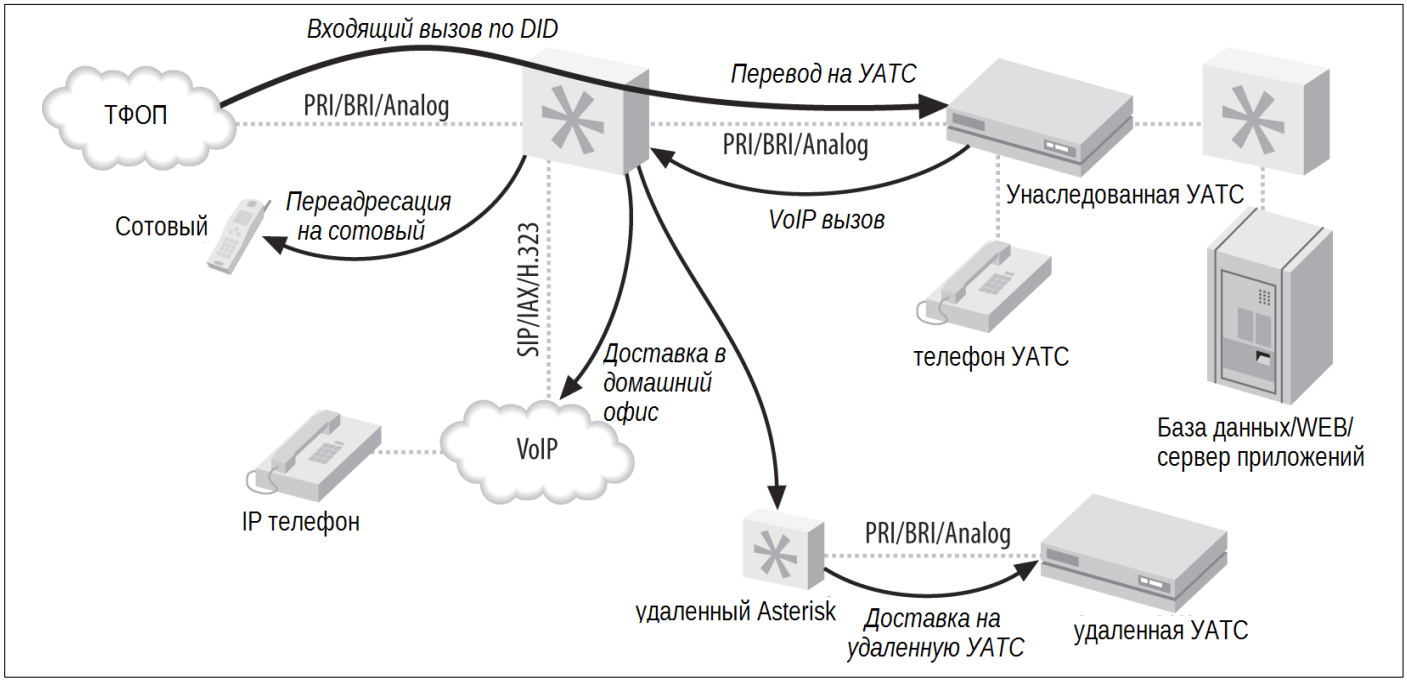


Рисунок 27-1. Asterisk как шлюз УАТС

Asterisk можно использовать как фантастический мост между старой АТС и будущим. Вы можете поместить его перед УАТС в качестве шлюза (и переносить пользователей с УАТС по мере необходимости) или вы можете поместить его за УАТС в качестве сервера периферийных приложений. Вы даже можете сделать и то и другое, как показано на Рисунке 27-1.

Вот некоторые из вариантов, которые можно реализовать:

Сохранить свою старую УАТС, но эволюционировать до IP

Компании, которые потратили огромные суммы денег за последние несколько лет на покупку собственного оборудования АТС, хотят выбраться из собственной тюрьмы, но они не могут переварить мысль о том, чтобы в этом случае выбросить все свое функционирующее оборудование. Нет проблем — Asterisk может решить все виды проблем, от замены системы голосовой почты до предоставления способа добавления пользователей на основе IP за пределы номинальной емкости системы.

Найди меня-следуй-сюда

Предоставьте АТС список номеров, по которым с вами можно связаться, и она будет звонить им всем при каждом звонке на ваш DID (прямой внутренний набор, aka телефон). Рисунок 27-2 иллюстрирует эту технологию.

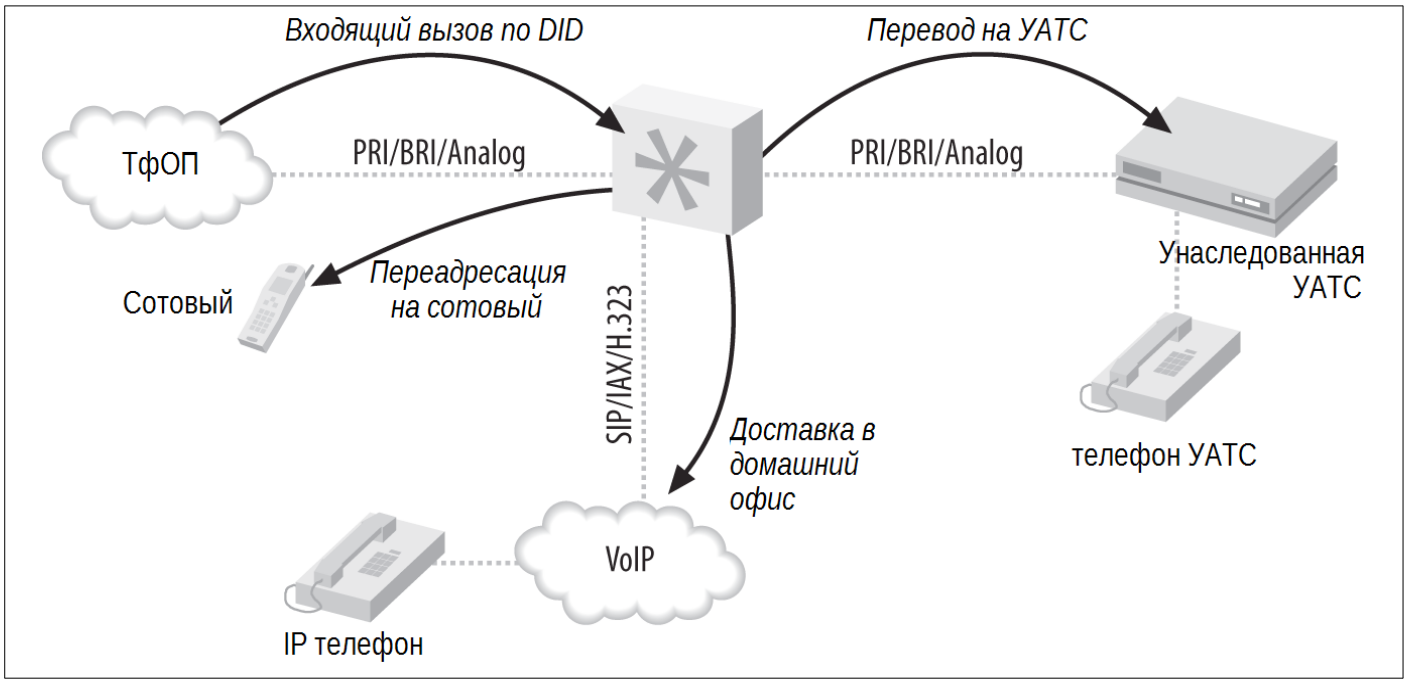


Рисунок 27-2. Найди-меня-следуйте-сюда

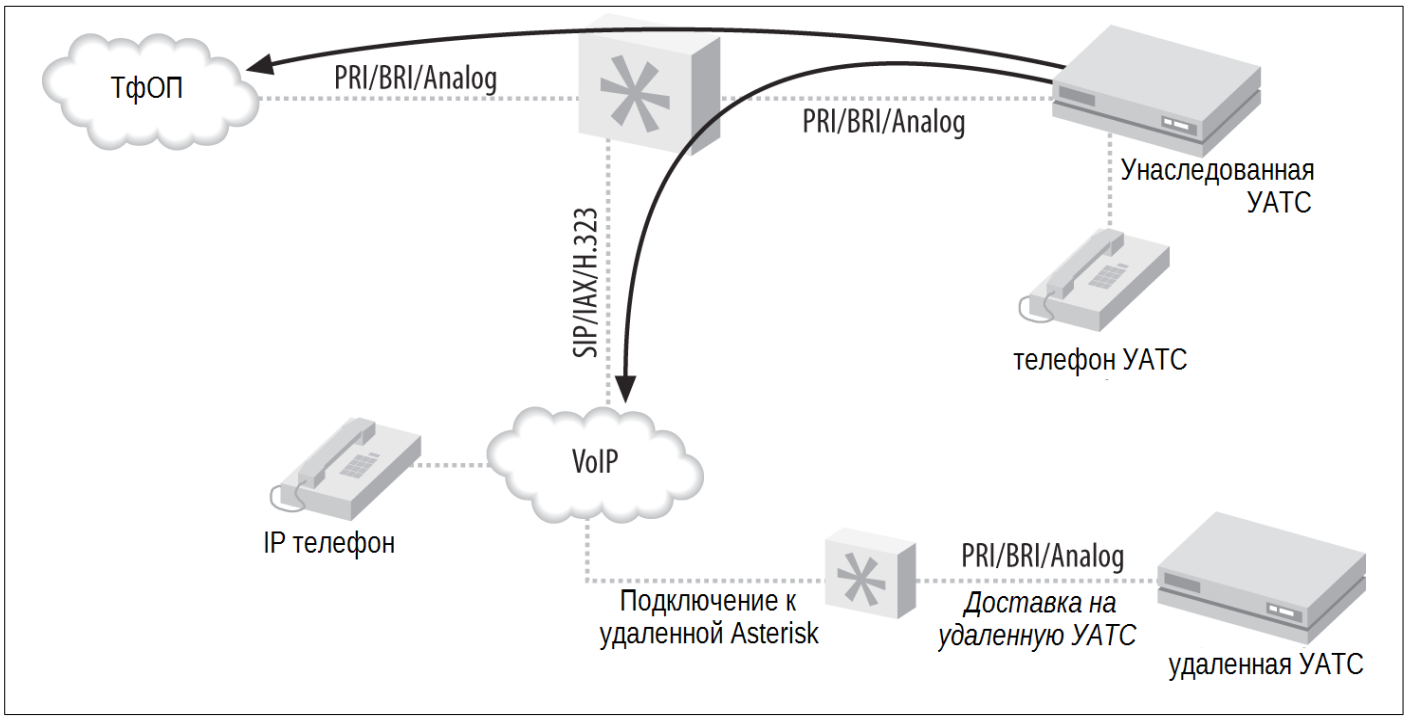


Рисунок 27-3. Поддержка VoIP устаревшей УАТС

Вызовы по VoIP

Если можно подключить устаревшее телефонное соединение от старой АТС к АТС Asterisk, Asterisk может предоставить доступ к услугам VoIP, в то время как старая АТС продолжает соединять с внешним миром как это было раньше. В качестве шлюза Asterisk просто нужно эмулировать функции PSTN и старая АТС не будет знать, что что-то изменилось. На Рисунке 27-3 показано, как использовать Asterisk для поддержки VoIP устаревшей УАТС.

Доступный IVR

Многие путают интерактивное голосовое меню (IVR) с автосекретарем (AA). Поскольку автоматизированный помощник (АвтоСекретарь) был первым, для чего использовалось IVR, это понятно. Тем не менее, для телекоммуникационной отрасли термин IVR представляет собой гораздо больше, чем АС. АС обычно делает больше, чем просто предоставляет абонентам способ быть соединенными с внутренними номерами, и он встроен в большинство проприетарных систем голосовой почты, но IVR — это намного больше.

Системы IVR, как правило, очень дороги — не только для покупки, но и для настройки. Пользовательская система IVR обычно требует подключения к внешней базе данных или приложению. Asterisk, возможно, является идеальным IVR, поскольку охватывает концепции подключения к базам данных и приложениям на самом глубоком уровне.

Вот несколько примеров относительно простых IVR-систем, где может быть использована Asterisk:

Отчет о погоде

Используя Интернет вы можете получить текстовые погодные отчеты со всего мира множеством способов. Захват этих отчетов и запуск их через специально построенный парсер (Perl, вероятно, съест это) позволит информации быть доступной для диалплана. Звуковая библиотека Asterisk уже содержит все необходимые подсказки, поэтому создание интерактивного меню для воспроизведения текущих прогнозов в любой точке мира не будет обременительной задачей.

Математические программы

Эд Гай (архитектор сети FWD Pulver) сделал презентацию для AstriCon 2004, в которой он рассказал о небольшой математической программе, которую он подготовил для своей дочери. Её написание заняло у него не больше часа. Он задал дочери несколько математических вопросов, ответы на которые она набрала в телефоне. Когда все вопросы были сведены в таблицу, система поставила ей оценку. Это чрезвычайно простое приложение Asterisk будет стоить десятки тысяч долларов для реализации на любой закрытой платформе АТС, если это вообще возможно. Как это часто бывает, вещи, которые просты для Asterisk, были бы невозможны или массово дороги с любой другой системой IVR.

Распределенный IVR

Стоимость собственной системы IVR такова, что когда компания со многими небольшими розничными точками хочет предоставить IVR, она вынуждена передавать абонентов на центральный сервер для обработки транзакций. С помощью Asterisk становится возможным распределять приложение на каждый узел и, таким образом, обрабатывать запросы локально. Буквально тысячи маленьких систем Asterisk, развернутых в торговых точках по всему миру, могли бы предоставлять функциональность IVR таким образом, что было бы невозможно добиться с любой другой системой. Нет больше междугородних переводов на центральный IVR-сервер, нет больше огромного транкинга, посвященного задаче — больше энергии с меньшими затратами.

Это три довольно простых примера потенциала Asterisk.

Конференц-связь

Этот маленький драгоценный камень в конечном итоге станет одной из убийственных функций Asterisk. В сообществе Asterisk люди все чаще используют конференц-залы для таких целей, как:

- Небольшие компании нуждаются в простом способе, чтобы собрать вместе всех партнеров для беседы.
- Команды продаж хотят иметь еженедельные встречи, где представители могут набирать номер, где бы они ни находились.
- Группы разработчиков должны определить общее место и время, чтобы информировать друг друга о достигнутом прогрессе.

С выпуском Asterisk 10 конференц-залы стали намного более захватывающими, добавив видеоконференции.[9]

9 Однако будьте бдительны, когда находитесь в конференц-зале. У нас есть друг, который участвовал в конференции. Кто-то еще работал по вызову из дома и был последним, кто заговорил перед тем, как подняться в туалет. К сожалению, дверь была видна камере, оставленной открытой, и в течение некоторого времени можно было видеть только пару колен со спущенными штанами.

Домашняя автоматизация

Asterisk по-прежнему слишком много инструмента über-geek, чтобы служить в рядовом доме, но с не более чем средними навыками Linux и Asterisk следующие вещи становятся правдоподобными:

Мониторинг детей

Родители, которые хотят проверить няню (или детей дома в одиночку) могут набрать контекст расширения, защищенный паролем. После аутентификации будет создана двусторонняя аудиосвязь со всеми IP-телефонами в доме, что позволит маме и папе всё прослушать. Жутко? Да. Но тем не менее интересная концепция.

Блокировка телефонов

Уходите на ночь? Не хотите чтобы няня пользовалась телефоном? Нет проблем! Простая настройка диалплана и единственные звонки, которые могут быть сделаны, это 911, ваш мобильный телефон и пиццерия. Любая другая попытка вызова включит запись "Мы платим вам, чтобы нянчить наших детей, а не делать личные звонки.”

Довольно зло, неправда ли?

Управление системой сигнализации

Вы получаете звонок во время отпуска от вашей мамы, которая хочет одолжить некоторые кухонные принадлежности. Она забыла ключ и стоит перед домом, дрожа от холода. Кусок пирога: вызов вашей системе Asterisk, быстрая строка цифр в контекст, который вы создали для этой цели, и ваша система сигнализации проинструктирована отключить сигнал тревоги в течение 15 минут. Маме лучше собрать свои вещи и поскорее убраться отсюда или появятся копы!

Управление звонками подростков

Как насчет ограничения по времени телефона для ваших подростков? Чтобы воспользоваться телефоном они должны ввести свои коды доступа. Они могут заработать дополнительные минуты, выполняя работу по дому, забивая все как есть, отгоняя этого раздражающего бомжа с плохой стрижкой — вы понимаете идею. Как только они израсходуют свои минуты ... нажмите ... вы получите свой телефон обратно.

Входящие вызовы также могут управляться через caller ID. "Донни, это отец Сьюзи. Она больше не заинтересована видеть вас, так как решила немного повысить свои стандарты. Кроме того, тебе стоит подумать о стрижке.”

Будущее Asterisk

Мы полюбили интернет потому что он богат контентом и дешев и, что возможно более важно, потому что он позволяет нам определять как мы общаемся. По мере того, как его способность нести более богатые формы медиа, мы будем все больше и больше использовать его. Как только интернет-голос обеспечит качество, которое будет конкурировать (или даже будет лучше) с ТфОП, телефонной компании лучше искать другую линию бизнеса. ТфОП перестанет существовать, вся её сложность будет поглощена интернетом, как еще одна технология. Как и большинство других интернет-технологий, технологии с открытым исходным кодом приведут к этой трансформации.

Обработка речи

Мечта о том, чтобы наши технические изобретения говорили с нами, старше самого телефона. Каждый технический прогресс порождает новую волну энергичных экспериментов. Как правило, результаты никогда не соответствуют ожиданиям, возможно потому, что как только машина говорит что-то, что *звучит* разумно, большинство людей считают ее разумной.

Люди, которые программируют и обслуживают компьютеры, осознают свои ограничения и, таким образом, склонны учитывать их недостатки. Все остальные ожидают, что их компьютеры и программное обеспечение просто будут работать. Количество мыслей, которые пользователь должен

приложить, чтобы взаимодействовать с компьютером, часто обратно пропорционально количеству мыслей, которое приложила команда разработчиков. Простые интерфейсы опровергают сложные проектные решения.

Поэтому задача состоит в том, чтобы разработать систему, которая предвосхитила бы самые распространенные желания своих пользователей, а также умело справлялась бы с неожиданными проблемами.

Festival и text-to-speech

Сервер text-to-speech Festival может преобразовывать текст в произносимые слова. Хотя этим очень весело играть, есть множество проблем, которые нужно преодолеть (подробнее об интеграции Festival с Asterisk см. в разделе “Утилиты Text-to-Speech” в Главе 18).

Для Asterisk очевидным преимуществом преобразования текста в речь может быть возможность чтения электронной почты с помощью телефонной системы. Если вы заметили несколько плохую грамматику, пунктуацию и орфографию, которые обычно встречаются в сообщениях электронной почты в наши дни, вы, возможно, оцените проблемы, которые они создают.

Нельзя не задаться вопросом — вдохновит ли появление text-to-speech новое поколение людей посвятить себя правильному письму. Видеть орфографические и пунктуационные ошибки на экране достаточно неприятно — слышать, как компьютер говорит такие вещи, потребует уровня дзадзэн, которым обладают немногие.

Распознавание речи

Если преобразование текста-в-речь — это ракетостроение, то распознавание речи — это научная фантастика.

Распознавание речи на самом деле может работать очень хорошо, но, к сожалению, это обычно верно только если вы предоставляете ему правильные условия, а правильные условия невозможны в телефонной сети. Даже идеальное соединение ТфОП считается самым низким допустимым пределом для точного распознавания речи. Добавьте сжатие и потери VoIP-соединения или сотовый телефон, и вы обнаружите гораздо больше ограничений, чем предполагаете.

Asterisk теперь имеет целый API речи, так что внешние компании (или даже проекты с открытым исходным кодом) могут связать свои механизмы распознавания речи в Asterisk. Одна компания, которая сделала — это LumenVox. Используя механизм распознавания речи LumenVox вместе с Asterisk вы можете создавать голосовые меню и IVR-системы в рекордные сроки! Дополнительные сведения см. на http://www.lumenvox.com.

В последнее время Apple и Google интегрируют распознавание речи в свои мобильные устройства. У Apple есть Siri, которая позволяет вам искать предметы рядом с вами, например: “Siri, где находится ближайшее кафе?"и Google, без особых фанфар, добавил голосовую функциональность в текстовый интерфейс для обмена сообщениями (для этого на вашей клавиатуре есть маленький микрофон). Кроме того, у Google есть небольшое приложение, которое даже позволит вам говорить на одном языке и воспроизводить то, что вы сказали на другом языке. Возможно, когда-нибудь у нас будут интерфейсы API для этих приложений, чтобы Asterisk мог воспользоваться ими.

Высококачественный голос

По мере того как мы получаем доступ ко все большей и большей пропускной способности, становится все труднее понять, почему мы все еще используем низкокачественные кодеки. Многие люди не понимают что Skype обеспечивает более высокое качество чем телефон; это весомая причина почему Skype имеет репутацию такого хорошего звучания.

Если бы вы когда-нибудь позвонили в CNN, разве вы не хотели бы услышать медлительный голос Джеймса Эрла Джонса, говорящий “This is CNN”, а не какую-то жестяную электронную запись? И если вы думаете, что Эллисон Смит[10] звучит хорошо по телефону, вы должны услышать ее лично!

В 2005 году мы писали "В будущем мы ожидаем и получим высококачественный голос через наше коммуникационное оборудование.

"По мере того, как все больше и больше поставщиков оборудования начинают реализовывать поддержку высококачественного голоса в своем VoIP-оборудовании, вы увидите больше поддержки в Asterisk для совершения звонков качественнее, чем ТфОП.”

В 2013 году любое новое развертывание системы Asterisk с новыми телефонными аппаратами (софтфонами или хардфонами) обеспечивает возможность передачи голоса HD через кодеки, такие как Speex и G.722. Почти все производители SIP-телефонов включают G.722 и даже некоторые ITSPs начинают предоставлять возможности G.722. К сожалению, качество теряется если вы разговариваете не локально в офисе или с кем-то другим на том же ITSP, у которого есть такие же голосовые возможности HD.

Компании сотовой связи тоже начали проникать в широкополосную игру кодеков. IPhone 5 имеет голосовые возможности HD, а сотовые компании в Канаде (такие как Bell и Telus) развернули его вместе со своими сетями передачи данных LTE. Возможно, дни колючего GSM остались позади?

Видео

Хотя большая часть этой книги посвящена аудио, видео также поддерживается многими способами в Asterisk. У нас была видео-голосовая почта в течение многих лет, и с выпуском Asterisk 10 видеоконференции были добавлены в приложение `ConfBridge()`.

Это не просто Asterisk, который узрел проникновение видео на рынок. Google создал решение для видеоконференций под названием Google Hangouts, которое позволяет нескольким участникам (до 10 человек) участвовать в видеоконференции. Горизонтально в окне выложен набор панелей, похожих на пленку, который содержит видео всех участников конференции, и, как и в приложении Asterisk `ConfBridge()`, детектирование говорящего используется для перевода основной видеопанели на говорящего человека.

Google Hangouts стал невероятно популярными в opensource-сообществе. Поскольку большинство opensource-проектов имеют разработчиков по всей стране и по всему миру, использование решения для видеоконференций (которое бесплатно для использования), в котором несколько участников могут обмениваться экранами и видео, позволяет им выполнять живые обзоры кода. Разработчики в Opscode выполняют еженедельные обзоры кода вокруг программного обеспечения автоматизации облачной инфраструктуры под названием Chef.

Кроме того, другие подкасты и еженедельные шоу конференц-связи начали интегрировать видео в свои прежние встречи без видео. Как Voice Users Conference (VUC), так и FoodFightShow начали использовать видео, YouTube и всевозможные другие социальные сети для создания всестороннего погружения социальной активности в своих программах.

Задача видеоконференций

Производители смартфонов также начали добавлять не только высококачественные камеры для съемки, но и фронтальные камеры, предназначенные для видеоконференций. Любой современный смартфон на базе Android или Apple включает в себя фронтальную камеру, также как и современные планшеты. Эти камеры предлагают бесконечные возможности для видео-приложений. Теперь все зависит от потребителя и от того, будут ли когда-либо реализованы обещания видео с начала 1960-х годов.

10 Эллисон Смит-это голос Asterisk—это ее голос во всех системных подсказках. Чтобы Эллисон создала ваше собственное приглашение, просто посетите http://www.theivrvoice.com.

Концепция видеоконференцсвязи существует с момента изобретения электронно-лучевой трубки. Телеком-индустрия десятилетиями обещает видеоконференционное устройство в каждом доме.

Как и многие другие коммуникационные технологии, если у вас дома есть видеоконференции, вы, вероятно, используете их через Интернет с помощью простой и недорогой веб-камеры. Тем не менее, кажется, что люди видят видеоконференции как хитрый трюк. Да, вы видите человека, с которым говорите, но чего-то не хватает.

Почему мы любим видеоконференции

Видеоконференции обещают более богатый опыт общения, чем телефония. Вместо того чтобы просто слышать бестелесный голос, Вы получаете доступ ко всем нюансам речи, которые возникают при общении лицом к лицу.

Почему видеоконференции никогда не смогут полностью заменить голос

Есть некоторые проблемы для преодоления и не все из них являются техническими.

Подумайте вот о чем: используя обычный телефон, люди, работающие в своих домашних офисах, могут вести деловые разговоры не надевая нижнего белья, положив ноги на стол, с кофе в руке — если они используют телефон. Подобный видео-разговор потребовал бы полчаса груминга, чтобы подготовиться, и не мог произойти на кухне, во внутреннем дворике или...ну, вы поняли идею.

Кроме того, обещание общения "глаза в глаза" по видео никогда не произойдет, пока фокусы участников не будут соответствовать камерам. Если вы посмотрите в камеру, ваша аудитория увидит, что вы смотрите на них, но вы их не увидите. Если вы посмотрите на экран, чтобы увидеть, с кем говорите, камера покажет вам, что вы смотрите на что-то, а не на аудиторию. Это выглядит безлично. Возможно, если бы видеофон был сконструирован как Tele-Prompt-R, где камера находится за экраном, это не было бы так неестественно.

WebRTC

С веб-сайта WebRTC "*WebRTC — это бесплатный открытый проект, который предоставляет браузерам и мобильным приложениям возможности связи в реальном времени (RTC) через простые API.*" Многие люди начали смотреть на WebRTC как на способ, которым они смогут начать размещать софтфон на веб-сайте; но на самом деле он представляет намного больше. Проект WebRTC позволит создать захватывающий Интернет, который нам еще предстоит испытать. Разработка все еще находится на ранних стадиях, но при поддержке нескольких браузеров (Google Chrome, Firefox, Opera и Internet Explorer) API WebRTC обязательно станет вездесущими в не слишком отдаленном будущем.

Asterisk 11 имеет некоторую раннюю поддержку WebRTC, но API в настоящее время разрабатывается и находится под постоянными изменениями. Однако вещи начнут материализовываться и API станет более стабильным. Когда это начнет происходить, вы сможете видеть большое развитие, происходящее по всему интернету. Во-первых, это, вероятно, обычные вещи, которые уже были сделаны, например, софтфон в вашем браузере. Эти разработки позволят разработчикам учиться, применяя то, что они уже умеют, к новой технологии. Но как только они будут выдержаны, мы подозреваем, что вы начнете видеть гораздо более захватывающие технологии, распространенные по всему интернету.

Джошуа Кольп и Тим Пантон говорили на AstriCon 2012 в Атланте о WebRTC и интеграции, которую включает Asterisk 11. Вы можете найти другие презентации AstriCon в видеоархиве.

Беспроводность

Поскольку Asterisk полностью поддерживает VoIP, беспроводная связь является частью пакета.

Wi-Fi

WiFi будет решением офисной мобильности для VoIP-телефонов. Эта технология уже достаточно зрелая. Самым большим препятствием является стоимость телефонов, которая, как можно ожидать, уменьшится, поскольку конкурентное давление со всего мира ведет к снижению цен.

WiMAX

Когда мы впервые написали эту главу, мы предсказывали:

"Поскольку мы так смело предсказываем так много вещей — нетрудно предсказать, что WiMAX означает начало конца для традиционных сотовых телефонных сетей.

"С беспроводным доступом в интернет в пределах досягаемости большинства сообществ, какая ценность будет недорогой сотовой связи?”

Можно с уверенностью сказать, что мы были далеки от истины. С тех пор поставщики услуг потратили значительные суммы на модернизацию инфраструктуры и начали развертывать сети следующего поколения, такие как LTE, которые предлагают значительные улучшения производительности сети по сравнению с тем, что мы имели в прошлом. Похоже, что WiMAX никогда не цеплялся за потребительские продукты — главный двигатель требований к беспроводной сети.

Мы можем представить альтернативную вселенную, где первый iPhone выкатился с поддержкой WiMAX и целый ряд новых сетевых носителей вышел в интернет чтобы обеспечить доступ к данным этим устройствам. Этого, конечно, никогда не происходило, и, похоже, WiMAX была отличной идеей без будущего.

Универсальная система обмена сообщениями

Это термин который был раздут телекоммуникационной отраслью в течение многих лет, но принятие было гораздо медленнее, чем прогнозировалось.

Единая система обмена сообщениями относится к концепции связывания голосовых и текстовых систем обмена сообщениями в одну. С Asterisk их не нужно искусственно комбинировать, так как Asterisk уже относится к ним одинаково.

Просто изучив термины, *универсальная* и *система обмена сообщениями*, мы можем увидеть, что интеграция электронной и голосовой почты должна быть только началом — единая система обмена сообщениями должна сделать намного больше, чем лишь это, если она заслуживает своего имени.

Возможно, нам нужно определить “систему обмена сообщениями" как общение, которое не происходит в реальном времени. Другими словами, когда вы отправляете сообщение, то ожидаете, что ответ может занять несколько секунд, минут, часов или даже дней. Вы придумываете то, что хотите сказать, и ваша аудитория должна написать ответ.

Сравните это с беседой, которая происходит в реальном времени. Когда вы говорите с кем-то по телефону, вы ожидаете не более чем несколько секунд задержки до получения ответа.

Несколько лет назад Тим О'Рейли выступил с речью под названием “Наблюдение за Alpha гиками: OS X и следующая большая вещь”, в которой он говорил о том, что кто-то пропускает IRC через механизм преобразования текста в речь. Можно было бы представить, как сделать обратное, позволяя нам присоединиться к IRC или чату мгновенных сообщений по WiFi-телефону, с нашей АТС Asterisk, предоставляющей перевод текст-в-речь-в-текст.

Пиринг

По мере того, как монопольные сети, такие как ТфОП, уступают место сетям на базе сообществ, таким как Интернет, придет время когда необходимо будет соединить их. Хотя традиционные

провайдеры предпочли бы, чтобы существующая модель была перенесена в новую парадигму, становится все более вероятным, что телефонные звонки станут не более чем еще одним приложением, которое интернет счастливо несет.

Но остается задача: как управлять планом нумерации телефонов, с которым мы все знакомы и довольны?

E.164

МСЭ определил план нумерации в своей спецификации E.164. Если вы звонили по телефону через ТфОП, то можете с уверенностью заявить, что знакомы с концепцией нумерации E.164. До появления общедоступного VoIP никто не заботился о E.164, кроме телефонных компаний — никому это не было нужно.

Теперь, когда звонки переходят из ТфОП в Интернет и черт знает куда ещё, некоторое внимание должно быть уделено E.164.

ENUM

В ответ на эту проблему IETF выступила спонсором рабочей группы по электронному сопоставлению номеров (Electronic NUmber Mapping - ENUM), цель которой заключается в сопоставлении номеров E.164 с системой доменных имен (DNS).

Хотя концепция ENUM является обоснованной, для достижения успеха требуется сотрудничество со стороны телекоммуникационной отрасли. Однако сотрудничество — это не то, чем славится телекоммуникационная индустрия, и до сих пор ENUM терпит крах.

e164.org

Народ в http://e164.org старается внести свой вклад в успех ENUM. Вы можете войти на этот сайт, зарегистрировать свой номер телефона и сообщить системе альтернативные способы общения с вами. Это означает, что кто-то, кто знает ваш номер телефона, может совершить вам VoIP-вызов, как http://e164.org зона DNS предоставит IP-адрес и информацию о протоколе, необходимом для подключения к вашей локации.

Поскольку все больше и больше людей публикуют информацию о подключении VoIP, все меньше и меньше вызовов будет подключаться через ТфОП.

freenum.org

Доморощенные решения, такие как freenum.org пытаются предложить решение stop-gap между набором URI SIP и традиционными ограничениями цифрового набора телефонии (а именно, большинство SIP-телефонов не поставляются с клавиатурой). Это может начать меняться, поскольку сенсорные экраны становятся повсеместными на всех устройствах. Но на данный момент, пиринг по-прежнему является проблемой. Маловероятно, что мы когда-нибудь увидим, как крупные телекоммуникационные компании используют пиринговую структуру, позволяющую людям обходить свои сети и прибыль — зачем им это?

Задачи

Как и в случае с любой стоящей вещью, Asterisk столкнется с проблемами. Давайте взглянем на некоторые из них.

Слишком много изменений, слишком мало стандартов

В наши дни интернет меняется так быстро и предлагает так много разнообразного контента, что даже самый внимательный гик не может быть в курсе всего этого. Хотя это так, как и должно быть, это также означает, что огромное количество технологий является неизбежной частью поддержания любой системы связи.

Телефонное мошенничество

Пока междугородние звонки стоят денег, будут преступники, которые хотят их украсть. Телефонное мошенничество не является чем-то новым, но со многими незащищенными системами Asterisk в интернете в настоящее время, популярность скриптов для поиска этих систем и их компрометации взорвалась. Администраторам телефонных систем, подключенных к Интернету, необходимо будет тщательно продумать свою систему безопасности, с тем чтобы все звонки, поступающие из их систем, совершались только авторизованными пользователями.

VoIP спам

Да, он приближается. Всегда найдутся люди, которые считают, что имеют право причинять неудобства и беспокоить других в погоне за деньгами. Предпринимаются усилия по решению этой проблемы, но только время покажет, насколько эффективными они будут.

Страх, неуверенность и сомнение

Индустрия совершает переход от невежества к устрашению. Если Ганди прав — мы можем ожидать что борьба начнется в ближайшее время.

По мере того, как их доходные потоки становятся все более угрожаемыми телефонией с открытым исходным кодом, традиционные игроки отрасли, несомненно, начнут кампанию устрашения в надежде подорвать революцию.

Проектирование узких мест

Ходят слухи, что крупные сетевые провайдеры искусственно калечат VoIP-трафик, помечая и приоритизируя трафик своих премиум-услуг перед VoIP и, что еще хуже, обнаруживают и сталкивают любой VoIP-трафик, генерируемый услугами, не одобренными ими.

Некоторые из них уже имеют место, когда поставщики услуг блокируют трафик определенных типов через свои сети, якобы как государственную услугу (например, блокирование популярных сервисов обмена файлами для защиты нас от пиратства). В Соединенных Штатах FCC заняла четкую позицию по этому вопросу и оштрафовала компании, которые занимаются такой практикой. В остальном мире регулирующие органы не всегда принимают VoIP.

Что кажется ясным, так это то, что сообщество и сеть найдут способы обойти блокировки, как и всегда.

Регламентные войны

Бывший председатель Федеральной комиссии США по связи Майкл Пауэлл сделал подарок, который вполне мог изменить путь революции VoIP. Вместо того, чтобы пытаться регулировать VoIP как телекоммуникационную услугу, он отстаивал концепцию, что VoIP представляет собой совершенно новый способ общения и требует своего собственного регулирующего пространства, в котором будет развиваться.

VoIP будет регулироваться, но не везде как услуга телефонии. Некоторые из правил, которые могут быть созданы включают:

Информация о присутствии для аварийно-спасательных служб

Одна из характеристик традиционной сети ТфОП заключается в том, что она всегда находится в одном и том же месте. Это очень полезно для аварийных служб, поскольку они могут точно определить местоположение вызывающего абонента, определив адрес сети, из которой был сделан вызов. Распространение сотовых телефонов сделало это гораздо труднее, так как мобильный телефон не имеет известного адреса. Мобильный телефон можно подключить к любой сети и зарегистрировать на любом сервере. Если телефон не идентифицирует свое физическое местоположение, экстренный вызов с него не даст никакой подсказки о том, где находится вызывающий. VoIP создает аналогичные проблемы.

Мониторинг вызовов для правоохранительных органов

Правоохранительным органам всегда удавалось получить прослушки на традиционных телефонных линиях с коммутацией каналов. Хотя в настоящее время принимаются нормативные акты, направленные на достижение той же цели в сети, технические проблемы, связанные с обеспечением этой функциональности, вероятно, никогда не будут полностью решены. Люди ценят свою личную жизнь, и чем больше правительства хотят ее задушить, тем больше усилий будет приложено для ее сохранения.

Антимонопольные практики

Такая практика уже наблюдается в США, где штрафы взимаются с сетевых провайдеров, которые пытаются фильтровать трафик на основе контента.

Когда дело доходит до регулирования, Asterisk является одновременно святым и дьяволом: святым, потому что он кормит бедных, и дьяволом, потому что он дает взломщиков и спамеров как ничто другое. Регулирование телефонии с открытым исходным кодом может частично определяться тем, насколько хорошо сообщество регулирует себя. Такие концепции как DUNDi, которые включают в себя процессы защиты от спама, являются отличным началом. С другой стороны, такие концепции, как подмена CallerID, изобилуют возможностями для злоупотреблений.

Качество обслуживания

Из-за лучших усилий реальности TCP/IP на основе Интернета еще не известно, как увеличение трафика VoIP в реальном времени повлияет на общую производительность сети. В настоящее время в магистрали так много избыточной полосы пропускания, что доставка максимальными усилиями, как правило, действительно хороша. Тем не менее, было доказано снова и снова, что всякий раз, когда мы получаем больше пропускной способности, мы находим способ использовать ее.

Возможно, следствие Закона Мура[11] будет применяться к пропускной способности сети. QoS может стать спорным из-за способности сети обеспечить адекватную производительность без какой-либо специальной обработки. Организации, которые требуют более высокого уровня надежности, могут выдавать премию за более высокий уровень обслуживания. Возможно, эпоха оплаты по минутам за междугородние соединения уступит место миллисекундой оплате за гарантированную низкую задержку или процентным пунктом за снижение потери пакетов. Премиум услуги будут предлагать надежность пять-девяток[12], которую традиционные телекоммуникационные компании всегда рекламировали как свое преимущество перед VoIP.

Трудоемкость

Открытые системы требуют новых подходов к проектированию решений. Только потому, что аппаратное и программное обеспечение дешево, не означает, что решение будет. Asterisk не выходит из коробки готовым к запуску; система Asterisk должна быть спроектирована и построена, а затем поддержана. Хотя базовое программное обеспечение является бесплатным, а затраты на аппаратное обеспечение будут основываться на ценах на товары, справедливо сказать, что затраты на конфигурацию для хорошо настроенной системы будут значительной частью общей стоимости решения. Фактически, во многих случаях, из-за высокой степени сложности и конфигурируемости Asterisk, стоимость будет больше, чем можно было бы ожидать при использовании традиционной АТС.

Эмпирическое правило обычно считается примерно так: если это можно сделать в диалплане, ресурсы, затраченные на проектирование системы, будут примерно такими же, как и для любой

11 Гордон Мур написал статью в 1965 году, в которой предсказал удвоение транзисторов на процессоре каждые несколько лет.

12 Этот термин относится к 99,999%, которая рекламируется как надежность традиционных телекоммуникационных сетей. Достижение пяти девяток требует, чтобы перерывы в обслуживании в течение всего года составляли не более 5 минут и 15 секунд. Многие люди считают, что VoIP необходимо будет достичь такого уровня надежности, прежде чем он сможет полностью заменить ТфОП. Многие другие люди считают, что ТфОП даже не приближается к надежности пяти девяток. Это мог бы быть отличный термин для описания высокой надежности, но отделы маркетинга злоупотребляют ею слишком часто.

аналогичной традиционной АТС. Кроме того, только опыт позволит точно оценить время, необходимое для создания системы.

Возможности

Телефония с открытым исходным кодом создает безграничные возможности. Вот некоторые из наиболее убедительных.

Индивидуальные частные телекоммуникационные сети

Некоторые люди скажут вам, что цена — это ключ, но мы считаем, что реальная причина успеха Asterisk заключается в том, что теперь можно построить телефонную систему как веб-сайт: с комплексной, полной настройкой каждого аспекта системы. Клиенты хотели этого годами. Только Asterisk может предоставить это.

Низкий барьер для входа

Каждый может внести свой вклад в будущее общения. Теперь кто-то со старым ПК за 200 долларов может разработать систему связи, которая имеет интеллект чтобы конкурировать с самыми дорогими проприетарными системами. Конечно, оборудование не было бы готово к продакшену, но нет причин, по которым программное обеспечение не могло бы быть. Это одна из причин, по которой закрытым системам будет трудно конкурировать. Количество людей, которые имеют доступ к необходимому оборудованию, возможно эквивалентны закрытому магазину.

Творческие возможности

Проектирование АТС всегда было своего рода формой искусства, но до Asterisk искусство заключалось в поиске творческих путей преодоления ограничений технологии. С безграничной технологией те же самые творческие навыки теперь могут быть правильно применены к задаче полного удовлетворения потребностей клиента. С открытым исходным кодом телефонии, двигатели, такие как Asterisk, позволили это. Телекоммуникационные проектировщики будут танцевать от радости, поскольку их значительные творческие способности теперь будут фактически служить потребностям их клиентов, а не быть сосредоточенными на управлении металлоломом.

Надлежащая интеграция коммуникационных технологий

В конечном счете, обещание открытого исходного кода ни к чему не приводит, если оно не может удовлетворить потребности людей в решении проблем. Закрытые отрасли потеряли клиента из виду и попытались приспособить его к продукту.

Телефония с открытым исходным кодом приводит голосовую связь в соответствие с другими информационными технологиями. Наконец, можно правильно начать задачу интеграции электронной почты, голоса, видео и всего остального, что мы могли бы представить себе через гибкие транспортные сети (будь то проводные или беспроводные) в ответ на потребности пользователя, а не прихоти монополий.

Добро пожаловать в будущее телекома!

Понимание телефонии

Полезность — это когда у вас один телефон, роскошь — когда у вас два, богатство — когда у вас три, а рай — когда нет ни одного.
- Даг Ларсон

В этом приложении мы поговорим о некоторых технологиях традиционной телефонной сети— особенно тех, которые люди чаще всего хотят подключить к Asterisk. (Мы обсудим Voice over IP в приложении B.)

В то время как тома можно было бы написать о технологиях, используемых в телекоммуникационных сетях, материал, изложенный здесь, был выбран на основе нашего опыта в сообществе, который помог нам определить конкретные элементы, которые могут быть наиболее полезными. Хотя эти знания могут не требоваться для настройки вашей системы Asterisk, они принесут большую пользу при подключении к системам (и разговоре с людьми) из мира традиционных телекоммуникаций.

Аналоговая телефония

Целью телефонной сети общего пользования (ТфОП) является установление и поддержание аудиосвязи между двумя конечными точками для передачи речи.

Хотя люди могут воспринимать звуковые колебания в диапазоне 20-20 000 Гц,[1] большинство звуков, которые мы издаем при разговоре, обычно находятся в диапазоне 250-3000 Гц. Поскольку целью телефонной сети является передача звуков говорящих людей,она была спроектирована с полосой пропускания где-то в диапазоне 300-3 500 Гц. Эта ограниченная пропускная способность означает, что некоторое качество звука будет потеряно (как может подтвердить любой, кто вынужден был слушать музыку на удержании), особенно на более высоких частотах.

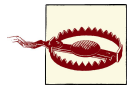
Части аналогового телефона

Аналоговый телефон состоит из пяти частей: звонка, номеронабирателя, гибридного (или сетевого) трансформатора, рычажного переключателя и телефонной трубки (оба подключаются к гибридному трансформатору). Звонок, номеронабиратель и гибридный трансформатор могут работать совершенно независимо друг от друга.

1 Если вы хотите поиграть с тем, как выглядят разные частоты на осциллографе, возьмите копию анализатора звуковой частоты из надежного программного обеспечения. Это очень простой и интересный способ визуализировать, как звук "выглядит". Спектрограф дает хорошую картину сложных гармоник, которые могут генерировать наши голоса, а также оценку фоновых звуков, которые всегда окружают нас. Вы также должны попробовать восхитительно раздражающий генератор тона NCH, от NCH Swift Sound.

Звонок

Когда центральный офис (CO) хочет сигнализировать о входящем вызове, он подает сигнал переменного тока (AC) напряжением примерно 90 Вольт к вашей цепи. Это вызовет звонок в вашем телефоне, чтобы произвести звенящий звук. (В электронных телефонах этот звонок может быть не колокольчиком, а маленькой электронной певуньей. В конечном счете, звонарем может быть все, что способно реагировать на напряжение звонка; в шумных средах, таких как заводы к примеру, часто используются стробоскопы .)



Напряжение звонка может быть опасным. Будьте очень осторожны при работе с внутренней телефонной линией.

Многие люди путают напряжение переменного тока (AC), которое вызывает звонок, с напряжением постоянного тока (DC), которое питает телефон. Помните, что звонок нуждается в переменном токе, чтобы колебаться (так же, как церковный колокол не будет звонить, если вы не обеспечиваете движение), и у вас это есть.

В Северной Америке количество звонков, которые вы можете подключить к своей линии, зависит от Ringer Equivalence Number (REN) ваших различных устройств. (REN должен быть указан на каждом устройстве.) Общее REN для всех устройств, подключенных к вашей линии не может превышать 5,0. REN 1.0 эквивалентен старомодному аналоговому аппарату с электромеханическим звонарем. Некоторые электронные телефоны имеют значение REN 0.3 или даже меньше. Если вы подключите слишком много устройств, требующих слишком много тока, вы обнаружите, что ни одно из них не сможет звонить.

Номеронабиратель

Когда вы звоните по телефону, вам нужно каким-то образом сообщить сети адрес стороны, с которой вы хотите связаться. Номеронабиратель - это часть телефона, которая обеспечивает эту функциональность. В первые дни ТфОП циферблаты были фактически поворотными устройствами, которые использовали импульсы для обозначения цифр. Это был довольно медленный процесс, поэтому телефонные компании в конечном итоге ввели тональный набор. С тональным набором-также известным как двухтональный мультичастотный набор (Dual-Tone Multi Frequency - DTMF) - наборная панель состоит из 12 кнопок. Каждой кнопке назначены две частоты (см. Таблицу A-1).

Таблица A-1. Цифры DTMF

	1209 Hz	1336 Hz	1477 Hz	1633 Hz[a]
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

[a] Обратите внимание, что этот столбец содержит буквы, которые обычно не присутствуют в качестве клавиш на телефонной панели. Тем не менее, они являются частью стандарта DTMF, и любой правильный телефон содержит электронику, необходимую для их создания, даже если у него нет самих кнопок. (Эти кнопки действительно существуют на некоторых телефонах, которые в основном используются в военных и правительственных учреждениях.)

Когда вы нажимаете кнопку на своей клавиатуре, две соответствующие частоты передаются по линии. Дальний конец может интерпретировать эти частоты и отметить, какая цифра была нажата.

Гибридный (или сетевой) трансформатор

Гибридный трансформатор, который регулирует потребность совместимости сигналов переданных и полученных, через одиночную пару проводов от ТфОП и 2 пары проводов в телефонной трубке. Одна из функций, которую выполняет гибридный трансформатор - регулирование местного эффекта (sidetone), который представляет собой количество передаваемого сигнала, возвращаемого в наушник; его цель-обеспечить более естественное звучание разговора. Если местного эффекта слишком много,

то ваш голос будет звучать слишком громко; если слишком мало вам будет казаться что линия отключена.

Рычажный переключатель (рычаг переключателя). Этот прибор сигнализирует положение телефонной цепи к CO. Когда вы поднимаете трубку телефона, рычажный переключатель замыкает петлю между вами и CO, что рассматривается как запрос на получение тонального сигнала готовности линии. Когда вы вешаете трубку, переключатель разрывает цепь, что указывает о завершении вызова.[2]

Рычажный переключатель можно также использовать для целей сигнализации. Некоторые электронные аналоговые телефоны имеют кнопку с надписью *Link* (Связь), которая вызывает событие, называемое *flash* (мгновенное события сброса). Flash можно выполнить вручную, нажав на рычажный переключатель в течение 200-1200 миллисекунд (номинально около 600 мс). Если вы удержите его дольше, коммутатор может предположить, что вы повесили трубку. Цель кнопки *Link* - обработать это время за вас. Если вы когда-либо использовали ожидание вызова или трехсторонний вызов по аналоговой линии, вы использовали flash переключатель для сигнализации сети.

Телефонная трубка. Телефонная трубка состоит из передатчика и приемника. Она выполняет преобразование между звуковой энергией, используемой людьми, и электрической энергией, используемой телефонной сетью. Все телефонные трубки (в том числе в цифровых и VoIP телефонах) являются аналоговыми.

Tip and Ring

В аналоговой телефонной сети есть два провода. В Северной Америке эти провода называются Tip (наконечником) и Ring (кольцом).[3] Эта терминология происходит от дней, когда телефонные звонки были соединены живыми операторами, сидящими за пультами. В вилках использовали два контакта —один на кончик вилки, а другой подключен к кольцу посередине (Рисунок A-1).

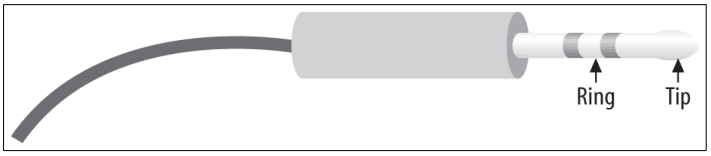


Рисунок A-1. Tip and Ring

Tip - это провод положительной полярности. В Северной Америке, этот провод обычно зеленый и обеспечивает обратный путь. Провод Ring - провод отрицательной полярности. В Северной Америке, этот провод обычно красный. Для современных кабелей Cat 5 и 6 Tip обычно представляет собой белый провод, а Ring - цветной. Когда ваш телефон находится на рычаге, провод Ring будет иметь потенциал -48V DC по отношению к Tip. При снятии трубки это напряжение падает примерно до -7V DC.

Цифровая телефония

Аналоговая Телефония почти мертва.

В ТфОП знаменитая последняя миля-это последний оставшийся кусок телефонной сети, все еще использующий технологию, впервые появившуюся более ста лет назад.[4]

---

2 Когда речь идет о состоянии аналоговой линии, люди часто говорят в терминах “занято” (off-hook) и “свободно" (on-hook). Когда ваша линия "занята" (off-hook), ваш телефон занят разговором - "включен" (on). Если ваша линия "свободна"(on-hook), телефон по существу ”выключен" (off) или свободен. On меняется на off и off меняется на on.

3 Они могут иметь другие названия в других странах (например, A и B).

4 ”Последняя миля" - это термин, который первоначально использовался для описания единственной части ТфОП, которая не была преобразована в волоконную оптику: связь между центральным офисом и клиентом. Последняя миля-это более широкое понятие, поскольку она также имеет значение как ценный актив традиционных телефонных компаний; они владеют подключением к вашему дому. Последняя миля становится все более и более трудной для описания в технических терминах, так как в настоящее время существует так много способов подключения сети к клиенту. Как вещь стратегической ценности для телекоммуникаций, кабелей и других коммунальных услуг, ее важность очевидна.

Одна из основных проблем при передаче аналоговых сигналов заключается в том, что всевозможные вещи могут мешать этим сигналам, приводя к низкой громкости, статике и всевозможным другим нежелательным эффектам. Вместо того, чтобы пытаться сохранить аналоговую форму волны на расстояниях, которые могут охватывать тысячи километров, почему бы просто не измерить характеристики исходного звука и отправить эту информацию в другой конец? Исходная форма волны туда не попадет, но вся информация, необходимая для ее восстановления передастся.

Это принцип всего цифрового аудио (включая телефонию): оцените характеристики формы волны источника, сохраните измеренную информацию численно, и отправьте эти данные в другой конец. Затем, на том конце, используйте переданную информацию для генерации совершенно нового аудиосигнала, который имеет те же характеристики, что и оригинал. Воспроизведение настолько хорошо,что человеческое ухо не сможет отличить.

Основное преимущество цифрового аудио заключается в том, что выборочные данные могут быть математически проверены на наличие ошибок по всему маршруту до места назначения, гарантируя, что идеальный дубликат оригинального сигнала прибывает в другой конец. Расстояние больше не влияет на качество, и взаимодействие можно обнаружить и исключить.

Имульсно-кодовая модуляция

Существует несколько способов цифрового кодирования звука, но наиболее распространенный метод (и тот, который используется в телефонии) известен как импульсно-кодовая модуляция (Pulse-Code Modulation, PCM). Чтобы проиллюстрировать, как это работает, давайте рассмотрим несколько примеров.

Цифровое кодирование аналогового сигнала

Принцип ИКМ заключается в том, что амплитуда[5] аналоговой формы волны отбирается через определенные интервалы, чтобы впоследствии ее можно было воссоздать. Объем детализации, который захватывается, зависит как от битового разрешения каждого образца, так и от того, как часто берутся образцы. Более высокое битовое разрешение и более высокая частота дискретизации обеспечат большую точность, но для передачи этой более подробной информации потребуется большая пропускная способность.

Чтобы лучше понять как работает ИКМ, рассмотрим форму волны, показанную на Рисунке A-2.

Чтобы кодировать волну в цифровом виде, ее необходимо отбирать на регулярной основе и измерять амплитуду волны в каждый момент времени. Процесс разрезания формы волны на моменты времени и измерения мощности в каждый момент называется *квантованием* или *выборкой*.

Образцы необходимо брать достаточно часто и необходимо будет захватить достаточно информации, чтобы гарантировать, что дальний конец может воссоздать достаточно похожую форму волны. Для достижения более точной выборки потребуется больше битов. Чтобы объяснить эту концепцию, мы начнем с очень низкого разрешения, используя 4 бита для представления нашей амплитуды. Это облегчит визуализацию как самого процесса квантования, так и влияния разрешения на качество.

5 Амплитуда-это, по существу, мощность или сила сигнала. Если вы когда-либо держали скакалку или садовый шланг и давали ему хлыст, вы видели результирующую волну. Чем выше волна, тем больше амплитуда.

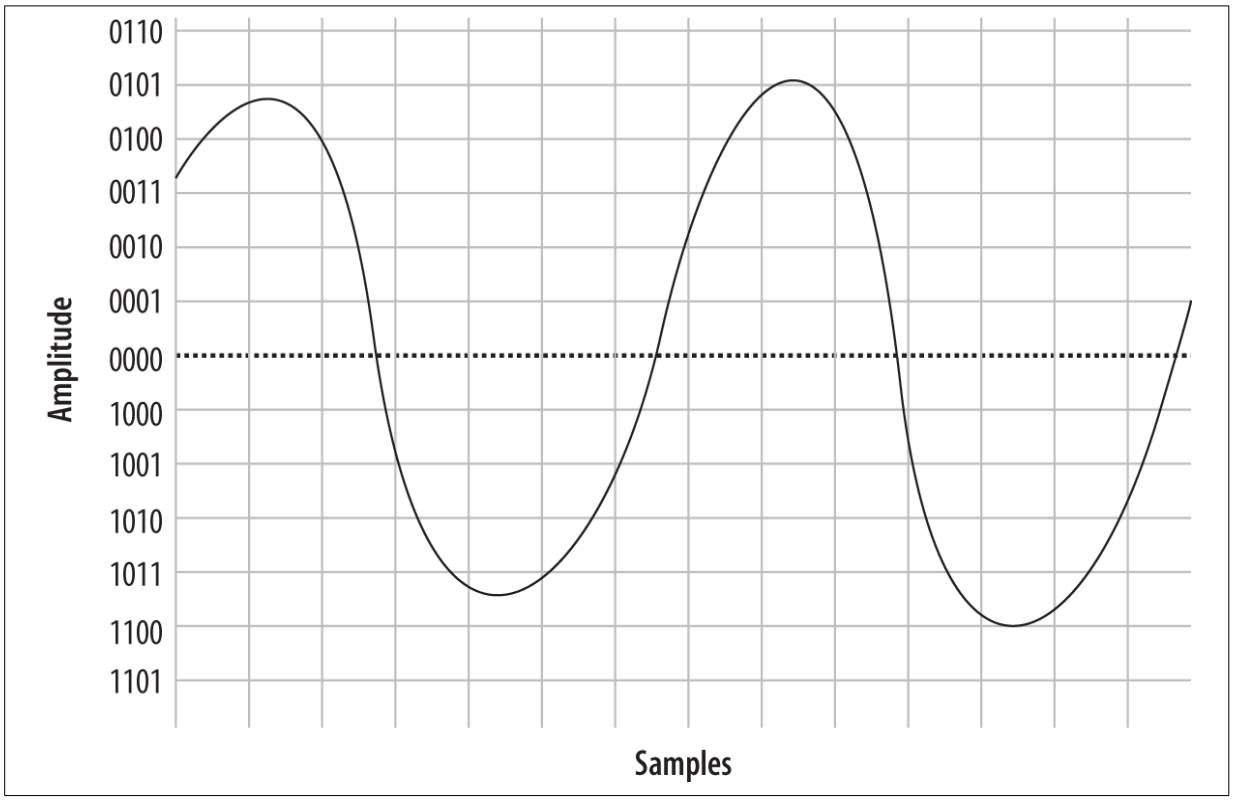


Рисунок A-2. Простая синусоидальная (гармоническая) волна

На Рисунке A-3 показана информация, которая будет захвачена при выборке синусоидальной волны с 4-битным разрешением.

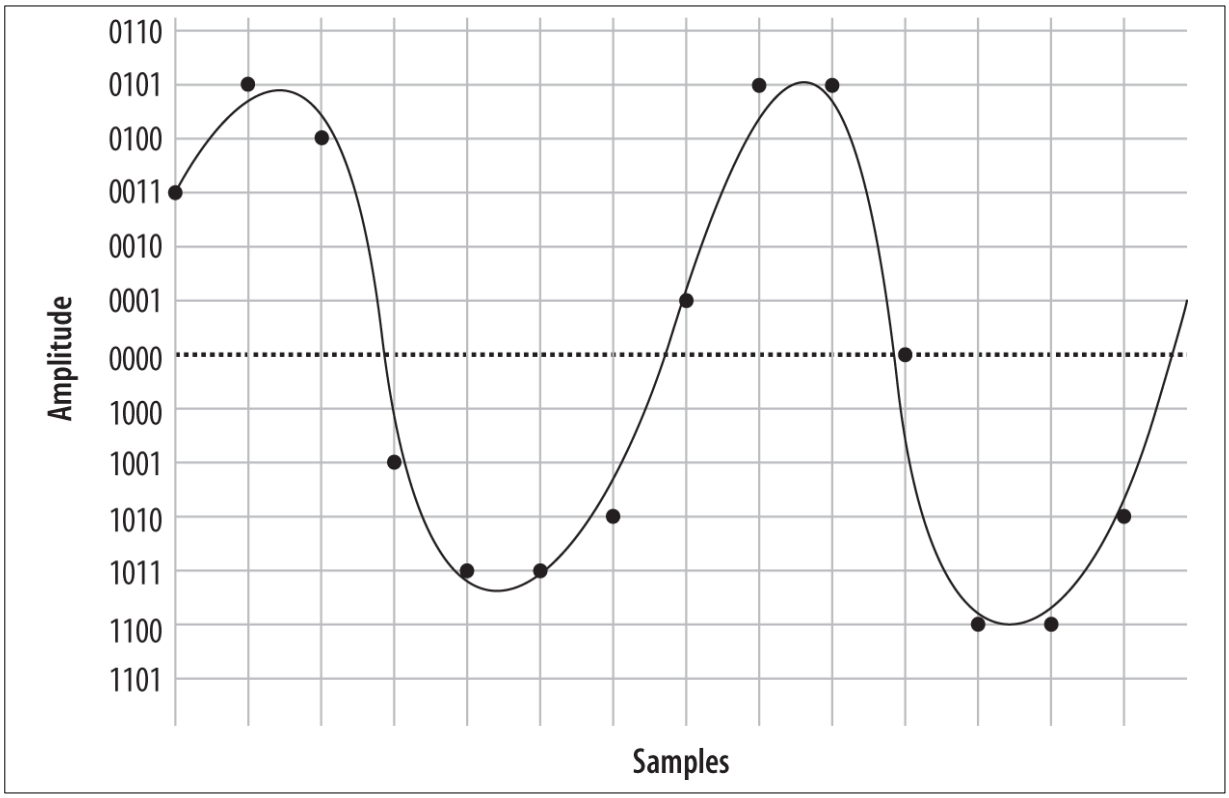


Рисунок A-3. Выборка нашей синусоидальной волны при использовании 4 бит

На каждом временном интервале мы измеряем амплитуду волны и записываем соответствующую интенсивность—другими словами, отбираем ее. Вы заметите, что 4-битное разрешение ограничивает нашу точность. Первый образец должен быть округлен до `0011`, а следующее квантование дает образец `0101`. Затем идет `0100`, затем `1001`, `1011` и так далее. Всего у нас 14 образцов (на самом деле нужно брать несколько тысяч образцов в секунду).

Если мы строим вместе все значения, мы можем отправить их на другую сторону как:

```
0011 0101 0100 1001 1011 1011 1010 0001 0101 0101 0000 1100 1100 1010
```

При передаче по проводам, этот код может выглядеть примерно как Рисунке А-4.

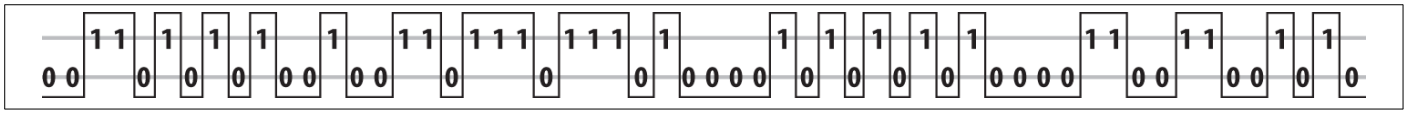


Рисунок А-4. ИКМ-кодированная волна

Когда цифро-аналоговый (D/A) преобразователь дальнего конца получает этот сигнал, он может использовать информацию для построения образцов, как показано на Рисунке А-5.

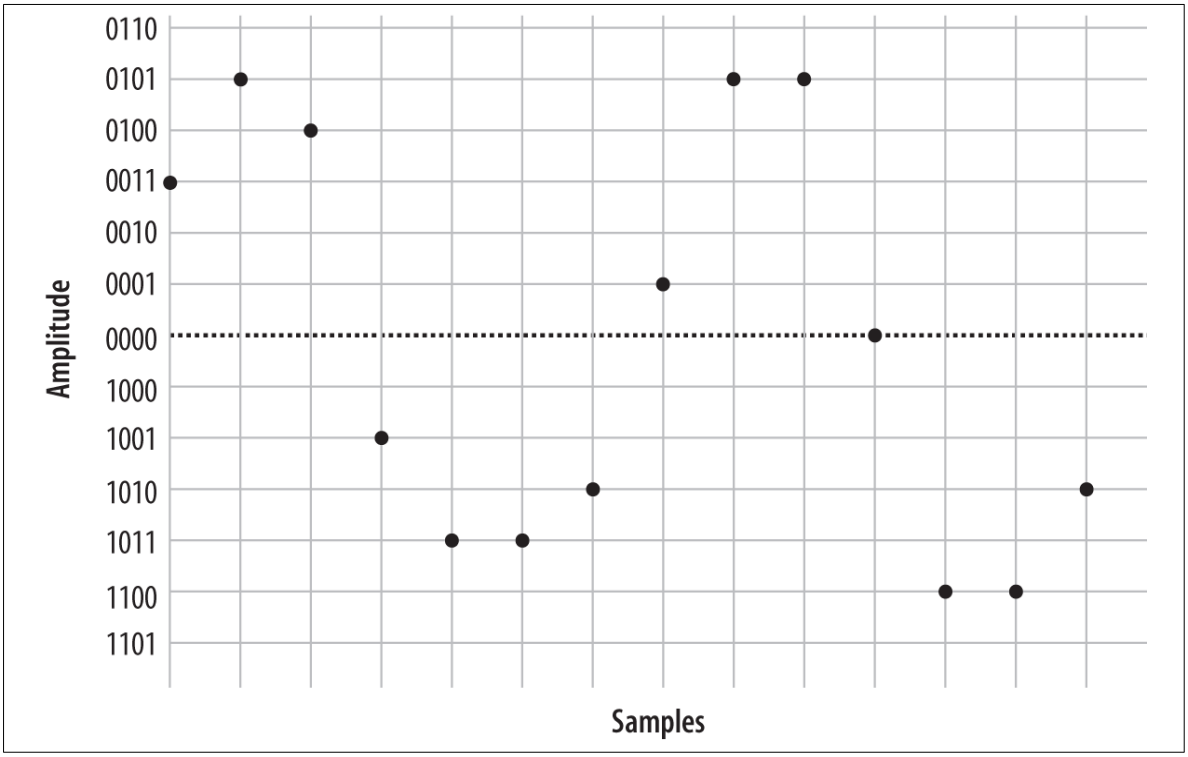


Рисунок А-5. График PCM сигнала

Из этой информации, сигнал может быть восстановлен (см. Рисунок А-6).

Как вы можете видеть, если вы сравните Рисунок А-2 с Рисунком А-6, эта реконструкция формы волны не очень точна. Это было сделано намеренно, чтобы продемонстрировать важный момент: качество цифровой кодированной формы волны зависит от разрядности и частоты, с которой она отбирается. При слишком низкой частоте дискретизации и слишком низком разрешении выборки качество звука будет неприемлемым.

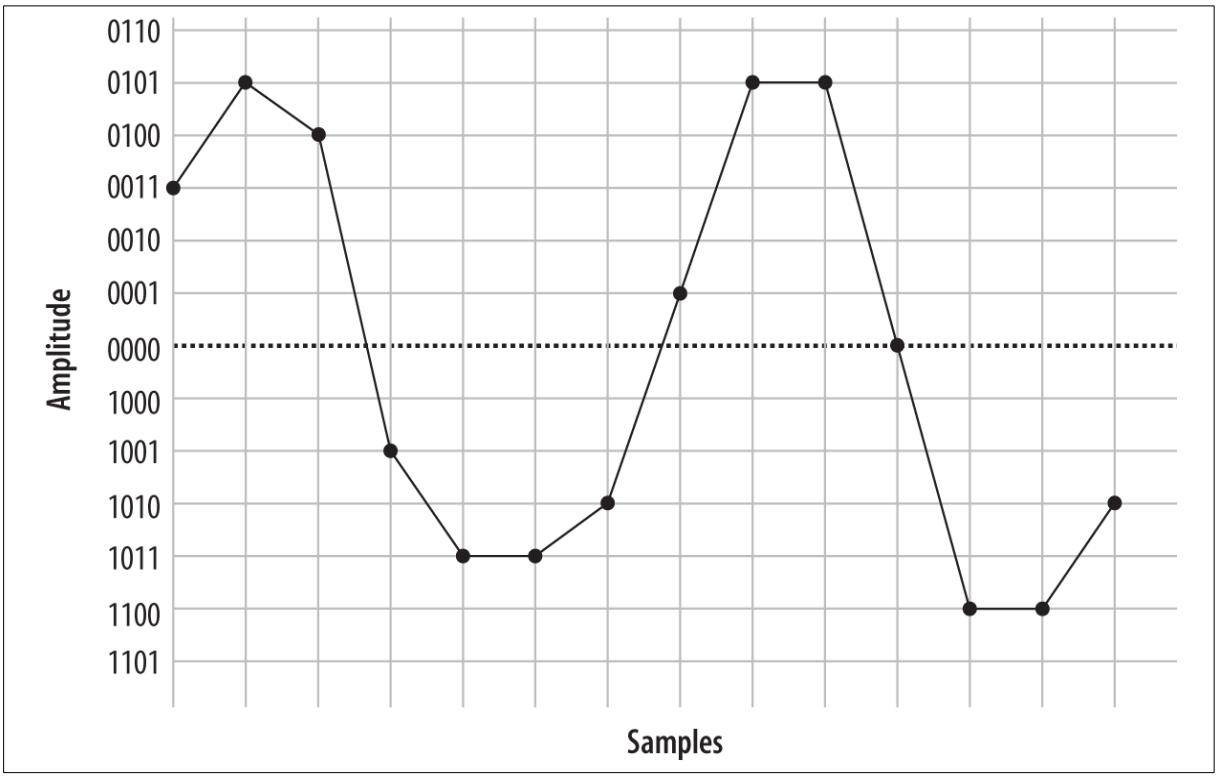


Рисунок A-6. Сигнал без сглаживания

Увеличение разрешения и частоты дискретизации

Давайте еще раз взглянем на наш исходный сигнал, на этот раз с использованием 5 битов для определения интервалов квантования (Рисунок A-7).

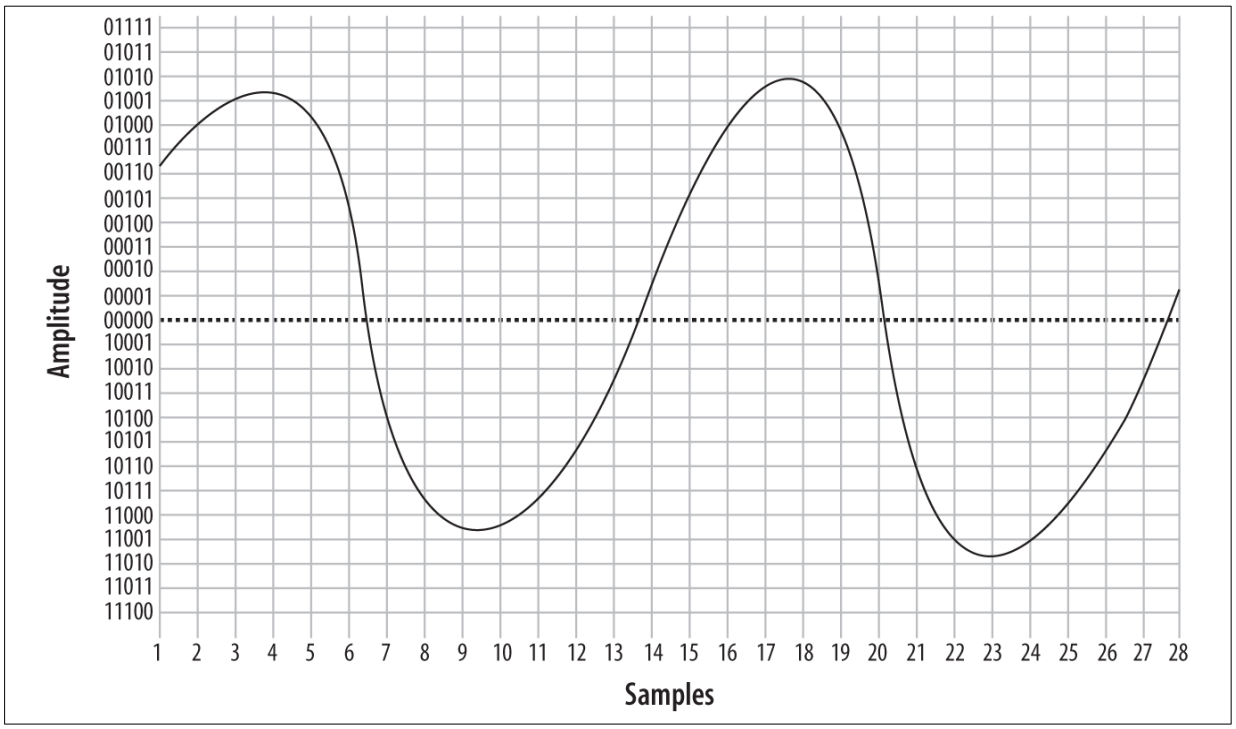


Рисунок A-7. Та же форма волны, на наложении с более высоким разрешением



На самом деле, нет такой вещи, как 5-битная ИКМ. В телефонной сети образцы ИКМ кодируются с использованием 8 бит. Другие цифровые аудио методы могут использовать 16 бит или более.

Мы также удвоим частоту дискретизации. Точки, построенные на этот раз, показаны на Рисунке A-8.

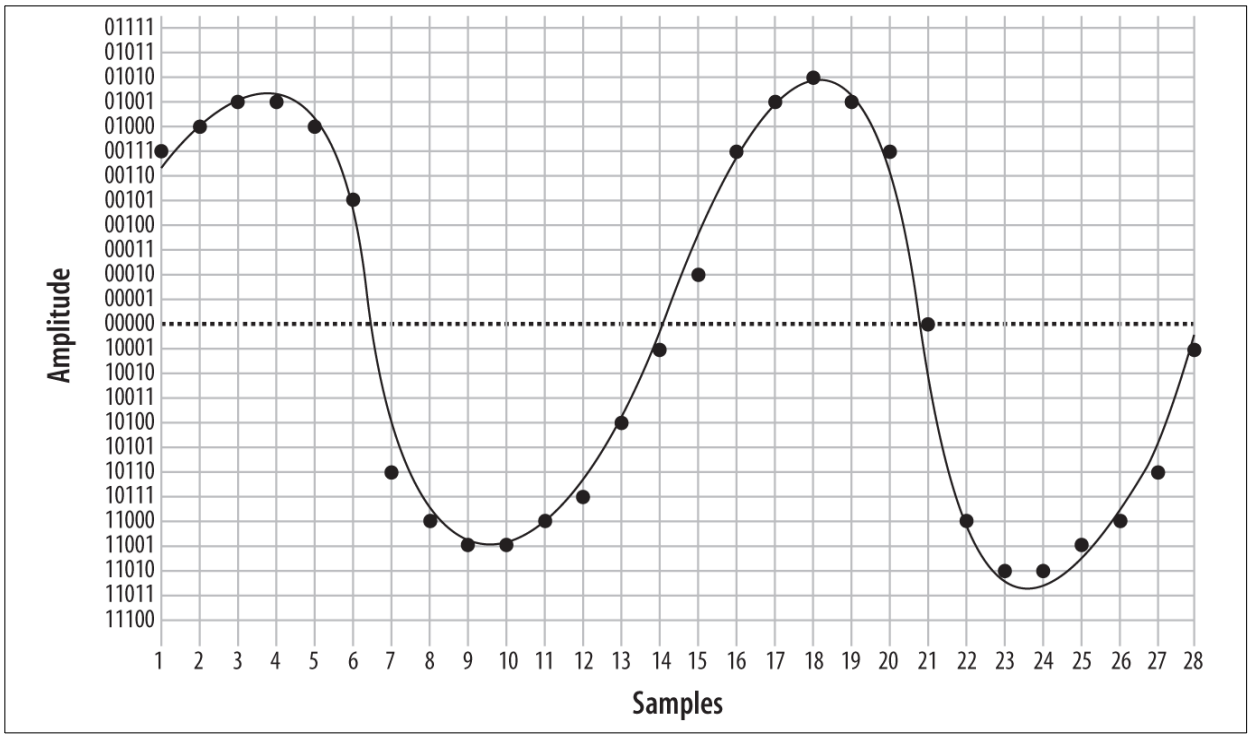


Рисунок A-8. Та же форма волны при двойном разрешении

Теперь у нас в два раза больше образцов, разрешение в два раза больше. Вот они:

```
00111 01000 01001 01001 01000 00101 10110 11000 11001 11001 11000 10111
10100 10001 00010 00111 01001 01010 01001 00111 00000 11000 11010 11010
11001 11000 10110 10001
```

Полученная на другом конце, эта информация теперь может быть нанесена на график, как показано на Рисунке A-9.

Из этой информации, сигнал, показанный на Рисунке A-10 может быть сгенерирован.

Как вы можете видеть, результирующая форма волны является гораздо более точным представлением оригинала. Однако, вы также можете видеть, что есть еще возможности для совершенствования.



Обратите внимание, что для кодирования сигнала с 4-битным разрешением требовалось 40 бит, а для передачи того же сигнала с 5-битным разрешением-156 бит (а также удвоение частоты дискретизации). Дело в том, что есть компромисс: чем выше качество звука, которое вы хотите кодировать, тем больше бит требуется для этого, и чем больше бит вы хотите отправить (в режиме реального времени, естественно), тем больше пропускной способности вам потребуется.

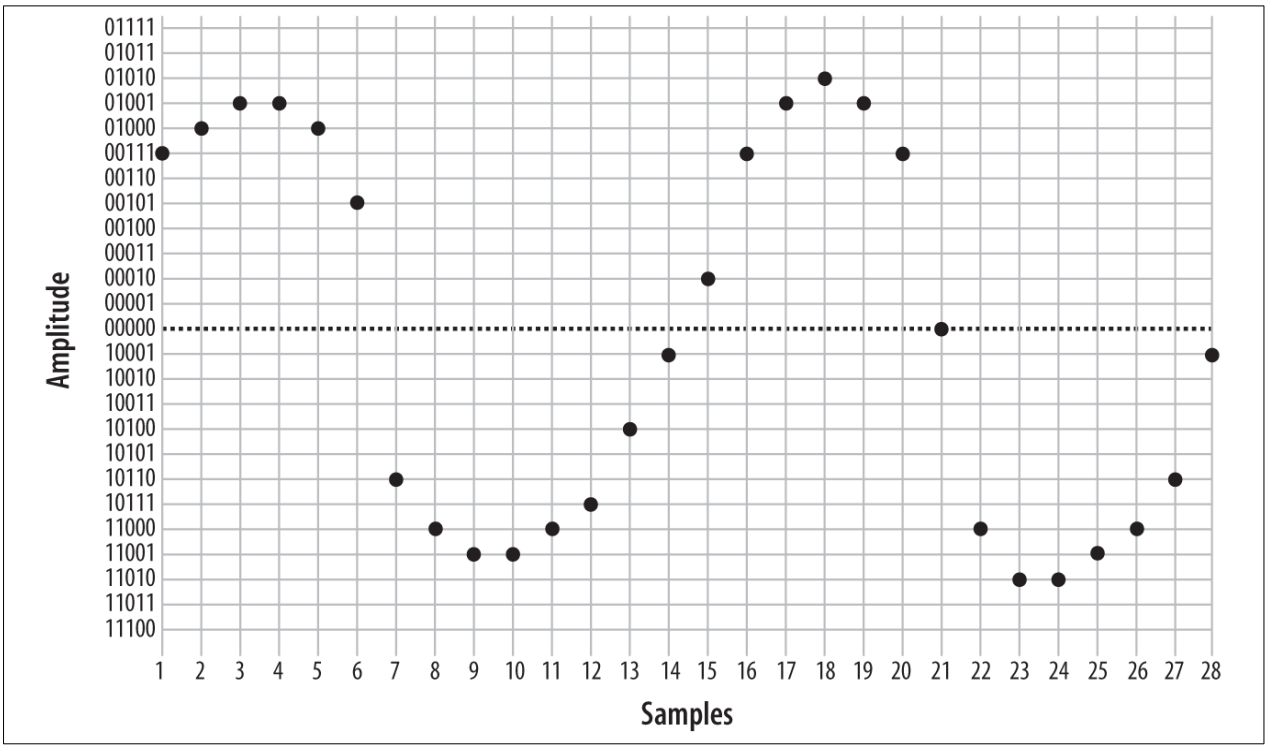


Рисунок A-9. Пятибитное представление ИКМ-сигнала

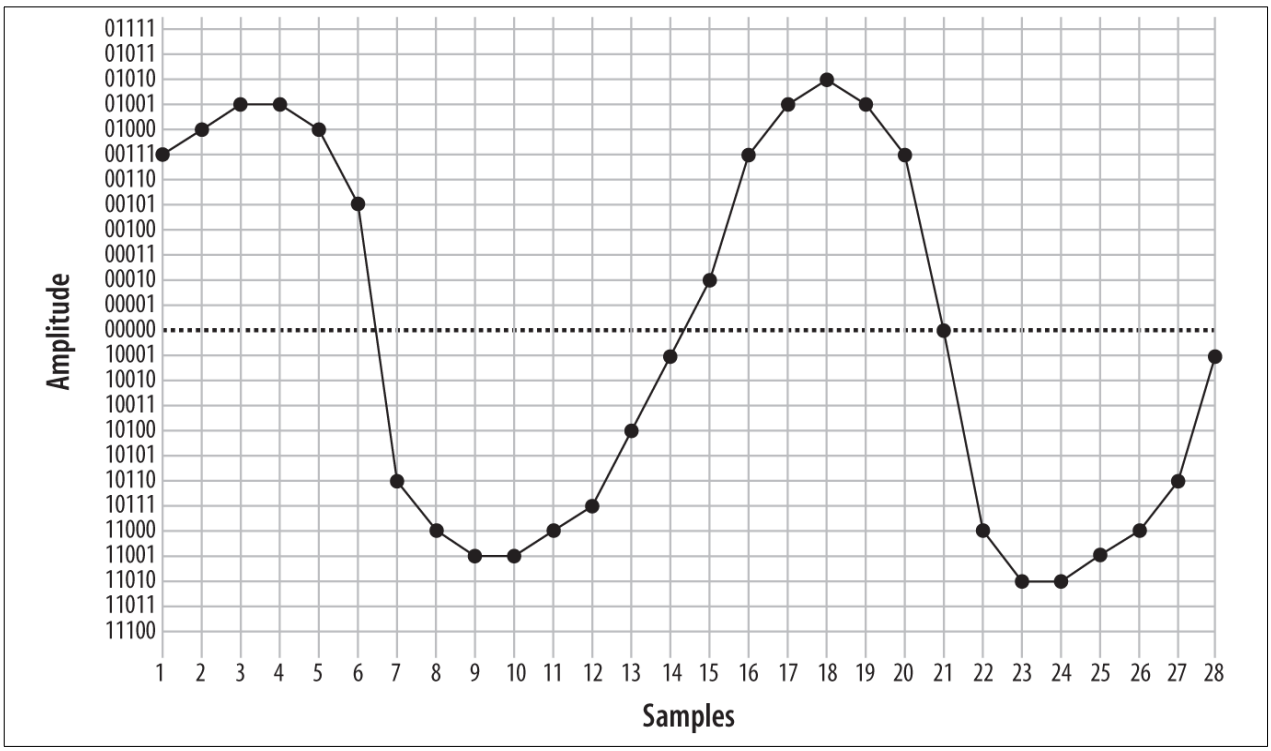


Рисунок A-10. Форма волны без сглаживания при 5-битной ИКМ

Теорема Найквиста

Сколько проб (сэмплов, частоты) достаточно? Этот же вопрос рассматривался в 1920-х годах инженером-электриком (и сотрудником AT&T/Bell) по имени Гарри Найквист. Теорема Найквиста гласит: "При дискретизации сигнала *частота дискретизации* должна быть больше, чем в два раза полосы пропускания входного сигнала, чтобы иметь возможность полностью восстановить оригинал из дискретизированной версии."[6]

6 Найквист опубликовал две статьи "Некоторые факторы, влияющие на скорость Телеграфа" (1924) и "Некоторые темы в теории телеграфной передачи" (1928), в которых он постулировал то, что стало известно как теорема Найквиста. Доказанная в 1949 году Клодом Шенноном ("Связь в присутствии шума"), она также упоминается как теорема выборки Найквиста-Шеннона.

По сути, это означает, что для точного кодирования аналогового сигнала вы должны замерять его вдвое чаще, чем общую пропускную способность, которую вы хотите воспроизвести. Поскольку телефонная сеть не будет нести частоты ниже 300 Гц и выше 4000 Гц, частота дискретизации 8000 проб в секунду будет достаточной для воспроизведения любой частоты в пределах полосы пропускания аналогового телефона. Имейте в виду, что 8000 образцов в секунду; мы поговорим об этом позже.

Логарифмическое компандирование

Мы рассмотрели основы квантования, и мы обсудили тот факт, что большее количество интервалов квантования (т.е. более высокая частота дискретизации) дают лучшее качество, но также требуют большей пропускной способности. Наконец, мы обсудили минимальную частоту дискретизации, необходимую для точного измерения диапазона частот, которые мы хотим передать (в случае телефона это 8000 Гц). Это все начинает складываться в справедливый бит данных, отправляемых по проводам, поэтому мы хотим поговорить о компандировании.

Компандирование - это метод улучшения динамического диапазона метода выборки без потери важной точности. Оно работает путем квантования более высоких амплитуд в гораздо грубой форме, нежели более низкие амплитуды. Другими словами, если вы кричите в свой телефон, вы не будете кодированы так чисто, как вы будете говорить нормально. Крик также не хорош для вашего кровяного давления, поэтому лучше его избегать.

Обычно используются два метода компандирования: μ-law[7] в Северной Америке и A-law в остальном мире. Они действуют по одним и тем же принципам, но в остальном несовместимы друг с другом.

Компандирование делит форму волны на *хорды*, каждая из которых имеет несколько *шагов*. Квантование включает соответствие измеренной амплитуды соответствующему шагу внутри хорды. Значением полосы и номера хорды (а также знака—положительного или отрицательного) становится сигнал. Следующие диаграммы дадут вам визуальное представление о том, что делает компандирование. Они не основаны на каком-либо стандарте, а скорее были составлены с целью иллюстрации (опять же, в телефонной сети компандирование будет осуществляться с 8-битным, а не 5-битным разрешением).

Рисунок A-11 иллюстрирует 5-разрядное компандирование. Как вы можете видеть, амплитуды вблизи точки пересечения нуля будут отбираться гораздо точнее, чем более высокие (положительные или отрицательные). Однако, поскольку человеческое ухо, передатчик и приемник также склонны искажать громкие сигналы, это не проблема.

7 μlaw часто называют "u-law", потому что, давайте посмотрим правде в глаза, у скольких из нас есть клавиши μ на клавиатурах? μ на самом деле греческая буква Мю; таким образом, вы также увидите, что μlaw написано (более правильно) как "Мю-лоу". Когда говорят, правильно уверенно говорить" мью-лоу“, но если люди смотрят на вас странно, и вы чувствуете себя глупо, вы можете помочь им и сказать им, что это "ю-лоу".- Многие просто не ценят пустяков.

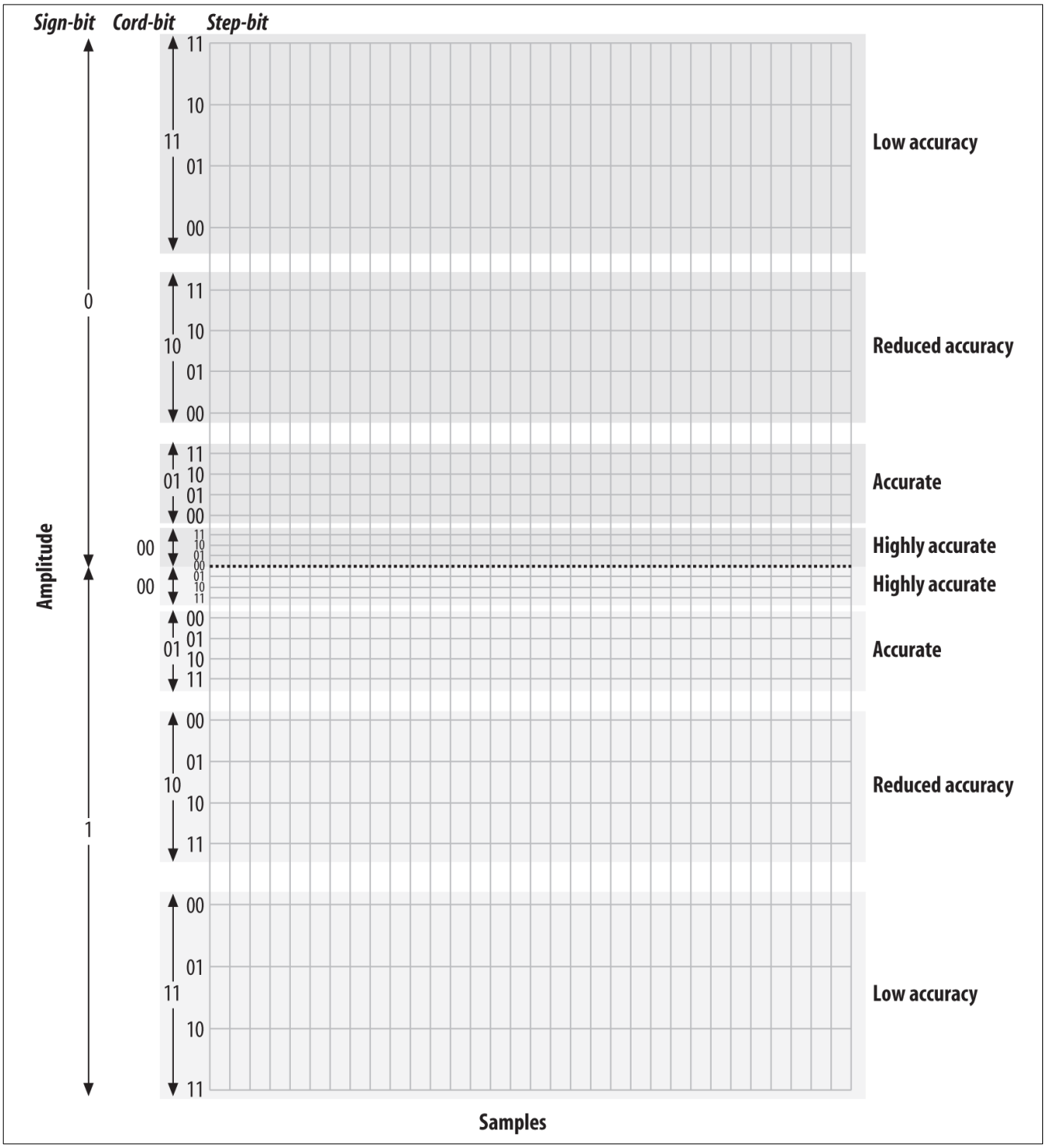


Рисунок A-11. Пятибитное компандирование

Квантованный образец может выглядеть как на Рисунке A-12. Он дает следующий битовый поток:

```
00000 10011 10100 10101 01101 00001 00011 11010 00010 00001 01000 10011
10100 10100 00101 00100 00101 10101 10011 10001 00011 00001 00000 10100
10010 10101 01101 10100 00101 11010 00100 00000 01000
```

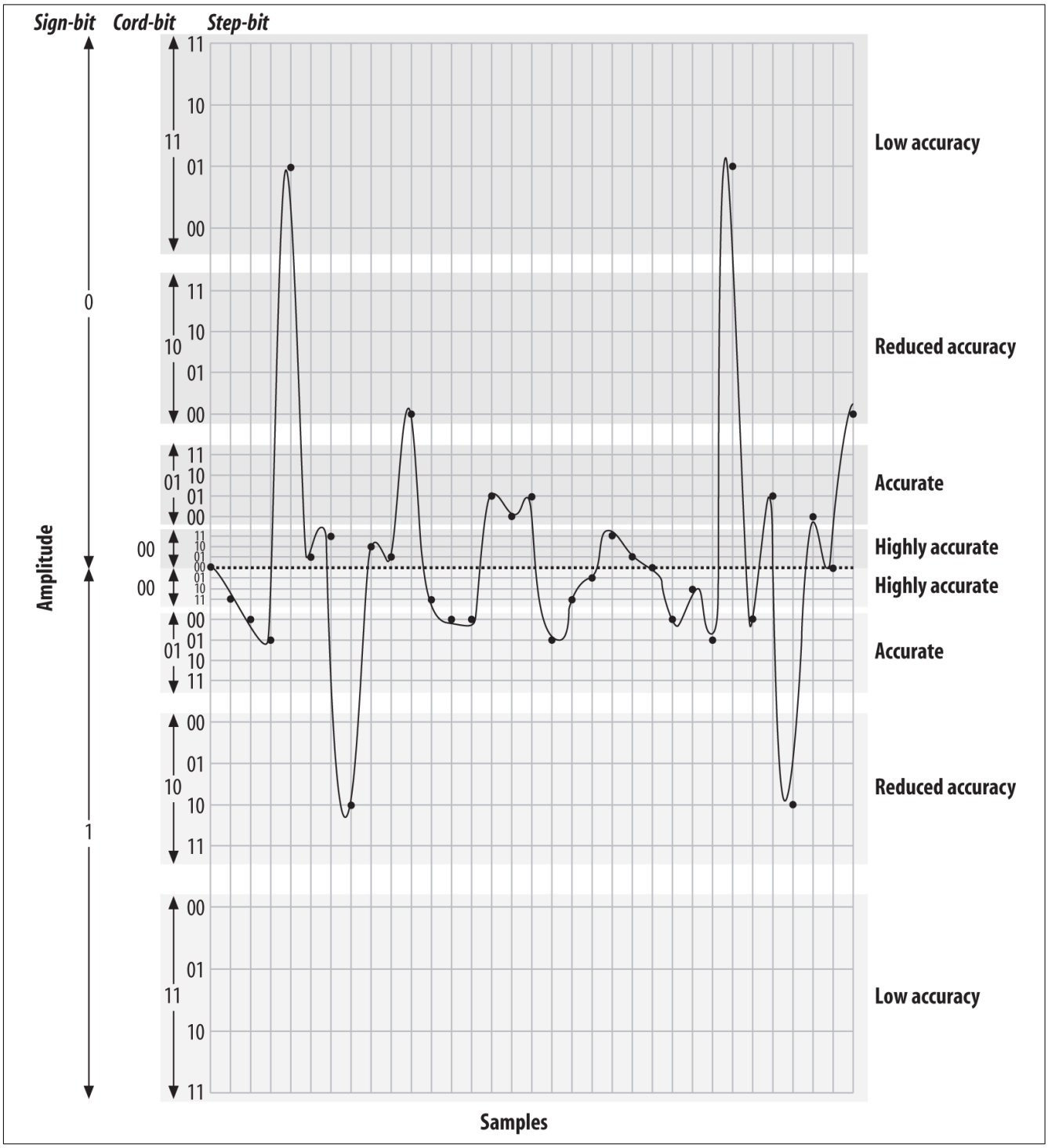


Рисунок A-12. Квантование и компандирование с 5-битным разрешением

Наложение

Если вы когда-нибудь видели, как колеса фургона поворачиваются назад в старом западном фильме, вы видели эффекты сглаживания. Частота кадров в кино не может идти в ногу с частотой вращения спиц, и это воспринимается как ложное вращение.

В цифровой аудиосистеме (которой, возможно, является современный ТфОП) сглаживание всегда происходит, если частоты, превышающие половину частоты дискретизации, представлены аналого-цифровому (A/D) преобразователю. В ТфОП это включает любые звуковые частоты выше 4000 Гц

(половина частоты дискретизации 8000 Гц). Эта проблема легко исправляется путем пропускания звука через фильтр нижних частот[8] перед передачей его в A/D конвертер.[9]

Цифровая коммутируемая телефонная сеть

На протяжении более ста лет, телефонные сети исключительно коммутируемые. Это означало, что для каждого телефонного звонка между двумя конечными точками устанавливалось специальное соединение с фиксированной полосой пропускания, выделенной для этого канала. Создание такой сети обходится дорого, а что касается расстояния, то использование этой сети также обходится дорого. Хотя мы все предсказываем конец сети с коммутацией каналов, многие люди все еще используют ее каждый день, и она действительно работает довольно хорошо.

Типы линий связи

В ТфОП, множество различных размеров линий служат различным потребностям сети. Между центральным офисом и абонентом обычно достаточно одной или нескольких аналоговых схем или нескольких десятков каналов, передаваемых по цифровой линии. Между офисами ТфОП (и с большим количеством клиентов), вообще используются оптоволоконные линии.

Простой DS-0 — это основа всего

Поскольку стандартным методом оцифровки телефонного звонка является запись 8-битной выборки 8000 раз в секунду, мы можем видеть, что телефонная линия, закодированная ИКМ, будет нуждаться в пропускной способности 8 раз 8000 бит в секунду, или 64 000 бит/с. Этот канал 64 Kbps называется *DS-0* (это”Dee-Ess-Zero", Ди-Эсс-Зиро). DS-0 является основным строительным блоком всех цифровых телекоммуникационных схем.

Даже вездесущая аналоговая схема переходит на DS-0 как можно скорее. Иногда это происходит в месте, где ваша линия заканчивается в центральном офисе, а иногда и задолго до этого.[10]

Линия с Т-несущей

Древний T1 является одним из наиболее признанных терминов цифровой телефонии. T1 - это цифровая схема, состоящая из 24 DS-0, мультиплексированных вместе в поток бит 1.544 Mbps.[11] Этот битовый поток правильно определен как DS-1. Голос кодируется на T1 с использованием алгоритма компандирования µlaw.



Европейская версия T1 была разработана Европейской конференцией администраций почтовых служб и служб связи (CEPT)[12] и впервые была названа CEPT-1. Теперь он называется E1.
E1 состоит из 32 DS-0, но метод кодирования PCM отличается: E1 использует компандирование a-law. Это означает, что для соединения между сетью на основе E1 и сетью на основе T1 всегда потребуется шаг перекодирования. Обратите внимание, что E1, хотя и

8 Фильтр нижних частот, как следует из его названия, пропускает только частоты, которые ниже его частоты среза. Другими типами фильтров являются фильтры высоких частот (которые удаляют низкие частоты) и полосовые фильтры (которые отфильтровывают как высокие, так и низкие частоты и пропускают только определенную полосу пропускания).

9 Если вам когда-либо придется делать аудиозаписи для системы, вы можете воспользоваться полосовым фильтром, встроенным в большинство телефонных аппаратов. Выполнение записи с использованием даже высококачественного записывающего оборудования может улавливать все виды фонового шума, который вы даже не слышите, пока не уменьшите, в этот момент фоновый шум производит сглаживание (что может звучать как все виды странных вещей). И наоборот, телефон уже захватывает звук в правильном формате, поэтому шум никогда не попадает в аудиопоток. Сказав все это, независимо от того, что вы используете для записи, избегайте сред с большим фоновым шумом. Типичные офисы могут быть намного шумнее, чем вы думаете, поскольку оборудование HVAC может производить шум, который мы даже не понимаем, есть.

10 Цифровые телефонные аппараты (включая IP-телефоны) выполняют аналого-цифровое преобразование в точке, где телефонная трубка подключается к телефону, поэтому DS-0 создается прямо на телефоне.

11 24 линий DS-0 используют 1.536 Мбит/с, а остальные .008 Mbps используются битами синхронизации.

12 Conférence Européenne des Administrations des Postes et des Télécommunications.

имеет 32 канала, также считается DS-1. Вполне вероятно, что E1 гораздо более широко развернут, поскольку он используется повсюду в мире, кроме Северной Америки и Японии.

Различные другие T-носители (T2, T3 и T4) кратны T1, каждый из которых основан на простом DS-0. В Таблице A-2 показаны взаимосвязи между различными цепями с T-несущей.

Таблица A-2. Линии с T-несущей

Несущая	Эквивалентный битрейт данных	Количество DS-0	Битрейт данных
T1	24 DS-0	24	1,544 Мбит/с
T2	4 T1	96	6,312 Мбит/с
T3	7 T2	672	44,736 Мбит/с
T4	6 T3	4 032	274,176 Мбит/с

Очень редко можно увидеть схему T-несущей при плотности выше T3. Для этих скоростей могут использоваться схемы оптоволоконных линий связи (optical carrier - OC).

SONET и носители OC

Синхронная оптическая сеть (Synchronous Optical Network- SONET) была разработана из желания вывести T-несущую систему на следующий технологический уровень: волоконную оптику. SONET основан на пропускной способности T3 (44.736 Мбит/с), а с небольшими накладными расходами 51.84 Мбит/с. Это называется OC-1 или STS-1. Как показано в Таблице A-3, все высокоскоростные схемы OC кратны этой базовой скорости.

Таблица A-3. OC линии

Несущая	Эквивалентный битрейт данных	Количество DS-0	Битрейт данных
OC-1	1 DS-3 (плюс накладные расходы)	672	51,840 Мбит/с
OC-3	3 DS-3	2016	155,520 Мбит/с
OC-12	12 DS-3	8064	622,080 Мбит/с
OC-48	48-DS-3	32256	2488,320 Мбит/с
OC-192	192 DS-3	129024	9953,280 Мбит/с

SONET был создан в попытке стандартизировать оптические схемы, но из-за его высокой стоимости в сочетании со значением, предлагаемым многими новыми схемами, такими как с плотным мультиплексированием с разделением по длине волны (DWDM), есть некоторые споры вокруг его будущего.

Цифровые сигнальные протоколы

Как и в любой линии, недостаточно, чтобы линии, используемые в ТфОП, просто несли данные (голосовые) между конечными точками. Необходимо также предусмотреть механизмы для передачи информации о состоянии канала между конечными точками. (ответ и разъединение - два примера базовой сигнализации, которая может потребоваться; caller ID - пример более сложной формы сигнализации.)

Сигнализация по выделенному каналу (Channel-associated signaling - CAS)

Также известный как обмен сигналами с резервированием битов, CAS используется для передачи голоса на T1, когда ISDN недоступен. Вместо того, чтобы воспользоваться преимуществами цифровой линии, CAS имитирует аналоговые каналы. CAS работает путем "кражи" битов из аудиопотока для целей сигнализации. Хотя влияние на качество звука не очень заметно, отсутствие мощного сигнального канала ограничивает гибкость.

При настройке CAS T1 параметры сигнализации на каждом конце должны совпадать. E&M (Ear

& Mouth or recEive & transMit) сигнализация является предпочтительной, по мере того как она предлагает наилучший контроль. Сказав это, в среде Asterisk наиболее вероятной причиной использования CAS будет для банка каналов, который обычно требует сигнализации FXS.

CAS очень редко используется на линиях ТфОП, больше из-за превосходства ISDN-PRI. Одним из ограничений CAS является то, что он не позволяет динамическое назначение каналов различным функциям. Кроме того, информация об идентификаторе вызывающего абонента (которая может даже не поддерживаться) должна быть отправлена как часть аудиопотока. CAS обычно используется на канале T1 в банках каналов.

ISDN

Цифровая сеть с интеграцией служб (ISDN) существует уже более 20 лет. Поскольку она отделяет каналы, которые несут трафик (каналы-носители или B-каналы) от канала, который несет сигнальную информацию (D-канал), ISDN позволяет предоставлять гораздо более богатый набор функций, чем CAS.

В начале ISDN обещала предоставить почти такую же функциональность, что и Интернет, включая расширенные возможности для передачи голоса, видео и данных. К сожалению, вместо того, чтобы ратифицировать стандарт и придерживаться его, соответствующие производители телекоммуникаций решили добавить свои собственные настройки в протокол, полагая, что их версии превосходят и в конечном итоге станут доминировать на рынке. В результате, получение двух ISDN-совместимых систем для подключения друг к другу часто было болезненной и дорогостоящей задачей. Поставщики услуг связи, которые должны были внедрять и поддерживать эту дорогостоящую технологию, в свою очередь, оценили ее так, чтобы она не была быстро принята. В настоящее время ISDN редко используется для гораздо большего, чем базовый транкинг—фактически, аббревиатура ISDN стала шуткой в телекоммуникационной отрасли: “It Still Does Nothing" (она по-прежнему ничего не делает).

Сказав это, ISDN стала довольно популярной для транкинга, и теперь она (в основном) соответствует стандартам. Если у вас есть УАТС с более чем десятком линий, подключенных к ТфОП, есть очень хороший шанс, что вы будете работать в схеме ISDN-PRI (первичный интерфейс скорости). Кроме того, в местах, где DSL и кабельный доступ к интернету недоступны (или слишком дороги), схема ISDN-BRI (Basic Rate Interface) может предоставить вам доступное соединение 128 Кб/с. В большей части Северной Америки использование BRI для подключения к Интернету было устаревшим в пользу DSL и кабельных модемов (и никогда не используется для голоса), но во многих европейских странах BRI почти полностью заменил аналоговые сети.

ISDN-BRI/BRA. Интерфейс базовой скорости (или доступ к базовой скорости) ISDN предназначен для обслуживания небольших конечных точек, таких как рабочие станции.

Этот колорит часто называют просто "ISDN", но это может быть источником путаницы, поскольку ISDN-это протокол, а не тип сети (не говоря уже о том, что сети PRI также правильно называются ISDN!).

Базовая скорость ISDN сети состоит из двух 64 Кбит/с B-каналов, управляемых 16 Кбит/с D-каналом, в общей сложности 144 Кбит/с.

Базовая ставка ISDN была источником большой путаницы в течение своей жизни из-за проблем с соблюдением стандартов, технической сложностью и плохой документацией. Тем не менее, многие европейские телеком-компании широко внедрили ISDN-BRI, и поэтому он более популярен в Европе, чем в Северной Америке.

ISDN-PRI/PRA. Интерфейс первичного уровня (или доступ к первичной скорости) ISDN используется для предоставления службы ISDN через более крупные сетевые подключения. Сеть ISDN-PRI использует один канал DS-0 в качестве сигнального канала (D-канал); остальные каналы служат B-каналами.

В Северной Америке ISDN-PRI обычно переносится на одну или несколько цепей T1. Поскольку T1 имеет 24 канала, североамериканская PRO-схема обычно состоит из 23 B-каналов и 1 D-канала. По этой причине PRI иногда называют 23B+D.[13]



В Европе используется 32-канальная схема E1, поэтому первичная схема ISDN называется 30B+D (конечный канал используется для синхронизации).

ISDN-PRI очень популярна, из-за своих технических преимуществ и вообще конкурентного ценообразовани на более высоких плотностях. Если вы считаете, что вам потребуется более дюжины или около того строк ТфОП, вы должны изучить ISDN-PRI.

С технической точки зрения ISDN-PRI всегда предпочтительнее CAS.

Signaling System 7

Signaling System 7 (SS7) - это система сигнализации, используемая поставщиками услуг связи. Он концептуально аналогичен ISDN, и играет важную роль в обеспечении механизма для несущих для передачи дополнительной информации, которую конечные точки ISDN обычно должны передавать. Однако технология SS7 отличается от технологии ISDN; одно большое различие заключается в том, что SS7 работает в совершенно отдельной сети, чем фактические магистрали, которые передают вызовы.

Поддержка SS7 в Asterisk находится на горизонте, так как существует большой интерес к обеспечению совместимости Asterisk с сетями поставщиков услуг. Существует версия SS7 с открытым исходным кодом, но работа по-прежнему необходима для полного соответствия SS7, и на момент написания этой статьи неизвестно, будет ли эта версия интегрирована с Asterisk. Еще один перспективный источник поддержки SS7 - Sangoma Technologies, которая предлагает функциональность SS7 во многих своих продуктах.

Следует отметить, что добавление поддержки SS7 в Asterisk будет не так просто, как написание правильного драйвера. Подключение оборудования к сети SS7 невозможно без того, чтобы это оборудование прошло чрезвычайно строгие процессы сертификации. Даже в этом случае представляется сомнительным, что какой-либо традиционный поставщик услуг будет спешить допустить такое, главным образом по стратегическим и политическим причинам.

Сети с коммутацией пакетов

В середине 1990-х годов производительность сети улучшилась до такой степени, что стало возможным передавать поток медиаинформации в режиме реального времени через сетевое соединение. Поскольку поток мультимедиа разделяется на сегменты, которые затем заключаются в конверт адресации, такие соединения называются *пакетными*. Задача, конечно, состоит в том, чтобы отправить поток этих пакетов между двумя конечными точками, гарантируя, что пакеты прибудут в том же порядке, в котором они были отправлены, менее чем за 150 миллисекунд, не потеряв ни один. В этом суть передачи голоса по IP.

Вывод

В настоящем примечании рассматриваются технологии, используемые в настоящее время в ТфОП. В Приложении B мы обсудим протоколы для VoIP: передачу телефонных соединений через IP-сети. Эти протоколы определяют различные механизмы ведения телефонных разговоров, но их значение гораздо больше. Включение телефонной сети в сеть передачи данных в конечном итоге приведет к стиранию границ между телефонами и компьютерами, что обещает революцию в нашем общении.

13 PRI на самом деле немного более гибкий, поскольку может охватить одну схему PRI через несколько DS1. Это может привести, например, к схеме 47B+D (где один D-канал служит двум T1) или схеме 46B+2D (где первичный и резервный D-каналы служат паре T1). Иногда PRI описывается как nB+nD, потому что число B-и D-каналов на самом деле весьма изменчиво. По этой причине вы никогда не должны ссылаться на T1, несущий PRI, как “PRI."Насколько вам известно, схема PRI охватывает несколько T1, как это часто бывает в больших развертываниях УАТС.

ПРИЛОЖЕНИЕ B

Протоколы VoIP

Интернет - это телефонная система,
которая стала наглой.
—Клиффорд Столл

Телекоммуникационная отрасль насчитывает более 100 лет, и Asterisk интегрирует большинство—если не все из основных технологий, которые она использовала за последнее столетие. Чтобы максимально использовать Asterisk, вам не нужно быть профессионалом во всех областях, но понимание различий между кодеками и протоколами даст вам большую оценку и понимание системы в целом.

Это приложение объясняет Voice через IP и то, что отличает VoIP-сети от традиционных голосовых сетей с коммутацией каналов, которые были темой Приложения A мы рассмотрим необходимость протоколов VoIP, изложив историю и потенциальное будущее каждого из них. Мы также рассмотрим соображения безопасности и возможности этих протоколов для работы в таких топологиях, как преобразование сетевых адресов (NAT). Будут обсуждаться следующие протоколы VoIP (некоторые более кратко, чем другие):

- IAX
- SIP
- H.323
- MGCP
- Skinny/SCCP
- UNISTIM

Кодеки — это средства, с помощью которых аналоговый голос может быть преобразован в цифровой сигнал и перенесен через Интернет. Пропускная способность в любом месте конечна, и количество одновременных разговоров, которые может нести любое соединение, напрямую связано с типом реализованного кодека. Мы также рассмотрим различия между следующими кодеками в отношении требований к пропускной способности (уровень сжатия) и качества:

- G.711
- G.726
- G.729A
- GSM
- iLBC
- Speex
- G.722
- MP3

Затем мы завершим приложение обсуждением того, как можно надежно маршрутизировать голосовой трафик, что вызывает эхо и как с ним бороться, и как Asterisk контролирует аутентификацию входящих и исходящих вызовов.

Необходимость протоколов VoIP

Основной предпосылкой VoIP является пакетизация[1] аудиопотоков для транспорта по сетям на основе интернет-протокола. Проблемы в достижении этого связаны с тем, как люди общаются. Сигнал должен поступать не только в том же виде, в каком он был передан, но и с задержкой менее чем 150 миллисекунд. Если пакеты теряются или задерживаются, качество общения ухудшается, а это означает, что двум людям будет трудно вести разговор.

Транспортные протоколы, которые в совокупности называются "интернет", изначально не были разработаны с учетом потокового мультимедиа в реальном времени. Предполагалось, что конечные точки будут уничтожать недостающие пакеты, не дожидаясь их прибытия, запрашивая повторную передачу или, в некоторых случаях, считая, что информация исчезла навсегда и передача просто продолжается без нее. В типичном голосовом потоке эти механизмы не сработают. Наши разговоры плохо адаптируются к потере букв или слов, а также к какой-либо заметной задержке между передачей и получением.

Традиционный ТфОП был конструирован специфически для передачи голоса, и он совершенно подходит к задаче с технической точки зрения. Однако с точки зрения гибкости его недостатки очевидны даже для людей с очень ограниченным пониманием технологии.

VoIP обещает включить голосовую связь во все другие протоколы, которые мы осуществляем в наших сетях, но из-за особых требований голосового трафика необходимы специальные навыки для проектирования, создания и обслуживания этих сетей.

Проблема с пакетной передачей голоса связана с тем, что способ, которым мы говорим, полностью несовместим со способом передачи данных по IP. Разговор и прослушивание состоят из ретрансляции потока аудио, в то время как интернет-протоколы предназначены для разбиения всего, инкапсуляции битов информации в тысячи пакетов, а затем доставки каждого пакета любым возможным способом в дальний конец. Очевидно, что необходим какой-то способ справиться с этим.

Протоколы VoIP

Механизм для передачи соединения VoIP обычно включает в себя серию сигнальных транзакций между конечными точками (и шлюзами между ними), кульминацией которых являются два постоянных медиа-потока (по одному для каждого направления), которые несут фактический разговор. Для этого существует несколько протоколов. В этом разделе мы обсудим некоторые, которые важны для VoIP в целом и для Asterisk в частности.

IAX (“Inter-Asterisk eXchange” протокол)

Если вы утверждаете, что являетесь одним из людей, знающих, когда дело доходит до Asterisk, ваш тест придет, когда вам нужно произнести имя этого протокола. Казалось бы, вы должны сказать “ай-эй-экс”, но это вряд ли очень хорошо сходит с языка.[2] К счастью, правильное произношение на самом деле "иикс".[3] IAX — это открытый протокол, то есть любой может скачать и разработать для него.[4]

В Asterisk IAX поддерживается модулем *chan_iax2.so*.

История

Протокол IAX был разработан Digium для связи с другими серверами Asterisk (отсюда и название Inter-Asterisk eXchange). Очень важно отметить, что IAX вовсе не ограничивается Asterisk. Стандарт

1 Это слово еще не попало в словарь, но оно становится все более распространенным. Оно относится к процессу разбиения устойчивого потока информации на дискретные куски (или пакеты), пригодные для доставки независимо друг от друга.

2 Звучит как название голландской футбольной команды.

3 Вперед. Скажите это. Это звучит намного лучше, не так ли?

4 Официально текущая версия-IAX2 (официально стандартизированная IETF в RFC 5456), но вся поддержка IAX1 была отброшена, поэтому, говорите ли вы “IAX” или “IAX2”, ожидается, что вы говорите о версии 2.

открыт для любого пользователя и поддерживается многими другими телекоммуникационными проектами с открытым исходным кодом, а также несколькими поставщиками оборудования. IAX-это транспортный протокол (очень похожий на SIP), который использует один UDP-порт (4569) как для сигнализации канала, так и для медиа-потоков. Как обсуждается ниже в этом приложении, это упрощает управление, когда NAT за брандмауэрами.

IAX также обладает уникальной способностью транка нескольких сеансов в один поток данных, что может привести к огромному преимуществу пропускной способности при отправке большого количества одновременных каналов на удаленный блок. Транкинг позволяет представлять несколько потоков мультимедиа с одним заголовком дейтаграммы, что снижает накладные расходы, связанные с отдельными каналами. Это помогает снизить задержку и уменьшить требуемую вычислительную мощность и пропускную способность, позволяя протоколу намного легче масштабироваться с большим количеством активных каналов между конечными точками. Если у вас есть большое количество IP-вызовов для передачи между двумя конечными точками, вы должны пристальней взглянуть на транкинг IAX.

Будущее

Поскольку IAX был оптимизирован для голоса, он получил некоторую критику за не лучшую поддержку видео, но на самом деле IAX имеет потенциал для переноса практически любого медиа—потока. Поскольку это открытый протокол, будущие типы медиа обязательно будут включены, как того пожелает сообщество.

Соображения безопасности

IAX включает возможность аутентификации тремя способами: обычный текст, хеширование MD5 и обмен ключами RSA. Это, конечно, ничего не делает для шифрования пути медиапотока или заголовков между конечными точками. Многие решения включают использование устройства или программного обеспечения виртуальной частной сети (VPN) для шифрования потока на другом уровне технологии, который требует, чтобы конечные точки предварительно установили метод настройки и открытия этих туннелей. Тем не менее, IAX теперь также может шифровать потоки между конечными точками с динамическим обменом ключами при настройке вызова (используя параметр конфигурации encryption=aes128), позволяя использовать автоматическое переключение ключа.

IAX и NAT

Протокол IAX2 был специально разработан для работы устройств, находящихся за NAT. Использование одного UDP-порта как для сигнализации, так и для передачи мультимедиа также позволяет свести к минимуму количество дыр в брандмауэре. Эти соображения помогли сделать IAX одним из самых простых протоколов (если не самым простым) для реализации в защищенных сетях.

SIP

Протокол установления сеанса (Session Initiation Protocol - SIP) взял телекоммуникационную отрасль штурмом. SIP в значительной степени сверг некогда предпочтительный H.323 в качестве протокола VoIP - конечно, в конечных точках сети. Предпосылка SIP заключается в том, что каждый конец соединения является одноранговым (пиром); протокол согласовывает возможности между ними. Что делает SIP привлекательным, так это то, что это относительно простой протокол с синтаксисом, подобным синтаксису других знакомых протоколов, таких как HTTP и SMTP. SIP поддерживается в Asterisk с помощью модуля *chan_sip.so*[5].

5 Только что назвав SIP простым, следует отметить, что он отнюдь не легкий. Было сказано, что если бы кто-то прочитал все PPC IETF, которые имеют отношение к SIP, у него было более 3000 страниц чтения. SIP быстро зарабатывает репутацию слишком раздутого, но это не уменьшает его популярности.

История

SIP был первоначально представлен целевой группе по разработке Интернета (IETF) в феврале 1996 года в качестве “draft-ietf-mmusic-sip-00"."Первоначальный проект не выглядел как SIP, который мы знаем сегодня, и содержал только один тип запроса: запрос на установку вызова. В марте 1999 года, после 11 ревизий, родился SIP RFC 2543.

Сначала SIP был полностью проигнорирован, так как H.323 считался предпочтительным протоколом для согласования транспорта VoIP. Однако, по мере того как рос шум, SIP начал приобретать популярность, и хотя многие различные факторы, возможно, ускорили его рост, мы хотели бы думать, что большая часть его успеха связана с его свободно доступной спецификацией.

SIP - это протокол сигнализации уровня приложения, который использует известный порт 5060 для связи. SIP можно транспортировать с помощью протоколов транспортного уровня UDP или TCP. SIP используется для " установки, изменения и завершения мультимедийных сеансов, таких как вызовы интернет-телефонии.”[6]

SIP не транспортирует медиапоток (т.е. голос) между конечными точками. Вместо этого для этой цели используется транспортный протокол реального времени (RTP). RTP использует пронумерованные, непривилегированные порты в Asterisk (по умолчанию 10000-20000).

Общая топология для иллюстрации SIP и RTP, обычно называемая “трапецией SIP”, показана на Рисунке B-1 . Когда Алиса хочет позвонить Бобу, телефон Алисы связывается с ее прокси-сервером, и прокси пытается найти Боба (часто подключаясь через его прокси). После того, как телефоны начали вызов, они общаются непосредственно друг с другом (если это возможно), так что данные не должны связывать ресурсы прокси.

SIP был не первым и не единственным протоколом VoIP, используемым сегодня (существуют также H.323, MGCP, IAX и т.д.), но в настоящее время он, похоже, набирает наибольшую популярность у поставщиков оборудования. Преимущества протокола SIP заключаются в его широком признании и архитектурной гибкости (и, как мы привыкли говорить, простотой!).

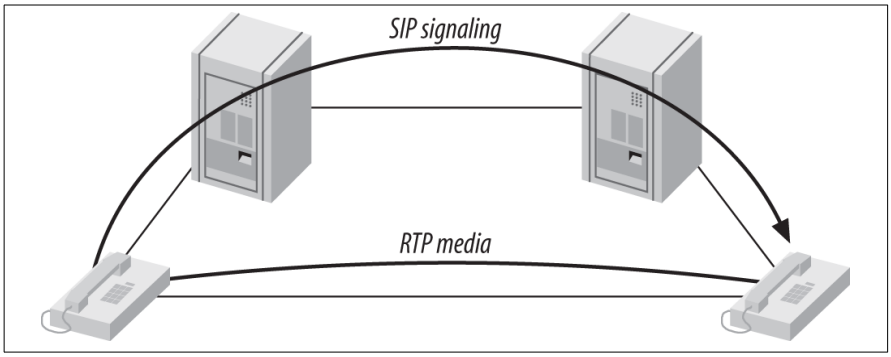


Рисунок B-1. Трапеция SIP

Будущее

SIP заслужил свое место в качестве протокола, который оправдал VoIP. Ожидается, что все новые пользовательские и корпоративные продукты будут поддерживать SIP, и любые существующие продукты теперь будут продаваться с трудом, если не будет предложен путь миграции на SIP. Ожидается, что SIP предоставит гораздо больше возможностей, чем VoIP, включая возможность передачи видео, музыки и любого типа мультимедиа в реальном времени. В то время как его использование в качестве вездесущего механизма общего назначения для транспортировки медиапотоков представляется сомнительным, SIP, бесспорно, готов предоставить большинство новых голосовых приложений в течение следующих нескольких лет.

6 RFC 3261, “SIP: Session Initiation Protocol,” p. 9, Section 2.

Соображения безопасности

SIP использует систему запрос/ответ для аутентификации пользователей. Первоначальный INVITE отправляется прокси-серверу, с которым конечное устройство желает связаться. Затем прокси-сервер отправляет сообщение 407 Proxy Authorization Request, которое содержит случайный набор символов, называемый nonce. Этот nonce используется вместе с паролем для создания хэша MD5, который затем отправляется обратно в последующем INVITE. Предполагая, что хэш MD5 совпадает с тем, который был создан прокси-сервером, клиент затем аутентифицируется.

Атаки на отказ в обслуживании (DoS), вероятно, являются наиболее распространенным типом атаки на VoIP-связь. DoS-атака может произойти, когда большое количество недопустимых запросов INVITE отправляются на прокси-сервер в попытке перегрузить систему. Эти атаки относительно просты в реализации и их воздействие на пользователей системы является незамедлительным. SIP имеет несколько методов минимизации последствий DoS-атак, но, в конечном счете, их невозможно предотвратить.

SIP реализует схему, гарантирующую что используется безопасный, зашифрованный механизм транспорта (а именно безопасность транспортного уровня или TLS) для установления связи между вызывающим абонентом и доменом вызываемого абонента. Кроме того, запрос безопасно отправляется на конечное устройство на основе локальной политики безопасности сети. Обратите внимание, что шифрование медиапотока (то есть потока RTP) выходит за рамки самого SIP и должно рассматриваться отдельно.

Дополнительные сведения о соображениях безопасности SIP, включая перехват регистрации, подмену сервера и разрыв сеанса, можно найти в Разделе 26 SIP RFC 3261.

SIP и NAT

Вероятно, самое большое техническое препятствие, которое SIP должен преодолеть, - это проблема проведения транзакций на уровне NAT. Поскольку SIP инкапсулирует информацию адресации в своих фреймах данных, а NAT находится на более низком сетевом уровне, информация адресации не изменяется автоматически, и, таким образом, медиа-потоки не будут иметь правильную информацию об адресации, необходимую для завершения соединения, когда NAT находится на месте. В дополнение к этому брандмауэры, обычно интегрированные с NAT, не будут рассматривать входящий поток мультимедиа как часть транзакции SIP и заблокируют соединение. Новые брандмауэры и пограничные контроллеры сеансов (session border controllers - SBC) знают о SIP, но это по-прежнему считается недостатком в этом протоколе, и не вызывает никаких проблем для сетевых специалистов, нуждающихся в подключении конечных точек SIP с использованием существующей сетевой инфраструктуры.

H.323

Этот протокол Международного союза электросвязи (МСЭ) был первоначально разработан для обеспечения механизма IP-транспорта для проведения видеоконференций. Он стал стандартом в IP-основе видеоконференционного оборудования, и он недолго пользовался известностью как протокол VoIP. В то время как существует много жарких дебатов о том, будет ли SIP или H.323 (или IAX) доминировать в мире протоколов VoIP, в Asterisk H.323 в значительной степени устарел в пользу IAX и SIP. H323 не пользуется большим успехом среди пользователей и предприятий, хотя он все еще может быть наиболее широко используемым протоколом VoIP среди провайдеров.

Три версии H.323, поддерживаемые в Asterisk, обрабатываются модулями *chan_h323.so* (поставляется с Asterisk), *chan_oh323.so* (доступно как бесплатное дополнение) и *chan_ooh323.so* (поставляется в *asterisk-addons*).



Вероятно, вы использовали H.323, даже не зная об этом—клиент Microsoft NetMeeting, возможно, является наиболее широко применяемым клиентом H.323.

История

H323 был разработан МСЭ в мае 1996 года в качестве средства передачи голоса, видео, данных и факсимильной связи по IP-сети при сохранении связи с ТфОП. С тех пор H.323 прошел через несколько версий и приложений (которые добавляют функциональность протоколу), что позволяет ему работать в чистых сетях VoIP и более распределенных сетях.

Будущее

Будущее H323 является предметом обсуждения. Если СМИ - это какая-то мера, то она не выглядит хорошо для H323; он почти никогда не упоминается (конечно, не с регулярностью SIP). H.323 часто рассматривается как технически превосходящий SIP, но такого рода вещи редко являются решающим фактором в том, пользуется ли технология успехом. Одним из факторов, делающих H.323 непопулярным, является его сложность (хотя многие утверждают, что когда-то простой SIP начинает страдать от той же проблемы).

H.323 по-прежнему несет на себе большую часть мирового трафика VoIP-операторов, но по мере того, как люди становятся менее зависимыми от традиционных операторов для своих телекоммуникационных потребностей, будущее H.323 становится все труднее предсказать с какой-либо уверенностью. Хотя H.323 может не быть протоколом выбора для новых реализаций, мы, безусловно, можем ожидать, что придется иметь дело с проблемами совместимости H.323 в течение некоторого времени.

Соображения безопасности

H.323 является относительно безопасным протоколом и не требует особых соображений безопасности, кроме тех, которые являются общими для любой сети, соединенной с Интернетом. Поскольку H.323 использует протокол RTP для медиа-коммуникаций, он изначально не поддерживает зашифрованные пути медиапотоки. Использование VPN или другого зашифрованного туннеля между конечными точками является наиболее распространенным способом безопасной инкапсуляции сообщений. Конечно, это имеет недостаток, когда требуется создание таких безопасных туннелей между конечными точками, что может быть не всегда удобно (или даже возможно). Поскольку VoIP все чаще используется для связи с финансовыми учреждениями, такими как банки, нам, вероятно, потребуются расширения наиболее часто используемых протоколов VoIP для поддержки надежных методов шифрования.

H.323 и NAT

Стандарт H.323 использует протокол RTP Инженерного совета Интернета (IETF) для транспортировки медиапотоков между конечными точками. Из-за этого H.323 имеет те же проблемы, что и SIP при работе с сетевыми топологиями с участием NAT. Самый простой способ-просто прокинуть соответствующие порты через устройство NAT внутреннему клиенту.

Для приема вызовов всегда необходимо перенаправить TCP-порт 1720 клиенту. Кроме того, вам нужно будет прокинуть UDP-порты для потоков RTP-media и RTP Control Protocol (RTCP) (см. руководство для вашего устройства для информации о диапазоне портов, который требуется). Для старых клиентов, таких как Microsoft NetMeeting, также потребуются TCP-порты, перенаправленные для туннелирования H.245 (опять же, см. руководство вашего клиента для диапазона номеров портов).

Если у вас есть несколько клиентов за NAT, вам нужно будет использовать *gatekeeper* (шлюз), работающий в режиме прокси. Шлюзу потребуется интерфейс, подключенный к частной IP-подсети и общедоступному интернету. Ваш клиент H.323 в частной IP-подсети затем зарегистрируется на шлюзе, который будет совершать прокси-вызовы от имени клиентов. Обратите внимание, что все внешние клиенты, которые захотят вам позвонить, также должны зарегистрироваться на прокси-сервере.

В настоящее время Asterisk не может действовать как шлюз H.323. Для этого вам придется использовать отдельное приложение, такое как свободный OpenH323 Gatekeeper.

MGCP

Протокол контроля медиашлюзов (MGCP) также разработан в IETF. Хотя развертывание MGCP более распространено, чем можно было бы подумать, оно быстро теряет почву для таких протоколов, как SIP и IAX. Тем не менее, Asterisk любит протоколы, поэтому, естественно, у него есть рудиментарная поддержка.

MGCP определен в RFC 3435[7] - он был разработан, чтобы сделать конечные устройства (такие как телефоны) максимально простыми и иметь всю логику вызова и обработку, обрабатываемую медиа-шлюзами и агентами вызовов. В отличие от SIP, MGCP использует централизованную модель. Телефоны MGCP не могут напрямую вызывать другие телефоны MGCP; они всегда должны проходить через какой-либо контроллер.

Asterisk поддерживает MGCP через модуль *chan_mgcp.so* и конечные точки, определенные в файле конфигурации *mgcp.conf*. Поскольку Asterisk предоставляет только базовые услуги агента вызовов, он не может эмулировать телефон MGCP (например, для регистрации на другом контроллере MGCP в качестве агента пользователя).

Если у вас есть телефоны MGCP, вы сможете использовать их с Asterisk. Если вы планируете запустить MGCP-телефоны в продакшен в Asterisk, имейте в виду, что сообщество перешло на более популярные протоколы, и поэтому вам необходимо будет соответствующим образом финансировать поддержку программного обеспечения. Если возможно (например, как с телефонами Cisco), вы должны обновить телефоны MGCP до SIP.

Проприетарные протоколы

Наконец, давайте рассмотрим два проприетарных протокола, которые поддерживаются в Asterisk.

Skinny/SCCP

Протокол управления Skinny клиентом (Skinny Client Control Protocol - SCCP) является проприетарным протоколом оборудования Cisco VoIP. Это протокол по умолчанию для конечных точек на УАТС Cisco Call Manager.[8] Skinny поддерживается в Asterisk, но если вы подключаете телефоны Cisco к Asterisk, чаще рекомендуется получить прошивку с поддержкой SIP для любых телефонов, в которых это возможно и подключаться через SIP вместо SCCP.

UNISTIM

Поддержка Asterisk проприетарного протокола VoIP Nortel, UNISTIM, делает его первой УАТС в истории, которая изначально поддерживает проприетарные IP-терминалы от двух крупнейших игроков в VoIP: Nortel и Cisco. Поддержка UNISTIM является полностью экспериментальной и пока не работает достаточно хорошо, чтобы запустить ее в продакшен, но тот факт, что кто-то взял на себя труд реализовать ее, демонстрирует мощь платформы Asterisk.

Кодеки

Под *кодеками*, как правило, понимаются различные математические модели, используемые для цифрового кодирования (и сжатия) аналоговой аудиоинформации. Многие из этих моделей учитывают способность человеческого мозга формировать впечатление из неполной информации. Мы все видели оптические иллюзии; точно так же алгоритмы сжатия голоса используют нашу склонность интерпретировать то, что как мы *полагаем*, должны услышать, а не то, что мы *на самом*

7 RFC 3435 делает устаревшим RFC 2705.

8 Cisco недавно объявила, что она будет мигрировать на SIP в своих будущих продуктах.

деле слышим.[9] Целью различных алгоритмов кодирования является достижение баланса между эффективностью и качеством.[10]

Первоначально термин *кодек* относился к КОдеру/ДЕКодеру: устройству, которое преобразует между аналоговым и цифровым. Теперь КОмпрессия/ДЕКомпрессия кажется более подходящим.

Прежде чем мы углубимся в отдельные кодеки, взгляните на Таблицу B-1—это краткий справочник, к которому вы можете обратиться.

Таблица B-1. Быстрая ссылка на кодеки

Кодек	Битрейт данных (Кбит/с)	Требуется лицензия
G.711	64 Кбит/с	Нет
G.726	16, 24, 32 или 40 Кбит/с	Нет
G.729A	8 Кбит/с	Да (нет для сквозного пропуска)
GSM	13 Кбит/с	Нет
iLBC	13,3 Кбит/с (фреймы по 30 мс) или 15,2 Кбит/с (фреймы по 20 мс)	Нет
Speex	Переменный (между 2.15 и 22.4 Кбит/с)	Нет
G.722	64 Кбит/с	Нет

G.711

G.711 является фундаментальным кодеком ТфОП. Фактически, если кто-то ссылается на PCM (обсуждается в Приложении A) в отношении телефонной сети, вам необходимо о G.711. Используются два метода компандирования: μlaw в Северной Америке и A-law в остальном мире. Он обеспечивает передачу 8-разрядного слова 8000 раз в секунду. Если вы посчитаете, то увидите, что для этого требуется 64 000 бит для передачи в секунду.

Многие люди скажут вам, что G.711-это кодек без сжатия. Это не совсем так, поскольку компандирование считается формой сжатия. Верно то, что G.711-это базовый кодек, из которого получены все остальные.

G. 711 накладывает минимальную (почти нулевую) нагрузку на процессор.

G.726

Этот кодек существует уже некоторое время (раньше это был G.721, который теперь устарел), и это один из оригинальных кодеков сжатия. Он также известен как адаптивная дифференциальная импульсно-кодовая модуляция (ADPCM), и он может работать на нескольких скоростях. Самые распространенные 16 Кбит/с, 24 Кбит/с и 32 Кбит/с. На момент написания этой статьи Asterisk поддерживает только скорость ADPCM-32, которая является самой популярной скоростью для этого кодека.

G.726 предлагает почти идентичное G.711 качество, но использует только половину полосы пропускания. Это возможно потому, что вместо отправки результата измерения квантования он отправляет только достаточную информацию для описания разницы между текущей выборкой и предыдущей. Популярность G.726 упала в 1990-х годах из-за его неспособности передавать модемные и факсимильные сигналы, но из-за своего коэффициента пропускной

9 Читаем следующее: “По рзелульаттам илссеовадний одонго анлигйсокго унвиертисета, не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве. Галвоне, чотбы преавя и плоендяя бквуы блыи на мсете. Осатьлыне бкувы мгоут селдовтаь в плоонм бсепордяке, все-рвано ткест чтаитсея без побрелм. Пичрионй эгото ялвятеся то, что мы чиатем не кдаужю бкуву по отдльенотси, а все солво цликеом.” (Источник этой цитаты неизвестен.) То же самое мы делаем со звуком: если информации достаточно, наш мозг может заполнить пробелы.

10 На аудио CD качество гораздо важнее, чем экономия полосы пропускания, поэтому звук квантован на 16 бит (2 раза, так как это стерео), с частотой дискретизации 44,100 Гц. Учитывая, что компакт-диск был изобретен в конце 1970-х годов, это был довольно впечатляющий материал. Телефонная сеть не требует такого уровня качества (и должна оптимизировать пропускную способность), поэтому телефонные сигналы кодируются с использованием 8 бит, с частотой дискретизации 8000 Гц.

способности/производительности процессора он теперь возвращается. G.726 особенно привлекателен тем, что не требует от системы большой вычислительной мощности.

G.729A

Учитывая, как мало пропускной способности он использует, G.729A обеспечивает впечатляющее качество звука. Он делает это с помощью линейного предсказания с алгебраическим кодом сопряженной структуры (Conjugate-Structure Algebraic-Code-Excited Linear Prediction - CS-ACELP).[11] Из-за патентов вы не можете использовать G.729A без уплаты лицензионного сбора; однако он чрезвычайно популярен и хорошо поддерживается на многих разных телефонах и системах.

Для достижения впечатляющей степени сжатия этот кодек требует столь же впечатляющих усилий от процессора. В системе Asterisk использование сильно сжатых кодеков быстро загрузит процессор.

G.729A использует 8 Кбит/с пропускной способности.

GSM

Кодек глобальной системы мобильной связи (Global System for Mobile Communications - GSM) является любимцем Asterisk. Этот кодек не обременен лицензионным требованием, как в случае с G.729A и он предлагает отличную производительность по отношению к нагрузке которую он несет для CPU. Качество звука обычно считается худшим чем у G.729A, но многое из этого сводится к личному мнению; обязательно попробуйте. GSM работает на 13 Кбит/с.

iLBC

Кодек с низким битрейтом для Интернета (iLBC) обеспечивает привлекательное сочетание низкой пропускной способности и качества, и он особенно хорошо подходит для поддержания разумного качества на потерях сетевых соединений.

Естественно, Asterisk поддерживает его (и поддержка в других местах растет), но он не так популярен, как кодеки МСЭ, и поэтому может быть несовместим со многими IP-телефонами и коммерческими системами VoIP. IETF RFCs 3951 и 3952 были опубликованы в поддержку iLBC, и iLBC находится на пути стандартов IETF.

Поскольку iLBC использует сложные алгоритмы для достижения высокого уровня сжатия, он имеет довольно высокую нагрузку на процессор в Asterisk.

Хотя вам разрешено использовать iLBC без уплаты взноса, владелец патента iLBC Global IP Sound (GIPS) хочет знать, когда вы используете его в коммерческих приложениях. Это можно сделать, загрузив и распечатав копию лицензии iLBC, подписав ее и вернув в GIPS. Если вы хотите прочитать о iLBC и его лицензии, вы можете сделать это по адресу *http://www.ilbcfreeware.org*.

iLBC оперируетотает 13,3 Кбит/с (кадры по 30 мс) и 15.2 Кбит/с (кадры по 20 мс).

Speex

Speex - это кодек с переменным битрейтом (VBR), что означает, что он может динамически изменять свой битрейт, чтобы реагировать на изменение условий сети. Он предлагается как в узкополосных, так и в широкополосных версиях, в зависимости от того какое качество вы хотите получить (телефонного разговора или лучше).

---

11 CELP-популярный метод сжатия речи. Математически моделируя различные способы, которыми люди издают звуки, можно построить кодовую книгу звуков. Вместо отправки фактического дискретизированного звука определяется код, соответствующий звуку. Кодеки CELP берут эту информацию (которая сама по себе производит звук, очень похожий на робота) и пытаются добавить индивидуальность обратно. (Конечно, дело не только в этом.) Страница кодирования речи Джейсона Вудворда является источником полезной информации для людей, не склонных к математике. Это довольно тяжелый материал, так что наденьте свою мыслительную шапочку.

Speex - совершенно бесплатный кодек, лицензированный под *Xiph.org* вариантом лицензии BSD.

Проект Speex доступен в интернете, и более подробную информацию о Speex можно найти на его домашней странице.

Speex может работать в любом месте от 2.15 до 22.4 Кбит/с, из-за его переменного битрейта.

G.722

G.722 является стандартным кодеком ITU-T, который был утвержден в 1988 году. Кодек G.722 производит гораздо более качественный голос в том же пространстве, что и G.711 (64 Кбит/с), и начинает становиться популярным среди производителей VoIP-устройств. Патенты на G.722 истекли, поэтому он свободно доступен. Если у вас есть доступ к устройствам, поддерживающим G.722, вы будете впечатлены улучшением качества.

MP3

Конечно, MP3 — это кодек. В частности, это группа экспертов Moving Picture Audio Layer 3 Encoding Standart.[12] С таким именем, неудивительно, что мы называем его MP3! В Asterisk кодек MP3 обычно используется для хранения музыки на ожидание (MOH). MP3 не является кодеком телефонии, поскольку он оптимизирован для музыки, а не голоса; тем не менее, он очень популярен в системах VoIP-телефонии как метод доставки MOH.



Имейте в виду, что музыка обычно не может транслироваться без лицензии. Многие люди считают, что нет никаких юридических проблем с подключением радиостанции или компакт-диска в качестве источника музыки на удержании, но это очень редко верно.

Quality of Service

Качество обслуживания, или QoS, как более часто называют, относится к задаче доставки потока данных с учетом времени по сети, которая была разработана для доставки данных в специальной, наилучшей форме. Хотя нет жесткого правила, общепринято, что если вы можете доставить звук, производимый динамиком, к слушателю в течение 150 миллисекунд, возможен нормальный поток разговора. Когда задержка превышает 300 миллисекунд, становится трудно избежать перебивания друг друга. За 500 миллисекунд нормальный разговор становится все более неловким и неприятным.

В дополнение к тому, чтобы получить его вовремя, также важно обеспечить, чтобы передаваемая информация поступала неповрежденной. Слишком большое количество потерянных пакетов помешают дальнему концу полностью воспроизвести дискретизированный звук, а пробелы в данных будут слышны как статические или, в тяжелых случаях, целые пропущенные слова или предложения. Даже потеря 5 процентов пакетов может серьезно препятствовать передачи голоса по сети.

TCP, UDP и SCTP

Если вы собираетесь отправлять данные по IP-сети, они будут транспортироваться с использованием одного из трех транспортных протоколов, рассмотренных здесь.

Transmission Control Protocol

Протокол управления передачей (TCP) почти никогда не используется для VoIP, поскольку, хотя у него есть механизмы для обеспечения доставки, он по своей сути не спешит это делать. Если между двумя конечными точками нет соединения с чрезвычайно низкой задержкой, TCP, как правило, вызывает больше проблем, чем решает.

12 Если вы хотите узнать все о MPEG audio, выполните веб-поиск статьи Дэвиса Пэна под названием "Учебник по сжатию MPEG/Audio."

Цель TCP - гарантировать доставку пакетов. Для этого реализовано несколько механизмов, таких как нумерация пакетов (для восстановления блоков данных), подтверждение доставки и повторный запрос потерянных пакетов. В мире VoIP скорость получения пакетов к конечной точке имеет первостепенное значение, но 20 лет сотовой телефонии обучили нас терпеть несколько потерянных пакетов.[13]

Высокие затраты на обработку TCP, управление состоянием и подтверждение прибытия хорошо работают для передачи больших объемов данных, но они просто неэффективны для связи в реальном времени.

User Datagram Protocol

В отличие от TCP, протокол пользовательских дейтаграмм (UDP) не предоставляет никаких гарантий доставки. Пакеты помещаются в линию как можно быстрее и выпускаются в мир, чтобы найти свой путь к конечному пункту назначения, без каких-либо упоминаний о том, попали они туда или нет. Поскольку UDP сам по себе не дает никаких гарантий того, что данные поступят адресату,[14] он достигает своей эффективности, затрачивая очень мало усилий на то, что он транспортирует.



TCP является более "социально ответственным" протоколом, потому что пропускная способность более равномерно распределена среди клиентов, подключающихся к серверу. По мере увеличения процента трафика UDP возможно, что сеть может быть перегружена.

Stream Control Transmission Protocol

Утвержденный IETF в качестве предлагаемого стандарта в RFC 2960, протокол передачи с управлением потоком (SCTP) является относительно новым транспортным протоколом. С самого начала он был разработан для устранения недостатков как TCP, так и UDP, особенно в отношении типов услуг, которые раньше предоставлялись по сетям телефонии с коммутацией каналов.

Некоторые из целей SCTP:

- Лучшие методы предотвращения перегрузок (в частности, предотвращение атак типа "отказ в обслуживании)
- Строгая последовательность доставки данных
- Более низкая латентность для улучшенной передачи в реальном времени

Устраняя основные недостатки TCP и UDP, разработчики SCTP надеялись создать надежный протокол для передачи SS7 и других типов ТфОП-сигналов по сети на основе IP.

Дифференцированное обслуживание

Дифференцированное обслуживание или DiffServ, является не столько механизмом QoS, сколько методом, с помощью которого трафик может быть помечен и дан конкретный режим. Очевидно, что DiffServ может помочь обеспечить QoS, позволяя некоторым типам пакетов иметь приоритет над другими. Хотя это, безусловно, увеличит вероятность быстрого прохождения пакета VoIP через каждое соединение, это ничего не гарантирует.

Гарантированный сервис

Окончательная гарантия QoS обеспечена ТфОП. Для каждого разговора канал 64 Кбит/с полностью посвящен вызову; полоса пропускания гарантирована. Аналогичным образом, протоколы, предлагающие гарантированное обслуживание, могут гарантировать, что для обслуживаемого

13 Порядок прибытия важен в голосовой связи, потому что звук будет обработан и отправлен вызывающему абоненту как можно скорее. Однако с буфером jitter'a порядок прибытия не так важен, поскольку он предоставляет небольшое окно времени, в котором пакеты могут быть переупорядочены перед передачей вызывающему.

14 Имейте в виду, что протоколы верхнего уровня или приложения могут реализовать свои собственные системы подтверждения пакетов.

соединения выделен необходимый объем пропускной способности. Как и в случае любой пакетной сетевой технологией, эти механизмы обычно работают лучше всего, когда трафик ниже максимального уровня. Когда соединение приближается к своим пределам, почти невозможно устранить деградацию.

MPLS

Multiprotocol Label Switching (MPLS) - это метод построения шаблонов сетевого трафика независимо от таблиц маршрутизации 3 уровня. Протокол работает путем назначения коротких меток (кадров MPLS) сетевым пакетам, которые маршрутизаторы затем используют для пересылки пакетов маршрутизатору выхода MPLS и, в конечном итоге, к их конечным назначениям. Традиционно маршрутизаторы принимают независимое решение о переадресации на основе поиска таблицы IP при каждом переходе в сети. В сети MPLS этот поиск выполняется только один раз, когда пакет входит в облако MPLS на входящем маршрутизаторе. Затем пакет назначается потоку, называемому путем с коммутацией меток (Label-Switched Path - LSP) и идентифицируется меткой. Метка используется в качестве индекса поиска в таблице пересылки MPLS, и пакет проходит LSP независимо от решений маршрутизации 3 уровня. Это позволяет администраторам больших сетей точно настраивать решения о маршрутизации и наилучшим образом использовать сетевые ресурсы.

Кроме того, информация может быть связана с меткой для приоритизации переадресации пакетов.

RSVP

MPLS не содержит метода динамической установки LSP, но можно использовать протокол резервирования (RSVP) с MPLS. RSVP - это протокол сигнализации, используемый для упрощения установки LSP и сообщения о проблемах маршрутизатору входа MPLS. Преимуществом использования RSVP в сочетании с MPLS является сокращение административных накладных расходов. Если вы не используете RSVP с MPLS, вам придется перейти к каждому маршрутизатору и настроить метки и каждый путь вручную. Использование RSVP делает сеть более динамичной, распределяя управление метками маршрутизаторам. Это позволяет сети стать более отзывчивой к изменяющимся условиям, потому что ее можно настроить для изменения путей на основе определенных условий, таких как падение определенного направления (возможно, из-за неисправного маршрутизатора). Конфигурация внутри маршрутизатора затем сможет использовать RSVP для распространения новых меток маршрутизаторам в сети MPLS без (или минимального) вмешательства человека.

Негарантированная доставка

Самый простой, наименее дорогостоящий подход к QoS — не предоставлять его вообще — метод "Best Effort" (негарантированной доставки). Хотя это может показаться плохой идеей, на самом деле это может работать очень хорошо. Любой VoIP-вызов, который пересекает общедоступный Интернет, почти наверняка будет наилучшим, поскольку механизмы QoS еще не распространены в этой среде.

Эхо

Вы можете этого не понимать, но эхо было проблемой в ТфОП с тех пор, как появились телефоны. Вы, вероятно, не часто испытывали её, потому что телекоммуникационная индустрия потратила большие суммы на разработку дорогих устройств эхоподавления. Кроме того, когда конечные точки физически близки—например, когда вы звоните своему соседу по улице—задержка настолько минимальна, что все, что вы передаете, будет возвращено так быстро, что будет неотличимо от местного эффекта[15], обычно происходящего в вашем телефоне. Итак, факт в том, что эхо на ваших местных вызовах занимают большую часть времени, но вы не можете воспринимать его с помощью обычного телефона, потому что оно происходит почти мгновенно. Что может помочь вам понять его, если вы считаете, что когда вы стоите в комнате и говорите, все, что вы говорите, отражается от стен

15 Как описано в Приложении A, местный эффект - это функция в вашем телефоне, которая возвращает часть того, что вы говорите, обратно в ваше ухо, чтобы обеспечить более естественный разговор.

и потолка (и, возможно, пола, если он не покрыт ковром), но не вызывает никаких проблем, потому что происходит так быстро, что вы не воспринимаете задержку.

Причина, по которой телефонные системы VoIP, такие как Asterisk, могут испытывать эхо, заключается в том, что добавление VoIP-телефона вводит небольшую задержку. Это занимает несколько миллисекунд для пакетов, чтобы путешествовать с телефона на сервер (и наоборот). Внезапно возникает заметная задержка, которая позволяет воспринимать эхо, которое было всегда, но никогда не было заметно.

Почему возникает эхо

Прежде чем мы обсудим меры по борьбе с эхом, давайте сначала посмотрим, почему оно происходит в аналоговом мире.

Если вы слышите эхо, это не ваш телефон вызывает проблему; это проблема в дальнем конце цепи. И наоборот, эхо, услышанное на дальнем конце, генерируется на вашем. Эхо может быть вызвано тем, что аналоговая локальная петлевая линия должна передавать и принимать по одной паре проводов. Если эта схема не является электрически сбалансированной, или если некачественный телефон подключен к концу линии, сигналы, которые он получает, могут отражаться назад, становясь частью обратной передачи. Когда этот отраженный сигнал вернется к вам, вы услышите слова, которые вы говорили только что. Люди будут воспринимать эхо за пределами определенной задержки (возможно, до 20 миллисекунд для некоторых людей). Это эхо будет раздражать по мере увеличения задержки.

В дешевом телефоне эхо может генерироваться внутри телефонной трубки. Вот почему некоторые дешевые IP-телефоны могут вызывать эхо, даже если все сквозное соединение не содержит аналоговой схемы.[16] В мире VoIP эхо обычно генерируется либо аналоговой схемой где-то в соединении, либо дешевой конечной точкой, отражающей часть сигнала (например, обратная связь через громкую связь или плохо спроектированную трубку или гарнитуру). Чем больше задержка в сети, тем больше раздражает это эхо.

Управление эхом на каналах DAHDI

Вы можете включить и отключить эхоподавление для интерфейсов DAHDI в файле *chan_dahdi.conf*. Конфигурация по умолчанию включает эхоподавление с `echocancel=yes`. `echocancelwhenbridged=yes` позволит отменить эхо для вызовов с временным разделением каналов (TDM). Хотя мостовые вызовы не должны требовать эхоподавления, это может улучшить качество вызова.

Когда эхоподавление включено, эхоподавитель узнает об эхо на линии, прослушивая его в течение всего времени вызова. Следовательно, эхо может быть услышано в начале вызова и уменьшиться через некоторое время. Чтобы избежать этой ситуации, вы можете использовать метод, называемый *тренировкой эхоподавления*, который отключит линию ненадолго в начале вызова и отправит тон, из которого можно определить количество эхо на линии.

Это позволяет Asterisk быстрее справляться с эхом. Тренировка эхоподавления может быть включена с помощью `echotraining=yes`.

Аппаратное эхоподавление

Самый эффективный способ обработки эхоподавления - не в программном обеспечении. Если вы планируете развернуть качественную систему, потратьте дополнительные деньги и купите карты для системы с аппаратным эхоподавлением. Эти карты немного дороже, но они быстро окупаются за счет снижения нагрузки на процессор, а также снижения нагрузки на вас из-за меньшего количества жалоб пользователей.

16 На самом деле, телефонная трубка в любом телефоне, будь то традиционный или VoIP, является аналоговым соединением.

Asterisk и VoIP

Неудивительно, что Asterisk любит разговаривать по VoIP. Но для этого Asterisk должен знать, какую функцию он должен выполнять: клиентскую, серверную или обе. Одним из самых сложных и часто запутанных понятий в Asterisk является конфигурация входящей и исходящей аутентификации.

Пользователи и пиры и друзья — о, боже!

Соединения, которые идентифицируются на нас, или где мы проходим аутентификацию, определяются в *iax.conf* и *sip.conf* как *users* и *peers*. Соединения, которые делают и то и другое определены как *friends*. При определении того, каким образом происходит аутентификация, всегда важно просматривать направление каналов с точки зрения Asterisk, так как соединения принимаются и создаются сервером Asterisk.

Users (пользователи)

Соединение, определенное как `user` - это любая система/пользователь/конечная точка, которую мы разрешаем подключить к нам. Имейте в виду, что определение `user` не предоставляет метод для вызова этого пользователя; тип `user` используется просто для создания канала для входящих вызовов.[17] Для определения пользователя потребуется определить имя контекста, чтобы указать, где входящий аутентифицированный вызов будет входить в диалплан (в *extensions.conf*).

Peers (пиры, одноранговые узлы)

Подключение определенное как `peer` тип исходящего соединения. Подумайте об этом так: *пользователи* делают звонки нам, в то время как мы делаем звонки нашим *пирам*. Поскольку пиры не вызывают нас, определение `peer` обычно не требует конфигурации имени контекста.

Однако есть одно исключение: если вызовы, исходящие из вашей системы, возвращаются в вашу систему в цикле, входящие вызовы (которые приходят от прокси-сервера SIP, а не агента пользователя) будут сопоставлены в определении `peer`. Контекст `default` должен обрабатывать эти входящие вызовы соответствующим образом, хотя предпочтительно, чтобы контексты определялись для них на одноранговой основе.

Чтобы знать, куда отправить вызов хосту, мы должны знать его местоположение по отношению к Интернету (то есть его IP-адрес). Местоположение пира может быть определено статически или динамически. Динамический пир настроен с `host=dynamic` под заголовком определения пира. Поскольку IP-адрес динамического пира может постоянно меняться, он должен регистрироваться в поле Asterisk, чтобы вызовы могли быть успешно перенаправлены на него. Если удаленный конец является другим полем Asterisk, необходимо использовать оператор `register`, как описано в следующем разделе.

Friends (друзья)

Определение типа как `friend` — это ярлык для определения его как `user`, так и `peer`. Однако соединения, являющиеся как пользователями, так и пирами, не всегда определяются таким образом, поскольку определение каждого направления создания вызова индивидуально (с определением пользователя и пира) позволяет повысить степень детализации и контроля над отдельными соединениями.

На Рисунке B-2 показан поток управления аутентификацией по отношению к Asterisk.

17 В SIP, это не *всегда* так. Если конечная точка является прокси-службой SIP (в отличие от агента пользователя), Asterisk будет аутентифицироваться на основе определения `peer`, сопоставляя IP-адрес и порт в поле `Contact` SIP-заголовка с именем хоста (и портом, если он указан), определенным для пира (если порт не указан, будет использоваться порт, определенный в разделе `[general]`).

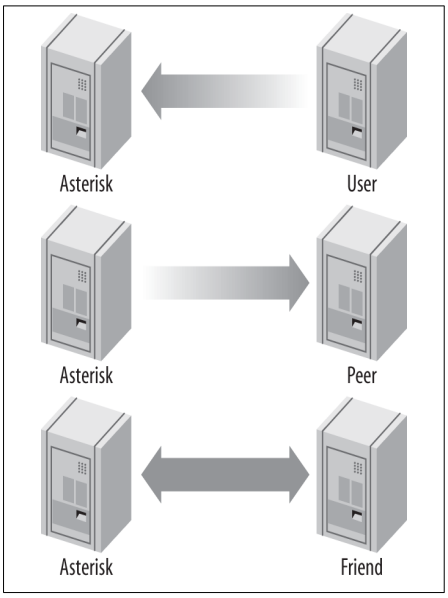


Рисунок B-2. Происхождение вызова в зависимости от отношения пользователей, пиров и друзей к Asterisk

Выражение register

Оператор `register` — это способ сообщить удаленному узлу, где находится блок Asterisk по отношению к интернету. Asterisk использует выражение `register` для проверки подлинности удаленных поставщиков услуг при использовании динамического IP-адреса или когда у поставщика услуг нет вашего IP-адреса в записи. Есть ситуации, когда оператор `register` не требуется, но чтобы продемонстрировать, когда он требуется давайте рассмотрим пример.

Скажем, у вас есть удаленный пир, который предоставляет вам услуги DID. Когда кто-то вызывает номер +1-800-555-1212, вызов идет по физической сети ТфОП к поставщику услуг и в его сервер Asterisk, возможно, через его соединение T1. Затем этот вызов направляется на сервер Asterisk через Интернет.

Ваш поставщик услуг будет иметь определение в его *sip.conf* или *iax.conf* (в зависимости от того, подключаетесь ли вы к протоколу SIP или IAX соответственно) для вашего сервера Asterisk. Если вы получаете вызовы только от этого поставщика, вы определяете его как `user` (если это другая система Asterisk, вы можете быть определены в его системе как `peer`).

Теперь предположим, что ваш блок находится в вашем домашнем интернет-соединении с динамическим IP-адресом. Ваш поставщик услуг имеет статический IP-адрес (или доменное имя), который вы помещаете в файле конфигурации. Поскольку у вас есть динамический адрес, поставщик услуг указывает `host=dynamic` в своем файле конфигурации. Чтобы узнать, куда направить ваш звонок +1-800-555-1212, ваш поставщик услуг должен знать, где вы находитесь по отношению к интернету. Здесь используется инструкция register.

Оператор `register` — это способ аутентификации и информирования вашего `peer` о вашем местонахождении. В разделе `[general]` файла конфигурации поместите оператор, подобный этому:

```
register => username:secret@my_remote_peer
```

Успешную регистрацию можно проверить с помощью команд *iax2 show registry* и *sip show registry* в консоли Asterisk.

Безопасность VoIP

Мы едва коснулись поверхности сложной проблемы безопасности VoIP в этом приложении, поэтому, прежде, чем мы начнем, хотим направить вас в направлении Альянса безопасности VoIP. Этот фантастический ресурс содержит отличный список рассылки, белые книги, инструкции и общий

сборник всех вопросов, связанных с безопасностью VoIP. Подобно тому, как электронной почтой злоупотребляли эгоисты и преступники, так же будет и с голосом. Прекрасные люди в VoIPSA делают все возможное, чтобы мы могли решить эти проблемы сейчас, прежде чем они станут эпидемией. В области книг на эту тему мы рекомендуем самые превосходные *Hacking Exposed VoIP* (Взлом доступных VoIP)[18] Дэвида Эндлера и Марка Колльера (McGraw-Hill Osborne Media, 2007). Если вы отвечаете за развертывание любой системы VoIP, вы должны знать об этом.

Спам через Интернет Телефонию (SPIT)

Мы не хотим думать об этом, но мы знаем, что это придет. Простой факт заключается в том, что в этом мире есть люди, которым не хватает определенных социальных навыков, и это в сочетании с какой-то бессмысленной жадностью означает, что эти люди ничего не думают о наводнении Интернета огромными объемами электронной почты. Эти же типы персонажей будут мало думать о том, чтобы сделать то же самое с голосом. Мы уже знаем, каково это - быть заваленным телемаркетингом; попытайтесь представить, что может случиться, когда эти телемаркетеры поймут, что они могут отправлять голосовой спам почти бесплатно. Регулирование не остановило спам электронной почты, и это, вероятно, не остановит голосовой спам, поэтому мы должны предотвратить его.

Шифрование аудио с помощью Secure RTP

Если вы можете проснифить (от англ. to sniff — нюхать) пакеты, выходящие из системы Asterisk, вы можете извлечь аудио из потоков RTP. Эти данные могут передаваться в автономном режиме в систему обработки речи, которая может прослушивать ключевые слова, такие как “номер кредитной карты” или “PIN-код”, и представлять собранные данные кому-то, кто в этом заинтересован. Поток также можно оценить, чтобы увидеть, есть ли в нем DTMF-тоны, что опасно, поскольку многие службы запрашивают пароли и информацию о кредитной карте для ввода через клавиатуру. В бизнесе стратегическую информацию можно также почерпнуть из записанного аудио.

Использование Secure RTP поможет бороться с этой проблемой путем шифрования потоков RTP. Более подробная информация о SRTP доступна в разделе "Шифрование SIP-вызовов" в Главе 7.

Спуфинг

В традиционной телефонной сети очень трудно успешно выдать себя за другую личность. Ваша деятельность может (и будет) прослеживаться, и власти быстро положат конец веселью. В мире IP гораздо проще оставаться анонимным. Таким образом, не будет трудностью представить, что есть полчища предприимчивых преступников, которые будут только рады сделать звонки в вашу компанию кредитных карт или банк, притворяясь вами. Если не обнаружен надежный механизм для борьбы со спуфингом, мы быстро узнаем, что не можем доверять VoIP-вызовам.

Что можно сделать?

Первое, что нужно иметь в виду при рассмотрении безопасности в системе VoIP, - это то, что VoIP основан на сетевых протоколах и должен оцениваться с этой точки зрения. Это не означает, что традиционную телекоммуникационную безопасность следует игнорировать, но мы должны обратить внимание на основную сеть.

Безопасность основной сети

Одна из самых эффективных вещей, которую можно сделать, - это обеспечить доступ к голосовой сети. Примеры того, как это может быть достигнуто, - использование брандмауэров и виртуальных LANs (VLANs). По умолчанию голосовая сеть должна быть доступна только тем вещам, которые в

18 По некоторым версиям "VoIP - легкая добыча для взломщиков: Секреты и решения в области передачи голоса по IP"

ней нуждаются. Например, если софтфоны не используются, не разрешайте клиентским ПК доступ к голосовой сети.

Разделение голосового и трафика данных. Если нет необходимости иметь голос и данные в одной сети, может иметь смысл в их разделении (это может иметь и другие преимущества, такие как упрощение конфигураций QoS). Это не неслыханно, чтобы построить внутреннюю голосовую сеть на совершенно отдельной локальной сети, используя существующие кабели Cat 3 и заканчивая на недорогих сетевых коммутаторах. Такая конфигурация может быть даже дешевле.

DMZ. Размещение вашей системы VoIP в демилитаризованной зоне (DMZ) может обеспечить дополнительный уровень защиты для вашей локальной сети, в то же время позволяя подключение для соответствующих приложений. Если ваша система VoIP будет скомпрометирована, будет намного сложнее использовать ее для запуска атаки на остальную часть вашей сети, поскольку ей не доверяют. Независимо от того, развертываете ли вы в DMZ или нет, любой ненормальный трафик, выходящий из системы, должен считаться подозрительным.

Укрепление сервера. Укрепление сервера Asterisk имеет решающее значение. Это не только повышение производительности (запуск несущественных процессов может съесть ценные ресурсы ЦП и ОЗУ), но и устранение всего, что не требуется, уменьшит вероятность того, что эксплуатируемая уязвимость в операционной системе может быть использована для получения доступа и запуска атаки на другие части вашей сети.

Запуск Asterisk как не-root является неотъемлемой частью укрепления системы. Дополнительную информацию см. в Главе 3.

Шифрование

Asterisk 1.8 и более поздние версии включают возможность использования SIP TLS для шифрования сигнализации и SRTP для шифрования медиаданных между конечными точками. Более подробную информацию о шифровании SIP-вызовов можно найти в разделе "Шифрование SIP-вызовов" в Главе 7. Asterisk также поддерживает шифрование между конечными точками с помощью IAX2 начиная с версии 1.4). Сведения о включении шифрования по магистралям IAX2 можно найти в разделе "Шифрование IAX" в Главе 7.

Физическая безопасность

Физическая безопасность не должна игнорироваться. Все терминирующее оборудование (например, коммутаторы, маршрутизаторы и сама АТС) должно быть защищено в среде, к которой могут получить доступ только уполномоченные лица. На пользовательском конце (например, рабочем столе) может быть сложнее обеспечить физическую безопасность, но если сеть реагирует только на устройства, с которыми она знакома (например, ограничение DHCP устройствами, чьи MAC-адреса известны), риск несанкционированных вторжений может быть несколько смягчен.

Вывод

За последние пару лет телекоммуникационная индустрия внедрила VoIP, что делает Asterisk довольно успешным. Хотя Asterisk занимается VoIP уже много лет (более десяти), интеграция VoIP и традиционной телефонии в единую мощную платформу сделала Asterisk крупным игроком в телекоммуникационной отрасли.

Подготовка системы для Asterisk

Очень рано я понял, что когда-нибудь
в каком-нибудь “идеальном” будущем там, за горизонтом,
компьютеры будут обрабатывать всю необходимую функциональность
обработки внутри, что делает необходимое внешнее оборудование
для подключения к телекоммуникационным интерфейсам очень недорогим
и, в некоторых случаях, тривиальным.

- Джим Диксон “История телефонии Zapata
и как она относится к АТС Asterisk”

К этому моменту, вы должны быть обеспокоены, чтобы получить собственную систему Asterisk и работать. Однако при развертывании, имеющем решающее значение для миссии, необходимо учитывать условия, в которых будет работать система Asterisk. Не ошибитесь: Asterisk, будучи очень гибким программным обеспечением, будет счастливо и успешно устанавливаться практически на любой платформе Linux, которую вы можете себе представить, а также на нескольких платформах, отличных от Linux.[1] Однако, чтобы вооружить вас пониманием типа операционной среды, в которой Asterisk будет действительно процветать, в этом приложении будут обсуждаться вопросы, которые вам нужно знать, чтобы обеспечить надежную, хорошо проработанную систему.

Потребности Asterisk в ресурсах аналогичны потребностям встроенного приложения в реальном времени. Это связано в значительной степени с ее необходимостью иметь приоритетный доступ к процессору и системным шинам. Поэтому крайне важно, чтобы любые функции в системе, не связанные непосредственно с задачами обработки вызовов Asterisk, выполнялись с низким приоритетом, если вообще выполнялись. В небольших системах и хобби-системах это может быть не такой большой проблемой.

Однако в высокопроизводительных системах недостатки производительности будут проявляться в качестве проблем качества звука для пользователей, часто ощущаемых как эхо, статика и тому подобное. Симптомы будут напоминать те, которые наблюдаются на мобильном телефоне при выходе из диапазона, хотя основные причины будут отличаться. По мере увеличения нагрузки, система будет иметь увеличение затруднения поддержки соединения. Для АТС такая ситуация не что иное, как катастрофа, поэтому тщательное внимание к требованиям производительности является критическим соображением в процессе выбора платформы.

1 Люди успешно скомпилировали и запустили Asterisk на WRAP платах, маршрутизаторах Linksys WRT54G, системах Soekris, Pentium 100s, КПК, Apple Macs, Sun SPARCs, ноутбуках и многом другом. Конечно, если вы хотите поставить такую систему в производство, это совсем другое дело. (На самом деле, AstLinux distribution, исполнителя Kristian Kielhofner, работает очень хорошо на борту оборудование soekris 4801. После того, как вы поняли основы Asterisk, это то, что стоит посмотреть дальше.)

В Таблице C-1 перечислены некоторые основные рекомендации, которые необходимо учитывать при планировании системы. В следующем разделе подробно рассматриваются различные вопросы проектирования и реализации, которые повлияют на его производительность. Имейте в виду, что ни одно руководство не может точно сказать, сколько вызовов может обрабатывать сервер. Невероятно большое количество переменных может повлиять на количество вызовов Asterisk. Единственный способ выяснить, сколько вызовов может обработать сервер, - это проверить его самостоятельно в своей собственной среде.



Размер системы Asterisk фактически определяется не количеством пользователей или аппаратов, а количеством одновременных вызовов, которые она будет поддерживать. Эти цифры весьма консервативны, поэтому не стесняйтесь экспериментировать и посмотреть, что работает для вас.

Таблица C-1. Руководящие принципы системных требований

Цель	Количество каналов	Рекомендуемый минимум
Хобби-система	Не более 5	400 MHz x86, 256 MB ОЗУ
SOHO-система (небольшой оффис/ домашний оффис - меньше 3 линий и 5 аппаратов)	От 5 до 10	1 GHz x86, 512 MB ОЗУ
Небольшая бизнесс-система	До 25	3 GHz x86, 1 GB ОЗУ
Средняя к большой системе	Более 25	Двухядерные процессоры, возможно, также несколько серверов в распределенной архитектуре

При больших установках Asterisk функциональные возможности обычно развертываются на нескольких серверах. Один или несколько центральных блоков будут предназначены для обработки вызовов; они будут дополнены одним или несколькими вспомогательными серверами, обрабатывающими периферийные устройства (такие как система баз данных, система голосовой почты, система конференц-связи, система управления, веб-интерфейс, брандмауэр и т. д.). Как и в большинстве сред Linux, Asterisk хорошо подходит для роста ваших потребностей: небольшая система, которая раньше могла обрабатывать все ваши вызовы и периферийные задачи, может быть распределена между несколькими серверами, когда повышенные требования превышают ее возможности. Гибкость является одной из ключевых причин, почему Asterisk является чрезвычайно экономически эффективным для быстро растущих предприятий; нет эффективного максимального или минимального размера, чтобы учитывать при составлении бюджета первоначальной покупки. Хотя в большинстве телефонных систем возможна некоторая масштабируемость, мы еще не слышали о такой, которая может это делать так же гибко, как Asterisk. Тем не менее, распределенные системы Asterisk не просты в проектировании — это не задача для новичка в Asterisk.



Если вы уверены, что вам нужно настроить распределенную систему Asterisk, вы захотите изучить протокол DUNDi, Asterisk Realtime Architecture (ARA), `func_odbc` и различные другие инструменты базы данных в вашем распоряжении. Это поможет вам абстрагировать данные, необходимые вашей системе, от логики диалплана, которую будут использовать ваши системы Asterisk, создавая общий набор логики дилалпна, который может использоваться в нескольких блоках. Это, в свою очередь, позволит вам масштабировать более просто, добавляя блоки в систему. Однако это выходит далеко за рамки данной книги и будет оставлено в качестве упражнения для читателя. Если вам нужен тизер некоторых инструментов, которые можно использовать для масштабирования, см. Главу 22.

Выбор оборудования сервера

Выбор сервера одновременно прост и сложен: прост, потому что, действительно, любой x86-платформы будет достаточно, но сложен, потому что надежная производительность вашей системы будет зависеть от тщательности, которая вкладывается в разработку платформы. При выборе оборудования необходимо тщательно продумать общую концепцию системы и функциональные возможности, которые необходимо поддерживать. Это поможет вам определить ваши требования к процессору, материнской плате и источнику питания. Если вы просто настраиваете свою первую систему Asterisk с целью обучения, вы можете смело игнорировать информацию в этом разделе.

Однако если вы создаете критически важную систему, пригодную для развертывания, эти вопросы требуют некоторого размышления.

Проблемы производительности

Среди прочих соображений при выборе оборудования для установки Asterisk необходимо иметь в виду такой важный вопрос: насколько мощной должна быть система? На этот вопрос нелегко ответить, поскольку то, как будет использоваться система, будет играть большую роль в потребляемых ею ресурсах. Нет такой вещи, как матрица производительности Asterisk, поэтому вам нужно будет понять, как Asterisk использует систему, чтобы принимать разумные решения о том, какие ресурсы потребуются. Вам нужно будет учитывать несколько факторов, в том числе:

Максимальное количество одновременных подключений, которые будет поддерживать система

Каждое подключение увеличивает нагрузку на систему.

Процент трафика, который потребует интенсивной загрузки процессора DSP от сжатых кодеков (таких как G.729 и GSM)

Работа с цифровой обработкой сигналов (DSP), которую Asterisk выполняет в программном обеспечении, может оказать ошеломляющее влияние на количество одновременных вызовов, которые она будет поддерживать. Система, которая могла бы успешно обрабатывать 50 одновременных вызовов G.711, могла бы быть поставлена на колени запросом на конференцию вместе 10 G.729-сжатых каналов. Мы говорим больше о G.729, GSM, G.711 и многих других кодеках в Приложении B.

Будет ли обеспечиваться конференц-связь и какой уровень конференц-связи ожидается

Будет ли система использоваться интенсивно? Конференц-связь требует, чтобы система перекодировала и микшировала каждый отдельный входящий аудиопоток в несколько исходящих потоков. Микширование нескольких аудиопотоков в режиме реального времени может создать значительную нагрузку на процессор.

Эхоподавление

Эхоподавление может потребоваться при любом вызове, в котором задействован интерфейс телефонной сети общего пользования (PSTN). Поскольку эхоподавление является математической функцией, чем больше ее должна выполнять система, тем выше будет нагрузка на процессор.[2] Некоторые поставщики оборудования телефонии предлагают аппаратное эхоподавление, чтобы снять нагрузку с ЦП хоста. Эхоподавление кратко обсуждается ниже в настоящем приложении и более подробно в Приложении B.

Логика сценариев Dialplan

Всякий раз, когда Asterisk должен передать управление вызовами внешней программе, существует плата за производительность. В диалплан должно быть встроено как можно больше логики. Если используются внешние скрипты, они должны быть разработаны с учетом производительности и эффективности в качестве критических соображений.

Что касается точного влияния этих факторов на производительность, трудно сказать наверняка. Эффект каждого из них известен в общих чертах, но точный калькулятор производительности еще не был успешно определен. Отчасти это связано с тем, что влияние каждого компонента системы зависит от множества переменных, таких как мощность процессора, набор микросхем материнской платы и общее качество, общая нагрузка на систему, оптимизация ядра Linux, сетевой трафик, количество и тип интерфейсов PSTN и трафик PSTN—не говоря уже о любых неастерисковых услугах, которые система выполняет одновременно. Давайте рассмотрим влияние нескольких ключевых факторов:

Кодеки и транскодирование

2 Примерно 30 МГц мощности процессора на канал.

Проще говоря, *codec* (сокращенно от coder/decoder или compression/decompression) - это набор математических правил, которые определяют, как будет оцифрована аналоговая форма волны. Различия между различными кодеками в значительной степени обусловлены уровнями сжатия и качества, которые они предлагают. Вообще говоря, чем больше сжатия требуется, тем больше работы DSP должен сделать для кодирования или декодирования сигнала. Несжатые кодеки, таким образом, несут гораздо меньше нагрузки на CPU (но требуют больше пропускной способности сети). Выбор кодека должен обеспечивать баланс между пропускной способностью и использованием процессора. Дополнительные сведения о кодеках см. в Приложении B.

Центральный блок обработки (и блок с плавающей запятой)

Центральный процессор состоит из нескольких компонентов, одним из которых является блок с плавающей запятой (FPU). Скорость процессора в сочетании с эффективностью его FPU будет играть значительную роль в количестве параллельных соединений, которые система может эффективно поддерживать. В следующем разделе (“Выбор процессора”) предлагаем некоторые общие рекомендации по выбору процессора, который будет отвечать потребностям вашей системы.

Другие процессы, выполняющиеся одновременно в системе

Будучи Unix-подобным, Linux предназначен для многозадачности нескольких различных процессов. Проблема возникает, когда один из этих процессов (например, Asterisk) требует от системы очень высокого уровня реагирования. По умолчанию Linux распределяет ресурсы справедливо для каждого приложения, которое их запрашивает. Если вы установите систему с множеством различных серверных приложений, эти приложения будут иметь право на равное использование ЦП. Поскольку Asterisk требует частого высокоприоритетного доступа к процессору, он не ладит с другими приложениями, и если Asterisk должен сосуществовать с другими приложениями, система может потребовать специальной оптимизации. Это в первую очередь предполагает назначение приоритетов различным приложениям в системе и, во время установки, тщательное внимание к тому, какие приложения устанавливаются в качестве служб.

Оптимизация ядра

Ядро, оптимизированное для работы одного конкретного приложения — это то, что по умолчанию предлагают очень немногие дистрибутивы Linux, и поэтому оно требует некоторого размышления. Как минимум-какой бы дистрибутив вы ни выбрали—вы должны загрузить и скомпилировать на своей платформе новую копию ядра Linux (доступную из архивов ядра Linux). Вы также можете получить исправления, которые улучшат производительность, но они считаются взломами официально поддерживаемых ядер.

Задержка IRQ

Задержка запроса прерывания (IRQ) - это в основном задержка между моментом, когда периферийная карта (например, телефонная интерфейсная карта) просит процессор остановить то, что он делает, и моментом, когда процессор фактически отвечает и готов справиться с задачей. Периферийные устройства Asterisk (особенно карты DAHDI) исторически были нетерпимы к задержке IRQ, хотя в DAHDI были значительные улучшения, чтобы помочь с этими проблемами. Это не связано с какой-либо проблемой с картами, а скорее является частью природы того, как должен работать программный движок TDM. Если мы буферизуем данные TDM и отправляем их на шину в виде большего пакета, это может быть более эффективным с точки зрения системы, но это создаст задержку между временем получения звука на карте и доставкой в ЦП. Это делает обработку данных TDM в реальном времени практически невозможной. В процессе разработки DAHDI было решено, что отправка данных каждые 1 мс создаст лучший компромисс, но побочным эффектом этого является то, что любая карта в системе, которая использует интерфейс DAHDI, попросит систему обрабатывать прерывание каждую миллисекунду. Раньше это было решающим фактором на старых материнских платах, но в значительной степени перестало быть причиной для беспокойства.



У Linux исторически были проблемы с его способностью быстро обслуживать IRQ; эта проблема вызвала достаточно проблем для разработчиков аудио, что было создано несколько патчей для устранения этого недостатка. До сих пор были некоторые споры о том, как включить эти исправления в ядро Linux.

Версия ядра

Asterisk официально поддерживается в Linux версии 2.6. Почти вся Asterisk сама по себе не очень заботится о версии ядра, но DAHDI требует 2.6.

Дистрибутив Linux

Дистрибутивы Linux многочисленны и разнообразны. Asterisk должен работать на всех из них. Выберите тот, который вам наиболее удобен.

Выбор процессора

Поскольку требования к производительности Asterisk обычно включают большое количество математических вычислений, важно выбрать процессор с мощным FPU. Обработка сигналов, которую выполняет Asterisk, может потребовать от ЦП ошеломляющего количества сложных математических вычислений. Эффективность, с которой выполняются эти задачи, будет определяться мощностью FPU внутри процессора.

На самом деле, упоминание лучшего процессора для Asterisk в этой книге противоречило бы закону Мура. Даже в период между написанием и публикацией этой книги скорость процессора будет быстро увеличиваться, как и поддержка Asterisk различных архитектур. Очевидно, что это хорошо, но это также делает предоставление советов по этой теме неблагодарной задачей. Естественно, чем мощнее FPU, тем больше параллельных задач DSP Asterisk сможет обрабатывать, и это самое главное. При выборе процессора сырая тактовая частота является только частью уравнения. Насколько хорошо он обрабатывает операции с плавающей запятой, будет ключевым отличием, так как операции DSP в Asterisk будут предъявлять большой спрос на этот процесс.

Процессоры Intel и AMD имеют мощные FPU. Можно ожидать, что чипы текущего поколения от любого из этих производителей будут работать хорошо.[3]

Очевидный вывод заключается в том, что вы должны получить самый мощный процессор, который позволит ваш бюджет. Однако не спешите покупать самый дорогой процессор. Вам нужно будет иметь в виду требования вашей системы; в конце концов, Формула 1 Ferrari плохо подходит для суровых условий движения в час пик. Более медленные процессоры часто работают лучше, так что вы можете построить более слабую, безвентиляторную систему Asterisk для небольшого офиса, которая может хорошо работать, например, в пыльной среде.

Чтобы попытаться предоставить вам точку отсчета, из которой вы можете рассматривать свое решение платформы, мы решили определить три размера систем Asterisk: малый, средний и большой.

Малые системы

Малые системы (до 10 телефонов) не защищены от требований к производительности Asterisk, но типичная нагрузка, размещенная на меньшей системе, как правило, подпадает под возможности современного процессора.

Если вы строите небольшую систему из старых компонентов, которые у вас есть, знайте, что результирующая система не может работать на том же уровне, что и более мощная машина, и производительность начнет ухудшаться под гораздо меньшей нагрузкой. Хобби-системы могут успешно работать на очень маломощном оборудовании, хотя это ни в коем случае не рекомендуется для тех, кто не является гением в настройке производительности Linux.[4]

3 Если вы хотите быть полностью в курсе того, какие процессоры лидируют в гонке производительности, перейдите к Tom's Hardware или AnandTech, где вы найдете множество информации о текущих и устаревших процессорах, материнских платах и чипсетах.

4 Грег Бонлейн однажды скомпилировал и запустил Asterisk в системе Pentium 133 МГц, но это было в целях эксперимента. Проблемы с производительностью гораздо более вероятны в таких условиях, и для правильной

Если вы настраиваете систему Asterisk для учебных целей, вы сможете построить полнофункциональную платформу, используя относительно маломощный процессор. Авторы этой книги запускают несколько лабораторных систем Asterisk с процессорами Celeron от 433 МГц до 700 МГц, но рабочая нагрузка этих систем минимальна (не более двух одновременных вызовов).

AstLinux и Asterisk на OpenWRT

Если вам действительно удобно работать с Linux на встроенных платформах, вы захотите присоединиться к списку рассылки AstLinux и запустить создание Кристиана Кильхофнера AstLinux или получить Linksys WRT54GL и установить версию Asterisk Брайана Капуча для этой платформы.

Эти проекты лишают Asterisk его основных компонентов и позволяют развертывать невероятно мощные приложения УАТС на очень недорогом оборудовании.

Хотя оба проекта требуют изрядного количества знаний и усилий с вашей стороны, они также разделяют огромный фактор крутизны, чрезвычайно популярны и имеют отличное качество.

Средние системы

Системы среднего размера (от 10 до 50 телефонов) - это то, где соображения производительности будут наиболее сложными для решения. Как правило, эти системы будут развернуты только на одном или двух серверах, и поэтому каждая машина должна будет выполнять более одной конкретной задачи. По мере увеличения нагрузок пределы платформы будут становиться все более напряженными. Пользователи могут начать воспринимать проблемы качества, не понимая, что система не неисправна, а просто превышает свои возможности. Эти проблемы будут становиться все хуже и хуже, поскольку все больше и больше нагрузки ложится на систему, а пользовательский опыт соответственно ухудшается. Важно, чтобы проблемы производительности были выявлены и устранены до того, как пользователи заметят их.

Мониторинг производительности этих систем и оперативное реагирование на любые развивающиеся тенденции является ключом к обеспечению качественной телефонной платформы.

Большие системы

Большие системы (более 120 каналов) могут быть распределены между несколькими системами и местоположениями, а проблемы производительности могут управляться путем добавления машин. Таким образом были созданы очень большие системы Asterisk.

Построение большой системы требует высокого уровня знаний в различных дисциплинах. Мы не будем подробно обсуждать это в этой книге, кроме того, что проблемы, с которыми вы столкнетесь, будут похожи на те, с которыми вы столкнетесь при любом развертывании нескольких серверов, обрабатывающих одну распределенную задачу.

Выбор материнской платы

Чтобы избавиться от предвкушения, мы также не можем рекомендовать конкретные материнские платы в этой книге. С появлением новых материнских плат на еженедельной основе любые рекомендации, которые мы могли бы сделать, стали бы спорными из-за устаревания до того, как опубликованная копия попала бы на полки. Не только это, но материнские платы похожи на автомобили: хотя все они очень похожи в принципе, разница в деталях. И поскольку Asterisk - это приложение производительности, детали имеют значение.
Поэтому мы дадим вам некоторое представление о типах материнских плат, которые могут хорошо работать с Asterisk, и о функциях, которые характеризуют хорошую материнскую плату. Важно, что

настройки такой системы требуется экспертное знание Linux. Мы не рекомендуем запускать Asterisk на чем-либо меньшем, чем 500 МГц (для производственной системы 2 ГГц может быть разумным минимумом). Тем не менее, мы считаем, что тот факт, что Asterisk настолько гибок, примечательным.

она должна иметь как стабильность, так и высокую производительность. Вот некоторые рекомендации, которым стоит следовать:

- Различные системные шины должны обеспечивать минимально возможную задержку. Если вы планируете PSTN-соединение с использованием аналоговых или PRI-интерфейсов (обсуждается ниже в этом приложении), наличие карт DAHDI в системе будет генерировать 1000 запросов прерываний в секунду. Наличие устройств на шине, которые мешают этому процессу, приведет к ухудшению качества вызова. Чипсеты от Intel (для процессоров Intel) и nVidia nForce (для процессоров AMD), похоже, набирают лучшие оценки в этой области. Просмотрите конкретный набор микросхем любой материнской платы, которую вы оцениваете, чтобы убедиться, что у него нет известных проблем с задержкой IRQ.
- Если вы используете карты DAHDI в своей системе, вы захотите убедиться, что ваш BIOS позволяет вам максимально контролировать назначение IRQ. Как правило, высококачественные материнские платы обеспечивают гораздо большую гибкость в отношении настройки BIOS; Платы с высокой стоимостью, как правило, предлагают очень мало контроля. Однако это может быть спорным вопросом, поскольку материнские платы с поддержкой APIC передают управление IRQ операционной системе.
- Материнские платы серверного класса обычно реализуют другой стандарт PCI, чем материнские платы рабочего места. Хотя есть много различий, наиболее очевидно и хорошо известно, что две версии имеют разные напряжения. В зависимости от того, какие карты вы покупаете, вам нужно будет знать, если вам нужны слоты 3.3 V или 5V PCI.[5] На Рисунке C-1 показаны визуальные различия между слотами 3.3 V и 5V. Большинство материнских плат сервера будут иметь оба типа, но рабочие станции, как правило, имеют только версию 5V.



Есть некоторые свидетельства того, что соединение двух совершенно отдельных однопроцессорных систем может дать гораздо больше преимуществ, чем простое использование двух процессоров на одной машине. Вы не только удваиваете мощность процессора, но и достигаете гораздо лучшего уровня избыточности по цене, аналогичной стоимости двухпроцессорной машины с одним шасси. Однако имейте в виду, что решение Asterisk с двумя серверами будет более сложным для разработки, чем решение с одной машиной.

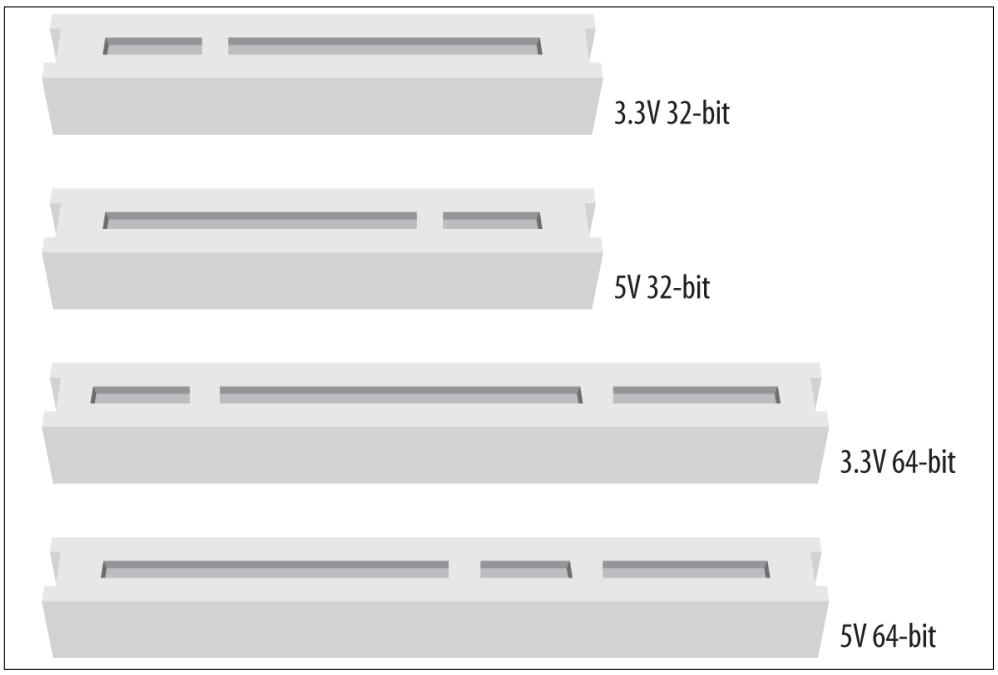


Рисунок C-1. Визуальная идентификация слотов PCI

5 С появлением PCI-X и PCI-Express становится все сложнее и сложнее выбрать материнскую плату с правильным типом слотов. Убедитесь, что выбранная вами материнская плата имеет правильный тип и количество слотов для вашего оборудования. Имейте в виду, что большинство компаний, производящих аппаратные карты для Asterisk, предлагают версии PCI и PCI-Express, но вы все равно должны убедиться, что они имеют смысл в любой комбинации материнской платы и шины, которую вы выбираете.

- Рассмотрите возможность использования нескольких процессоров или процессоров с несколькими ядрами. Это позволит улучшить способность системы справляться с несколькими задачами. Для Asterisk это будет иметь особое преимущество в области операций с плавающей запятой.
- Если вам нужен модем, установите внешний блок, который подключается к последовательному порту. Если у вас должен быть внутренний модем, вам нужно будет убедиться, что это не так называемый “Win-модем”-это должен быть полностью самодостаточный блок (обратите внимание, что их очень сложно, если не невозможно, найти).
- Учтите, что при наличии встроенной сети в случае сбоя сетевого компонента необходимо будет заменить всю материнскую плату. С другой стороны, при установке карты периферийного сетевого интерфейса (NIC) может увеличиться вероятность сбоя из-за дополнительных механических соединений. Также может быть полезно иметь отдельные сетевые карты, обслуживающие устройства и пользователей (внутренняя сеть) и поставщиков VoIP и внешние местоположения (внешняя сеть). NICs дешевы; мы предлагаем всегда иметь по крайней мере два.
- Стабильность и качество вашей системы Asterisk будет зависеть от компонентов, которые вы выберете для ее архитектуры. Asterisk - зверь, и он ожидает, что его будут кормить лучше всего. Как и почти все, высокая стоимость не всегда синонимична качеству, но вы захотите стать знатоком компьютерных компонентов.

Сказав все это, нам нужно вернуться к исходной точке: Asterisk может и с радостью установится практически на любую систему, которая будет работать под управлением Linux. Лабораторные системы, используемые для написания этой книги, например, включали в себя все: от Linksys WRT до Локомотива dual-Xeon.[6] Мы не испытывали никаких проблем с производительностью или стабильностью, при работе с менее чем пятью параллельными телефонными соединениями. Для целей обучения не бойтесь устанавливать Asterisk на любую систему, которую вы можете раздобыть. Однако, когда вы будете готовы запустить свою систему в продакшен, вам нужно будет понять последствия выбора, который вы делаете в отношении своего оборудования.

Требования к электропитанию

Одним из часто упускаемых компонентов в ПК является источник питания (и поставщик питания). Для телекоммуникационной системы[7] эти компоненты могут играть важную роль в качестве пользовательского опыта.

Питание компьютера

Электропитание, которое вы выбираете для вашей системы сыграет жизненно важную роль в стабильности всей платформы. Asterisk не является особенно энергоемким приложением, но все, что связано с мультимедиа (будь то телефония, профессиональное аудио, видео или тому подобное), обычно чувствительно к *качеству* питания.

Этот часто забытый компонент может превратить систему высшего класса в плохого исполнителя. Точно так же первоклассный источник питания может позволить дешевому ПК работать как чемпион.

Мощность, подаваемая в систему, должна обеспечивать не только энергию, необходимую системе для выполнения ее задач, но и стабильные, чистые сигнальные линии для всех напряжений, которые система ожидает от нее.

Потратьте деньги и получите первоклассный источник питания (геймеры довольно увлечены такими вещами, поэтому есть много вариантов).

6 Ладно, *на самом деле* это был не Локомотив, но звучало похоже. Кто-нибудь знает, где получить тихие вентиляторы процессора для Xeon? В лаборатории становится слишком шумно.

7 Или любая система, которая должна обрабатывать аудио.

Резервный источник питания

В среде операторского уровня или высокой доступности обычно развертываются серверы, использующие резервный источник питания. По сути, он включает в себя два полностью независимых источника питания, каждый из которых способен удовлетворять требованиям к питанию системы.

Если это важно для вас, имейте в виду, что лучшие практики предполагают, что для правильного резервирования эти источники питания должны быть подключены к полностью независимым источникам бесперебойного питания (UPSs), которые, в свою очередь, питаются полностью отдельными электрическими цепями. В действительно критически важных средах (таких как больницы) даже основные электрические каналы в здании являются резервными, а дизельные генераторы находятся на месте для выработки электроэнергии во время длительных сбоев питания.

Окружающая среда

Окружающая среда системы состоит из всех тех факторов, которые на самом деле не являются частью самого сервера, но тем не менее играют важнейшую роль в надежности и качестве, которые следует ожидать от системы. Электроснабжение, комнатная температура и влажность, источники помех и безопасность - все это факторы, которые следует учитывать.

Питание отвечающее стандартам и источники бесперебойного питания

При выборе источников питания для вашей системы следует учитывать не только количество энергии, которую будет использовать система, но и способ, которым эта мощность подается.

Питание это не просто напряжение, поступающее от розетки в стене, и вы никогда не должны просто подключать производственную систему к любому электрическому источнику под рукой.[8] Уделение некоторого внимания поставке энергии в вашу систему может гарантировать, что вы обеспечите гораздо более стабильную энергетическую среду, что приведет к гораздо более стабильной системе.

Одним из преимуществ чистой энергии является снижение температуры, что означает меньшую нагрузку на компоненты, что приводит к увеличению продолжительности жизни.

Правильно заземленное, соответствующее стандартам, питание подаваемое на высококачественный блок питания, обеспечит чистое *логическое заземление* (aka 0 Вольт)[9] для системы и сведет электрический шум на материнской плате к минимуму. Это стандартная передовая практика для оборудования такого типа, которым не следует пренебрегать. Относительно простой способ достичь этого - использование источника бесперебойного питания (ИБП) с поддержанием требуемого качества электроэнергии.[10]

ИБП с поддержанием требуемого качества электроэнергии

ИБП хорошо известен своей ролью в качестве резервного аккумулятора, но преимущества поддержания требуемого качества электроэнергии, которые также обеспечивают высококачественные ИБП, менее понятны.

Подготовка электроэнергии может обеспечить ценный уровень защиты от электрической окружающей среды путем регенерирации чистой электроэнергии через изолирующий

8 Хорошо, слушайте, вы можете подключить его, где хотите, и он, вероятно, будет работать, но если у вашей системы есть странные проблемы со стабильностью, пожалуйста, прочтите этот раздел еще раз.

9 В электронных устройствах двоичный ноль (0) обычно связан с сигналом 0 Вольт, в то время как двоичная (1) может быть представлена многими различными напряжениями (обычно от 2,5 до 5 вольт). Ссылка на заземление, которую система будет рассматривать как 0 Вольт, часто называется *логическим заземлением*. Плохо заземленная система может иметь электрический потенциал на логическом заземлении до такой степени, что электроника ошибочно принимает двоичный ноль за двоичную 1. Это может нанести ущерб способности системы обрабатывать инструкции.

10 Это распространенное заблуждение, что все UPSs обеспечивают поддержание требуемого качества электронергии. Это совсем не так.

трансформатор. Качественное поддержание электроэнергии в вашем ИБП устранит большинство электрических шумов от подачи питания и поможет обеспечить стабильную подачу питания в вашу систему.

К сожалению, не все блоки ИБП сделаны таковыми; многие из менее дорогих блоков не обеспечивают чистой электроэнергии. Что еще хуже, производители этих устройств часто обещают все виды защиты от скачков, всплесков, перенапряжений и переходных процессов. Хотя такие устройства могут защитить вашу систему от поджаривания в грозу, они не будут очищать электроэнергию, подаваемую в вашу систему, и, таким образом, не будут способствовать стабильности.

Убедитесь, что ваш ИБП оснащен *поддержанием требуемого качества электроэнергии*. Если именно об этом не говорится, то это не так.

Заземление

Напряжение определяется как разность электрических потенциалов между двумя точками. При рассмотрении *земли* (которая в основном представляет собой не более чем электрический путь к Земле) общее предположение заключается в том, что она представляет 0 Вольт. Но если мы не определяем эти 0В *относительно* к чего-то, то это может оказаться совсем не так. Если вы измерите напряжение между двумя точками заземления, то можете обнаружить, что между ними существует потенциал. Этот потенциал между точками заземления может быть достаточно значительным, чтобы вызвать логические ошибки или даже повреждение в системе, где присутствует более одного пути к Земле.



Один из авторов вспоминает, как однажды сжёг звуковую карту, которую пытался подключить к стереосистеме друга. Несмотря на то, что и компьютер, и стереосистема находились в одной комнате, между заземляющими проводами двух электрических розеток, к которым они были подключены, было обнаружено более 6 Вольт разницы! Провод между стерео и ПК (через звуковую карту) обеспечивал путь, по которому поступало напряжение, таким образом, поджаривая звуковую карту, которая не была предназначена для обработки такого большого тока на своих сигнальных выводах. Подключение ПК и стерео к одной и той же розетке исправило проблему.

При рассмотрении электрических правил целью заземления является прежде всего безопасность человека. В компьютере земля используется в качестве логической ссылки 0В. Электрическая система, которая обеспечивает надлежащую безопасность, не всегда обеспечивает надлежащую логическую ссылку — на самом деле цели безопасности и качества электроэнергии иногда расходятся. Естественно, когда выбор должен быть сделан, безопасность должна иметь приоритет.



Поскольку разница между двоичными нулем и единицей представлена в компьютерах разницами напряжения иногда менее 3В, вполне возможно, что нестабильные условия питания, вызванные плохим заземлением или электрическим шумом, вызывают все виды прерывистых системных проблем. Некоторые сторонники питания и заземления считают, что более 80 процентов необъяснимых компьютерных сбоев можно проследить от качества питания. Большинство из нас обвиняют Microsoft.

Современные импульсные источники питания несколько изолированы от проблем с качеством питания, но любая высокопроизводительная система всегда должна пользоваться хорошо продуманной энергетической средой. В мейнфреймах, проприетарных АТС и других дорогостоящих вычислительных платформах заземление системы никогда не оставляется на волю случая. Электроника и корпуса этих систем всегда обеспечены специальным заземлением, которое не зависит от защитных заземлений, поставляемых с электропитанием.

Независимо от того, сколько вы готовы вложить в заземление, при указании электропитания на любую АТС, убедитесь, что электрическая цепь полностью предназначена для вашей системы (как описано в следующем разделе) и предусмотрен изолированный заземляющий проводник. Это может

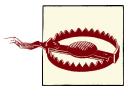
быть дорого, но это будет в значительной степени способствовать качественной среде питания для вашей системы.[11]

Важно также, чтобы все периферийные устройства, подключаемые к вашей системе, были подключены к одной и той же электрической розетке (или, более конкретно, к одному и тому же заземлению). Это сократит возникновение контуров заземления, которые могут вызвать что-либо от жужжания и гудения до поврежденного или разрушенного оборудования.

Электрическая цепь

Если вы когда-нибудь видели, как гаснет свет, когда включается электроприбор, вы видели, как энергозатратное устройство влияет на электрическую цепь. Если бы вы посмотрели на эффекты множества таких устройств, каждое из которых по-своему потребляет мощность, вы бы увидели, как гармонически идеальная синусоидальная волна 50 или 60 Гц, которую вы можете представить, под действием этой силы, превращается во что угодно, но не идеальную синусоиду. Гармонический шум весьма общий на электрических цепях, и он может сеять хаос на чувствительной радиотехнической аппаратуре. Для АТС эти проблемы могут проявляться в виде проблем со звуком, логических ошибок и нестабильности системы.

В идеале никогда не следует устанавливать сервер в электрическую цепь, которая используется совместно с другими устройствами. В цепи должна быть только одна розетка, и вы должны подключить к ней только свою телефонную систему (и связанные периферийные устройства). Провода (включая заземление) должны быть подключены непосредственно к электрическому щиту. Заземляющий проводник должен быть отделен и изолирован. Существует слишком много историй о фотокопировальных машинах, кондиционерах и пылесосах, сеющих хаос с чувствительной электроникой, чтобы игнорировать это эмпирическое правило.



Электрические правила в вашей области всегда должны иметь приоритет над любыми идеями, представленными здесь. Если вы сомневаетесь, проконсультируйтесь с экспертом по качеству электроэнергии в этой области о том, как обеспечить соблюдение электрических правил. Помните, что электрические правила учитывают тот факт, что безопасность человека намного важнее безопасности оборудования.

Комната оборудования

Условия окружающей среды могут привести к разрушению систем, однако довольно часто критические системы развертываются практически без внимания к этим вопросам. Когда система установлена, все работает хорошо, но всего через полгода компоненты начинают отказывать. Поговорите с любым, кто имеет опыт обслуживания серверов и систем, и станет очевидным, что внимание к факторам окружающей среды может играть значительную роль в стабильности и надежности систем.

Влажность

Проще говоря, влажность — это вода в воздухе. Вода является катастрофой для электроники по двум основным причинам: 1) Вода является катализатором коррозии, и 2) вода является достаточно проводящей, что может вызвать короткое замыкание. Не устанавливайте электронное оборудование в зонах повышенной влажности без предоставления средств для удаления влаги.

Температура

Тепло — враг электроники. Чем прохладней вы содержите свою систему, тем надежнее она будет работать, и тем дольше она будет жить. Если вы не можете обеспечить должным образом

11 В хобби-системе это, вероятно, слишком много, чтобы просить, но если вы планируете использовать Asterisk для чего-то важного, по крайней мере, не забудьте дать ему шанс на борьбу; не ставьте ничего, как кондиционеры, фотокопировальные аппараты, лазерные принтеры или двигатели в одну цепь. Напряжение, которое такие потребители создают на вашем источнике питания, сократит его продолжительность жизни.

охлаждаемое помещение для вашей системы, как минимум убедитесь, что оно расположено в месте, обеспечивающем стабильную подачу чистого, прохладного воздуха. Также, держите температуру устойчивой. Изменение температуры может привести к конденсату и другим повреждающим изменениям.

Пыль

Старая пословица в компьютерной индустрии гласит, что пыльным кроликам внутри компьютера повезло. Давайте рассмотрим некоторые из реалий пыльных кроликов:

- Значительное нарастание пыли может ограничить воздушный поток внутри системы, приводя к увеличению температуры.
- Пыль может содержать металлические частицы, которые в достаточном количестве могут способствовать ухудшению сигнала или замыканию на монтажных платах.

Поместите критические серверы в фильтрованную среду и регулярно чистите пыльных кроликов.

Безопасность

Безопасность сервера, естественно, включает в себя защиту от сетевых вторжений, но окружающая среда также играет роль в безопасности системы. Телефонное оборудование всегда должно быть заперто, и только лица, которым необходимо получить доступ к оборудованию, должны быть допущены к нему.

Телефонное оборудование

Если вы собираетесь подключить Asterisk к любому традиционному телекоммуникационному оборудованию, вам понадобится правильное оборудование. Требуемое оборудование будет определяться тем, чего вы хотите достичь.

Подключение к ТфОП

Asterisk позволяет беспрепятственно связывать телекоммуникационные сети с коммутацией каналов[12] с сетями с пакетной коммутацией данных[13].

Благодаря открытой архитектуре Asterisk (и открытому исходному коду) в конечном счете можно подключить любое совместимое со стандартами оборудование интерфейса. Выбор интерфейсных плат телефонии с открытым исходным кодом в настоящее время ограничен, но по мере роста интереса к Asterisk это быстро изменится.[14] На данный момент одним из наиболее популярных и экономически эффективных способов подключения к ТфОП является использование интерфейсных карт, которые появились в результате работы проекта телефонии Zapata, который превратился в DAHDI.

Аналоговые интерфейсные карты

Если вам не нужно много каналов (или у вас есть много денег, чтобы тратить каждый месяц на телекоммуникационные средства), есть вероятность, что ваш интерфейс ТфОП будет состоять из одной или нескольких аналоговых линий, каждая из которых потребует порт FXO.

Digium, компания, которая спонсирует разработку Asterisk, производит аналоговые интерфейсные карты для Asterisk. Ознакомьтесь с его веб-сайтом для получения подробной информации о его обширной линейке аналоговых карт, включая почтенный TDM400P, новейший TDM800P и

12 Часто называемые TDM-сетями, из-за мультиплексирования с временным разделением, используемого для передачи трафика через PSTN.

13 Популярно называемые VoIP-сетями, хотя Voice over IP - не единственный метод передачи голоса по пакетным сетям (Voice over Frame Relay был очень популярен в конце 1990-х годов).

14 Эволюция недорогих аппаратных средств телефонии лишь немного отстает от революции программного обеспечения телефонии. Новые компании появляются еженедельно, каждая из которых выводит на рынок новые и недорогие устройства на основе стандартов.

высокоплотный TDM2400P. Например, TDM800P - это восьмипортовая базовая карта, которая позволяет вставлять до двух дочерних карт, каждая из которых предоставляет либо четыре порта FXO, либо четыре порта FXS.[15] TDM800P можно приобрести с предустановленными этими модулями, а также добавить аппаратный эхоподавитель.

Другие компании, производящие аналоговые карты, совместимые с Asterisk, включают:

- Rhino
- Sangoma
- Voicetronix
- Pika Techologies

Цифровые интерфейсные карты

Если вам требуется более 10 линий или цифровое подключение, скорее всего, вы будете искать на рынке для карты T1 или E1[16]. Однако, имейте в виду, что ежемесячные платежи за цифровую ТфОП линию сильно различаются. В некоторых местах только пять линий могут оправдать цифровую линию; в иных случаях технология никогда не может быть оправдана стоимостью. Чем больше конкуренция в вашей области, тем больше у вас шансов найти хорошее предложение. Не забудьте пройтись по магазинам.

Проект телефонии Zapata первоначально произвел карту T1 - Tormenta, которая является предком большинства совместимых с Asterisk карт T1. Оригинальные карты Tormenta теперь считаются устаревшими, но они все еще работают с Asterisk.

Digium делает несколько различных карт интерфейса цифровой линии. Функции карт одинаковы; основные различия заключаются в том, предоставляют ли они интерфейсы T1 или E1 и сколько диапазонов обеспечивает каждая карта. Digium выпускает карты DAHDI для Linux дольше, чем кто-либо другой; он был глубоко вовлечен в разработку DAHDI (ранее Zaptel) на Linux и был движущей силой развития DAHDI на протяжении многих лет.

Sangoma, которая уже много лет производит WAN-карты с открытым исходным кодом, добавила поддержку Asterisk для своих карт T1/E1 несколько лет назад.[17] Rhino уже некоторое время имеет оборудование T1 для Asterisk, и есть много других компаний, которые предлагают цифровые интерфейсные карты для Asterisk.

Банки каналов

Банк каналов в общих чертах определяется как устройство, которое позволяет демультиплексировать цифровую схему в несколько аналоговых схем (и наоборот). Более конкретно, банк каналов позволяет подключать аналоговые телефоны и линии в систему через линию T1. Рисунок C2 показывает, как банк каналов вписывается в типичную телефонную систему офиса.

15 FXS и FXO относятся к противоположным концам аналоговой линии. Какой из них вам нужен, будет определяться тем, к чему вы хотите подключиться. В Приложении A они рассматриваются более подробно.

16 T1 и E1-схемы цифровой телефонии. Мы обсудили их в Приложении A.

17 Следует отметить, что карта Sangoma Frame Relay сыграла определенную роль в первоначальной разработке Asterisk; Sangoma имеет долгую историю поддержки WAN-интерфейсов с открытым исходным кодом с Linux.

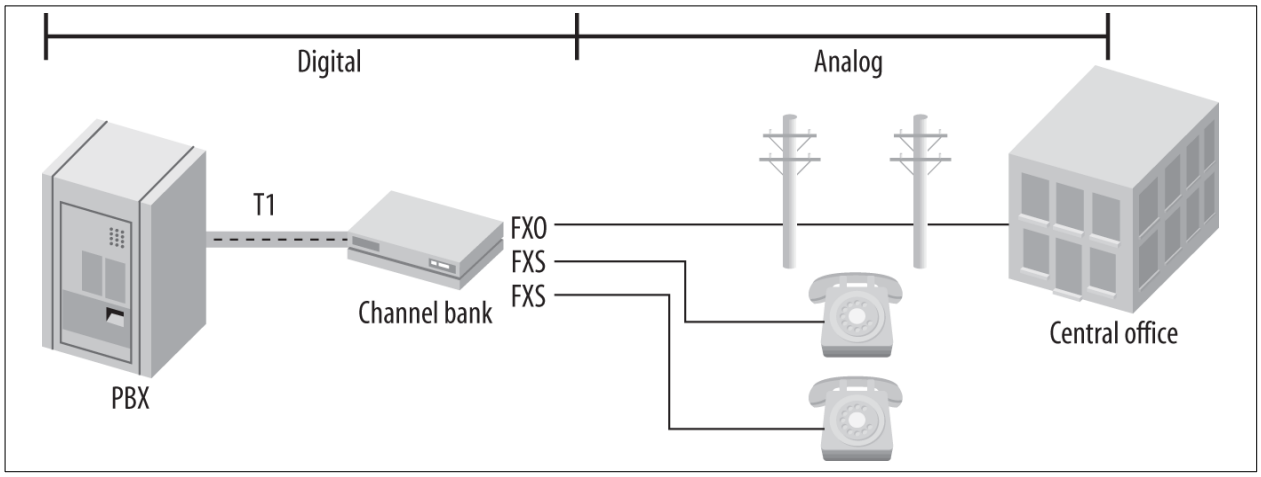


Рисунок C-2. Один из способов подключения банка каналов

Хотя они могут стоить дорого, многие люди очень сильно ощущают, что единственный правильный способ интегрировать аналоговые схемы и устройства в Asterisk - это через банк каналов. Правда это или нет, зависит от многих факторов, но если у вас есть бюджет, они могут быть очень полезны.[18] Вы часто можете купить использованные банки каналов на eBay. Ищите устройства от Adtran и Carrier Access Corp. (Rhino делает отличные банки каналов, и они очень конкурентоспособны по цене, но их может быть трудно найти.) Не забывайте,что для подключения банка каналов к Asterisk вам понадобится карта T1.

Другие типы интерфейсов ТфОП

Существует много шлюзов VoIP, которые можно настроить для обеспечения доступа к линиям ТфОП. Вообще говоря, они будут наиболее полезны в малой системе (одна или две линии). Они также могут быть очень сложными в настройке, так как понимание взаимодействия между различными сетями и устройствами требует глубокого понимания основ телефонии и VoIP. По этой причине мы не будем подробно обсуждать эти устройства в этой книге. Однако их стоит изучить; популярные продукты производятся Sipura, Grandstream, Digium и многими другими компаниями.

Другой способ подключения к ТфОП - использование схем ISDN интерфейса базовой скорости (BRI). BRI цифровой стандарт телекоммуникаций который определяет двухканальную цепь которая может передавать до 144 Кбит/с трафика.[19] Из-за разнообразия способов внедрения этой технологии и отсутствия испытательного оборудования мы не будем подробно обсуждать BRI в этой книге.

Подключение исключительно к сети с коммутацией пакетов

Если подключение к ТфОП не требуется, Asterisk не требует другого оборудования, кроме сервера с NIC. Однако вам все равно может потребоваться установить модули ядра DAHDI, так как DAHDI требуется для использования приложения `MeetMe()` для конференц-связи.

Эхоподавление

Одной из проблем, которые могут возникнуть при использовании аналоговых интерфейсов в системе VoIP, является эхо. *Эхо* - это просто голос, что вы произносите, отражаясь возвращается к вам через короткое время. Эхо вызвано дальним концом, но вы - тот, кто его слышит. Малоизвестным фактом является то, что эхо было бы огромной проблемой в ТфОП, если бы не тот факт, что поставщики услуг используют сложные (и дорогие) стратегии для ее устранения. Мы предлагаем вам рассмотреть возможность добавления оборудования эхоподавления к любой карте, которую вы покупаете для использования в качестве интерфейса ТфОП. Хотя Asterisk может выполнять некоторую работу с эхо

18 Мы используем банки каналов для имитации телефонной станции. Один 24-портовый банк каналов от системы Asterisk может обеспечить до 24 аналоговых линий - идеально подходит для класса или лаборатории.

19 BRI очень редко используется в Северной Америке, но очень популярен в Европе, и Digium произвел карту B410P для удовлетворения этой потребности.

в программном обеспечении, он не обеспечивает достаточной мощности для решения проблемы. Кроме того, эхоподавление в программном обеспечении накладывает нагрузку на процессор; аппаратные эхоподавители, встроенные в карту PSTN, снимают эту нагрузку с процессора.

Аппаратное эхоподавление может добавить несколько сотен долларов к стоимости вашего оборудования, но если вы серьезно относитесь к системе качества, инвестируйте дополнительные деньги сейчас, а не страдайте позже. Проблемы с эхо совсем не приятны, и ваши пользователи будут ненавидеть систему, если они ее испытывают.



Дополнительные сведения по теме эхоподавления см. в Приложении B.

Недавно стали доступны несколько программных эхоподавителей. У нас не было возможности оценить их, но мы знаем, что они используют те же алгоритмы, что и аппаратные эхокомпенсаторы. Если у вас есть недавно приобретенная аналоговая карта Digium, вы можете вызвать Digium sales для кода ключа, чтобы его новейшее программное обеспечение Echo canceler заработало с вашей системой.[20] Для других типов карт доступны другие варианты программного обеспечения, но для их использования может потребоваться приобрести лицензию.[21] Имейте в виду, что использование программных эхоподавителей обходится дорого. Они будут осуществлять существенную нагрузку CPU, которую необходимо учитывать при разработке системы с использованием этих технологий.

Типы телефонов

Мы все знаем, что такое телефон, но будет ли он таким же через пять лет? Частью революции, которой способствует Asterisk, является эволюция телефона, от простого устройства аудиосвязи до мультимедийного коммуникационного терминала, обеспечивающего все виды функций, которые еще предстоит представить.

В качестве введения к этой захватывающей концепции мы кратко обсудим различные типы устройств, которые мы в настоящее время называем “телефонами” (любой из которых может быть легко интегрирован с Asterisk). Мы также обсудим некоторые идеи о том, во что эти устройства могут превратиться в будущем (устройства, которые также легко интегрируются с Asterisk).

Физические телефоны

Любое физическое устройство, основной целью которого является терминирование по требованию линии аудиосвязи между двумя точками, может быть классифицировано как физический телефон. Как минимум, такое устройство имеет телефонную трубку и клавиатуру; он также может иметь функциональные клавиши, экран дисплея и различные аудиоинтерфейсы.

В этом разделе кратко рассматриваются различные пользовательские (или конечные) устройства, которые можно подключить к системе Asterisk. Мы более глубоко исследуем механику аналоговой и цифровой телефонии в Приложении A.

Аналоговые телефоны

Аналоговые телефоны существуют с момента изобретения телефона. Ещё лет 20 назад все телефоны были аналоговыми. Хотя аналоговые телефоны имеют некоторые технические различия в разных странах, все они работают на схожих принципах.

Когда человек говорит, голосовые связки, язык, зубы и губы создают сложное разнообразие звуков. Целью телефона является захват этих звуков и преобразование их в формат, подходящий для

20 Это программное обеспечение не является частью обычной загрузки Asterisk, потому что Digium должен заплатить, чтобы лицензировать его отдельно. Тем не менее, он дедушка его во всех его карт, поэтому он доступен бесплатно для всех, кто имеет аналоговую карту Digium, которая все еще находится под гарантией. Если вы используете аналоговую карту не-Digium, вы можете приобрести код для этого программного обеспечения echo canceler с веб-сайта Digium.

21 Sangoma также предлагает бесплатное программное обеспечение для эхоподавления на аналоговых картах (до шести каналов).

передачи по проводам. В аналоговом телефоне передаваемый сигнал *аналогичен* звуковым волнам, производимым говорящим человеком. Если бы вы могли видеть звуковые волны, идущие от рта к микрофону, они были бы пропорциональны электрическому сигналу, который вы могли бы измерить на проводе.

Аналоговые телефоны являются единственным видом телефонов, которые обычно доступны в любом розничном магазине электроники. Можно ожидать, что в ближайшие несколько лет ситуация кардинально изменится.

Проприетарные цифровые телефоны

По мере развития цифровых коммутационных систем в 80-х и 90-х годах телекоммуникационные компании разрабатывали цифровые частные УАТС и клавишные телефонные системы (KTS). Патентованные телефоны, разработанные для этих систем, полностью зависели от систем, к которым они были подключены, и не могли использоваться ни в каких других системах. Даже телефоны, произведенные одним и тем же производителем, не были кросс-совместимыми (например, набор Nortel Norstar не будет работать на АТС Nortel Meridian 1). Собственнический характер цифровых телефонов ограничивает их будущее. В эту новую эру основанных на стандартах коммуникаций они быстро будут выброшены на свалку истории.

Телефонная трубка в цифровом телефоне обычно идентична функции трубки в аналоговом телефоне, и они часто совместимы друг с другом. Цифровой телефон отличается тем, что внутри него аналоговый сигнал дискретизируется и преобразуется в цифровой, т.е. получается цифровое представление аналогового сигнала. Мы более подробно обсудили цифровые сигналы в Приложении А; пока достаточно сказать, что основное преимущество цифрового сигнала заключается в том, что он может передаваться на неограниченные расстояния без потери качества сигнала.

Шансы на то, что кто-либо когда-либо сделает проприетарный цифровой телефон, напрямую совместимый с Asterisk, невелики, но такие компании, как Citel[22], создали шлюзы, которые преобразуют проприетарные сигналы в протокол инициации сеанса (SIP).[23]

Телефоны ISDN

До VoIP, ближе всего к стандартному цифровому телефону был терминал ISDN-BRI. Разработанный в начале 1980-х годов, ISDN должен был революционизировать телекоммуникационную отрасль точно так же, как VoIP обещает, наконец, достичь сегодня.



Существует два типа ISDN: *Primary Rate Interface* (PRI) и *Basic Rate Interface* (BRI). PRI обычно используется для обеспечения транкинга между УАТС и ТфОП и широко используется во всем мире. BRI совсем не популярен в Северной Америке, но распространен в Европе.

Хотя ISDN была широко развернута телефонными компаниями, многие считают, что стандарт был провалом, поскольку он, как правило, не оправдал своих обещаний. Высокие затраты на внедрение, периодические сборы и отсутствие сотрудничества между основными игроками отрасли способствовали созданию условий, которые вызывали больше проблем, чем решали.

BRI предназначался для обслуживания терминальных устройств и небольших местоположений (шлейф BRI обеспечивает две цифровые линии). Было разработано множество устройств BRI, но BRI в основном устарела в пользу более быстрых и менее дорогих технологий, таких как ADSL, кабельные модемы и VoIP.

---

22 Citel произвела фантастический продукт, но он ограничен тем, что он слишком дорог. Если у вас есть старые проприетарные телефоны АТС, и вы хотите использовать их с вашей системой Asterisk, технология Citel может сделать эту работу, но убедитесь, что вы понимаете, как складывается стоимость каждого порта этих устройств против замены старых аппаратов чистыми телефонами VoIP.

23 SIP в настоящее время является самым известным и популярным протоколом для VoIP. Мы обсудили это в Приложении В.

BRI все еще очень популярен для использования в оборудовании для видеоконференций, поскольку оно обеспечивает соединение с фиксированной пропускной способностью. Кроме того, BRI не имеет проблем с качеством обслуживания, которые могут возникнуть при подключении VoIP, поскольку он имеет коммутацию каналов.

BRI по-прежнему иногда используется вместо аналоговых линий для обеспечения транкинга к АТС. Является ли это хорошей идеей или нет, зависит в основном от того, как ваша местная телефонная компания оценивает услугу и какие функции она готова предоставить.[24]

IP-телефоны

IP-телефоны являются предвестниками самых захватывающих изменений в телекоммуникационной отрасли. Уже сейчас в розничных магазинах доступны стандартные IP-телефоны. Богатство возможностей, присущих этим устройствам, вызовет взрыв интересных приложений, от видеотелефонов до высококачественных широковещательных устройств, беспроводных мобильных решений до специально построенных комплектов для конкретных отраслей промышленности до гибких мультимедийных систем "все в одном".

Революция, которую породят IP-телефоны, не имеет ничего общего с новым типом проводов для подключения вашего телефона, и все это связано с предоставлением вам возможности общаться так, как вы хотите.

IP-телефоны ранних моделей, которые были доступны в течение нескольких лет, не представляют будущего этих захватывающих устройств. Это всего лишь ступенька, знакомый пакет, в который можно завернуть фантастический новый образ мышления.

Будущее гораздо более многообещающее.

Софтфоны

Софтфон — это программное обеспечение, которое обеспечивает функциональность телефона на не-телефонном устройстве, таком как ПК или КПК. Так как же мы узнаем такого зверя? Вроде бы на первый взгляд простой вопрос на самом деле возникает очень часто. Софтфон, вероятно, должен иметь своего рода клавиатуру, и он должен обеспечить интерфейс, который напоминает пользователям телефон. Но так ли будет всегда?

Можно ожидать, что термин *софтфон* будет быстро развиваться, поскольку наша концепция того, что такое телефон, претерпевает революционные метаморфозы.[25] В качестве примера этой эволюции рассмотрим следующее: правильно ли мы определяем популярные коммуникационные программы, такие как Instant Messenger, как софтфоны? IM предоставляет возможность инициировать и получать соединения VoIP на основе стандартов. Разве это не квалифицирует его как софтфон? Ответ на этот вопрос требует знания будущего, которым мы еще не обладаем. Достаточно сказать, что, хотя на данный момент ожидается, что софтфоны будут выглядеть и звучать как традиционные телефоны, эта концепция, вероятно, изменится в ближайшем будущем.

По мере развития стандартов и перехода от традиционного телефона к культуре мультимедийной коммуникации граница между софтфонами и физическими телефонами будет действительно размываться. Например, мы можем приобрести коммуникационный терминал, который будет служить телефоном, и установить на него программу софтфона, чтобы обеспечить нужные нам функции.

Замутнив таким образом воду, лучшее, что мы можем сделать на данном этапе - это определить, к чему будет относиться термин *софтфон* по отношению к этой книге, при том понимании, что можно ожидать, что значение этого термина претерпит массовые изменения в течение следующих нескольких лет. Для наших целей мы определим софтфон как любое устройство, которое работает на персональном компьютере, представляет внешний вид телефона и обеспечивает в качестве своей

24 Если вы находитесь в Северной Америке, откажитесь от этой идеи, если у вас не хватает терпения и денег и вы не мазохист.

25 Когда-нибудь слышали о Skype?

основной функции возможность совершать и принимать полнодуплексные аудиозвонки (ранее известные как “телефонные звонки”)[26] через адресацию E. 164.[27]

Телефонные адаптеры

Телефонный адаптер (обычно называемый ATA или аналоговым терминальным адаптером) может быть описан как устройство конечного пользователя, преобразующее линии связи из одного протокола в другой. Чаще всего эти устройства используются для преобразования цифрового (IP или проприетарного) сигнала в аналоговое соединение, к которому можно подключить стандартный телефон или факс.

Эти адаптеры можно описать как шлюзы, поскольку это их функция. Однако популярное использование термина телефонный шлюз, вероятно, лучше всего описывает многопортовый телефонный адаптер, обычно с более сложными функциями маршрутизации.

Телефонные адаптеры будут с нами до тех пор, пока существует необходимость подключения несовместимых стандартов и старых устройств к новым сетям. В конце концов, наша зависимость от этих устройств исчезнет, как и наша зависимость от модема - из-за неактуальности.

Терминалы связи

Терминал связи - это старый термин, который исчез на десятилетие или два и снова появился здесь, очень возможно, по той причине, что его нужно обсудить, чтобы он мог в конечном итоге исчезнуть снова - до тех пор как он снова станет вездесущим.

Сначала немного истории. Когда цифровые АТС были впервые выпущены, производители этих машин поняли, что они не могут называть свои конечные точки телефонами — их собственный характер не позволял им подключаться к ТфОП. Поэтому их называли терминалами, или станциями. Пользователи, конечно, не имели ничего от этого. Он выглядел как телефон и действовал как телефон, и поэтому это был телефон. Вы все еще иногда найдете комплекты АТС, называемые терминалами, но по большей части они называются телефонами.

Обновленная актуальность термина терминал связи не имеет ничего общего с чем-либо проприетарным - скорее, наоборот. По мере того как мы разрабатываем более творческие способы общения друг с другом, мы получаем доступ ко многим различным устройствам, которые позволят нам присоединиться. Рассмотрим следующие сценарии:

- Если я подключу видеокамеру к компьютеру, подключусь к веб-сайту компании и запрошу чат с представителем службы поддержки клиентов, мой компьютер теперь телефон?
- Если я использую IP-телефон на кухне для поиска рецептов, это телефонный звонок?

Дело просто в том, что мы, вероятно, всегда будем “звонить” друг другу, но всегда ли мы будем использовать для этого “телефоны”?

Особенности Linux

Если вы спросите кого-нибудь в Free Software Foundation, они скажут вам, что то, что мы знаем как Linux, на самом деле GNU/Linux. Все этимологические аргументы в сторону, есть некая ценная истина в этом утверждении. Хотя ядро операционной системы действительно Linux, подавляющее большинство утилит, установленных в системе Linux и используемых регулярно, на самом деле являются утилитами GNU. "Linux" - это, вероятно, только 5 процентов Linux, возможно, 75 процентов GNU и, возможно, 20 процентов всего остального.

26 Так ты думаешь, что знаешь, что такое телефонный звонок? Мы тоже. Давай просто подождем несколько лет, хорошо?

27 E.164 - это стандарт МСЭ, определяющий порядок присвоения телефонных номеров. Если вы хоть раз пользовались телефоном, вы пользовались адресацией E.164.

Почему это так важно? Ну, гибкость Linux-это одновременно и благословение, и проклятие. Это благословение, потому что с Linux вы действительно можете создать свою собственную операционную систему с нуля. Поскольку очень немногие люди когда-либо делают это, проклятие в значительной степени связано с ответственностью, которую вы должны нести при определении того, какие утилиты GNU установить и как настроить систему.

Вывод

В этом приложении мы обсудили всевозможные вопросы, которые могут способствовать стабильности и качеству установки Asterisk. Сколько времени и усилий вы должны посвятить выполнению рекомендаций, приведенных в этом приложении, зависит от того, какую работу вы ожидаете от сервера Asterisk и какое качество и надежность должна обеспечить ваша система. Если вы экспериментируете с Asterisk, не волнуйтесь слишком; просто имейте в виду, что любые проблемы, которые у вас есть, могут быть не по вине системы Asterisk.

То, что мы попытались сделать в этом приложении, - это дать вам представление о лучших практиках, которые помогут гарантировать, что ваша система Asterisk будет построена на надежной, стабильной платформе. Asterisk вполне готова работать в гораздо худших условиях, но количество усилий и внимания, которое вы решите посвятить этим вопросам, сыграет свою роль в стабильности вашей АТС. Ваше решение должно зависеть от того, насколько критичной будет ваша система Asterisk.

Алфавитный указатель

А
автосекретарь 325
  проектирование 325
    главное меню 327
    набор добавочного номера 329
    неверный обработчик 329
    приветствие 327
    тайм-аут 328
  создание 329
    входящие вызовы 332
    диалплан 331
    запись подсказок 330
  сравнение с IVR 325
агенты, определение 259
адаптеры, телефонные адаптеры 621
аккаунты
  пример базы данных 473
  сканирование действительных учетных записей 540
акцент подсказок, интернационализация 178
аналоговая телефония 571
  линии ТфОП 119
  части 571
  tip and ring 573
аналоговые интерфейсные карты 615
аналоговые линии, линии ТфОП 126
аналоговые телефоны
  интернационализация 173
  о 91, 618
аналоговые транки, пример системы клавиш 312
аппаратные средства 27, 604
  заземление 613
  комната оборудования 614
  питание отвечающее стандартам и источники бесперебойного питания 612
  подключение исключительно к сети с коммутацией пакетов 617
  сервер 605
    материнская плата 609
    производительность 606
    процессор 608
    электропитание 611
  телефоны 618
    аналоговые телефоны 618
    софтфоны 620
    телефонные адаптеры 621
    телефоны ISDN 619
    терминалы связи 621
    цифровые телефоны 619
    IP-телефоны 620
  ТфОП 615
    электрическая цепь 614
    эхо 617
аргументы
  вызов подпрограммы в диалплане 195
  макросы диалплана 193
архитектура 7
  аппаратные средства 27
  диалплан 27
  модули 8
    дополнительные модули 25
    драйверы каналов 15
    интерпретаторы формата 18
    конверторы кодеков 17
    модули ресурсов 21
    модули соединений 14
    модули УАТС 21
    модули CDR 14
    модули CEL 15
    приложения 10
    тестовые модули 26
    функции диалплана 19
  открытая архитектура 557
  управление версиями 28
    предыдущие методологии выпуска 28
    текущая методология версий 29
    упрощение номеров версий 30
  файловая структура 26
    внешние источники данных 27
    логирование 27
    модули 26
    файлы конфигурации 26
    spool 27
атаки на отказ в обслуживании
  распределенные 256
  IAX2 548
аутентификация
  безопасность 542
  AMI через HTTP 448
Б
база данных 333
  голосовая почта 169
  кластеризация 481
    единая база данных 481
    реплицирование базы данных 482
  поиск 243
  пример аккаунта базы данных 473
  AstDB 200
    использование 201
    получение 200
    пример горячего стола 202
    удаление 201
    хранение 200

банки каналов 616
барьер для входа 570
безопасность 254, 540
  аппаратное обеспечение 615
  аутентификация 542
  диалплан 547
  другие меры по снижению риска 550
  информация доступна на Asterisk wiki 99
  инъекция SQL 344
  мошенничество с оплатой 255
  разрешения CLI 551
  распределенные атаки на отказ в обслуживании 256
  ресурсы 552
  сеть API 548
  сканирование действительных учетных записей 540
  спам по интернет-телефонии (SPIT) 255
  укрепление сервера 603
  факсы 429
  фишинг 256
  шифрование 547
  fail2ban
    конфигурация 543
    установка 542
  Fail2ban 542
  H.323 592
  IAX 589
  IAX2 отказ в обслуживании 548
  SIP 591
  VoIP 601
    разделение голосового и трафика данных 603
    спуфинг 602
    физическая безопасность 603
    шифрование 603
    DMZ 603
    SPIT 602
    SRTP 602
беспроводная телефония 565
библиотека ресурсов 27
библиотека FreeTDS
  cdr_tds 518
  cel_tds 527
библиотеки
  внешние источники ресурсов 27
  LibPRI 46
будильник 393
В
взвешивание очереди 280
взлом indications.conf 69
видеоконференции
  будущее 564
  включение 234
виртуальные устройства 297
включения (инклюды), диалплан 116
влажность,комната оборудования 614
внешние источники данных 27
внешние сообщения 415
  конфигурирование диалплана 417
  sip.conf 416
  xmpp.conf 415
внешние устройства, интернационализация 171
внешний пейджинг 220
внутреннее хранение, голосовая почта 164
внутренние номера
  имена 100
  концепция 76
  набор добавочного номера, автосекретарь 329
  пример общего внутреннего номера 320
  статусы 299
    проверка 300
    хинты 299
  extension, диалплан 99
возобновление 265
время, NTP 38, 41
входящие вызовы, автосекретарь 332
вызов
  макрос диалплана 192
  подпрограммы диалплана 195
вызовы
  автосекретарь 332
  двухсторонний вызов 528
  запись информации о вызове в календарь 398
  запуск будильника 393
  неавторизованные вызовы 137
  односторонний вызов 527
  планирование 394
  управление вызовами на основе информации календаря 397
  Google Voice 414
  SIP URI 239
выражение register 601
выражения, диалплан 182
высококачественный голос, будущее 563
Г
гавное меню, автосекретарь 327
гибридные телефонные системы 479
гибридный трансформатор 572
глобальные переменные 110
голос
  высококачественный голос 563
  против видеоконференций 565
голосовая почта
  автономный сервер голосовой почты 165
    SIP 165
    SMDI 168

внутреннее хранение 164
диалплан 159
каталог набор-по-имени 162
приложение VoiceMail() 159
приложение VoiceMailMain() 162
jitterbuffer'ы 163
интеграция базы данных 169
проверка паролей 150
Comedian Mail 145
раздел [general]146
раздел [zonemessages] 154
раздел контекстов 154
стандартные клавиши голосовой почты 158
voicemail.conf 158
IMAP 401
голосовая почта ODBC 369
компиляция app_voicemail 370
макет таблицы хранения 372
настройка voicemail.conf 373
тестирование 374
MySQL 376
PostgreSQL 375
тип Large Object для PostgreSQL 370
группировка карт приложений 217
Д
двусторонний вызов
CDR 518
CEL 528
дейтвие, кодирование сообщений 448
делегированные зоны 252
демилитаризованная зона (DMZ), VoIP 603
детализация логирования 508
диалплан 97, 182
автосекретарь 331
безопасность 547
включения 116
внешние подключения 118
выражения 182
голосовая почта 159
каталог набор-по-имени 162
приложение VoiceMail() 159
приложение VoiceMailMain() 162
jitterbuffer 163
конференц-связь
приложение ConfBridge() 209
приложение MeetMe() 208
кратко 27
локальные (Local) каналы 198
макрос 191
аргументы 193
вызов 192
определение 192
неверные значения и тайм-ауты 106
операторы 183
очереди 380
парковка вызова 207
переменные 110
приложение ConfBridge() 228, 233
приложение Dial() 107
приложение GoSub() 194
аргументы 195
возврат 196
вызов подпрограмм 195
определение подпрограмм 194
приложение GotoIf() 186
приложение GotoIfTime() 189
приложение Zapateller() 206
приложения 10
приложения Goto(), Background() и WaitExten() 104
пример Hello World 103
синтаксис 97
контекст 98
приложения 102
приоритеты 100
расширения (extensions) 99
совпадение шаблонов 112
канальная переменная ${EXTEN} 115
пример 113
синтаксис 112
тестирование пользовательских устройств 94
управления участниками очереди 264
файлы конфигураций каналов 80
функции 19, 185
функция диалплана CURL() 385
AstDB 200
CDR 512
CEL 521
cURL 386
DUNDi
поиск 503
сопоставление 501
func_odbc 344
include 116
SIP и Asterisk как автономный сервер голосовой почты 167
SIP URI 240
XMPP 407
подключение к XMPP серверу 407
приложение JabberSend() 409
функция диалплана JABBER_RECEIVE() 410
динамический Realtime
краткие сведения 362
queues.conf378
динамическое добавление добавочных номеров 501
дифференцированный сервис 597

добавочные номера, DUNDi 501
документация 4
  проект документации Asterisk 6
  AMI (Asterisk Manager Interface) 47
домашняя автоматизация, open source телефония 562
дополнительные модули 25
драйверы каналов 15
3
зависимости
  календарь и зависимости RHEL 390
  календарь и зависимости Ubuntu 390
  модули 56
  программного обеспечения 42
загрузка 43
  Subversion 43
  wget 44
заземление 613
запись подсказок 329
звонок 572
звук
  высококачественный голос 563
  конференц-связь 227
И
имена
  концепции именования телефонов 76
  расширений 100
импульсно-кодовая модуляция
  компандирование 580
  наложение 582
  разрешение и частота дискретизации 577
  теорема Найквиста 579
  цифровое кодирование аналогового сигнала 574
имульсно-кодовая модуляция 574
имя системы, глобальный уникальный идентификатор 366
инициирование вызова 454, 456
интеграция
  календарь 22
  телефонии 570
интернационализация 170
  внешние устройства 171
  ТфОП, DAHDI, карты Digium и аналоговые телефоны 173
  шпаргалка 181
  штампы время/дата 179
  Caller ID 177
интерфейсы
  синхронизации 22
  menuselect 54
интерфейсы синхронизации 22
источники бесперебойного питания 612
К
календарь 389
  интеграция календаря 22
  компиляция 390
    зависимости RHEL 390
    зависимости Ubuntu 390
  напоминания 393
    будильник 393
    дополнительные функции 400
    запись информации о вызове 398
    конференц-связь 396
    планирование вызовов 394
    управление вызовами 397
  настройка 391
каналы
  каналы Local и очереди 290
  локальные (Local) каналы 198
канальная переменная
  наследование 215
  сводка 111
карты Digium, интернационализация 173
каталог
  каталог набор-по-имени 162
  фамилии 157
каталог набор-по-имени 162
кдирование manager 450
кластеризация 477
  базы данных 481
    единая база данных 481
    реплицирование базы данных 482
  гибридные телефонные системы 479
  нераспределенный чистый Asterisk 480
  очереди 486
  предоставление статуса устройств
    по LAN 484
    по WAN 485
  предоставление статуса устройства 484
  УАТС 478
кодеки 593
  G.711 594
  G.722 596
  G.726 594
  G.729A 595
  GSM 595
  iLBC 595
  MP3 71, 596
  Speex 595
кодирование mxml 451
кодирование rawman 450
команды оболочки 50
команды CLI, участники очереди 262
комбинации пейджинга 223
комедийная почта 145
  раздел [general] 146
  раздел [zonemessages] 154
  раздел контекстов 154

стандартные клавиши голосовой почты 158
voicemail.conf 158
комната оборудования 614
компандирование 580
компиляция
голосовой почты IMAP 401
календаря 390
зависимости RHEL 390
зависимости Ubuntu 390
модулей ODBC 342
app_voicemail 370
LDAP 420
res_odbc.conf 342
spandsp 427
XMPP 407
конверторы кодеков 17
контексты
диалплан 98
DUNDi 497
конференц-связь 208, 224
видеоконференции 234
источников синхронизации 22
маркированный пользователь 231
напоминания календаря 396
приложение ConfBridge() 209, 228, 232
приложение MeetMe() 208
профили пользователей 225
профили соединения 226
раздел [general] 224
open source телефония 561
PIN 229
конфигурация
аналоговых линий 126
бэкенд, модули ресурсов 21
диалплана внешних сообщений 417
календарь 390
конфигурирование Asterisk для LDAP 420
конфигурирование res_ldap.conf 421
extconfig.conf 421
sip.conf для realtime 422
пользовательских устройств 75
аналоговые телефоны 91
диалплан проверки устройств 94
загрузка новых конфигураций канала 90
концепции именования телефонов 76
регистрация 91
телефоны Digium 90
телефоны, софтфоны и ATA 77
Asterisk 79
файлы конфигурации 26
цифровых линий 122
AMI 441
http.conf 445
manager.conf 442
Asterisk 47, 63
дополнительные файлы конфигурации 73
пользовательских устройств 79
asterisk.conf 48, 63
Corosync 304
indications.conf 48, 69
menuselect 54
modules.conf 51, 67
musiconhold.conf 53, 70
SIP и состояния устройств 301
XMPP 309
CEL 521
Corosync 305
DUNDi 491
контексты сопоставления 496
определение пиров 494
ответов 500
поиск 503
раздела [general] 493
удаленных подключений 498
dundi.conf 491
Fail2ban 543
MySQL 336
ODBC 337
для Microsoft SQL 340
для MySQL 339
для PostgreSQL 338
OpenLDAP 418
PostgreSQL 335
sip.conf 416, 422
SLA 312, 321
SNMP использование OpenNMS 532
установка OpenNMS 532
res_snmp.conf 532
snmpd.conf 533
voicemail.conf 373
VoIP-транков 137
IAX-транки между системами Астериск 142
SIP-транков между Астериск-системами 137
xmpp.conf 415
конфигурация клавиш, голосовая почта 158
Л
линии ОС 584
линии PSTN (ТфОП) 119
аналоговая телефония 119
аналоговые линии 126
установка 121
BRI ISDN 125
DAHDI 121
MFC/R2 125
PRI ISDN 123

цифровая телефония 120
линия с Т-несущей 583
логирование 27
логические операторы 183
локальные (Local) каналы
очереди 290
Локальные (Local) каналы
диалплан 198
очереди 290
M
макет таблицы хранения, голосовая почта ODBC 372
макрос, диалплан 191
аргументы 193
вызов 192
определение 192
маркированный пользователь, конференц-связь 231
математические операторы 184
материнская плата 609
меню
главное меню, автосекретарь 327
приложение ConfBridge() 232
управление громкостью 232
dialplan_exec 233
меню управления громкостью 232
метки, метки приоритетов 101
метрики 360
многоадресный пейджинг 221
многоадресный RTP 23
модернизация Asterisk 61
модули 8
дополнительные модули 25
драйверы канала 15
зависимости 56
интерпретаторы формата 18
конверторы кодеков 17
модули ресурсов 21
интеграция календаря 22
интерфейсы синхронизации 22
конфигурирование 21
обработчики атрибутов формата 23
прочие модули ресурсов 24
расширения CLI 24
реализация RTP 23
модули соединений 14
модули УАТС 21
модули CDR 14
модули CEL 15
приложения 10
тестовые модули 26
файловая структура 26
функции диалплана 19
DAHDI 121
модули ресурсов 21
интеграция календаря 22
интерфейсы синхронизации 22
конфигурирование 21
обработчики атрибутов формата 23
прочие модули ресурсов 24
расширения CLI 24
реализация RTP 23
мониторинг Asterisk с помощью OpenNMS 534
мошенничество
будущее 568
сведения 255
NANP 114
H
набор добавочного номера, автосекретарь 329
набор SIP URI 245
назначение, приложение Dial() 107
наилучшие усилия, QoS 598
наложение 582
напоминания
календарь 393
будильник 393
дополнительные функции 400
запись информации о вызове 398
конференц-связь 396
планирование вызовов 394
управление вызовами 397
наследование переменных канала 215
настольный пейджинг 220
неавторизованные вызовы 137
неверные значения и тайм-ауты в диалплане 106
неверный обработчик, автосекретарь 329
ненумерованные приоритеты 100
номеронабиратель 572
O
обмен сообщениями 415
будущее 566
обновление
диалплан 109
Asterisk 58
обработка речи, будущее 562
оператор same => 101
операторы регулярных выражений 184
операторы, диалплан 183
отключение
Digium Fax for Asterisk 429
spandsp 428
очереди 258
кластеризация 486
web-интерфейсы 538
Очереди автоматического распределения вызовов (ACD)
диалплан 380
каналы Local 290

Очереди автоматического распределения вызовов (ACD) **258**
переполнение
переполнение 288
присоединение и выход из очереди 289
тайм-ауты 289
пример 259
приоритет очереди 280
управление объявлениями 284
участники очереди
диалплан 264
команда CLI 262
несколько очередей 266
приоритет 282
состояния устройств 269
участники очереди 262
queues.conf 264
agents.conf 278
queuerules.conf 283
queues.conf271, 378
П
парковка вызовов 207
парковочные места 217
пароли, проверка паролей голосовой почты 150
пауза 265
пейджинг 218
внешний пейджинг 220
комбинации пейджинга 223
многоадресный пейджинг 221
настольный пейджинг 220
пример 223
VoIP-пейджинговые адаптеры 223
переменные
диалплан 110
кавычки в условных ветвлениях 188
переменные канала 215
переменные среды 111
переполнение, очереди 288
присоединение и выход из очереди 289
тайм-ауты 289
пиринг, будущее 566
пиры
DUNDi
контексты сопоставления 497
определение 494
VoIP 600
питание отвечающее стандартам и источники бесперебойного питания 612
пицца, IVR 384
планирование вызовов 394
подавление, эхоподавление 599, 617
подсказки
автосекретарь 329
интернационализация 178
кавычки переменных в условных ветвлениях 188
IVR 386
поиск
база данных 243
DUNDi и диалплан 503
пользователи
группы 6
профили, конференц-связь 225
VoIP 600
потолочный, пейджинг 218
приветствие, автосекретарь 327
приложение Answer() 102
приложение Background() 105
приложение ConfBridge() 209, 228, 232
приложение Dial() 107
назначение 107
обновление диалплана 109
строка опций 109
тайм-аут 108
URI 109
приложение GoSub() 194
аргументы 195
возврат 196
вызов подпрограмм 195
определение подпрограмм 194
приложение Goto() 104
приложение GotoIf() 186
приложение GotoIfTime() 189
приложение Hangup() 103
приложение JabberSend() 409
приложение MeetMe() 208
приложение Playback() 102
приложение Progress() 102
приложение r2test 125
приложение WaitExten() 104
приложение Zapateller() 206
приложения
диалплан
интерактивность 104
о приложениях 102
список 10
CDR 512
CEL 521
приложение ConfBridge 209, 228, 232
приложение Dial() 107
назначение 107
обновление диалплана 109
пустые аргументы 109
строка опций 109
тайм-аут 108
URI 109
приложение GoSub 194
приложение GoSub()
аргументы 195

возврат 196
вызов подпрограмм 195
определение подпрограмм 194
приложение GotoIf() 186
приложение GotoIfTime() 189
приложение JabberSend() 409
приложение MeetMe() 208
приложение Progress() 102
приложение r2test 125
приложение VoiceMail() 159
приложение VoiceMailMain() 162
приложение Zapateller() 206
SLA 311
пример общего внутреннего номера 320
пример одностороннего вызова
CDR 518
CEL 527
пример перенаправления вызова 455
пример системы клавиш
с аналоговыми транками 312
с SIP-транками 315
пример альтернативы 317
extensions.conf 316
sla.conf 315
пример системы клавиш с SIP-транками
extensions.conf 316
пример слепого трансфера 529
пример Hello World, диалплан 103
приоритет участника очереди 282
приоритетная очередь 280
приоритеты, диалплан 100
проверка
паролей голосовой почты 150
ODBC-коннектор 341
проект телефонии Zapata 2
производительность, серверы 606
протокол резервирования (RSVP) 598
профили соединения, конференц-связь 226
процессоры 608
пыль, комната оборудования 615
P
разбор файлов 242
разработка фреймворков
AGI 476
AMI 461
разрешение и частота дискретизации 577
разрешения
файлы 47
CLI 551
разрешения CLI 551
распознавание речи
будущее 563
IVR 387
распределенные атаки на отказ в обслуживании 256
распространение состояний устройств 304
распространение статусов устройств 484
Corosync 304
конфигурация 305
тестирование 307
установка 304
XMPP 308
конфигурация Asterisk 309
по LAN 484
по WAN 485
тестирование 310
установка 308
учетные записи 309
расширение s 127
расширения CLI 24
регистрация, тестирование устройств 91
регламентирование, VoIP и телефония 568
резервирование, требования к электропитанию 612
реляционная база данных 333
голосовая почта ODBC 369
компиляция app_voicemail 370
макет таблицы хранения 372
настройка voicemail.conf 373
тестирование 374
тип Large Object для PostgreSQL 370
инъекция SQL 344
компиляция модулей ODBC 342
настройка
MySQL 336
ODBC 337
PostgreSQL 335
res_odbc 342
очереди 377
диалплан 380
queue_log 380
queues.conf 378
проверка ODBC-коннектора 341
установка
MySQL для RHEL 334
MySQL для Ubuntu 334
ODBC 337
PostgreSQL для RHEL 333
PostgreSQL для Ubuntu 334
устранение неполадок 343
ARA 358
динамический realtime 362
статический realtime 359
CDR 365
func_odbc 344
репозитории, сторонние репозитории 42
ротация, logger.conf 510
рычажный переключатель 573
C
серверы

безопасность 603
материнская плата 609
производительность 606
процессоры 608
требования к электропитанию 611
сессии
AGI467
AMI через HTTP 448
сетевой трансформатор 572
сети с коммутацией пакетов586
сеть с коммутацией пакетов, подключение 617
сигнализация по выделенному каналу (CAS) 584
системное время, NTP 38, 41
сканирование действительных учетных записей 540
скрипты, menuselect 57
служба обратного звонка, состояния устройств 323
соблюдение стандартов
будущее 567
традиционная телефония555
open sorce телефония 558
события
журнал очереди 295
кодирование сообщений 447
события диспетчера и AMI через HTTP 451
CEL 519
события диспетчера, AMI через HTTP 451
совпадение по шаблонам
сведения 112
DUNDi 500
сообщения
приложение JabberSend() 409
функция диалплана JABBER_RECEIVE() 410
сообщество 4
группы пользователей Asterisk 6
проект документации Asterisk 6
сайт wiki 6
списки рассылки Asterisk 5
IRC-каналы 6
open source телефония 558
сопоставления, DUNDi и функции диалплана 501
состояние устройств 297
пользовательские состояния устройств 302
распространение состояний устройств 304
Corosync 304
XMPP 308
сводка 297
служба обратного звонка 323
статусы внутренних номеров 299
проверка 300
хинты 299
участников очереди 269
SIP 301
SLA 311
конфигурация 312, 321
ограничения 322
пример общего номера 320
пример системы клавиш с аналоговыми транками 312
пример системы клавиш с SIP-транками 315
установка приложения 311
софтфоны 77, 620
Спам через Интернет Телефонию (SPIT)
VoIP 602
спам, VoIP 568
списки рассылки, сообщество Asterisk 5
спуфинг, VoIP602
статический realtime
сведения 359
queues.conf378
сторонние репозитории 42
структура ветвления 30
T
тайм-ауты
автосекретарь 328
очереди 289
приложение Dial() 108
текст-в-речь
будущее 563
утилиты 422
IVR387
телефон 77
телефония 554, 571
аналоговая телефония 571
части 571
Tip and Ring 573
барьер для входа 570
беспроводность 565
видео 564
высококачественный голос 563
интеграция 570
качество обслуживания 569
мошенничество 568
обработка речи 562
пиринг 566
проектирование узких мест 568
регулирование 568
сдвиг парадигмы 556
соблюдение стандартов 567
страх, неуверенность и сомнение 568
творческие возможности 570
традиционная телефония554
закрытое мышление 555

прошлое и будущее 556
соблюдение стандартов 555
цикл выпусков 555
трудоемкость 569
универсальная система обмена сообщениями 566
цифровая коммутируемая телефонная сеть 583
типы линий связи 583
цифровые сигнальные протоколы 584
цифровая телефония 573
имульсно-кодовая модуляция 574
частные сети 570
open source телефония 556
домашняя автоматизация 562
конференц-связь 561
новые технологии 558
открытая архитектура 557
сведения 557
соблюдение стандартов 558
сообщество 558
шлюз миграции традиционной УАТС 559
IVR 560
VoIP спам 568
WebRTC 565
телефонная трубка 573
телефонный адаптер 621
телефоны
аналоговые 91
аналоговые телефоны 618
концепции именования телефонов 76
софтфон 620
телефонный адаптер 621
телефоны ISDN 619
терминал связи 621
цифровые телефоны 619
IP-телефоны 620
телефоны Digium 90
температура, комната оборудования 614
теорема Найквиста 579
терминал связи 621
терминалы, терминалы связи 621
тестирование
голосовая почта ODBC 374
MySQL 376
PostgreSQL 375
устройства пользователей
использование диалплана 94
регистрация 91
Corosync 307
ITAD 252
XMPP 310
тестовые модули 26
типа Large Object, PostgreSQL 370
типы статуса, menuselect 9
транки
основы 117
VoIP 137
IAX-транки между Asterisk-системами 142
SIP-транки между Asterisk-системами 137
трансфер
пример вызова 455
пример слепого трансфера 529
требования к электропитанию 611
ТфОП
инициирование 135
интернационализация 173
подключение к 615
терминация, VoIP 134
У
УАТС
модули 21
сравнение по кластеризации 478
шлюз миграции традиционной УАТС 559
удаленные подключения, настройка DUNDi 498
удаленный брандмауэр 130
узкие места 568
универсальная система обмена сообщениями, будущее 566
управление версиями 28
предыдущие методологии выпуска 28
текущая методология выпуска 29
упрощение номеров версий 30
управление объявлениями, очереди ACD 284
условные ветвления 188
установка
линии ТфОП 121
BRI ISDN 125
DAHDI 121
MFC/R2 125
PRI ISDN 123
приложения SLA 311
Asterisk 31
зависимости программного обеспечения 42
загрузка 43
конфигурация 47
модернизация 61
обновление 58
права доступа к файлам 47
проблемы 59
процедура установки 44
шпаргалка 33
DAHDI 44
LibPRI 46
RHEL сервер 36

Ubuntu сервер 39
Corosync 304
cURL 385
Fail2ban 542
Festival 423
MySQL
RHEL 334
Ubuntu 334
ODBC 337
PostgreSQL
RHEL 333
Ubuntu 334
res_xmpp 407
SNMP 531
spandsp 427
XMPP 308
устранение неполадок с базой данных 343
устройства 75
интернационализация 171
NAT 129
устройства пользователей
аналоговые телефоны 91
загрузка конфигураций новых каналов 90
концепции именования телефонов 76
телефоны Digium 90
телефоны, софтфоны и телефонные адаптеры 77
тестирование
используя диалплан 94
регистрации 91
Asterisk 79
участники очереди 262
Ф
фаерволл
преобразование сетевых адресов 85
удаленный 130
файлы 26
права доступа 47
разбор 242
структура 26
внешние источники ресурсов 27
Журналирование (логирование) 27
модули 26
файлы конфигурации 26
Spool 27
файлы вызовов и AMI 452
формат файла для отправки факсов 433
формат, файл подсказки 329
файлы вызовов, AMI 452
файлы конфигурации
краткая сводка 26
agents.conf 278
asterisk.conf 48, 63
cdr_adaptive_odbc 513
cdr.conf 512
cel.conf 521
chan_dahdi.conf 437
dundi.conf 491
extconfig.conf 421
extensions.conf 97, 314, 316, 320
features.conf 211
indications.conf 48
logger.conf 507, 509p.
manager.conf 442, 467
modules.conf 51, 67, 361
motif.conf 413
musiconhold.conf 53, 70
queuerules.conf 283
queues.conf 264, 271, 378
res_ldap.conf 421
res_snmp.conf 532
sip.conf 81, 240, 416, 422
sla.conf 313, 315, 320
snmpd.conf 533
voicemail.conf 158, 373, 404
xmpp.conf 415
файлы конфигурации каналов
диалпланом 80
изменение файлов для вашей среды 89
файлы лога
безопасность 545
cel_custom 524
факс
входящий 429
обнаружение факса 431
факс в Email 431
факс в TIFF 430
T.38 432
исходящий 433
передача 433
формат файла 433
email в факс 434
определение 426
сквозной 437
Digium Fax For Asterisk 428
spandsp 427
фамилии, директории 157
фишинг 256
формат
интерпретаторы 18
обработчики атрибутов 23
файла для отправки факсов 433
файлы подсказок 329
musiconhold.conf 70
функции, диалплан 19, 185, 501
функция диалплана CURL() 385
функция диалплана JABBER_RECEIVE() 410
функция ARRAY() 350
X
хакеры 4

хинты, состояния внутренних номеров 299
Ц
цикл выпусков, традиционная телефония 555
цифровая коммутируемая телефонная сеть 583
  типы линий связи 583
  цифровые сигнальные протоколы 584
цифровая телефония
  импульсно-кодовая модуляция
    компандирование 580
    наложение 582
    разрешение и частота дискретизации 577
    теорема Найквиста 579
    цифровое кодирование аналогового сигнала 574
  имульсно-кодовая модуляция 574
  линии ТфОП 120
цифровые интерфейсные карты 616
цифровые линии, настройка 122
цифровые телефоны 619
Ч
частные телекоммуникационные сети 570
Ш
шифрование
  безопасность 547
  IAX 143
  SIP-вызовов 139
  VoIP 603
шлюз миграции традиционной УАТС 559
штампы время/дата, интернационализация 179
штрафы, очереди 283
Щ
щтампы время/дата, интернационализация 179
Э
экстренный набор 143
электрическая цепь 614
эхо 598, 617
Я
языки, интернационализация 178
/
/var точка монтирования 37
$
${EXTEN} канальная переменная 115
A
A2Billing, web-интерфейсы 539
AGI (Asterisk Gateway Interface) 385, 463
  быстрый старт 463
  команды и ответы 469
    async AGI 471
    process-based AGI/fastAGI 471
  разработка фреймворков 476
  сеансы
    завершение 472
    настройка 467
  сессии
    завершение 472
  async AGI 466
  deadAGI 465
  fastAGI 466
  process-based AGI 464
AMI (Asterisk Manager Interface) 385, 439
  аналоговые интерфейсные карты 615
  безопасность 548
  быстрый старт 439
    AMI через HTTP 440
    AMI через TCP 440
  документация 47
  конфигурация 441
    http.conf 445
    manager.conf 442
  примеры 454
    инициирование вызова 454, 456
    перенаправление вызова 455
    flash-панель оператора 461
  протоколы 445
    кодировка сообщений 447
    AMI через HTTP 448
  разработка фреймворков 461
  файлы вызовов 452
  async AGI 466
  cel_manager 525
a
app_voicemail, компиляция 370
A
ARA (Asterisk Realtime Architecture) 358
  динамический realtime 362
  статический realtime 359
a
ast_tls_cert 141
A
AstDB (Asterisk database)
  база данных Asterisk 200
  извлечение данных 200
  использование 201
  пример горячего стола 202
  удаление 201
  хранение данных 200
Asterisk проект документации 6
Asterisk CLI 90
Asterisk-Biz, Asterisk-Dev, Asterisk-Users 5
a
asterisk.conf 48, 63
  раздел [compat] 67
  раздел [directories] 63
  раздел [files] 67
  раздел [options] 64
async AGI 466, 468, 471, 473

A
ATA 77
B
B2BUA 95
BLF 301
BRI ISDN 125
c
caller ID, интернационализация 177
cat 51
C
CDR (Call detail records) 365, 510
  веб-интерфейсы 539
  двухсторонний вызов 528
  модули CDR 14
  односторонний вызов 527
  поля 511
  предостережения 519
  приложения диалплана 512
  cdr_adaptive_odbc 513
  cdr_adaptive_odbc.conf 368
  cdr_csv 514
  cdr_custom 515
  cdr_manager 515
  cdr_mysql 516
  cdr_odbc 516
  cdr_pgsql 516
  cdr_radius 517
  cdr_sqlite и cdr_sqlite3_custom 517
  cdr_syslog 518
  cdr_tds 518
  cdr.conf 512
CEL (Channel event logging)
  модули 15
  поля 520
  приложения диалплана 521
  пример двухстороннего вызова 528
  пример одностороннего вызова 527
  пример слепого трансфера 529
  типы событий 519
  cel_custom 524
  cel_odbc 522
  cel_pgsql 526
  cel_radius 526
  cel_sqlite3_custom 526
  cel_tds 527
CEL (Channel Event Logging) 519
Cepstral 424
c
chan_dahdi.conf 437
chan_gtalk 412
chan_jingle 412
chan_motif
  классы, musiconhold.conf 72
  Google Talk 413p.
  Google Voice 414p.
  motif.conf 413
comebacktoorigin 213
comedian mail 145
C
Computer-Supported Telecommunications Applications (CSTA), AMI 461
Corosync 304
  конфигурация 305
  тестирование 307
  установка 304
CSTA (Computer-Supported Telecommunications Applications), AMI 461
c
cURL, IVR 385
D
DAHDI (Digium Asterisk Hardware Device Interface) 44
  интернационализация 173
  модули 121
  эхо 599
  indications.conf 70
d
dahdi_genconf 123, 125p., 176
dahdi_hardware 123, 126
dbsecret, iax.conf 499
D
DeadAGI 465
DevOps: автоматическое развертывание 477
d
dialplan_exec 233
D
Digium факс для Asterisk 428
DMZ (demilitarized zone), VoIP 603
DNS (Domain Name System)
  ITAD 251
  URI 238
DS-0 583
DTMF 211
DUNDi 489
  контексты сопоставления 496
  конфигурация 491
  краткие сведения 489
  определение узлов 494
  ответы 500
  поиск 503
  раздел [general] 493
  удаленные подключения 498
  dundi.conf 491
E
E.164 246, 567
e
e164.org 567
E
EAGI (Enhanced AGI) 465
e

email
факс в email 431
email по факсу 434
Fail2ban 543
E
ENUM
будущее 567
назначение 246
e
extconfig.conf 421
extensions.conf 97, 314, 316, 320
F
Fail2ban 542
конфигурация 543
установка 542
FastAGI
безопасность 548
завершение сеанса 472
команды 471
определение 466
переменные 467
f
features.conf 211
группировка сопоставлений приложений 217
парковочные места 217
раздел [applicationmap] 214
раздел [featuremap] 214
раздел [general] 212
F
Festival, сервер текст-в-речь 423, 563
Flash-панель оператора,пример 461
FOP (Flash Operator Panel) 537
f
freenum.org 237, 567
F
FreePBX GUI диалплана 537
Friends (друзья), VoIP 600
f
func_odbc 344, 385
F
FXO 119
FXS 119
G
G.711 594
G.722 596
G.726 594
G.729A 595
Global System for Mobile Communications (GSM) 595
Gmail, IMAP 403
Google Talk 413p.
Google Voice 414p.
GSM (Global System for Mobile Communications) 595
H
H.323 591
h
hot-desking 76, 203, 346
H
HTTP
AMI через HTTP 448
аутентификация и обработка сеанса 448
быстрый старт 440
кодирование manager 450
кодирование mxml 451
кодирование rawman 450
события диспетчера 451
h
http.conf 445
I
IANA (Internet Assigned Numbers Authority) 250
IAX (Inter-Asterisk eXchange)
сведения 588
транки между системами Астериск 142
i
iax.conf
сведения 86
dbsecret 499
I
IAX2 отказ в обслуживании 548
i
iLBC (Internet Low Bitrate Codec) 595
I
IMAP (Internet Message Access Protocol), голосовая почта 164, 401
i
indications.conf 48, 69
I
IP-телефоны 620
i
iptables 543
I
IPv6
доступа к базе данных PostgreSQL 335
sip.conf 82
IRC-каналы, сообщество Asterisk 6
ISDN (Integrated Services Digital Network)
сведения 585
телефоны 619
ISN (ITAD Subscriber Number) 249
ITADs (IP Telephony Administrative Domains)
записи DNS 251
сведения 250
тестирование 252
ISN 250
IVR (Interactive Voice Response)
запись приглашений 386
компоненты 382

модули Asterisk 384
распознавание речи 387
соображения по проектированию 384
сравнение с автосекретарем 325
телефония open source 560
что это такое 382
CURL 385
text-to-speech 387
j
jitterbuffer 163
L
LAN, распространение состояний устройств 484
LDAP 418
компиляция 420
конфигурирование Asterisk 420
extconfig.conf 421
res_ldap.conf 421
sip.conf для realtime 422
конфигурирование OpenLDAP 418
l
libpath, spandsp 427
L
LibPRI 46
Linux 39
сведения 621
файловая система голосовой почты 164
DAHDI-Linux 45
logger.conf и демон syslog 509
l
logger.conf 507
демон Linux syslog 509
журналы Asterisk 509
проверка логирования 510
ротация логов 510
lsdahdi 123, 126
m
manager.conf 442, 467
M
Media Gateway Control Protocol (MGCP) 593
m
menuselect 54
дополнительные модули 25
интерфейсы 54
использование 54, 56
компиляция и установка модулей 68
скрипты 57
типы статусов 9
M
MFC/R2 125
MGCP (Media Gateway Control Protocol) 593
Microsoft SQL, настройка ODBC 340
m
modules.conf 51, 67, 361
motif.conf 413
M
MP3
сведения 596
RHEL 71
MPLS (Multiprotocol Label Switching) 598
m
musiconhold.conf 70
сведения 53
файл по-умолчанию 72
формат 70
M
MySQL
голосовая почта ODBC, тестирование 376
настройка 336
ODBC 339
установка
RHEL 334
Ubuntu 334
cdr_mysql 516
N
NANP (North American Numbering Plan)
мошенничество с тарифами 114
сведения 247
NAT (network address translation) 128
обработка RTP 132
устройства 129
фаерволл 85
Asterisk 131
H.323 592
IAX 589
SIP 591
NTP
RHEL сервер 38
Ubuntu сервер 41
O
ODBC
голосовая почта 164
компиляция
для Asterisk 342
res_odbc 342
настройка 337
для Microsoft SQL 340
для MySQL 339
для PostgreSQL 338
проверка ODBC-коннектора 341
установка 337
cdr_odbc 516
cel_odbc 522
OpenLDAP, конфигурирование 418
OpenNMS
SNMP 532
мониторинг 534
установка OpenNMS 532
res_snmp.conf 532
snmpd.conf 533

OpenR2 125
P
PDF, факсы 431
PIN-код, конференц-связь 229
PostgreSQL
голосовая почта ODBC
тестирование 375
настройка 335
ODBC 338
тип Large Object 370
установка
для RHEL 333
для Ubuntu 334
cdr_pgsql 516
cel_pgsql 526
PRI ISDN 123
p
process-based AGI 464, 467, 471p.
P
Python, пример инициирования вызова 456
Q
QoS (Quality of Service) 596
дифференцированное обслуживание 597
негарантированная доставка 598
сведения 569
MPLS 598
RSVP 598
SCTP 597
TCP 596
UDP 597
q
queue_log 294, 380
queuerules.conf 283
queues.conf 264, 271, 378
R
RADIUS сервер
cdr_radius 517
cel_radius 526
r
realtime 358
sip.conf 422
res_ldap.conf 421
res_odbc, компиляция 342
res_snmp 532
res_xmpp 407
R
RHEL
голосовая почта IMAP 401
компиляция календаря 390
компиляция LDAP 420
компиляция XMPP 407
установка MySQL334
установка PostgreSQL 333
установка SNMP 531
Festival 423
MP3 71
musiconhold.conf 71
RHEL сервер 36
включение NTP 38
добавление пользователя в систему 38
обновление базовой системы 37
установка базовой системы 36
RSVP (Reservation Protocol) 598
RTP
реализация23
NAT 132
SRTP 602
s
safe_asterisk 51
S
SCCP (Skinny Client Control Protocol) 593
SCTP (Stream Control Transmission Protocol) 597
Signaling System 7 (SS7) 586
Simplified Message Desk Interface (SMDI) 168
SIP (Session Initiation Protocol)
каналы 292
неавторизованные вызовы 137
пример системы клавиш с SIP-транками 315
пример альтернативы 317
extensions.conf 316
sla.conf 315
распределенные атаки на отказ в обслуживании 256
сведения 75
состояния устройств 301
Asterisk в качестве автономного сервера голосовой почты 165
URI238
вызов 245
вызовы 239
сведения 238
SRV записи 238
VoIP 137, 589
SIP URI
вызовы
диалплан 240
поиск в базе данных 243
разбор файлов 242
sip.conf 240
s
sip.conf 81, 240, 416, 422
S
Skinny Client Control Protocol (SCCP) 593
SLA (Shared Line Appearances) 311
конфигурация 312, 321
ограничения 322
пример общего внутреннего номера 320

пример системы клавиш с аналоговыми транками 312
пример системы клавиш с SIP-транками 315
пример альтернативы 317
extensions.conf 316
sla.conf 315
установка приложений 311
s
sla.conf 313, 315, 320
S
SMDI (Simplified Message Desk Interface) 168
SNMP (Simple Network Management Protocol) 531
мониторинг с помощью OpenNMS 534
настройка для OpenNMS 532
установка 532
res_snmp.conf 532
snmpd.conf 533
установка 531
s
snmpd.conf 533
S
SONET (Synchronous Optical Network) 584
s
spandsp 427
S
Speex 595
SPIT (Spam over Internet Telephony)
сведения 255
s
spool 27
S
SQL 516
диалплан 353
инъекция SQL 344
SQLite
cdr_sqlite и cdr_sqlite3_custom 517
cel_sqlite3_custom 526
SRTP, VoIP 602
SS7 (Signaling System 7) 586
StarPy, пример инициирования вызова 456
Stream Control Transmission Protocol (SCTP) 597
Subversion 43
s
sudo 38
S
Synchronous Optical Network (SONET) 584
s
syslog
cdr_syslog 518
logger.conf 509
T
T.38
обработка исходящих факсов 433
сквозной факс 437
TCP (Transmission Control Protocol)
сведения 596
AMI через TCP 440
FastAGI 466
TIFF, факс в 430
Tip and Ring 573
U
Ubuntu
голосовая почта IMAP 401
компиляция календаря 390
компиляция LDAP 420
комплияция XMPP 407
установка MySQL 334
установка PostgreSQL 334
установка SNMP 531
Festival 423
Ubuntu сервер 39
включение NTP 41
обновление базовой системы 41
установка базовой системы 39
musiconhold.conf 71
UDP (User Datagram Protocol) 597
UNISTIM 593
URI
приложение Dial() 109
SIP 238
вызов 245
вызовы 239
сведения 238
SRV записи 238
User Datagram Protocol (UDP) 597
v
voicemail.conf
голосовая почта ODBC 373
Comedian Mail 158
IMAP 404
V
VoIP (Voice over IP) 128, 587
безопасность
разделение голосового и трафика данных 603
спуфинг 602
укрепление сервера 603
физическая безопасность 603
шифрование 603
DMZ 603
SPIT 602
SRTP 602
в VoIP 136
кодеки 593
G.711 594
G.722 596
G.726 594

G.729A 595
GSM 595
iLBC 595
MP3 596
Speex 595
пейджинговые адаптеры 223
провайдеры 433
протоколы
сведения 588
H.323 591
IAX 588
MGCP 593
SCCP 593
SIP 589
UNISTIM 593
регламент 568
спам 568
транки 137
IAX-транки между Asterisk-системами 142
SIP-транки между Астериск-системами 137
ТфОП
инициирование135
терминация 134
что это? 2
эхо 598
Asterisk 600
NAT 128
устройства 129
Asterisk 131
RTP 132
QoS596
дифференцированное обслуживание597
негарантированная доставка 598
MPLS 598
RSVP 598
SCTP 597
TCP 596
UDP 597
VoIP спам 568
W
WAN, распространение состояний устройств 485
w
web-интерфейсы 537
состояние очереди и отчеты 538
A2Billing 539
CDR 539
FOP 537
W
WebRTC, будущее 565
w
wget 44
W
Wi-Fi, будущее 566
w
wiki-сайт, Сообщество Asterisk 6
W
WiMAX, будущее 566
X
XMPP (eXtensible Messaging and Presence Protocol) 308, 406
диалплан 407
подключение к XMPP серверу 407
приложение JabberSend() 409
функция диалплана JABBER_RECEIVE() 410
компиляция 407
конфигурация Asterisk 309
тестирование 310
установка 308
учетные записи 309
chan_motif 412
Google Talk 413p.
Google Voice 414p.
motif.conf 413
x
xmpp.conf 415

Об авторах

Рассел Брайант — главный инженер-программист в Red Hat, где он работает над проектом OpenStack. Начиная с 2004 года Рассел провел семь лет, работая на Digium в проекте Asterisk. Роль Рассела в Digium началась как разработчика программного обеспечения и в заключение, будучи лидером проекта Asterisk и инженерным менеджером, команда сосредоточилась на разработке Asterisk.

Лейф Мэдсен впервые связался с сообществом Asterisk когда искал решение для голосовых конференций. Однажды он узнал, что официального документа Asterisk нет-он был соучредителем проекта Asterisk Documentation. Лейф в настоящее время работает в Thinking Phone Networks, возглавляя команду Unified communications backend. Вы можете узнать больше о нем по адресу *http://www.leifmadsen.com*.

Джим Ван Меггелен — президент и технический директор Core Telecom Innovations, канадской компании поставщика решений для телефонии с открытым исходным кодом. Он имеет 15-летний опыт работы предпринимательства в сфере телекоммуникаций в таких компаниях как Nortel, Williams и Telus, и имеет обширные знания как устаревшего, так и VoIP-оборудования от таких производителей, как Nortel, Cisco и Avaya. Джим является одним из основных участников проекта документации Asterisk и является соавтором бестселлера О'Рейли *Asterisk: Будущее телефонии*.

Концовка

Животное на обложке *Asterisk: Окончательное руководство* — Морская звезда (*Asteroidea*), группа иглокожих (иглокожие беспозвоночные, встречающиеся только в море). Большинство морских звезд имеют пятикратную радиальную симметрию (ветви или лучи, ответвляющиеся от диска центрального тела кратны пяти), хотя некоторые виды имеют четыре или девять ветвей. Существует более 1500 видов морских звезд.

Морские звезды живут на дне моря и в приливных бассейнах, цепляясь за скалы и двигаясь (медленно), используя сосудистую систему на водной основе, чтобы манипулировать сотнями крошечных трубчатых ног, называемых *подиями*. Маленькая луковица или *ампула* в верхней части трубки сжимается, выталкивая воду и расширяя ногу морской звезды. Ампула расслабляется и нога втягивается. На кончике каждой ноги находится присоска, которая позволяет морской звезде вскрывать раковины моллюсков, устриц или мидий. Морские звезды — плотоядные животные; они едят кораллы, рыбу и улиток, а также двустворчатых моллюсков.

Морские звезды могут сгибать и манипулировать своими руками, чтобы помещаться в небольших местах. В конце каждой руки находится глазное яблоко — примитивный датчик, который обнаруживает свет и помогает морской звезде определять направление. Морские звезды также обладают способностью регенерировать недостающую конечность. Некоторые виды могут даже вырастить полную, новую морскую звезду из отрубленной руки.

Изображение обложки взято из Дуврского живописного архива. Про шрифты не будем.