Asterisk

Stefan Wintermeyer

Asterisk

Stefan Wintermeyer

Kapitel 1. Howto zu diesem Buch

Versionsgeschichte
Version Rechtschreiblektorat 21.12.2006

Erst einmal vielen Dank, dass Sie dieses Buch gekauft haben! Dadurch wird das ganze Projekt erst ermöglicht. Dieses Kapitel gibt Ihnen eine grundlegende Einstiegshilfe zu diesem Buch.

1. Was überhaupt ist Asterisk?

Asterisk ist eine Telefonanlage. Diese wird allerdings nicht - wie bei klassischen Telefonanlagen - in einem Gehäuse geliefert, sondern ist als Software im Internet downloadbar. Diese Software muss installiert und danach konfiguriert werden.

Asterisk kann sowohl Analog-, ISDN- und natürlich auch VoIP-Telefonie verarbeiten und sogar miteinander verbinden. So kann ein Mitarbeiter im VoIP angebundenen Homeoffice genauso über den Firmen-ISDN-Anschluss telefonieren, wie jemand, der in der Firma sitzt. Die Möglichkeiten von Asterisk sind fast grenzenlos und im Vergleich zu klassischen Telefonanlagen kann man oft viel Geld sparen. Allerdings ist der Einstieg in Asterisk auch schwierig und der Alltag eines Asterisk-Administrators auch nicht einfach.

1.1. Unterschied zwischen klassischer Telefonanlage und Asterisk

Um den Unterschied zwischen der uns allen bekannten klassischen Telefonanlage und einer Telefonanlagensoftware wie Asterisk zu erklären, bediene ich mich eines Bildes aus unserer Kindheit. Eine klassische Telefonanlage ist vergleichbar mit einem *Playmobil*-Spielzeug:

Asterisk hingegegen kann man eher vergleichen mit einem *Lego*-Spielzeug:

Beide Spielzeugvariante haben ihre Vor- und Nachteile. Und beide haben ihre Fans.

Die Hauptvorteile von Playmobil (also der klassischen Telefonanlage):

- Auspacken und fertig! Man kann direkt loslegen.
- Keine oder nur eine minimale Bauanleitung ist notwendig.

Die Hauptvorteile von Lego (also Asterisk):

- Ich kann selber neue Lösungen generieren. Dazu benötige ich nur eine neue Bauanleitung.
- Mehrere Baukästen sind zu einem großen Baukasten kombinierbar. Ich kann zwei Lego-Flugzeuge kaufen und daraus ein großes Flugzeug bauen. Wenn ich zwei Playmobil-Flugzeuge kaufe, habe ich zwei Flugzeuge. Man kann jedoch aus zwei Playmobil-Flugzeugen kein neues großes Playmobil-Flugzeug bauen.

Der Hauptnachteil von Lego (also Asterisk):

- Nach dem Auspacken hat man erst mal nur Einzelteile. Das Zusammenbauen erfordert Geduld und etwas Geschick. Es gibt kein schnelles Erfolgserlebnis.

Genauso wie es Playmobil- und Legofans gibt, die von dem jeweilig anderen Spielzeug nichts wissen wollen, so gibt es Fans der klassischen Telefonanlage und von Asterisk.

2. Welche Kapitel soll ich lesen und womit anfangen?

Nicht jeder muss bei Adam und Eva anfangen und auf der anderen Seite will auch nicht jeder zum absoluten Asterisk-Profi werden. Dieses Buch ist bewusst sowohl für Anfänger als auch für Fortgeschrittene geschrieben. Allerdings sollten gerade Anfänger sich an Abschnitt 2.1, „Absoluter Asterisk-Anfänger“ halten.

2.1. Absoluter Asterisk-Anfänger

Für den absoluten Asterisk-Anfänger ist es am einfachsten, systematisch das Buch durchzulesen, da alle Kapitel aufeinander aufbauen. Sie können also einfach bei Kapitel 2, *Installation und "Hello World"* anfangen und das Buch bis zum Ende durchlesen. Ich empfehle auf jeden Fall Kapitel 2, *Installation und "Hello World"*, Kapitel 3, *Die große Telefonanlage* und Kapitel 4, *Programmieren im Dialplan* durchzuarbeiten. Danach sind Sie schon in der Lage, komplexere Telefonanlagen mit Asterisk aufzusetzen. Mit Kapiteln wie Kapitel 7, *Asterisk-Datenbank (AstDB)* können Sie sich dann bei entsprechend konkretem Bedarf beschäftigen.

2.1.1. Die Anfänger-FAQ

Im Laufe des Beta-Testes zu diesem Buch sind einige Anfängerfragen aufgetaucht, die sich nur schwer in das Buch einsortieren ließen. Deshalb gibt es an dieser Stelle eine diesbezügliche FAQ.

2.1.1.1. Welchen "guten" VoIP-Anbieter gibt es? Auf was sollte man achten?

Einen bestimmten VoIP-Provider an dieser Stelle zu empfehlen, würde man immer als Schleichwerbung empfinden. Der Asterisk-Anfänger sollte einfach irgendeinen großen SIP-Provider nehmen, der kostenlose Accounts anbietet. Damit kann man dann rumspielen. Im Buch finden Sie Gutscheine für Gratisguthaben von VoIP-Providern. Das ist ein guter Start. Welchen Provider Sie langfristig nehmen sollten, hängt von vielen Faktoren ab. Hier nur einige:

- Bietet Ihr Provider nur SIP oder auch IAX an? Benötigen Sie IAX?
- Wie gut ist der Support des Providers?
- Wie sind die Preise?

2.1.1.2. Welche "normalen" Endgeräte gibt es und was kosten diese?

Es gibt in der VoIP-Telefonie kein "normales" Endgerät. Sie haben folgende Möglichkeiten, ein Telefon mit Asterisk zu betreiben:

- Ein SIP-Telefon (die einfachste Möglichkeit)
- Ein analoges Telefon an einem ATA Gateway (das ist ein Analog to VoIP-Adapter)
- Ein ISDN-Telefon an einem internen ISDN-Bus. Das ist meistens teurer als ein SIP-Telefon, ohne mehr zu bieten. Empfehlung: SIP-Telefon
- Exotische Konstrukte. Man kann z.B. einen AB-Wandler an einen internen ISDN-Bus hängen und an diesen AB-Wandler dann ein analoges Telefon. Sollte man nicht gerade die Hardware dafür herumliegen haben, ist von dieser Konstellation abzuraten.

Zu den Preisen kann ich keine Auskunft geben. Dies wäre bei einem Buch auch wenig sinnvoll, da sich gerade bei den VoIP-Telefonen die Preise monatlich ändern.

2.1.1.3. Was ist mit Fax?

Lesen Sie dazu Kapitel 11, *Asterisk als Faxserver*

2.2. Leser mit etwas Asterisk-Erfahrung

Es ist schwierig bei Lesern mit etwas Asterisk-Erfahrung vorherzusehen, was bereits bekannt ist und was nicht. Selbst bei Dingen, die man sehr gut beherrscht, finden sich oft einzelne Aspekte, auf die man bisher nicht gestoßen ist. Falls Sie bereits mit Asterisk praktische Erfahrungen gesammelt haben, können Sie Kapitel 2, *Installation und "Hello World"* überspringen oder einfach querlesen. Ich persönlich kann nur empfehlen, es auch in diesem Fall zu lesen - es lohnt sich fast immer.

So richtig "spannend" wird es für diese Gruppe ab Kapitel 7, *Asterisk-Datenbank (AstDB)* (das sollte man sich nicht entgehen lassen). Danach empfehle ich, einfach mal die verschiedenen Kapitel durchzublättern. Beim Schreiben des Buches wurde auf viele Praxisbeispiele Wert gelegt. Allein das Durchlesen dieser Beispiele bringt den Leser häufig schon auf neue Ideen.

2.3. Der Asterisk-Profi

Sie können Kapitel 2, *Installation und "Hello World"* übergehen und einfach auf Entdeckungstour gehen. Sie werden viele Beispiele finden, die Sie direkt in der Praxis übernehmen können (z.B. Abschnitt 6, „Anwendungsbeispiel CallForwarding“). Vielleicht bringt Sie das eine oder andere Beispiel aus der Praxis auch auf neue Ideen.

Darüber hinaus wird das Buch auch als Nachschlagewerk gute Dienste leisten. Alle wichtigen Konfigurationsparameter werden erklärt, meist mit Beispiel und das dazugehörige Kapitel hilft, sich eingehender mit einzelnen Aspekten zu beschäftigen.

3. Updates zum Buch

Auf der Webseite *http://www.das-asterisk-buch.de* steht immer die aktuelle Version des Buches. Dort finden Sie auch Updates.

4. Mithilfe und Feedback

Dieses Buch ist kein abgeschlossener Text, es soll "lebendig" bleiben und ständig verbessert werden. Aus diesem Grund steht der gesamte Text unter der GNU Free Documentation License (siehe Appendix A, *GNU Free Documentation License*). Daher finden Sie auf der Webseite *http://www.das-asterisk-buch.de* stets die aktuelle Version des Buches mit dem DocBook-Quellcode.

Wenn Sie Verbesserungsvorschläge haben, dann schreiben Sie an `<stefan.wintermeyer@amooma.de>` eine E-Mail mit dem entsprechendem Feedback. Falls Sie bereits ausformulierte Veränderungsvorschläge haben, fügen Sie den entsprechenden Textabschnitt bitte als ASCII-Text oder direkt im *DocBook*-Format der E-Mail bei.

Schon an dieser Stelle möchte ich mich für Ihre Hilfe bei diesem Projekt bedanken! :-)

4.1. FAQ zum Beta-Test

In regelmäßigen Abständen findet ein allgemeiner Beta-Test der nächsten Buchversion statt. Dieser Beta-Test wird auf der *http://www.das-asterisk-buch.de* Homepage bekannt gegeben.

4.1.1. Was soll gemeldet werden?

4.1.1.1. Sollen Rechtschreib- und Zeichensetzungsfehler gemeldet werden?

> Da der Text noch nicht durch das Rechtschreib- und Grammatiklektorat gegangen ist, gibt es hunderte Rechtschreibfehler. Dessen sind wir uns bewusst. Sollte ein Fehler in diese Kategorie fallen, so braucht er nicht gemeldet zu werden (da eine entsprechende Kontrolle ganz zum Schluss noch einmal gemacht wird). Allerdings gibt es auch Fehler, die technischer Natur sind: Ist z.B. ein Befehl in einem Programm oder auch im Text falsch geschrieben, so benötigen wir diese Info und sind dankbar für Ihre Hilfe. Solche Fehler sollten also gemeldet werden.

4.1.1.2. Ich habe etwas aus dem Buch ausprobiert. Es funktioniert aber nicht wie abgedruckt. Soll ich das melden?

Ja, unbedingt! Bitte mit Angabe der URL und einer evt. Fehlermeldung.

4.1.1.3. Wie kann ich einen Fehler melden?

Am einfachsten per E-Mail an den Autor Stefan Wintermeyer `<stefan.wintermeyer@amooma.de.>`

4.1.2. Fragen zur Technik

4.1.2.1. In welchem Format wird das Buch geschrieben?

Der "Quell"text des Buches wird im DocBook-Format erstellt. Informationen zu DocBook finden Sie auf *http://www.docbook.org*.

4.1.2.2. Warum gibt es kein Wiki, in dem ich einfach Fehler ändern kann?

Das Buch muss zu einem bestimmten Zeitpunkt gedruckt werden. Dazu benötigt der Setzer einen strukturierten Text. Die Arbeit des Setzers ist mit DocBook am einfachsten. Außerdem zwingt DocBook den Autor eines Texts zu klaren Stukturen.

Wenn der ganze Text in einem Wiki stehen würde, dann würde sich der Export aus diesem Wiki sehr schwierig gestalten. Mediawiki hat einfach keine für eine technische Dokumentation sinnvollen Tags.

Es kann sein, dass wir als Kompromiss in Zukunft *Annotate* von Dominik Brettnacher einsetzen werden. Aber erst mal muss der Beta-Test abgeschlossen und das Buch gedruckt sein.

4.1.2.3. Mit welchem Editor kann man am besten DocBook schreiben?

Das ist eine religiöse Frage. ;-) Für die einen ist es emacs und für die anderen vi. Der Autor benutzt meistens den XMLmind XML Editor, der sich von *http://www.xmlmind.com/xmleditor/* herunterladen lässt.

4.1.2.4. Kann ich auch die Original-DocBook-Dateien bekommen?

Ja, sollten Sie in einem Kapitel sehr viel verändern wollen, so können Sie dafür gerne die Original-Dateien erhalten. Schreiben Sie dazu bitte eine E-Mail an den Autor Stefan Wintermeyer `<stefan.wintermeyer@amooma.de>`.

Wir planen langfristig die Quellen in einem öffentliches CVS oder SVN-System zu veröffentlichen. Bis dahin ist der Autor ein menschliches CVS. ;-)

4.1.2.5. Warum sind die Webseiten so schrecklich formatiert?

Das ist zur Zeit des Beta-Testes ein Standard DocBook to HTML Export. Wir arbeiten an einem besseren CSS. Falls jemand einen entsprechenden Vorschlag hat, soll er sich bitte melden. Ansonsten kann man sich natürlich auch das schön formatierte Buch kaufen.

4.1.2.6. Was ist mit dem Thema XYZ? Das habe ich noch nicht gefunden und es ist doch so wichtig.

Bei ganz offensichtlichen Lücken können Sie davon ausgehen, dass das entsprechende Kapitel schon existiert, aber noch nicht am Beta-Test teilnimmt. Bei exotischen Themen schreiben Sie uns bitte eine E-Mail. Falls Sie zu einem solchen Thema selber ein Kapitel schreiben wollen, so ist das natürlich jederzeit gern gesehen.

4.1.3. Allgemeine Fragen

4.1.3.1. Gibt es das Buch auch als PDF?

Noch nicht. Wir arbeiten dran.

4.1.3.2. Mir fehlt bei den Installationsbeschreibungen mein Lieblings-Betriebssystem. Was kann man da machen?

Idealerweise schreiben Sie das entsprechende Kapitel und schicken es an Stefan Wintermeyer <stefan.wintermeyer@amooma.de>. Falls Sie nicht genügend Know-how dazu besitzen, schreiben Sie bitte eine entsprechende Anfrage.

4.1.3.3. Werden Beta-Tester irgendwo aufgelistet?

Jeder, der beim Beta-Test mitmacht und nützliches Feedback gibt, wird in Abschnitt 3, „Helfer beim Beta-Test“ aufgeführt. Falls dies nicht gewünscht wird, bitte dies beim Einreichen eines Bugs mitteilen.

5. Hilfe, ich habe immer noch ein Problem mit Asterisk!

Sie haben ein Problem und finden in diesem Buch keine entsprechende Antwort? Dann gibt es folgende Möglichkeiten:

- *http://www.voip-info.org*

  Das ist eine Webseite, die sich dem Thema VoIP und Asterisk widmet. Dort findet man bei vielen Problemen eine passende Antwort.

- IRC Channel

  Auf dem IRC-Server *irc.freenode.net* gibt es den Channel `#asterisk`, der in englischer Sprache geführt wird. Dort kann man fast immer schnell Hilfe erhalten, selbst bei komplizierten Problemen.

  Wenn Sie noch nie mit IRC gearbeitet haben, finden Sie auf *http://de.wikipedia.org/wiki/Internet_Relay_Chat* eine Beschreibung.

- Mailinglisten

  Die englischen Digium-Mailinglisten sind nicht jedermanns Sache, da gerade der ungeübte Mailinglistenbenutzer schnell den Überblick verliert. Allerdings hat man hier den Vorteil, dass immer auch Digium-Mitarbeiter mitlesen und Fragen beantworten. Informationen zu diesen Mailinglisten finden Sie unter *http://www.asterisk.org/support/* und *http://lists.digium.com/*.

- E-Mail an Stefan Wintermeyer

  Wenn alle Stricke reißen oder Sie Verständnissprobleme bei den englischen Beschreibungen haben, können Sie mir natürlich auch jederzeit eine E-Mail an <stefan.wintermeyer@amooma.de> schreiben. Bitte geben Sie mir ein paar Tage Zeit zum Beantworten.

Wichtig: Bitte geben Sie mir Feedback! Wenn es sich bei Ihrer Frage um ein allgemeines Problem handelt und es in diesem Buch keine ausreichende Erklärung dazu gibt, dann schreiben Sie mir bitte an <stefan.wintermeyer@amooma.de> eine E-Mail.

Kapitel 2. Installation und "Hello World"

1. Einleitung

Ein nicht immer ganz zu Unrecht bestehendes Vorurteil gegenüber Asterisk ist die Behauptung, dass man mindestens 2 bis 3 Tage Webseiten und Dokumentation studieren muss, um überhaupt auch nur irgendetwas Lauffähiges zu Stande zu bekommen. Diese Hürde ist für viele Interessierte zu hoch oder es ist die Mühe und Zeit nicht wert. Wer zu denen gehört, die sich nicht lange mit einem theoretischen Unterbau beschäftigen möchten, sondern möglichst schnell ein Ergebnis sehen wollen, ist in diesem Kapitel genau richtig aufgehoben. In 30 Minuten haben Sie Ihre erste funktionierende Asterisk-Telefonanlage. Versprochen!

Bei Programmierbüchern gibt es ganz am Anfang meistens ein „Hello World“-Beispiel. Die Aufgabe des „Hello World“-Programmes ist die Ausgabe des Strings „Hello World“ auf dem Bildschirm. Es ist quasi ein Standardbeispiel, um die grundlegende Form und Syntax einer Programmiersprache aufzuzeigen und ein schnelles erstes Erfolgserlebnis zu vermitteln. Dieses Kapitel versucht, diese Praxis auf das Thema Telefonanlagen mit Asterisk zu übertragen.

2. Eine Minimal-Telefonanlage

Wie sieht die einfachst mögliche Telefonanlage aus? Zwei Telefone und eine Black-Box, die diese beiden Telefone sinnvoll miteinander verbindet. Die Black-Box ist ein gewöhnlicher PC, auf dem wir die Asterisk-Software installieren und der von nun an unsere Telefonanlage sein wird. Die beiden Telefone sind so genannte „Softphones“, also Telefone in Software, die wir ebenfalls auf einem PC ausführen. Somit genügt für das folgende Beispiel ein einfacher PC, auf dem Sie eine aktuelle Linux-Distribution installieren, als Grundlage für unseren *Kick-Start*. [1]

2.1. Die Aufgabe

- Einen frisch mit Linux installierten PC zu einer Mini-Telefonanlage umfunktionieren
- Zwei VoIP-Telefone einrichten und die Nummern 2000 und 2001 zuweisen
- Vom Telefon 2000 das Telefon 2001 anrufen und umgekehrt

2.2. Voraussetzungen

Sie benötigen für die Installation von Asterisk einen halbwegs aktuellen PC mit genügend Speicher. Es geht auch mit älterer Hardware, allerdings kann ich dann mein 30-Minuten Versprechen nicht einlösen und es macht auch weniger Spaß, damit zu arbeiten. Konkret sollte es mindestens ein 500 MHz Pentium-System mit 256 MByte Arbeitsspeicher (RAM) und einer 10 GByte Festplatte sein. Asterisk läuft auf allen gängigen Linux-Distributionen. Diese können Sie entweder im Geschäft kaufen oder aus dem Internet herunterladen.[2]

Bitte achten Sie darauf, dass die Linux-Distribution aktuell ist, und verwenden Sie nicht eine 2 Jahre alte Heft-CD, die noch irgendwo im Schrank liegt.

[1] Unser „Hello World“-Beispiel macht noch mehr Spaß, falls Sie über zwei oder mehr PCs (oder Laptops) verfügen, die Sie in einem gemeinsamen Netzwerk verbunden haben. Dann können Sie einen Rechner als Asterisk-Telefonanlage verwenden und die anderen PCs als Softphones.
[2] Ein paar URLs zu im Internet frei verfügbaren Linux-Distributionen:

- Debian: *http://www.debian.org*
- SuSE Linux (Open SuSE): *http://www.opensuse.org*
- Fedora (Redhat): *http://www.fedora.org*

Die einzelnen Installations- und Konfigurationsanweisungen sind in diesem Buch exemplarisch für ein Linux-System auf Basis einer Debian-Distribution[3] beschrieben. Weiterhin werden die entsprechenden Anweisungen auf der Kommandozeile einer Shell eingegeben.[4]

2.3. Welche Asterisk-Version ist empfehlenswert?

Für 99 Prozent aller Asterisk-Installationen wird es völlig egal sein, ob man ein Asterisk 1.2.x oder ein Asterisk 1.4.x verwendet. Deshalb empfehle ich bis Mitte 2007 nur die Version 1.2.x und danach die Version 1.4.x einzusetzen. Gerade bei solchen wichtigen Applikationen sollte immer die Stabilität das wichtigste Entscheidungskriterium sein, und die Version 1.4 wird bis Mitte 2007 brauchen, um eine ähnliche Stabiliät wie Version 1.2 zu erhalten.

2.4. Welche Linux-Distribution ist die richtige für einen Asterisk-Server?

Die Diskussion über die ideale Linux-Distribution bekommt immer sehr schnell technisch-religiöse Züge. In diesem Buch haben wir uns für ein Debian Linux (stable) entschieden. Trotzdem haben wir gerade in diesem Kapitel sehr viel Platz für Installationsanleitungen auf anderen Linux-Distributionen und anderen Betriebssystemen reserviert. Bei späteren Kapiteln (z.B. ISDN) kann diese Vielfalt nicht mehr gewährleistet werden. Zum Teil gibt es für andere Betriebssysteme keine passenden Treiber oder die Installation ist so kompliziert, dass der Rahmen dieses Buches völlig gesprengt werden würde. Aus diesem Grund gehen wir, wenn nichts anders gesagt wird, immer von einem Debian Linux aus.

2.5. Warum keine fertigen Asterisk-Pakete mit rpm oder apt-get installieren?

Es gibt einen ganz einfachen Grund, warum an dieser Stelle im Buch bei den Installationsanleitungen für Asterisk nicht auf die in den Distributionen bereits enthaltenen Asterisk-Pakete zurückgegriffen wird: Und der heißt: Aktualität!

Keine Standard-Linux-Distribution hat auch nur halbwegs aktuelle Asterisk-Pakete in ihren Stable-Zweigen. Da Asterisk ein sehr stark wachsendes Softwareprojekt ist, ist es aber auch nicht sinnvoll, eine 1.0-er Version zu installieren, wenn bereits eine 1.2-er lange auf dem Markt und einfach viel besser ist.

Anmerkung

Da von Distributionen gebaute Pakete enorme Vorteile haben und ein Update-Zyklus einfacher zu realisieren ist, hoffe ich darauf, dass dieses Kapitel irgendwann einmal auf fertige Standardpakete aus den Stable-Versionen zugreifen kann. Das liegt aber noch in der Zukunft.

Als Lohn für die Arbeit mit dem Übersetzen der Quellpakete winkt Ihnen das Wissen, Ihr Asterisk-System auf dem neuesten Stand zu haben, ohne dass Sie dafür auf aktuellere Pakete für Ihre Distribution warten müssen.

2.6. Asterisk 1.2

2.6.1. Installation Asterisk 1.2.x auf Debian Linux

Diese Installationsanleitung setzt ein frisch installiertes Debian GNU/Linux 3.1 (a.k.a sarge) voraus. Ein ISO-Image für eine entsprechende i386er Debian Linux-Installation befindet sich unter *http://cdimage.debian.org/debian-cd/3.1_r4/i386/iso-cd/debian-31r4-i386-netinst.iso*. Eine Debian GNU/Linux-Installationsanleitung steht unter *http://www.debian.org/releases/stable/i386/* und ein allgemeines Anwenderhandbuch unter *http://debiananwenderhandbuch.de*.

[3] Die aktuelle Debian Stable-Version.

[4] Falls Sie mit einer grafischen Benutzeroberfläche des X-Window Systems arbeiten, benötigen Sie eine Textkonsole für die Eingabe, beispielweise "xterm" oder "konsole".

Nach erfolgreicher Installation des Debian Linux-Systems melden Sie sich als Benutzer root in das System an und führen Sie die hier nachfolgend angegebenen Befehle aus. Bitte achten Sie darauf, keinen Schritt auszulassen und die Schritte als Benutzer root durchzuführen.

Als Erstes stellen Sie mit einem **apt-get update** sicher, dass `apt-get` alle aktuellen Paketlisten zur Verfügung hat:

```
debian:~# apt-get update
OK   http://ftp.de.debian.org stable/main Packages
OK   http://ftp.de.debian.org stable/main Release
OK   http://security.debian.org stable/updates/main Packages
OK   http://security.debian.org stable/updates/main Release
OK   http://security.debian.org stable/updates/contrib Packages
OK   http://security.debian.org stable/updates/contrib Release
OK   http://ftp.de.debian.org stable/main Sources
OK   http://ftp.de.debian.org stable/main Release
Paketlisten werden gelesen... Fertig
```

Um sicherzugehen, dass alle Pakete auf dem neuesten Stand sind, führen Sie zur Sicherheit noch ein **apt-get -y upgrade** aus:

```
debian:~# apt-get -y upgrade
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
```

Jetzt müssen Sie mit **apt-get install openssl zlibc libkrb5-dev libncurses5-dev libssl-dev zlib1g-dev** noch ein paar Pakete nachinstallieren, damit das Kompilieren von Asterisk erfolgreich durchgeführt werden kann:

```
debian:~# apt-get -y install openssl zlibc libkrb5-dev libncurses5-dev
libssl-dev zlib1g-dev
```

Auf der Asterisk-Homepage *http://www.asterisk.org/* finden Sie die notwendigen Quellen, um Asterisk zu kompilieren. Nehmen Sie bitte eine **stabile** und keine Entwickler-Version und laden Sie diese mit **wget http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz** in das Verzeichnis `/usr/src/` herunter:

```
debian:~# cd /usr/src
debian:/usr/src# wget http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
--18:30:46--  http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
           => `asterisk-1.2-current.tar.gz'
Auflösen des Hostnamen »ftp.digium.com«.... 69.16.138.164
Verbindungsaufbau zu ftp.digium.com[69.16.138.164]:80... verbunden.
HTTP Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 10,584,113 [application/x-gzip]

100%[====================================>] 10,584,113   224.71K/s    ETA 00:00

18:31:36 (207.67 KB/s) - »asterisk-1.2-current.tar.gz« gespeichert [10584113/10584113]
```

Die komprimierte Datei muss jetzt mit **tar xvzf asterisk-1.2-current.tar.gz** entpackt werden:

```
debian:/usr/src# tar xvzf asterisk-1.2-current.tar.gz
asterisk-1.2.13/
asterisk-1.2.13/build_tools/
asterisk-1.2.13/build_tools/make_svn_branch_name
asterisk-1.2.13/build_tools/mkdep
asterisk-1.2.13/build_tools/make_build_h
asterisk-1.2.13/build_tools/make_version_h
asterisk-1.2.13/build_tools/make_defaults_h
asterisk-1.2.13/aesopt.h
asterisk-1.2.13/utils.c
asterisk-1.2.13/keys/
[...]
asterisk-1.2.13/cli.c
asterisk-1.2.13/io.c
asterisk-1.2.13/ulaw.c
asterisk-1.2.13/logger.c
debian:/usr/src#
```

Die entpackten Dateien müssen jetzt kompiliert werden.[5]Dies geschieht mit dem Befehl **make** im gerade entpackten Verzeichnis.

Warnung

Bitte achten Sie darauf, dass die hier in der Dokumentation benutzte Version 1.2.13 vielleicht zum Zeitpunkt Ihrer Installation nicht mehr aktuell ist. Entsprechend werden sich die Verzeichnisnamen ändern.

Übrigens ist dies jetzt ein guter Moment, sich einen Kaffee oder Tee zu kochen. Der Kompiliervorgang kann je nach Rechenleistung des PCs wenige Minuten oder bis zu einer Stunde dauern.

```
debian:/usr/src# cd asterisk-1.2.13
debian:/usr/src/asterisk-1.2.13# make
if cmp -s .cleancount .lastclean ; then echo ; else \
        make clean; cp -f .cleancount .lastclean;\

[...]

make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/usr/src/asterisk-1.2.13/stdtime'
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, but +
 + cannot be run before being installed by   +
 + running:                                  +
 +                                           +
 +               make install                +
 +-------------------------------------------+
```

Die Software ist jetzt kompiliert und muss nur noch mit **make install** installiert (sprich an die vorgesehenen Stellen im Dateisystem kopiert) werden:

```
debian:/usr/src/asterisk-1.2.13# make install
build_tools/make_version_h > include/asterisk/version.h.tmp
if cmp -s include/asterisk/version.h.tmp include/asterisk/version.h ;
then echo;

[...]

 +---- Asterisk Installation Complete -------+
 +                                           +
 +    YOU MUST READ THE SECURITY DOCUMENT    +
 +                                           +
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
 +                                           +
 +               make samples                +
 +                                           +
 +-----------------  or ---------------------+
 +                                           +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                           +
 +              make progdocs                +
 +                                           +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +-------------------------------------------+
```

Geschafft! Die soeben kompilierten Programmdateien sind jetzt auf dem System installiert. Allerdings finden wir im Asterisk-Konfigurationsverzeichnis `/etc/asterisk/` noch gähnende Leere vor. In diesem Verzeichnis liegen die Konfigurationsdateien von Asterisk und da wir nicht von Null beginnen möchten, lassen wir uns die Standard-dateien erstellen. Dies erreichen wir mit **make samples**:

[5]Die Linux-Cracks unter den Lesern mögen mir diese starke Vereinfachung verzeihen.

```
debian:/usr/src/asterisk-1.2.13# make samples
build_tools/make_version_h > include/asterisk/version.h.tmp
if cmp -s include/asterisk/version.h.tmp include/asterisk/version.h ;
then echo;

[...]

done
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **asterisk -V** können Sie die installierte Version herausfinden (bitte achten Sie auf das großgeschriebene V):

```
debian:/usr/src/asterisk-1.2.13# asterisk -V
Asterisk 1.2.13
```

2.6.2. Installation Asterisk 1.2.x auf Ubuntu Linux

Da Ubuntu auch von der CD als Live-System gebootet werden kann, müssen Sie es nicht zwangsläufig auf der Festplatte installieren (gut zum Testen). Diese Anleitung wurde mit einem Ubuntu 6.06.1 LTS (Dapper Drake) ausgeführt bzw. setzt dieses voraus. Das entsprechende ISO-Image finden Sie unter *http://de.archive.ubuntu.com/ubuntu-releases/6.06/ubuntu-6.06.1-desktop-i386.iso*. Wir benutzen die Desktop-CD, der gleiche Installationsweg funktioniert aber auch bei der Server-CD. Hierbei sollte aber vor der Installation von Asterisk noch ein **sudo apt-get update** und ein **sudo apt-get -y upgrade** durchgeführt werden, um sicherzustellen, dass alle aktuellen Security Fixes installiert sind.

Weitere Informationen zu Ubuntu finden Sie auf *http://www.ubuntu.com*.

Tipp

Eine Liste von FTP-Mirrors zum Download des ISO-Images stehen auf *http://www.ubuntu.com/products/GetUbuntu/download#lts*.

Da auf einem Ubuntu-System root-Befehle immer über sudo (siehe http://de.wikipedia.org/wiki/Sudo) erfolgen, können Sie diese Installationsanleitung mit dem Benutzer ubuntu ausführen. Bitte öffnen Sie nach dem Starten von Ubuntu einfach ein Terminal-Fenster (im Menü oben links -> Applications -> Accessories -> Terminal).

Als Erstes stellen Sie mit einem **sudo apt-get update** sicher, dass `apt-get` alle aktuellen Paketlisten zur Verfügung hat:

```
ubuntu@ubuntu:~$ sudo apt-get update
Get:1 http://security.ubuntu.com dapper-security Release.gpg [191B]
Hit http://security.ubuntu.com dapper-security Release
Get:2 http://archive.ubuntu.com dapper Release.gpg [189B]
Get:3 http://archive.ubuntu.com dapper-updates Release.gpg [191B]
Hit http://security.ubuntu.com dapper-security/main Packages
Hit http://archive.ubuntu.com dapper Release
Hit http://security.ubuntu.com dapper-security/restricted Packages
Hit http://security.ubuntu.com dapper-security/main Sources
Hit http://archive.ubuntu.com dapper-updates Release
Hit http://security.ubuntu.com dapper-security/restricted Sources
Hit http://archive.ubuntu.com dapper/main Packages
Hit http://archive.ubuntu.com dapper/restricted Packages
Hit http://archive.ubuntu.com dapper/main Sources
Hit http://archive.ubuntu.com dapper/restricted Sources
Hit http://archive.ubuntu.com dapper-updates/main Packages
Hit http://archive.ubuntu.com dapper-updates/restricted Packages
Hit http://archive.ubuntu.com dapper-updates/main Sources
Hit http://archive.ubuntu.com dapper-updates/restricted Sources
Fetched 3B in 1s (2B/s)
Reading package lists... Done
ubuntu@ubuntu:~$
```

Jetzt müssen Sie mit **sudo apt-get -y install make gcc g++ libncurses5-dev libssl-dev zlib1g-dev** noch ein paar Pakete nachinstallieren, damit das Kompilieren von Asterisk erfolgreich durchgeführt werden kann:

```
uubuntu@ubuntu:~$ sudo apt-get -y install make gcc g++ libncurses5-dev libssl-dev zlib1g-dev
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  binutils cpp cpp-4.0 g++-4.0 gcc-4.0 libc6-dev libstdc++6-4.0-dev
  linux-kernel-headers
Suggested packages:
  binutils-doc cpp-doc gcc-4.0-locales gcc-4.0-doc lib64stdc++6 manpages-dev
  autoconf automake1.9 libtool flex bison gcc-doc libc6-dev-amd64 lib64gcc1
  glibc-doc libstdc++6-4.0-doc stl-manual
Recommended packages:
  libmudflap0-dev
The following NEW packages will be installed:
  binutils cpp cpp-4.0 g++ g++-4.0 gcc gcc-4.0 libc6-dev libncurses5-dev
  libssl-dev libstdc++6-4.0-dev linux-kernel-headers make zlib1g-dev
0 upgraded, 14 newly installed, 0 to remove and 0 not upgraded.
Need to get 15.6MB of archives.
After unpacking 58.9MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com dapper/main cpp-4.0 4.0.3-1ubuntu5 [1987kB]
Get:2 http://security.ubuntu.com dapper-security/main binutils 2.16.1cvs20060117-1ubuntu2.1 [1407kB]

[...]

Setting up g++-4.0 (4.0.3-1ubuntu5) ...
Setting up libstdc++6-4.0-dev (4.0.3-1ubuntu5) ...
Setting up g++ (4.0.3-1) ...

ubuntu@ubuntu:~$
```

Auf der Asterisk-Homepage *http://www.asterisk.org/* finden Sie die notwendigen Quellen, um Asterisk zu kompilieren. Nehmen Sie bitte eine stabile und keine Entwickler-Version und laden Sie diese mit **wget http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz** in das Verzeichnis `/usr/src/` herunter:

```
ubuntu@ubuntu:~$ cd /tmp
ubuntu@ubuntu:/tmp$ wget http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
--09:36:42--  http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
           => `asterisk-1.2-current.tar.gz'
Resolving ftp.digium.com... 69.16.138.164
Connecting to ftp.digium.com|69.16.138.164|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10,584,113 (10M) [application/x-gzip]

100%[====================================>] 10,584,113   109.65K/s    ETA 00:00

09:38:29 (97.21 KB/s) - `asterisk-1.2-current.tar.gz' saved [10584113/10584113]

ubuntu@ubuntu:/tmp$
```

Die komprimierte Datei muss jetzt mit **tar xvzf asterisk-1.2-current.tar.gz** entpackt werden:

```
ubuntu@ubuntu:/tmp$ tar xvzf asterisk-1.2-current.tar.gz
asterisk-1.2.13/
asterisk-1.2.13/build_tools/
asterisk-1.2.13/build_tools/make_svn_branch_name
asterisk-1.2.13/build_tools/mkdep
asterisk-1.2.13/build_tools/make_build_h
asterisk-1.2.13/build_tools/make_version_h
asterisk-1.2.13/build_tools/make_defaults_h
asterisk-1.2.13/aesopt.h
asterisk-1.2.13/utils.c
asterisk-1.2.13/keys/
[...]
asterisk-1.2.13/cli.c
asterisk-1.2.13/io.c
asterisk-1.2.13/ulaw.c
asterisk-1.2.13/logger.c
ubuntu@ubuntu:/tmp$
```

Die entpackten Dateien müssen jetzt kompiliert werden.[6]Dies geschieht mit dem Befehl **make** im gerade entpackten Verzeichnis.

Warnung

Bitte achten Sie darauf, dass die hier in der Dokumentation benutzte Version 1.2.13 vielleicht zum Zeitpunkt Ihrer Installation nicht mehr aktuell ist. Entsprechend werden sich die Verzeichnisnamen ändern.

Übrigens ist dies jetzt ein guter Moment, sich den obligatorischen Kaffee oder Tee zu kochen. Der Kompiliervorgang kann je nach Rechenleistung des PCs wenige Minuten oder bis zu einer Stunde dauern.

```
ubuntu@ubuntu:/tmp$ cd asterisk-1.2.13/
ubuntu@ubuntu:/tmp/asterisk-1.2.13$ make
if cmp -s .cleancount .lastclean ; then echo ; else \
               make clean; cp -f .cleancount .lastclean;\


[...]

make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/tmp/asterisk-1.2.13/stdtime'
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, but +
 + cannot be run before being installed by   +
 + running:                                  +
 +                                           +
 +              make install                 +
 +---------------------------------------------+
ubuntu@ubuntu:/tmp/asterisk-1.2.13$
```

Die Software ist jetzt kompiliert und muss nur noch mit **sudo make install** installiert (sprich in die richtigen Verzeichnisse kopiert) werden:

```
ubuntu@ubuntu:/tmp/asterisk-1.2.13$ sudo make install
build_tools/make_version_h > include/asterisk/version.h.tmp
if cmp -s include/asterisk/version.h.tmp include/asterisk/version.h ; then echo;

[...]

make[1]: Leaving directory `/tmp/asterisk-1.2.13/stdtime'
 +---- Asterisk Installation Complete -------+
 +                                           +
 +    YOU MUST READ THE SECURITY DOCUMENT    +
 +                                           +
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
 +                                           +
 +               make samples                +
 +                                           +
 +-----------------  or ---------------------+
 +                                           +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                           +
 +              make progdocs                +
 +                                           +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +-------------------------------------------+

ubuntu@ubuntu:/tmp/asterisk-1.2.13$
```

Die eben kompilierten Programmdateien sind jetzt auf dem System installiert. Allerdings finden wir im Asterisk-Konfigurationsverzeichnis `/etc/asterisk/` noch gähnende Leere vor. In diesem Verzeichnis liegen die Konfigu-

[6]Die Linux-Cracks unter den Lesern mögen mir diese starke Vereinfachung verzeihen.

rationsdateien von Asterisk und da wir nicht von Null beginnen möchten, lassen wir uns die Standarddateien erstellen. Dies erreichen wir mit einem **sudo make samples**:

```
ubuntu@ubuntu:/tmp/asterisk-1.2.13$ sudo make samples
build_tools/make_version_h > include/asterisk/version.h.tmp
if cmp -s include/asterisk/version.h.tmp include/asterisk/version.h ; then echo;

[...]

for x in vm-theperson digits/1 digits/2 digits/3 digits/4 vm-isonphone; do \
                cat /var/lib/asterisk/sounds/$x.gsm >> /var/spool/asterisk/voicemail/default/1234/busy.gs
        done
ubuntu@ubuntu:/tmp/asterisk-1.2.13$
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **sudo asterisk -V** können Sie die installierte Version herausfinden (bitte achten Sie auf das großgeschriebene V):

```
ubuntu@ubuntu:/tmp/asterisk-1.2.13$ sudo asterisk -V
Asterisk 1.2.13
ubuntu@ubuntu:/tmp/asterisk-1.2.13$
```

Warnung

Dadurch, dass Ubuntu normalen root-Zugriff nur über sudo zulässt, müssen Sie bei allen Beispielen in diesem Buch bei Aufrufen von Asterisk immer ein sudo davorschreiben. Wenn Sie als Benutzer ubuntu Asterisk starten, werden Sie eine Fehlermeldung bekommen, da der Benutzer ubuntu nicht in allen Verzeichnissen schreiben kann.

Der richtige Aufruf zum Starten von Asterisk muss also immer wie folgt aussehen: **sudo asterisk -c** oder **sudo asterisk**

2.6.3. Installation Asterisk 1.2.x auf OpenSUSE Linux

Diese Installationsanleitung setzt ein aktuelles OpenSUSE 10.2 aus dem stabilen Entwicklungszweig voraus. Falls Sie nicht bereits eine fertig installierte OpenSUSE-Distribution haben, können Sie sich die entsprechenden Installationsmedien auf folgender Seite herunterladen: *http://de.opensuse.org/Stabile_Version*.

Um Asterisk zu kompilieren, benötigen Sie einige zusätzliche Pakete. Diese können Sie mit dem SuSE Administrationstool YaST installieren. Bitte starten Sie dazu YaST.

Wählen Sie im Installationsfenster als Filter "Schemata" aus, dort finden Sie dann als installierbare Schemata unter "Entwicklung" die Einträge "Grundlegende Entwicklung", "C/C++ Entwicklung" und "Linux-Kernel-Entwicklung".

Wechseln Sie dann wieder auf "Suchen" als Filter und suchen Sie nach folgenden Stichworten: "openssl, krb5, ncurses, zlib". Installieren Sie jeweils das gleichnamige Paket, als auch das mit dem Zusatz "devel", also z.B. "openssl" und "openssl-devel". Nach erfolgreicher Einrichtung der Buildumgebung können dann im nächsten Schritt die Asterisk-Sourcen heruntergeladen und kompiliert werden.

Auf der Asterisk-Homepage *http://www.asterisk.org/* finden Sie die notwendigen Quellen, um Asterisk zu kompilieren. Nehmen Sie bitte eine stabile und keine Entwickler-Version und laden Sie diese mit **wget http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz** in das Verzeichnis `/usr/src/` herunter:

```
linux:~# cd /usr/src
linux:/usr/src# wget http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
--18:30:46--  http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
           => `asterisk-1.2-current.tar.gz'
Auflösen des Hostnamen »ftp.digium.com«.... 69.16.138.164
Verbindungsaufbau zu ftp.digium.com[69.16.138.164]:80... verbunden.
HTTP Anforderung gesendet, warte auf Antwort... 200 OK
```

```
Länge: 10,584,113 [application/x-gzip]

100%[====================================>] 10,584,113   224.71K/s    ETA 00:00

18:31:36 (207.67 KB/s) - »asterisk-1.2-current.tar.gz« gespeichert [10584113/10584113]
```

Die komprimierte Datei muss jetzt mit **tar xvzf asterisk-1.2-current.tar.gz** entpackt werden:

```
linux:/usr/src# tar xvzf asterisk-1.2-current.tar.gz
asterisk-1.2.13/
asterisk-1.2.13/build_tools/
asterisk-1.2.13/build_tools/make_svn_branch_name
asterisk-1.2.13/build_tools/mkdep
asterisk-1.2.13/build_tools/make_build_h
asterisk-1.2.13/build_tools/make_version_h
asterisk-1.2.13/build_tools/make_defaults_h
asterisk-1.2.13/aesopt.h
asterisk-1.2.13/utils.c
asterisk-1.2.13/keys/
[...]
asterisk-1.2.13/cli.c
asterisk-1.2.13/io.c
asterisk-1.2.13/ulaw.c
asterisk-1.2.13/logger.c
linux:/usr/src#
```

Die entpackten Dateien müssen jetzt kompiliert werden.[7]Dies geschieht mit dem Befehl **make** im gerade entpackten Verzeichnis.

Warnung

Bitte achten Sie darauf, dass die hier in der Dokumentation benutzte Version 1.2.13 vielleicht zum Zeitpunkt Ihrer Installation nicht mehr aktuell ist. Entsprechend werden sich die Verzeichnisnamen ändern.

Übrigens ist dies jetzt vielleicht ein guter Moment, sich einen Kaffee oder Tee zu kochen. Der Kompiliervorgang kann je nach Rechenleistung des PCs wenige Minuten oder bis zu einer Stunde dauern.

```
linux:/usr/src# cd asterisk-1.2.13
linux:/usr/src/asterisk-1.2.13# make
if cmp -s .cleancount .lastclean ; then echo ; else \
        make clean; cp -f .cleancount .lastclean;\

[...]

make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/usr/src/asterisk-1.2.13/stdtime'
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, but +
 + cannot be run before being installed by   +
 + running:                                  +
 +                                           +
 +               make install                +
 +-------------------------------------------+
```

Die Software ist jetzt kompiliert und muss nur noch mit **make install** installiert (sprich in die richtigen Verzeichnisse kopiert) werden:

```
linux:/usr/src/asterisk-1.2.13# make install
build_tools/make_version_h > include/asterisk/version.h.tmp
if cmp -s include/asterisk/version.h.tmp include/asterisk/version.h ;
then echo;

[...]

 +---- Asterisk Installation Complete -------+
 +                                           +
```

[7]Die Linux-Cracks unter den Lesern mögen mir diese starke Vereinfachung verzeihen.

```
+    YOU MUST READ THE SECURITY DOCUMENT    +
+                                           +
+ Asterisk has successfully been installed. +
+ If you would like to install the sample   +
+ configuration files (overwriting any      +
+ existing config files), run:              +
+                                           +
+               make samples                +
+                                           +
+-----------------  or ---------------------+
+                                           +
+ You can go ahead and install the asterisk +
+ program documentation now or later run:   +
+                                           +
+              make progdocs                +
+                                           +
+ **Note** This requires that you have      +
+ doxygen installed on your local system    +
+-------------------------------------------+
```

Die eben kompilierten Programmdateien sind jetzt auf dem System installiert. Allerdings finden wir im Asterisk-Konfigurationsverzeichnis `/etc/asterisk/` noch gähnende Leere vor. In diesem Verzeichnis liegen die Konfigurationsdateien von Asterisk und da wir nicht von Null beginnen möchten, lassen wir uns die Standarddateien erstellen. Dies erreichen wir mit einem **make samples**:

```
linux:/usr/src/asterisk-1.2.13# make samples
build_tools/make_version_h > include/asterisk/version.h.tmp
if cmp -s include/asterisk/version.h.tmp include/asterisk/version.h ;
then echo;

[...]

done
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **asterisk -V** können Sie die installierte Version herausfinden (bitte achten Sie darauf, dass es sich um ein großgeschriebenes V handelt):

```
linux:/usr/src/asterisk-1.2.13# asterisk -V
Asterisk 1.2.13
```

2.6.4. Installation Asterisk 1.2.x auf FreeBSD

Diese Installationsanleitung setzt ein frisch installiertes FreeBSD (aktuell ist Version 6.1) voraus. ISO-Images für eine entsprechende i386er FreeBSD-Installation finden Sie unter *ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/6.1/*. Eine Installationsanleitung befindet sich bei *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install.html*.

Nach erfolgreicher Installation melden Sie sich als Benutzer root am System an und führen Sie die hier nachfolgend angegebenen Befehle aus. Bitte achten Sie darauf, keinen Schritt auszulassen und die Schritte als Benutzer root durchzuführen.

Warnung

Da FreeBSD seit dem Release der Version 6.0 aktuelle Asterisk-Pakete beinhaltet hat, beschreibt diese Anleitung nicht die Installation aus den Asterisk-Quellen, sondern aus dem FreeBSD Portage-Tree.

Die Installation erfolgt dabei ganz einfach, einen aktuellen Portage-Tree vorausgesetzt, und zwar wie folgt:

Tipp

Es gibt verschiedene Möglichkeiten, den Portage-Tree aktuell zu halten. Eine Variante besteht in einem Supfile /etc/supfile-ports mit folgendem Inhalt:

```
*default host=cvsup6.de.FreeBSD.org
*default base=/usr
```

```
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix
ports-all
```

Danach kann wie folgt der Portsdb-Index aktualisiert werden:

```
# cvsup -L 2 -g /etc/supfile-ports
# portsdb -Uu
```

Nach der Aktualisierung des Portage-Trees kann Asterisk dann einfach mit dem folgenden Befehl installiert werden:

```
# cd /usr/ports/net/asterisk
# make install distclean
```

Nach der Installation des Basispaketes werden dann die Asterisk-Add-ons installiert:

```
# cd /usr/ports/net/asterisk-addon
# make install distclean
```

Warnung

Wenn ISDN genutzt werden soll, dann empfiehlt sich der Einsatz des net/asterisk-bristuff-Paketes anstelle von net/asterisk. Allerdings ist die Installation von ISDN und von chan_capi unter FreeBSD nicht trivial und wird derzeitig vom Portage System nicht unterstützt. Eine genaue Besprechung der Vorgehensweise würde den Rahmen dieses Buches sprengen und wird daher hier nicht beschrieben. Digium unterhält eine Asterisk-BSD Mailingliste, auf der man weitere Hilfe bekommen kann: *http://lists.digium.com/mailman/listinfo/asterisk-bsd*.

Soll Asterisk beim Booten automatisch starten, so ist in der Datei `/etc/rc.conf` die folgende Zeile einzutragen:

```
asterisk_enable="YES"
```

Warnung

Unter FreeBSD werden die Konfigurationsdateien der nachinstallierten Softwarepakete unter `/usr/local/etc/` gespeichert, um diese von den Systemdateien unter `/etc/` zu trennen. Daher liegen die Asterisk-Konfigurationsdateien nicht wie oft in diesem Buch beschrieben in `/etc/asterisk/`, sondern in `/etc/local/etc/asterisk/`.

2.6.5. Installation Asterisk 1.2.x auf MacOS X

Asterisk kann auf dem Mac ab OS X 10.2.8 („Jaguar“) zum Laufen gebracht werden. Empfehlenswert ist jedoch die aktuellste Version (z.Zt. 10.4.x, „Tiger“). Eine Installation auf MacOS X Server ist ebenfalls möglich.

Anmerkung

Unterstützung für PCI-Karten ist in Asterisk auf MacOS X bisher noch nicht möglich. Will man also seinen Server direkt mit dem öffentlichen Telefonnetz verbinden, muss man auf einen externen Gateway-Provider zurückgreifen. [8]Das „Unicall for MacOS X“-Projekt arbeitet an einer Lösung.

Asterisk ist für verschiedene Anwendungen (z.B. MeetMe-Konferenzen) zu Timing-Zwecken auf Zaptel-Hardware angewiesen. Als Alternative kann man den mitgelieferten `ztdummy`-Treiber kompilieren - was aber auf MacOS X kaum möglich sein dürfte.

[8]Eine Liste finden Sie auf http://www.voip-info.org/wiki/index.php?page=VoIP+Gateways oder auch http://www.voip-info.org/wiki/index.php?page=VoiP+Service+Providers.

Es kann in Ausnahmefällen auch zu Problemen mit den Formaten WAV und WAV49 kommen (GSM funktioniert).

Asterisk ist auf MacOS X insgesamt noch keine ganz „runde“ Lösung. Zu Testzwecken oder für den Heimgebrauch zwar problemlos möglich, für größere Installationen aber (noch?) nicht empfehlenswert, wenn man auf die genannten Funktionen angewiesen ist.

Möglicherweise stehen bald unter http://www.mactelephony.net/ (ehemals http://www.astmasters.net/, inzwischen nicht mehr erreichbar) „Binaries“, also vorkompilierte Pakete, zum Download bereit. [9]Bezüglich der Aktualität der Software schadet es aber nicht, Asterisk neu zu kompilieren.

Vorausetzung sind die bei MacOS X mitgelieferten oder im Internet von Apple verfügbaren Developer Tools („Xcode“).

Tipp

Falls Sie die DVD nicht greifbereit haben, können Sie Xcode auch von der Apple-Webseite http://developer.apple.com/tools/xcode/ downloaden.

Installieren Sie diese, falls Sie das noch nicht getan haben (dieses Paket legt das Verzeichnis `/Developer` auf Ihrer Festplatte an). Danach steht z.B. der Compiler **gcc** und weitere Tools zur Verfügung.

Laden Sie von http://www.asterisk.org/ (oder direkt http://ftp.digium.com/pub/) die Asterisk-Quellen (Sources) in der gewünschten Version herunter. Sie können das im Browser machen oder auf der Kommandozeile (statt `http://` ist auch `ftp://` möglich):

```
$  curl -O http://ftp.digium.com/pub/asterisk/asterisk-1.2-current.tar.gz
```

Danach wird das komprimierte Archiv entpackt:

```
$  tar -xvzf asterisk-1.2-current.tar.gz
asterisk-1.2.13/
asterisk-1.2.13/build_tools/
asterisk-1.2.13/build_tools/make_svn_branch_name
asterisk-1.2.13/build_tools/mkdep
asterisk-1.2.13/build_tools/make_build_h
[...]
```

Jetzt müssen wir in das gerade entpackte Verzeichnis wechseln und Asterisk mit **make** kompilieren. Bitte beachten Sie, dass 1.2.13 vielleicht nicht mehr die aktuellste Version ist; passen Sie also die Befehle entsprechend an.

```
$  cd asterisk-1.2.13/
$  make
```

Da dieser Vorgang je nach Leistung Ihres Rechners einige Zeit dauern kann, empfehle ich an dieser Stelle eine Tasse Tee[10] oder Kaffee. [11]Der Testrechner war allerdings schon nach etwa 3 Minuten fertig, was sich in dieser erfreulichen Botschaft ausdrückt:

```
[...]
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, and +
 + can be installed by running:              +
 +                                           +
 +               make install                +
 +-------------------------------------------+
```

Die Software ist jetzt kompiliert. Der nächste Schritt muss mit Root-Rechten ausgeführt werden. Hierzu bedienen wir uns auf Mac OS X des Programms ***sudo***.

Jetzt nur noch alle Dateien automatisch mit **sudo make install** installieren, also in die richtigen Verzeichnisse kopieren lassen:

[9]Zur Drucklegung fand sich auch ein Paket auf http://www.dacrib.net/rant/ (ggf. auf der Seite nach „Asterisk“ suchen).
[10]Vorzugsweise einen Darjeeling Finest Tippy Golden Flowery Orange Pekoe von der *Teekampagne*
[11]Jedes kleine Kind weiß, dass man Kaffee nicht unbeaufsichtigt rumstehen lässt!

```
$  sudo make install
[...]
 +---- Asterisk Installation Complete -------+
 +                                            +
 +    YOU MUST READ THE SECURITY DOCUMENT     +
 +                                            +
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
 +                                            +
 +               make samples                 +
 +                                            +
 +-----------------  or ----------------------+
 +                                            +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                            +
 +              make progdocs                 +
 +                                            +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +--------------------------------------------+
```

Und da wir sofort loslegen wollen, lassen wir auch noch mit **sudo make samples** die Beispiel-Konfiguration erstellen:

```
$  sudo make samples
[...]
done
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **asterisk -V** können Sie die installierte Version herausfinden (bitte achten Sie auf das großgeschriebene V):

```
$  asterisk -V
Asterisk 1.2.13
```

Ihren Asterisk-Server können Sie mit **sudo asterisk** starten und mit **sudo asterisk -rx "stop now"** beenden.

2.6.5.1. Asterisk automatisch beim Hochfahren starten

Unsere Installation hat noch eine Schwäche: Wenn Sie Ihren Computer neustarten, läuft der Asterisk-Server nicht mehr. Auf einem „Production“-Server ist es natürlich wünschenswert, dass Asterisk immer automatisch startet (während man das auf einem Arbeitsplatzrechner eher nicht will). Unter Mac OS X ist das Vorgehen etwas anders, als man es von anderen *nixen kennt. Man richtet sich am besten ein Startup Item ein, indem man im Verzeichnis `/Library/StartupItems/` ein neues Verzeichnis `Asterisk/` anlegt. Darin erstellt man mit einem Text-Editor seiner Wahl eine Datei `StartupParameters.plist` mit dem Inhalt:

```
{
  Description = "Asterisk PBX";
  Provides = ("Asterisk");
  OrderPreference = "Last";
  Messages =
  {
    start = "Starting Asterisk";
    stop  = "Stopping Asterisk";
  };
}
```

Diese Datei sagt dem **SystemStarter**, was unser Startup-Item bereitstellt und in welcher Rehenfolge es relativ zu den anderen Startup-Items geladen werden soll.

Jetzt brauchen wir noch eine zweite Datei mit dem Namen `Asterisk`, nämlich ein Shell-Skript, das dem System-Starter erklärt, wie Asterisk zu starten und zu beenden ist:

```
#!/bin/sh
```

```
# ohne Parameter geben wir aus, wie dieses Skript aufzurufen ist:
#
if [ -z $1 ] ; then
  echo "Usage: $0 [start|stop|restart] "
  exit 1
fi

# die Funktionen für Startup-Skripte einbinden:
#
test -r /etc/rc.common || exit 1
. /etc/rc.common

# das ausführbare Asterisk-Programm finden:
#
if [ -x /usr/sbin/asterisk ]; then
  ASTERISK="/usr/sbin/asterisk"
else
  if [ -x /opt/sbin/asterisk ]; then
    ASTERISK="/opt/sbin/asterisk"
  else
    if [ -x asterisk ]; then
      ASTERISK="asterisk"
    else
      ConsoleMessage "Cannot find asterisk"
      exit 1
    fi
  fi
fi

# Funktion zum Starten von Asterisk:
#
StartService ()
{
  if [ ! -f /var/run/asterisk.pid ]; then
    ConsoleMessage "Starting Asterisk"
    $ASTERISK
  else
    ConsoleMessage "Asterisk already running"
  fi
}

# Funktion zum Stoppen von Asterisk:
#
StopService ()
{
  if [ -f /var/run/asterisk.pid ]; then
    ConsoleMessage "Stopping Asterisk"
    $ASTERISK -rx "stop now"
  else
    ConsoleMessage "Asterisk not running"
  fi
}

# Funktion zum Neu-Starten von Asterisk:
#
RestartService ()
{
  if [ -f /var/run/asterisk.pid ]; then
    ConsoleMessage "Reloading Asterisk configuration"
    $ASTERISK -rx reload
  else
    StartService
  fi
}

# je nach Parameter die entsprechende oben definierte Funktion
# aufrufen:
#
RunService "$1"
```

Es ist wichtig, dass diese Dateien dem Benutzer `root` und der Gruppe `wheel` gehören und kein anderer Schreibrechte an diesen Dateien hat. Wir machen also Folgendes im Terminal:

```
$  cd /Library/StartupItems/
$  sudo chown -R root:wheel Asterisk
$  sudo chmod -R 754 Asterisk
```

Fertig. Wenn wir jetzt unseren Rechner hochfahren, wird Asterisk automatisch gestartet. Wir können das selbst im Terminal testen:

```
$  sudo SystemStarter start Asterisk
$  ps ax | grep ast
  521  ??  Ss     0:00.13 /usr/sbin/asterisk
  528  p1  S+     0:00.00 grep ast
$  sudo SystemStarter stop Asterisk
  538  p1  S+     0:00.00 grep ast
```

2.7. Asterisk 1.4

2.7.1. Installation Asterisk 1.4.x auf Debian Linux

Diese Installationsanleitung setzt ein frisch installiertes Debian GNU/Linux 3.1 (a.k.a sarge) voraus. Ein ISO-Image für eine entsprechende i386er Debian Linux-Installation finden Sie unter *http://cdimage.debian.org/debian-cd/3.1_r4/i386/iso-cd/debian-31r4-i386-netinst.iso*. Eine Debian GNU/Linux-Installationsanleitung befindet sich unter *http://www.debian.org/releases/stable/i386/*. Ein allgemeines Anwenderhandbuch steht unter *http://debiananwenderhandbuch.de.*

Bitte logen Sie sich als Benutzer root in das System ein und führen Sie alle hier angegebenen Befehle mit diesem Benutzer aus.

Als Erstes stellen Sie mit einem **apt-get update** sicher, dass `apt-get` alle aktuellen Paketlisten zur Verfügung hat:

```
debian:~# apt-get update
OK   http://ftp.de.debian.org stable/main Packages
OK   http://ftp.de.debian.org stable/main Release
OK   http://security.debian.org stable/updates/main Packages
OK   http://security.debian.org stable/updates/main Release
OK   http://security.debian.org stable/updates/contrib Packages
OK   http://security.debian.org stable/updates/contrib Release
OK   http://ftp.de.debian.org stable/main Sources
OK   http://ftp.de.debian.org stable/main Release
Paketlisten werden gelesen... Fertig
```

Um sicherzugehen, dass alle Pakete auf dem neuesten Stand sind, führen Sie noch ein **apt-get -y upgrade** aus:

```
debian:~# apt-get -y upgrade
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
```

Jetzt müssen Sie mit **apt-get -y install libncurses5-dev** noch ein Paket nachinstallieren, damit das Kompilieren von Asterisk erfolgreich durchgeführt werden kann:

```
debian:~# apt-get -y install libncurses5-dev
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Die folgenden NEUEN Pakete werden installiert:
  libncurses5-dev
0 aktualisiert, 1 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen noch 0B von 1038kB Archiven geholt werden.
Nach dem Auspacken werden 5296kB Plattenplatz zusätzlich benutzt.
Wähle vormals abgewähltes Paket libncurses5-dev.
(Lese Datenbank ... 21874 Dateien und Verzeichnisse sind derzeit installiert.)
Entpacke libncurses5-dev (aus .../libncurses5-dev_5.4-4_i386.deb) ...
```

```
Richte libncurses5-dev ein (5.4-4) ...
debian:~#
```

Auf der Asterisk-Homepage *http://www.asterisk.org/* finden Sie die notwendigen Quellen, um Asterisk zu kompilieren. Nehmen Sie bitte eine **stabile** und keine Entwickler-Version und laden Sie diese mit **wget http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz** in das Verzeichnis `/usr/src/` herunter:

```
debian:~# cd /usr/src
debian:/usr/src# wget http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
--18:53:00--  http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
           => `asterisk-1.4-current.tar.gz'
Auflösen des Hostnamen »ftp.digium.com«.... 69.16.138.164
Verbindungsaufbau zu ftp.digium.com[69.16.138.164]:80... verbunden.
HTTP Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 10,928,902 [application/x-gzip]

100%[====================================>] 10,928,902   230.71K/s    ETA 00:00

18:53:47 (227.00 KB/s) - »asterisk-1.4-current.tar.gz« gespeichert [10928902/10928902]
```

Die komprimierte Datei muss jetzt mit **tar xvzf asterisk-1.4-current.tar.gz** entpackt werden:

```
debian:/usr/src# tar xvzf asterisk-1.4-current.tar.gz
asterisk-1.4.0/
asterisk-1.4.0/build_tools/
asterisk-1.4.0/build_tools/mkpkgconfig
asterisk-1.4.0/build_tools/get_moduleinfo
asterisk-1.4.0/build_tools/mkdep
asterisk-1.4.0/build_tools/get_makeopts
asterisk-1.4.0/build_tools/make_version

[...]

asterisk-1.4.0/menuselect/mxml/COPYING
asterisk-1.4.0/menuselect/mxml/CHANGES
asterisk-1.4.0/menuselect/mxml/mxml-file.c
asterisk-1.4.0/menuselect/mxml/install-sh
asterisk-1.4.0/menuselect/mxml/mxml.pc
debian:/usr/src#
```

Asterisk 1.4 ist die erste Version, die den allgemein üblichen autoconf-Mechanismus benutzt. Als Erstes muss also im Asterisk-Verzeichnis ein **./configure** aufgerufen werden:

```
debian:/usr/src# cd asterisk-1.4.0/
debian:/usr/src/asterisk-1.4.0# ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for uname... /bin/uname
checking for gcc... gcc

[...]

configure: creating ./config.status
config.status: creating build_tools/menuselect-deps
config.status: creating makeopts
config.status: creating channels/h323/Makefile
config.status: creating include/asterisk/autoconfig.h

               .$$$$$$$$$$$$$$$=..
           .$7$7..          .7$$7:.
         .$$:.                 ,$7.7
       .$7.     7$$$$           .$$77
     ..$$.      $$$$$            .$$$7
    ..7$   .?.   $$$$$   .?.       7$$$.
   $.$.   .$$$7. $$$$7 .7$$$.      .$$$.
 .777.   .$$$$$77$$$77$$$$$7.      $$$,
 $$$~      .7$$$$$$$$$$$$$7.       .$$$.
.$$7          .7$$$$$$$:          ?$$$.
$$$          ?7$$$$$$$$$I        .$$$7
```

```
$$$       .7$$$$$$$$$$$$$$$$      :$$$.
$$$       $$$$$$7$$$$$$$$$$$$    .$$$.
$$$        $$$   7$$$7  .$$$    .$$$.
$$$$              $$$$7         .$$$.
7$$$7             7$$$$        7$$$
 $$$$$                         $$$
  $$$$7.                        $$  (TM)
   $$$$$$$.           .7$$$$$$  $$
     $$$$$$$$$$$$7$$$$$$$$.$$$$$$
       $$$$$$$$$$$$$$$$.

configure: Package configured for:
configure: OS type  : linux-gnu
configure: Host CPU : i686
debian:/usr/src/asterisk-1.4.0#
```

Asterisk muss jetzt kompliliert werden.[12]Dies geschieht mit dem Befehl **make**.

Warnung

Bitte achten Sie darauf, dass die hier in der Dokumentation benutzte Version vielleicht zum Zeitpunkt Ihrer Installation nicht mehr aktuell ist. Entsprechend werden sich die Verzeichnisnamen ändern.

Übrigens ist dies jetzt ein guter Moment, sich einen Kaffee oder Tee zu kochen. Der Kompiliervorgang kann je nach Rechenleistung des PCs wenige Minuten oder bis zu einer Stunde dauern.

```
debian:/usr/src/asterisk-1.4.0# make
make[1]: Entering directory `/usr/src/asterisk-1.4.0/menuselect'
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes

[...]

   [LD] abstract_jb.o acl.o aescrypt.o aeskey.o aestab.o alaw.o app.o ast_expr2.o ast_expr2f.o asterisk.o
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, and +
 + can be installed by running:              +
 +                                           +
 +               make install                +
 +-------------------------------------------+
debian:/usr/src/asterisk-1.4.0#
```

Die Software ist jetzt kompiliert und muss nur noch mit **make install** installiert (sprich: in die vorgesehenen Verzeichnisse kopiert) werden:

```
debian:/usr/src/asterisk-1.4.0# make install
make[1]: Für das Ziel »depend« ist nichts zu tun.
make[1]: Für das Ziel »depend« ist nichts zu tun.

[...]

for x in ; do /usr/bin/install -c -m 755 $x /usr/lib/asterisk/modules ; done
make[1]: Leaving directory `/usr/src/asterisk-1.4.0/main'
 +---- Asterisk Installation Complete -------+
 +                                           +
 +    YOU MUST READ THE SECURITY DOCUMENT    +
 +                                           +
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
```

[12]Die Linux-Cracks unter den Lesern mögen mir diese starke Vereinfachung verzeihen.

```
 +                                          +
 +               make samples               +
 +                                          +
 +-----------------  or ---------------------+
 +                                          +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                          +
 +              make progdocs               +
 +                                          +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +-------------------------------------------+
debian:/usr/src/asterisk-1.4.0#
```

Die eben kompilierten Programmdateien sind jetzt auf dem System installiert. Allerdings finden wir im Asterisk-Konfigurationsverzeichnis `/etc/asterisk/` noch gähnende Leere vor. In diesem Verzeichnis liegen die Konfigurationsdateien von Asterisk und da wir nicht von Null beginnen möchten, lassen wir uns die Standarddateien erstellen. Dies erreichen wir mit einem **make samples**:

```
debian:/usr/src/asterisk-1.4.0# make samples
mkdir -p /etc/asterisk
for x in configs/*.adsi; do \
        if [ ! -f /etc/asterisk/$x ]; then \
                /usr/bin/install -c -m 644 $x /etc/asterisk/`/usr/bin/basename $x` ; \
        fi ; \

[...]

for x in vm-theperson digits/1 digits/2 digits/3 digits/4 vm-isonphone; do \
        cat /var/lib/asterisk/sounds/$x.gsm >> /var/spool/asterisk/voicemail/default/1234/busy.gsm ; \
done
debian:/usr/src/asterisk-1.4.0#
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **asterisk -V** können Sie die installierte Version rausfinden (bitte achten Sie auf das großgeschriebene V):

```
debian:/usr/src/asterisk-1.4.0# asterisk -V
Asterisk 1.4.0
debian:/usr/src/asterisk-1.4.0#
```

2.7.2. Installation Asterisk 1.4.x auf Ubuntu Linux

Da Ubuntu auch von der CD als Live-System gebootet werden kann, müssen Sie es nicht zwangsläufig auf der Festplatte installieren (gut zum Testen). Diese Anleitung wurde mit einem Ubuntu 6.06.1 LTS (Dapper Drake) ausgeführt bzw. setzt dieses voraus. Das entsprechende ISO-Image finden Sie unter *http://de.archive.ubuntu.com/ubuntu-releases/6.06/ubuntu-6.06.1-desktop-i386.iso*. Wir benutzen die Desktop-CD. Der gleiche Installationsweg funktioniert aber auch bei der Server-CD. Hierbei sollte aber vor der Installation von Asterisk noch ein **sudo apt-get update** und ein **sudo apt-get -y upgrade** durchgeführt werden, um sicherzustellen, dass alle aktuellen Security Fixes installiert sind.

Weitere Informationen zu Ubuntu finden Sie auf *http://www.ubuntu.com*.

Tipp

Eine Liste von FTP-Mirrors zum Download des ISO-Images finden Sie auf *http://www.ubuntu.com/products/GetUbuntu/download#lts*.

Da auf einem Ubuntu-System root-Befehle immer über sudo (siehe http://de.wikipedia.org/wiki/Sudo) erfolgen, können Sie diese Installationsanleitung mit dem Benutzer ubuntu ausführen. Bitte öffnen Sie nach dem Starten von Ubuntu einfach ein Terminal-Fenster (im Menü oben links -> Applications -> Accessories -> Terminal).

Als Erstes stellen Sie mit einem **sudo apt-get update** sicher, dass `apt-get` alle aktuellen Paketlisten zur Verfügung hat:

```
ubuntu@ubuntu:~$ sudo apt-get update
Get:1 http://security.ubuntu.com dapper-security Release.gpg [191B]
Hit http://security.ubuntu.com dapper-security Release
Get:2 http://archive.ubuntu.com dapper Release.gpg [189B]
Get:3 http://archive.ubuntu.com dapper-updates Release.gpg [191B]
Hit http://security.ubuntu.com dapper-security/main Packages
Hit http://archive.ubuntu.com dapper Release
Hit http://security.ubuntu.com dapper-security/restricted Packages
Hit http://security.ubuntu.com dapper-security/main Sources
Hit http://archive.ubuntu.com dapper-updates Release
Hit http://security.ubuntu.com dapper-security/restricted Sources
Hit http://archive.ubuntu.com dapper/main Packages
Hit http://archive.ubuntu.com dapper/restricted Packages
Hit http://archive.ubuntu.com dapper/main Sources
Hit http://archive.ubuntu.com dapper/restricted Sources
Hit http://archive.ubuntu.com dapper-updates/main Packages
Hit http://archive.ubuntu.com dapper-updates/restricted Packages
Hit http://archive.ubuntu.com dapper-updates/main Sources
Hit http://archive.ubuntu.com dapper-updates/restricted Sources
Fetched 3B in 1s (2B/s)
Reading package lists... Done
ubuntu@ubuntu:~$
```

Jetzt müssen Sie mit **apt-get -y install make gcc g++ libncurses5-dev** noch ein paar Pakete nachinstallieren, damit das Kompilieren von Asterisk erfolgreich durchgeführt werden kann:

```
ubuntu@ubuntu:~$ sudo apt-get -y install make gcc g++ libncurses5-dev
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  binutils cpp cpp-4.0 g++-4.0 gcc-4.0 libc6-dev libstdc++6-4.0-dev
  linux-kernel-headers
Suggested packages:
  binutils-doc cpp-doc gcc-4.0-locales gcc-4.0-doc lib64stdc++6 manpages-dev
  autoconf automake1.9 libtool flex bison gcc-doc libc6-dev-amd64 lib64gcc1
  glibc-doc libstdc++6-4.0-doc stl-manual
Recommended packages:
  libmudflap0-dev
The following NEW packages will be installed:
  binutils cpp cpp-4.0 g++ g++-4.0 gcc gcc-4.0 libc6-dev libncurses5-dev
  libstdc++6-4.0-dev linux-kernel-headers make
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 13.1MB of archives.
After unpacking 52.6MB of additional disk space will be used.
Get:1 http://security.ubuntu.com dapper-security/main binutils 2.16.1cvs20060117-1ubuntu2.1 [1407kB]

[...]

Setting up g++-4.0 (4.0.3-1ubuntu5) ...
Setting up libstdc++6-4.0-dev (4.0.3-1ubuntu5) ...
Setting up g++ (4.0.3-1) ...

ubuntu@ubuntu:~$
```

Auf der Asterisk-Homepage *http://www.asterisk.org/* finden Sie die notwendigen Quellen, um Asterisk zu kompilieren. Nehmen Sie bitte eine stabile und keine Entwickler-Version und laden Sie diese mit **wget http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz** in das Verzeichnis `/tmp` herunter:

```
ubuntu@ubuntu:/tmp$ wget http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
--11:16:56--  http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
           => `asterisk-1.4-current.tar.gz'
Resolving ftp.digium.com... 216.27.40.102, 69.16.138.164
Connecting to ftp.digium.com|216.27.40.102|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10,928,902 (10M) [application/x-gzip]

100%[====================================>] 10,928,902   228.90K/s    ETA 00:00

11:17:44 (228.19 KB/s) - `asterisk-1.4-current.tar.gz' saved [10928902/10928902]
```

```
ubuntu@ubuntu:/tmp$
```

Die komprimierte Datei muss jetzt mit **tar xvzf asterisk-1.4-current.tar.gz** entpackt werden:

```
ubuntu@ubuntu:/tmp$ tar xvzf asterisk-1.4-current.tar.gz
asterisk-1.4.0/
asterisk-1.4.0/build_tools/
asterisk-1.4.0/build_tools/mkpkgconfig
asterisk-1.4.0/build_tools/get_moduleinfo
asterisk-1.4.0/build_tools/mkdep
asterisk-1.4.0/build_tools/get_makeopts
asterisk-1.4.0/build_tools/make_version

[...]

asterisk-1.4.0/menuselect/mxml/COPYING
asterisk-1.4.0/menuselect/mxml/CHANGES
asterisk-1.4.0/menuselect/mxml/mxml-file.c
asterisk-1.4.0/menuselect/mxml/install-sh
asterisk-1.4.0/menuselect/mxml/mxml.pc
ubuntu@ubuntu:/tmp$
```

Asterisk 1.4 ist die erste Version, die den allgemein üblichen autoconf-Mechanismus benutzt. Zuerst muss also im Asterisk-Verzeichnis ein **./configure** aufgerufen werden:

```
ubuntu@ubuntu:/tmp$ cd asterisk-1.4.0/
ubuntu@ubuntu:/tmp/asterisk-1.4.0$ ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for uname... /bin/uname
checking for gcc... gcc


[...]

configure: creating ./config.status
config.status: creating build_tools/menuselect-deps
config.status: creating makeopts
config.status: creating channels/h323/Makefile
config.status: creating include/asterisk/autoconfig.h

               .$$$$$$$$$$$$$$$=..
           .$7$7..          .7$$7:.
         .$$:.                 ,$7.7
       .$7.     7$$$$           .$$77
     ..$$.      $$$$$            .$$$7
    ..7$   .?.   $$$$$   .?.       7$$$.
   $.$.   .$$$7. $$$$7 .7$$$.      .$$$.
 .777.   .$$$$$$77$$$77$$$$$7.      $$$,
 $$$~      .7$$$$$$$$$$$$$7.       .$$$.
.$$7          .7$$$$$$$:          ?$$$.
$$$          ?7$$$$$$$$$I        .$$$7
$$$       .7$$$$$$$$$$$$$$$      :$$$.
$$$       $$$$$$7$$$$$$$$$$$    .$$$.
$$$        $$$   7$$$7  .$$$    .$$$.
$$$$             $$$$7          .$$$.
7$$$7            7$$$$        7$$$
 $$$$$                        $$$
  $$$$7.                       $$  (TM)
   $$$$$$$.           .7$$$$$$  $$
     $$$$$$$$$$$$$7$$$$$$$$$.$$$$$$
       $$$$$$$$$$$$$$$$.

configure: Package configured for:
configure: OS type  : linux-gnu
configure: Host CPU : i686
ubuntu@ubuntu:/tmp/asterisk-1.4.0$
```

Die entpackten Dateien müssen jetzt kompiliert werden.[13]Dies geschieht mit dem Befehl **make** im gerade entpackten Verzeichnis.

Warnung

Bitte achten Sie darauf, dass die hier in der Dokumentation benutzte Version vielleicht zum Zeitpunkt Ihrer Installation nicht mehr aktuell ist. Entsprechend werden sich die Verzeichnisnamen ändern.

Übrigens ist dies jetzt ein guter Moment, sich den obligatorischen Kaffee oder Tee zu kochen. Der Kompiliervorgang kann je nach Rechenleistung des PCs wenige Minuten oder bis zu einer Stunde dauern.

```
debian:/usr/src/asterisk-1.4.0# make
make[1]: Entering directory `/usr/src/asterisk-1.4.0/menuselect'
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes

[...]

   [LD] abstract_jb.o acl.o aescrypt.o aeskey.o aestab.o alaw.o app.o ast_expr2.o ast_expr2f.o asterisk.o
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, and +
 + can be installed by running:              +
 +                                           +
 +               make install                +
 +-------------------------------------------+
debian:/usr/src/asterisk-1.4.0#
```

Die Software ist jetzt kompiliert und muss nur noch mit **sudo make install** installiert (sprich in die richtigen Verzeichnisse kopiert) werden:

```
ubuntu@ubuntu:/tmp/asterisk-1.4.0$ sudo make install
make[1]: Nothing to be done for `depend'.
make[1]: Nothing to be done for `depend'.

[...]

for x in ; do /usr/bin/install -c -m 755 $x /usr/lib/asterisk/modules ; done
make[1]: Leaving directory `/tmp/asterisk-1.4.0/main'
 +---- Asterisk Installation Complete -------+
 +                                           +
 +    YOU MUST READ THE SECURITY DOCUMENT    +
 +                                           +
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
 +                                           +
 +               make samples                +
 +                                           +
 +-----------------  or ---------------------+
 +                                           +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                           +
 +              make progdocs                +
 +                                           +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +-------------------------------------------+
ubuntu@ubuntu:/tmp/asterisk-1.4.0$
```

[13]Die Linux-Cracks unter den Lesern mögen mir diese starke Vereinfachung verzeihen.

Die eben kompilierten Programmdateien sind jetzt auf dem System installiert. Allerdings finden wir im Asterisk-Konfigurationsverzeichnis `/etc/asterisk/` noch gähnende Leere vor. In diesem Verzeichnis liegen die Konfigurationsdateien von Asterisk und da wir nicht von Null beginnen möchten, lassen wir uns die Standarddateien erstellen. Dies erreichen wir mit einem **sudo make samples**:

```
ubuntu@ubuntu:/tmp/asterisk-1.4.0$ sudo make samples
mkdir -p /etc/asterisk
for x in configs/*.adsi; do \
                if [ ! -f /etc/asterisk/$x ]; then \
                        /usr/bin/install -c -m 644 $x /etc/asterisk/`/usr/bin/basename $x` ; \
                fi ; \


[...]

for x in vm-theperson digits/1 digits/2 digits/3 digits/4 vm-isonphone; do \
                cat /var/lib/asterisk/sounds/$x.gsm >> /var/spool/asterisk/voicemail/default/1234/busy.gs
        done
ubuntu@ubuntu:/tmp/asterisk-1.4.0$
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **sudo asterisk -V** können Sie die installierte Version rausfinden (bitte achten Sie auf das großgeschriebene V):

```
ubuntu@ubuntu:/tmp/asterisk-1.4.0$ sudo asterisk -V
Asterisk 1.4.0
ubuntu@ubuntu:/tmp/asterisk-1.4.0$
```

2.7.3. Installation Asterisk 1.4.x auf Fedora Linux

Diese Installationsanleitung setzt ein frisch installiertes Fedora Core 6 "Zod" voraus. Eine Liste von Mirrors zum Download des ISO-Images findet man unter *http://fedora.redhat.com/Download/mirrors.html*. Auf einem Mirror Ihrer Wahl laden Sie dann das CD- oder besser gleich das DVD-Image herunter. Mit der gebrannten DVD können Sie nun den Rechner booten und Fedora installieren.

Bitte logen Sie sich danach als Benutzer root in das System ein und führen Sie alle hier angegebenen Befehle mit diesem Benutzer aus.

Als Erstes stellen Sie mit einem **yum -y update** sicher, dass Ihr Fedora Linux-System nur aktuelle Pakete installiert hat:

```
[root@localhost ~]# yum -y update
Loading "installonlyn" plugin
Setting up Update Process
Setting up repositories
core                      100% |=========================| 1.1 kB    00:00
extras                    100% |=========================| 1.1 kB    00:00
updates                   100% |=========================| 1.2 kB    00:00
Reading repository metadata in from local files

[...]

bs.i386 0:0.7.40-1.fc6 system-config-soundcard.noarch 0:2.0.5-2.fc6 system-config-users.noarch 0:1.2.47-1
Replaced: bluez-pin.i386 0:0.30-5
Complete!
[root@localhost ~]#
```

Jetzt müssen Sie mit **yum -y groupinstall "Development Tools"** noch einige Pakete installieren:

```
[root@localhost ~]# yum -y groupinstall "Development Tools"
Loading "installonlyn" plugin
Setting up Group Process
Setting up repositories
Setting up repositories
Reading repository metadata in from local files
Package make - 1:3.81-1.1.i386 already installed and latest version

[...]
```

```
  Installing: automake17                   ######################## [56/56]

Installed: autoconf.noarch 0:2.59-12 automake.noarch 0:1.9.6-2.1 automake14.noarch 0:1.4p6-13 automake15.
Dependency Installed: cairo-java.i386 0:1.0.5-3.fc6 elfutils-libs.i386 0:0.123-1.fc6 glib-java.i386 0:0.2
Complete!
[root@localhost ~]#
```

Zum Schluss noch ein **yum -y install libtermcap-devel ncurses-devel openssl-devel** ausführen, damit das Kompilieren von Asterisk erfolgreich durchgeführt werden kann:

```
[root@localhost ~]# yum -y install libtermcap-devel ncurses-devel openssl-devel
Loading "installonlyn" plugin
Setting up Install Process
Setting up repositories

[...]

Installed: libtermcap-devel.i386 0:2.0.8-46.1
Complete!
[root@localhost ~]#
```

Auf der Asterisk-Homepage *http://www.asterisk.org/* finden Sie die notwendigen Quellen, um Asterisk zu kompilieren. Nehmen Sie bitte eine stabile und keine Entwickler-Version und laden Sie diese mit **wget http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz** in das Verzeichnis `/usr/src/` herunter:

```
[root@localhost ~]# cd /usr/src
[root@localhost src]# wget http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
--16:33:35--  http://ftp.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz
AuflÃ¶sen des Rechnernamens Â»ftp.digium.comÂ«.... 69.16.138.164, 216.27.40.102
Verbindungsaufbau mit ftp.digium.com[69.16.138.164]:80... verbunden.
HTTP-Anfrage gesendet, warte auf Antwort... 200 OK
LÃ¤nge: 10928902 (10M) [application/x-gzip]
Speichere nach: Â»asterisk-1.4-current.tar.gzÂ«

100%[======================================>] 10.928.902   142K/s   in 72s

16:34:48 (149 KB/s) - Â»asterisk-1.4-current.tar.gzÂ« gespeichert [10928902/10928902]

[root@localhost src]#
```

Die komprimierte Datei muss jetzt mit **tar xvzf asterisk-1.4-current.tar.gz** entpackt werden:

```
[root@localhost src]# tar xvzf asterisk-1.4-current.tar.gz
asterisk-1.4.0-beta3/
asterisk-1.4.0-beta3/build_tools/
asterisk-1.4.0-beta3/build_tools/mkpkgconfig
asterisk-1.4.0-beta3/build_tools/get_moduleinfo
asterisk-1.4.0-beta3/build_tools/mkdep

[...]

asterisk-1.4.0-beta3/menuselect/mxml/CHANGES
asterisk-1.4.0-beta3/menuselect/mxml/mxml-file.c
asterisk-1.4.0-beta3/menuselect/mxml/install-sh
asterisk-1.4.0-beta3/menuselect/mxml/mxml.pc
[root@localhost src]#
```

Asterisk 1.4 ist die erste Version, die den allgemein üblichen autoconf-Mechanismus benutzt. Zuerst muss also im Asterisk-Verzeichnis ein **./configure** aufgerufen werden:

```
[root@localhost src]# cd asterisk-1.4.0
[root@localhost asterisk-1.4.0]# ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking for uname... /bin/uname
checking for gcc... gcc

[...]
```

```
configure: creating ./config.status
config.status: creating build_tools/menuselect-deps
config.status: creating makeopts
config.status: creating channels/h323/Makefile
config.status: creating include/asterisk/autoconfig.h

               .$$$$$$$$$$$$$$$=..
            .$7$7..          .7$$7:.
          .$$:.                 ,$7.7
        .$7.     7$$$$           .$$77
     ..$$.       $$$$$            .$$$7
    ..7$   .?.   $$$$$   .?.       7$$$.
   $.$.   .$$$7. $$$$7 .7$$$.      .$$$.
 .777.   .$$$$$$77$$$77$$$$$7.      $$$,
 $$$~      .7$$$$$$$$$$$$$7.       .$$$.
.$$7          .7$$$$$$$7:          ?$$$.
$$$          ?7$$$$$$$$$I         .$$$7
$$$       .7$$$$$$$$$$$$$$$      :$$$.
$$$       $$$$$$7$$$$$$$$$$$    .$$$.
$$$        $$$   7$$$7  .$$$    .$$$.
$$$$             $$$$7          .$$$.
7$$$7            7$$$$        7$$$
 $$$$$                        $$$
  $$$$7.                       $$  (TM)
   $$$$$$$.           .7$$$$$$  $$
     $$$$$$$$$$$$$7$$$$$$$$$.$$$$$$
       $$$$$$$$$$$$$$$$.

configure: Package configured for:
configure: OS type  : linux-gnu
configure: Host CPU : i686
[root@localhost asterisk-1.4.0]##
```

Asterisk muss jetzt kompiliert werden.[14]Dies geschieht mit dem Befehl **make**.

Warnung

Bitte achten Sie darauf, dass die hier in der Dokumentation benutzte Version vielleicht zum Zeitpunkt Ihrer Installation nicht mehr aktuell ist. Entsprechend werden sich die Verzeichnisnamen ändern.

Übrigens ist dies jetzt ein guter Moment, sich einen Kaffee oder Tee zu kochen. Der Kompiliervorgang kann je nach Rechenleistung des PCs wenige Minuten oder bis zu einer Stunde dauern.

```
[root@localhost asterisk-1.4.0]# make
make[1]: Entering directory `/usr/src/asterisk-1.4.0/menuselect'
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes

[...]

   [LD] abstract_jb.o acl.o aescrypt.o aeskey.o aestab.o alaw.o app.o ast_expr2.o ast_expr2f.o asterisk.o
 +--------- Asterisk Build Complete ---------+
 + Asterisk has successfully been built, and +
 + can be installed by running:              +
 +                                           +
 +               make install                +
 +-------------------------------------------+
[root@localhost asterisk-1.4.0]#
```

[14]Die Linux-Cracks unter den Lesern mögen mir diese starke Vereinfachung verzeihen.

Die Software ist jetzt kompiliert und muss nur noch mit **make install** installiert (sprich: in die vorgesehenen Verzeichnisse kopiert) werden:

```
[root@localhost asterisk-1.4.0]# make install
make[1]: Für das Ziel »depend« ist nichts zu tun.
make[1]: Für das Ziel »depend« ist nichts zu tun.

[...]

for x in ; do /usr/bin/install -c -m 755 $x /usr/lib/asterisk/modules ; done
make[1]: Leaving directory `/usr/src/asterisk-1.4.0/main'
 +---- Asterisk Installation Complete -------+
 +                                           +
 +    YOU MUST READ THE SECURITY DOCUMENT    +
 +                                           +
 + Asterisk has successfully been installed. +
 + If you would like to install the sample   +
 + configuration files (overwriting any      +
 + existing config files), run:              +
 +                                           +
 +               make samples                +
 +                                           +
 +-----------------  or ---------------------+
 +                                           +
 + You can go ahead and install the asterisk +
 + program documentation now or later run:   +
 +                                           +
 +              make progdocs                +
 +                                           +
 + **Note** This requires that you have      +
 + doxygen installed on your local system    +
 +-------------------------------------------+
[root@localhost asterisk-1.4.0]#
```

Die eben kompilierten Programmdateien sind jetzt auf dem System installiert. Allerdings finden wir im Asterisk-Konfigurationsverzeichnis `/etc/asterisk/` noch gähnende Leere vor. In diesem Verzeichnis liegen die Konfigurationsdateien von Asterisk und da wir nicht von Null beginnen möchten, lassen wir uns die Standarddateien erstellen. Dies erreichen wir mit einem **make samples**:

```
[root@localhost asterisk-1.4.0]# make samples
mkdir -p /etc/asterisk
for x in configs/*.adsi; do \
        if [ ! -f /etc/asterisk/$x ]; then \
                /usr/bin/install -c -m 644 $x /etc/asterisk/`/usr/bin/basename $x` ; \
        fi ; \

[...]

for x in vm-theperson digits/1 digits/2 digits/3 digits/4 vm-isonphone; do \
        cat /var/lib/asterisk/sounds/$x.gsm >> /var/spool/asterisk/voicemail/default/1234/busy.gsm ; \
done
[root@localhost asterisk-1.4.0]#
```

Fertig! Asterisk ist auf Ihrem System installiert. Mit dem Befehl **asterisk -V** können Sie die installierte Version rausfinden (bitte achten Sie auf das großgeschriebene V):

```
[root@localhost asterisk-1.4.0]# asterisk -V
Asterisk 1.4.0
[root@localhost asterisk-1.4.0]#
```

2.7.4. Installation von Asterisk 1.4.x mit AsteriskNOW

Der Hersteller von Asterisk (die Firma *Digium*) hat für die Asterisk-Version 1.4 erstmalig eine eigene Linux-Distribution herausgebracht, die unter dem Namen AsteriskNOW auf der Webseite *http://www.asterisknow.org* zum Download angeboten wird. Diese Installation ist natürlich im Vergleich zu den anderen hier beschriebenen extrem einfach, allerdings ist man dafür später teilweise auf Digium-Hardware eingeschränkt.

Das ISO-Image für die Installations-CD können Sie auf *http://www.asterisknow.org* herunterladen.

Bitte stellen Sie vor der Installation sicher, dass auf dem entsprechenden Rechner keine wichtigen Daten mehr vorhanden sind. Die Festplatte wird während der Installation neu partitioniert und die Partitionen dann formatiert.

Nach dem Einlegen der CD und dem Booten (hierfür muss im BIOS die Einstellung "zuerst von CD booten" aktiviert sein) sehen Sie folgenden Bildschirm:

Um ein neues System zu installieren, können Sie hier einfach **ENTER** drücken.

Danach können Sie entscheiden, welche Art der Installation Sie wünschen:

Das Installationsprogramm wird danach noch ein paar Einstellungen zum System (wie z.B. die Zeitzone) abfragen und dann mit der Installation beginnen:

Ist die Installation abgeschlossen, wird das System gestartet und zeigt das AsteriskNOW Console Menu an:

Mit der Tastenkombinationen **ALT-F2** erhalten Sie eine normale Konsole, auf der Sie sich mit dem User `admin` einlogen können. Für Rootrechte müssen Sie auch bei dieser Distribution mit dem Programm **sudo** arbeiten.

Tipp

Mit der Tastenkombination **ALT-F9** kommen Sie ins Asterisk CLI.

Warnung

Die Benutzung von AsteriskNOW ist für den Asterisk-Anfänger an dieser Stelle wenig geeignet. Nehmen Sie lieber ein Debian Linux! Wer Asterisk verstanden hat, der wird die Vor- und Nachteile von AsteriskNOW abschätzen können. Für den Anfänger überwiegen die Nachteile, da in diesem Buch der Schwerpunkt in der Programmierung von Dialplänen in der `extensions.conf` liegt.

2.8. Den Asterisk-Server konfigurieren

Im Verzeichnis `/etc/asterisk/` finden wir jetzt alle Konfigurationsdateien in einer einfachen Standardausführung:

```
debian:/usr/src/asterisk-1.2.13# cd /etc/asterisk
debian:/etc/asterisk# ls
adsi.conf           cdr_tds.conf     indications.conf  privacy.conf
adtranvofr.conf     codecs.conf      logger.conf       queues.conf
agents.conf         dnsmgr.conf      manager.conf      res_odbc.conf
alarmreceiver.conf  dundi.conf       meetme.conf       rpt.conf
alsa.conf           enum.conf        mgcp.conf         rtp.conf
asterisk.adsi       extconfig.conf   misdn.conf        sip.conf
asterisk.conf       extensions.ael   modem.conf        sip_notify.conf
cdr.conf            extensions.conf  modules.conf      skinny.conf
cdr_custom.conf     features.conf    musiconhold.conf  telcordia-1.adsi
cdr_manager.conf    festival.conf    osp.conf          voicemail.conf
cdr_odbc.conf       iax.conf         oss.conf          vpb.conf
cdr_pgsql.conf      iaxprov.conf     phone.conf        zapata.conf
debian:/etc/asterisk#
```

Dies ist jetzt eine recht umfangreiche Liste, aber -- keine Angst -- für unsere Mini-Telefonanlage müssen wir uns nur mit zwei Dateien näher befassen. Die mit **make samples** erstellen Defaultdateien verschieben wir dazu erst einmal nach `/etc/asterisk/backup/` (damit wir später bei Bedarf darauf zugreifen können):

```
debian:/etc/asterisk# mkdir backup
debian:/etc/asterisk# mv sip.conf backup/
debian:/etc/asterisk# mv extensions.conf backup/
debian:/etc/asterisk#
```

Legen Sie mit Ihrem Lieblingseditor[15] die Datei /etc/asterisk/sip.conf mit folgendem Inhalt an:[16]

```
[general]
port = 5060
bindaddr = 0.0.0.0
context = sonstige

[2000]
type=friend
context=meine-telefone
secret=1234
host=dynamic

[2001]
type=friend
context=meine-telefone
secret=1234
host=dynamic
```

In die Datei `/etc/asterisk/extensions.conf` schreiben wir einen einfachen Wählplan:

```
[sonstige]

[meine-telefone]
exten => 2000,1,Dial(SIP/2000)
exten => 2001,1,Dial(SIP/2001)
```

Und diese 21 Zeilen Konfiguration sollen ausreichen, um eine komplette Telefonanlage zu konfigurieren? Dabei heißt es doch immer, dass Asterisk so kompliziert sei. Probieren wir es einmal aus! Mit dem Befehl **asterisk -c** starten Sie Asterisk:

```
debian:/etc/asterisk# asterisk -c
Asterisk 1.2.13, Copyright (C) 1999 - 2005 Digium.
Written by Mark Spencer <markster@digium.com>
=========================================================================
[ Booting...Nov 20 18:59:28 NOTICE[14937]: cdr.c:1185 do_reload: CDR
simple logging enabled.
.........................................................................
........................................... ]
Asterisk Ready.
*CLI>
```

Mit dem erfolgreichen Start von Asterisk erhalten wir ebenfalls eine Konsole, auf der man mit dem laufenden Asterisk-Prozess kommunizieren kann. Vor uns sehen wir jetzt das Command Line Interface (CLI) von Asterisk. Hier können wir aktiv in das Geschehen eingreifen und als erste Handlung die eben gestartete Telefonanlage wieder herunterfahren:

```
*CLI> stop now
debian:/etc/asterisk#
```

2.9. Die SIP-Telefone konfigurieren

Jetzt müssen wir zwei SIP-Telefone mit der Anlage verbinden. Wenn Sie kein SIP-Telefon besitzen, können Sie auch zwei Software-Telefone, die Sie aus dem Internet herunterladen, benutzen. Die User-Daten können Sie der Datei `/etc/asterisk/sip.conf` entnehmen. Falls Sie die IP-Adresse Ihres Asterisk-Linux-Systems nicht kennen, finden Sie diese mit folgendem Befehl heraus:

[15]Wenn Sie keinen Editor kennen, den Sie in einem Terminalfenster verwenden können, empfehle ich Ihnen **nano**. Unter Debian ist dieser einfach als Benutzer root mit **apt-get install nano** zu installieren. Danach öffnen Sie mit **nano dateiname** die Datei. In Nano selbst sehen Sie im unteren Bereich des Bildschirms die wichtigsten Befehle.

[16]Diese einfachen Passwörter sind natürlich nur für den Testbetrieb gedacht, für den produktiven Einsatz sollten Sie entsprechend bessere Passwörter vergeben.

```
debian:/etc/asterisk# ifconfig | grep Bcast | sed s/Bcast.*//
          inet addr:215.160.52.161
```

In diesem Fall ist es also die `215.160.52.161`. Wir starten Asterisk erneut, diesmal jedoch in einem "gesprächigeren" Modus, um mehr Informationen über die laufenden Aktionen zu erhalten. Dies erreichen wir, indem wir Asterisk mit dem Parameter -vvvvvc starten (die 5 v stehen für "verbose"). Dann können wir verfolgen, wie sich das SIP-Telefon an der Anlage anmeldet:

```
debian:/etc/asterisk# asterisk -vvvvvc
[...]
Asterisk Ready.
*CLI>
```

Nachdem Sie Ihre SIP-Telefone konfiguriert haben (Daten siehe `/etc/asterisk/sip.conf`), können Sie jetzt den Anmeldevorgang starten (einfach das Telefon einschalten). Der Asterisk-Prozess teilt uns die Anmeldung der Telefone mit:

```
*CLI>     -- Registered SIP '2000' at 87.143.3.144 port 5060
expires 120 -- Unregistered SIP '2000'

*CLI>     -- Registered SIP '2001' at 87.143.3.145 port 5060
expires 120 -- Unregistered SIP '2001'
```

Nachdem die Telefone am System angemeldet sind, können wir jetzt von einem Telefon das andere anrufen, z.B. einfach die 2001 vom 2000-er anwählen. Wenn Sie sich jetzt über die Telefone unterhalten können, haben Sie es geschafft --- Ihre erste Mini-Telefonanalage mit Asterisk funktioniert.

Tipp

Falls Sie im Eifer des Gefechts die Registered SIP Ausgabe im CLI übersehen haben oder sich nicht mehr sicher sind, können Sie mit **sip show peers** alle konfigurierten SIP-Telefone auflisten. Mit **sip show peer 2000** bekommen Sie sogar ganz detaillierte Informationen zum Peer 2000.

Sie können Asterisk jederzeit mit dem Kommando **stop now** im CLI beenden.

2.10. Und jetzt mit einem Anrufbeantworter

Asterisk beinhaltet bereits ein fertiges Voicemail-Modul, wir müssen es lediglich in der Datei `/etc/asterisk/voicemail.conf` aktivieren. Als ersten Schritt verschieben wir diese Default-Datei in unser Verzeichnis mit den Sicherungskopien der Konfigurationsdateien:

```
debian:/# cd /etc/asterisk
debian:/etc/asterisk# mv voicemail.conf backup/
```

Daraufhin erstellen wir eine neue `/etc/asterisk/voicemail.conf` und versehen diese mit folgendem Inhalt:

```
[general]
format = wav

[default]
2000 => 4711,Hans Mustermann,hansi@company.de
2001 => 0815,Ute Beispiel,ute.beispiel@company.de
```

Damit sind die Voicemailboxen grundsätzlich eingerichtet (ja, so einfach geht das!). Jetzt müssen wir aber noch in der `/etc/asterisk/extensions.conf` ein paar weitere Zeilen eintragen, um die Voicemail-Funktionalität unseren Telefonen zuzuweisen. Bitte vergessen Sie dabei nicht den Eintrag *",20,j"* bei der Dial-Applikation.

```
[sonstige]

[meine-telefone]
exten => 2000,1,Dial(SIP/2000,20,j)
exten => 2000,2,VoiceMail(u2000)
exten => 2000,102,VoiceMail(b2000)

exten => 2001,1,Dial(SIP/2001,20,j)
```

```
exten => 2001,2,VoiceMail(u2001)
exten => 2001,102,VoiceMail(b2001)

exten => 2999,1,VoiceMailMain(s${CALLERID(num)})
```

Fertig! Asterisk nur noch mit **asterisk -vvvvvc** neu starten

Warnung

Sie müssen Asterisk an dieser Stelle neu starten, da sonst die Voicemailboxen nicht angelegt werden.
und mit einem Telefon das andere anrufen. Nachdem es 20 Sekunden lang geklingelt hat (deshalb die *20* und die Option *j* am Ende des Dial-Befehls), kommt man auf die VoiceMailbox. Ist die Gegenstelle besetzt, wird man direkt auf die VoiceMailbox geleitet. Deshalb gibt es eine zweite Regel mit einem b2000 (das b steht für busy). Sie können die Mailboxen abhören, indem Sie die Nummer 2999 anrufen. Dann bekommen Sie Ihre Voicemailbox als Menü vorgespielt. Wenn Sie das Ganze noch mit einer Passwortabfrage absichern wollen oder eine Auflistung des Menüs suchen, finden Sie dazu in Kapitel 6, *Voicemailsystem* die nötigen Informationen.

2.11. Was haben wir gerade gemacht?

Nach unserem ersten Erfolg gehen wir nun Schritt für Schritt die Konfigurationsdateien durch, um mehr über die Funktionsweise von Asterisk zu erfahren. Fangen wir mit der Datei `/etc/asterisk/sip.conf` an.

```
[general]
port = 5060
bindaddr = 0.0.0.0
context = sonstige
```

In diesem ersten Abschnitt [general] werden allgemeine Variablen definiert. Der Standard-Port wird auf 5060 gesetzt, d.h. Asterisk wickelt die Verbindungen über diese Portnummer ab. Mit bindaddr = 0.0.0.0 beachtet Asterisk einkommenden IP-Pakete auf allen IP-Adressen auf diesem System. Üblicherweise haben Rechner meist nur eine IP-Adresse, falls Sie einem Rechner mehrere IP-Adressen zugewiesen haben und vielleicht sogar mehrere Asterisk-Instanzen betreiben, können Sie hiermit eine bestimmte IP-Adresse einer jeweiligen Asterisk-Instanz zuordnen. Das Context-Konstrukt folgt nicht unbedingt einer intuitiven Logik bei der Konfiguration von Asterisk. Auf diesen Konfigurationsparameter werde ich noch an geeigneterer Stelle eingehen. Im weiteren Verlauf der Anleitungen wird nach und nach das Verständnis dafür geschaffen, was ein Context im Asterisk-Sinne bedeutet und wie er eingesetzt wird.

```
[2001]
type=friend
context=meine-telefone
secret=1234
host=dynamic
```

Im Abschnitt `[2001]` wird der SIP-Anschluss mit der Kennung 2001 definiert. Dass hierbei eine Zahl (also 2001) verwendet wurde, ist dabei eher der üblichen Erwartungshaltung geschuldet, da die meisten Nutzer Telefone mit Nummern verbinden. Ein SIP-Anschluss kann aber auch mit einem alphanumerischen Wert definiert werden, also beispielsweise `[Rezeption-1]`. Der Parameter type=friend bestimmt vereinfacht gesagt, dass dieser SIP-Anschluss sowohl eingehende, als auch ausgehende Verbindungen aufbauen darf.[17]

Und wieder stoßen wir auf den ominösen `context`. Diesen werden wir gleich in der Datei `/etc/asterisk/extensions.conf` aufgreifen und dann wird das zugrundeliegende System in der Verwendung auch klarer.

[17] Der Eintrag `type=` kennt drei verschiedene Werte (auf die in einem späteren Kapitel genau eingegangen wird):

- `friend`: kann anrufen und angerufen werden.
- `peer`: kann nur anrufen.
- `user`: kann nur angerufen werden.

Mit `secret` wird das Passwort gesetzt. Wir benutzen an dieser Stelle Zahlen, da man diese mit einem Telefon leichter eintippen kann. `host=dynamic` sagt aus, dass es für Asterisk ohne Bedeutung ist, ob der SIP-Client wechselnde IP-Adressen hat.

2.11.1. extensions.conf - der Dialplan

Die Datei `/etc/asterisk/extensions.conf` ist das Herz jeder Asterisk-Konfiguration (siehe auch Kapitel 4, *Programmieren im Dialplan*). Sie ist vergleichbar mit einer Schaltzentrale bei frühen Telefonanlagen. In dieser Konfigurationsdatei wird bestimmt, welches Telefon mit welchem Telefon Kontakt aufnehmen darf.

```
[sonstige]
```

Diese erste Sektion der Konfiguration ist für den Context `sonstige` bestimmt. Da wir dieses in diesem Beispiel nicht benötigen, ist sie leer.

```
[meine-telefone]
exten => 2000,1,Dial(SIP/2000,20,j)
exten => 2000,2,Voicemail(u2000)
exten => 2000,102,Voicemail(b2000)
```

Der Context wird von Asterisk immer benutzt, wenn ein Telefon die Nummer eines anderen Telefons wählt. Der Name ist dabei beliebig. Wichtig ist, dass in der Datei `/etc/asterisk/sip.conf` bei der Definition eines SIP-Gerätes eine entsprechende Bezeichnung für den Context definiert wurde und dieselbe Bezeichnung in der Datei `/etc/asterisk/extensions.conf` wieder als `context`-Bezeichnung verwendet wird (dann könnte er auch `Apfelmuss` heißen). Dieser Context ist von entscheidender Bedeutung für die Konfiguration des Telefons! Einfach gesagt legen die im Context definierten Regeln exakt fest, welche Nummern das Telefon wählen darf und welche Aktionen dadurch gestattet werden. *Der Context der angerufenen Nummer spielt keinerlei Rolle! Merken Sie sich bitte diesen Zusammenhang, er ist wesentlich für einen erfolgreichen Umgang mit den Konfigurationsdateien.*

Die Syntax der Einträge ist immer wie folgt:

```
exten => Nummer, Prioritaet, Applikation
```

Wird eine Nummer gewählt, die einen entsprechenden Eintrag (Regel) in dieser Form aufweist, dann wird diese Regel gelesen und übernommen. Ist mehr als eine Regel für eine Nummer definiert, wird als Erstes die Regel mit der Priorität 1 durchgeführt. Die mit einer Regel verbundene Aktion steht an dritter Stelle ("Applikation"). Auf der Grundlage unserer Konfigurationsdateien (s.o.) geschieht Folgendes, wenn vom Telefon 2001 die Nummer 2000 gewählt wird:

- Asterisk sieht in der Datei `/etc/asterisk/sip.conf` nach, welcher Context der anrufenden Nummer (2001) zugewiesen wurde. In unserem Beispiel haben wir der Nummer 2001 den Context `meine-telefone` zugewiesen. Dadurch werden die Regeln gelesen, die in der entsprechenden Context-Bezeichnung `context=meine-telefone` der Datei `/etc/asterisk/extensions.conf` definiert sind.

- Nachdem Asterisk die Regeln im Context für die anrufende Nummer (2001) gelesen hat, führt es die Regeln aus, die mit der angerufenen Nummer (2000) definiert sind und zwar in der durch die Priorität festgelegten Reihenfolge. (In unserem Beispiel sind im Context der anrufenden Nummer (2001) für die angerufene Nummer (2000) drei Regeln hinterlegt.)

- Da die Regel mit dem Befehl `Dial(SIP/2000,20)` die Priorität 1 besitzt, kommt diese zur Ausführung. Dadurch wird das Programm Dial gestartet, welches wiederum in der Datei `/etc/asterisk/sip.conf` nach dem Eintrag 2000 sucht und diesen Apparat dann 20 Sekunden lang anruft (dafür die 20 als zweiter Parameter des Dial-Befehls).

- Wenn nach 20 Sekunden niemand den Anruf annimmt und die Option j für Priorityjumping gesetzt ist, wird automatisch die Priorität um 101[18] auf dann 102 hochgesetzt und es wird wieder nach einem entsprechenden Eintrag (jetzt mit der Priorität 102) gesucht. Falls dieser nicht existiert, wird keine weitere Aktion durchgeführt.

[18]Der Wert 101 wurde vom Asterisk-Entwicklerteam so gewählt. Er macht genauso viel oder wenig Sinn wie jede andere Zahl. Warum man einen solchen Sprung benötigt, werden Sie in späteren Kapiteln noch lernen.

- Die Zeile `exten => 2000,102,Voicemail(b2000)` ist also an der Reihe und startet das Programm Voicemail mit dem Parameter *b2000*. Das *b* steht dabei für busy (also besetzt) und die *2000* für die Nummer der Voicemailbox. Wie Sie sich sicherlich schon denken können, ist auch hier die 2000 nur der Einfachheit halber genommen worden. Die Voicemailbox könnte auch 5555 heißen.

- Warum ist nun aber noch eine Zeile mit der Priorität 2 vorhanden? Diese Zeile fängt den Fall ab, dass kein SIP-Telefon mit dieser Nummer angemeldet ist oder das entsprechende Telefon gerade keine Netzwerkverbindung hat. Dann ist nämlich der Teilnehmer "unavailable" und deshalb steht beim Parameter ein *u* vor der *2000*. Kann der Befehl Dial nicht durchgeführt werden, weil kein entsprechendes Telefon aktuell am System angemeldet ist, bricht Asterisk diesen Vorgang ab und führt die nächste Regel (um 1 erhöhte Priorität) aus.

Eigentlich müssten wir die Programmierschritte noch mit einem Hangup-Befehl beenden, damit Asterisk auch wieder auflegt. Dies übernimmt aber in diesem Fall der Befehl Voicemail schon für uns. Damit wir später auch an unsere Sprachnachrichten kommen, wird in der letzten Zeile die Nummer 2999 für das Aufrufen der Voicemailbox definiert.

```
exten => 2999,1,VoicemailMain(${CALLERID(num)})
```

Hierbei kommen wir zum ersten Mal mit der Benutzung von Variablen in der `/etc/asterisk/extensions.conf` in Berührung. Das aufgerufene Programm heißt VoicemailMain und ihm wird als Parameter die Funktion *${CALLERID(num)}* übergeben. *${CALLERID(num)}* gibt die Nummer des Anrufers aus. Deshalb weiß das Programm VoicemailMain, welche Voicemailbox gerade abgefragt wird. Wird dieser Parameter nicht angegeben, fragt VoicemailMain als Erstes den Anrufer nach dem Namen der Voicemailbox.

2.11.2. voicemail.conf - der Anrufbeantworter

Das Voicemail-Modul (siehe auch Kapitel 6, *Voicemailsystem*) wird in der Datei `/etc/asterisk/voicemail.conf` konfiguriert und mit dem Wissen, das wir bisher bereits erworben haben, ist sie fast schon selbsterklärend:

```
[general]
format = wav

[default]
2000 => 4711,Hans Mustermann,hansi@company.de
2001 => 0815,Ute Beispiel,ute.beispiel@company.de
```

In der Sektion `[general]` wird das Aufzeichnungsformat definiert (hier WAV) und in der Sektion `[default]` werden die beiden Mailboxen 2000 und 2001 mit den entsprechenden Passwörtern (4711 und 0815) definiert. Hinter dem Passwort findet sich dann noch ein Eintrag für den Namen des Benutzers und danach seine E-Mail-Adresse. Sprachnachrichten werden so automatisch als E-Mail-Attachments im WAV-Format zugestellt.

2.12. Ins öffentliche Telefonnetz telefonieren

Jetzt werden Sie vielleicht einwenden, dass Sie zwar eine fertige Telefonanlage haben, was an und für sich spannend ist, aber ohne den Anschluss an die große weite Welt noch nicht wirklich nützlich. Wenn Sie mir weitere 10 Minuten Zeit geben und Ihre Mini-Telefonanlage eine Internetverbindung besitzt, lässt sich auch diese Aufgabe lösen und Sie erhalten in Kürze Anschluss ans Festnetz. Auf der Innenseite des Heftumschlages dieses Buches finden Sie einen Gutscheincode mit einem Gesprächsguthaben für einen VoIP-Provider. Wer bereits einen Account bei einem VoIP-Provider besitzt, kann natürlich diesen Account benutzen.

Jetzt konfigurieren wir Asterisk so, dass Sie mit den Telefonen 2000 und 2001 ins deutsche Festnetz telefonieren können. Als Erstes muss der Provider-Account in der `/etc/asterisk/sip.conf` eingetragen werden:

```
[general]
port = 5060
bindaddr = 0.0.0.0
context = sonstige

register => 5587572:UHDZJD@mein-voip-provider.de/5587572
;              ^       ^       ^                    ^
;              |       |       |                    |
;            User Passwort  Provider              User
```

```
[2000]
type=friend
context=meine-telefone
secret=1234
host=dynamic

[2001]
type=friend
context=meine-telefone
secret=1234
host=dynamic

[ext-sip-account]
type=friend
context=von-voip-provider
username=5587572
fromuser=5587572
secret=UHDZJD
host=mein-voip-provider.de
fromdomain=mein-voip-provider.de
qualify=yes
insecure=very
nat=yes
```

Den Usernamen (im Beispiel *5587572*) und Ihr Passwort (im Beispiel *UHDZJD*) können Sie der SIP-Provider-Webseite entnehmen. Damit kann Asterisk diesen Account benutzen. Allerdings müssen wir jetzt noch in der `/etc/asterisk/extensions.conf` eine Regel zum Rauswählen einfügen:

```
[sonstige]

[meine-telefone]
exten => 2000,1,Dial(SIP/2000,20,j)
exten => 2001,1,Dial(SIP/2001,20,j)

exten => _0[1-9].,1,Dial(SIP/${EXTEN}@ext-sip-account)
```

Nachdem die Einträge in den Konfigurationsdateien erstellt sind, starten Sie Asterisk, wie bekannt, mit **asterisk -vvvvvc** in das CLI und warten ein paar Sekunden, bis sich eines der SIP-Telefone angemeldet hat. Danach einfach eine beliebige Nummer im deutschen Festnetz (mit Vorwahl) wählen und warten (im CLI können Sie beobachten, was passiert). Nach kurzer Zeit hören Sie auch schon das Freizeichen und können mit einem Festnetz-Teilnehmer telefonieren. Wie die obige Konfiguration genau funktioniert, wird an dieser Stelle noch nicht verraten.[19]

2.13. Gespräche vom öffentlichen Netz entgegennehmen

Der letzte Schritt ist ein kleiner. Wir wollen über unsere Rufnummer beim VoIP-Provider auf dem Telefon mit der Nummer 2000 angerufen werden können. Dazu müssen wir nur noch einen weiteren Context in die Konfigurationsdatei `/etc/asterisk/extensions.conf` hinzufügen:

```
[von-voip-provider]
exten => 5587572,1,Dial(SIP/2000)
```

Fertig! :-)

Die *5587572* ist Ihre interne Rufnummer bei Ihrem VoIP-Provider. Bitte schauen Sie auf der Webseite des Providers in Ihren Kundendaten diese Nummer nach. Dass die Rufnummer hier dieselbe ist wie der Username, ist prinzipiell Zufall.

In diesem Einführungs-Kapitel wollte ich Ihnen zeigen, wie schnell und einfach man mit Asterisk eine funktionierende Telefonanlage aufbauen kann. Im verbleibenden Teil des Buches gehe ich nun tiefer in die Details und zeige Ihnen, was man mit Asterisk alles machen kann.

[19] Nur so viel an dieser Stelle: Die Variable `${EXTEN}` speichert die vom Benutzer gewählte Nummer ab (siehe Kapitel 4, *Programmieren im Dialplan*).

Kapitel 3. Die große Telefonanlage

Versionsgeschichte
Version Rechtsschreiblektorat 23.12.2006

1. Einleitung

Nachdem Sie in Kapitel 2, *Installation und "Hello World"* gesehen haben, wie man schnell und einfach eine kleine Asterisk-Telefonanlage aufsetzen kann, werden im folgenden Kapitel am Beispiel einer fiktiven Firma die umfangreichen Möglichkeiten und Funktionalitäten von Asterisk aufgezeigt und beschrieben.

2. Die Apfelmus GmbH

Unsere frei erfundene Firma produziert und verkauft Apfelmus. Im neuen Firmensitz - für den wir eine neue Telefonanlage planen sollen - gibt es folgende Bereiche:

- Geschäftsführung (2 Personen)
- Sekretariat (3 Personen)
- Verkauf National (6 Personen)
- Verkauf Ausland (4 Personen)
- Produktion (50 Personen)
- Versand (10 Personen)
- IT (2 Personen)
- Hausmeister (1 Person)

Jeder Mitarbeiter hat sein eigenes Telefon.

3. Der Rufnummernplan

Obwohl wir prinzipiel mit 2-stelligen Telefonnummern auskommen würden, werden wir unseren Rufnummernplan direkt 3-stellig aufsetzen. Damit haben wir im Fall einer Erweiterung ausreichend Platz im Nummernraum. Außerdem wird mancher Mitarbeiter noch ein Fax erhalten oder bestimmte Gruppen werden eine Sammelrufnummer benötigen.

Rufnummern	**Beschreibung**	**Begründung**
110, 112, 911	Notrufnummern	Im Notfall sollten wichtige Nummern völlig intuitiv sein. Wenn es brennt, sollte kein Mitarbeiter überlegen müssen, welche Nummer er wählen soll. Die 110 und die 911 gehen zur Polizei, die 112 zur Feuerwehr.
150 - 159	Hausmeister	Der Hausmeister spielt eine zentrale Rolle und bekommt deshalb auch eine Nummer im 100er Block.
160 - 169	IT	Da auch die EDV-Abteilung eine zentrale Rolle im Unternehmen spielt, bekommt sie einen Teil des 100er Blocks.

200 - 229	Geschäftsführung	Die Geschäftsführung bekommt einen Bereich aus 29 Nummern.
230 - 269	Sekretariat	Das Sekretariat bleibt auch im 200er Block. Der Bereich von 270 bis 299 bleibt zur Reserve frei.
300 - 349	Verkauf National	Der 300er Block gehört ganz dem Verkauf und wird mit reichlich Reserve ausgestattet. Die erste Hälfte geht an den Inlandsvertrieb.
350 - 399	Verkauf Ausland	Die obere Hälfte des 300er Blocks ist für den Auslandsvertrieb.
400 - 449	Versand	Der Versand bekommt am Anfang des 400er Blocks auch reichlich Reserve.
500 - 599	Produktion	Die Produktion ist am größten und bekommt deshalb den ganzen 500er Block.
800 - 899	Interne Servicenummer	z.B. Voicemailsystem oder Konferenzräume

4. Auswahl der Infrastruktur

Da in dem neuen Gebäude bereits eine strukturierte und moderne Netzwerkverkabelung vorliegt, ist es eine naheliegende Idee, jeden Arbeitsplatz mit einem SIP-Telefon auszustatten. So müssen nicht extra Kabel für das Telefonnetz verlegt werden. Die Faxe und schnurlosen Telefone werden über SIP-to-Analog-Adapter angeschlossen. Der Asterisk-Server wird in einem ordentlich ausgestatteten Serverraum mit entsprechender Klimatisierung untergebracht, mit Sicherheitseinrichtungen und Zugangskonzept. Durch das Zugangskonzept und die Sicherheitseinrichtungen haben auch nur diejenigen Personen Zugriff auf die Anlage, die dazu berechtigt sind.[1]

4.1. Netzwerk

Die wichtigste Regel, der man sich stets bewusst sein sollte, ist, dass Telefonie ein synchrones Medium ist, d.h. die Informationen müssen für den Sender und den Empfänger praktisch "gleichzeitig" vorliegen und dies über die Dauer der gesamten Kommunikation. Während wir in einem Telefongespräch Verzögerungen bis ca. 300 ms noch einigermaßen tolerieren, werden Pausen von einer halben Sekunde und mehr als sehr störend empfunden und kaum akzeptiert. Bei der IP-Telefonie wird ein synchrones Kommunikationsmittel über eine asynchrone Technik transportiert. Die Funktionsweise des TCP/IP-Protokolls des Internets wurde auf Störfestigkeit und Robustheit hin ausgelegt und nicht auf einen Datentransport in Echtzeit. Für den Transport von E-Mails zum Beispiel ist es völlig unerheblich, ob im Datentransport mehrere Unterbrechungen von einigen Sekunden liegen. Dass es überhaupt möglich ist, Telefonie via IP unter diesen Bedingungen erfolgreich umzusetzen, liegt an der Leistungsfähigkeit der heutigen Netzwerktechnologie. Dadurch, dass die zu übertragene Datenmenge im Verhältnis zur vorhandenen Übertragungskapazität deutlich geringer ist, können die Anforderungen quasi in Echtzeit umgesetzt werden. Bei den meisten Netzwerken liegen jedoch häufig unterschiedliche Lastzustände vor, d.h. die zur Verfügung stehende Bandbreite wird zu unterschiedlichen Zeiten mal stärker und mal weniger stark ausgenutzt. In fast allen Unternehmen kann man beobachten, wie die Auslastung des Netzwerks dem Rhythmus des Arbeitstages folgt. Zwischen 8 Uhr morgens und 18 Uhr abends steigt die Auslastung in der Regel, da die Mitarbeiter zum Beispiel E-Mails schreiben, im Web surfen, Daten herunterladen etc. Zu den Zeiten, in denen die Mitarbeiter intensiv arbeiten, telefonieren sie aber auch, d.h. unsere IP-Telefone konkurrieren mit dem restlichen Netzwerkverkehr um die zur Verfügung stehende Bandbreite. Und während es für die Zustellung einer E-Mail kein größeres Problem darstellt, wenn die Verbindungsgeschwindigkeit sinkt -- es dauert halt etwas länger --, kann es für unsere IP-Telefone kritisch werden, wenn sie unter den Mindestbedarf fällt. Die Sprachqualität leidet dann deutlich, es kommt zu Aussetzern und

[1] Noch vor 10 Jahren war es bei vielen Telefonanlagenherstellern üblich, sich "Hintertüren" in große Anlagen einzubauen. So konnte man ohne großen Aufwand bei Problemen schnell helfen, indem man sich über einen offenen Port von außen in die Anlage einwählte. Böse Geister nutzten diese Möglichkeit allerdings auch, um auf Kosten der Eigentümer (meistens große Behörden, in denen die Übersicht fehlte) Fern- oder Auslandsgespräche zu führen.

Sprachfetzen. Wie störend das ist, kennt sicherlich jeder von Telefonaten mit dem Mobiltelefon bei einer schlechten Verbindung. Um mit den unterschiedlichen Anforderungen der jeweiligen Verbindungsarten besser umgehen zu können, verfügen professionelle Netzwerkrouter über die Möglichkeit, Netzwerkverkehr je nach Typ zu priorisieren.[2]

Diese Maßnahme ist zwar geeignet, den Netzwerkverkehr nach innen und außerhalb des eigenen Netzwerkanschlusses zum Internet-Provider entsprechend zu regeln, jedoch hat dies keinen Einfluss auf die Lastsituation im Netz des Providers. Erst wenn der Provider entsprechende Garantien für Übertragungskapazitäten für bestimmte Verbindungstypen[3] anbietet, kann die Kette lückenlos geschlossen werden. Falls Sie also mehrere Standorte in Ihrem Unternehmen via IP-Telefonie verbinden möchten, sollten Sie dies in Ihrer Planung berücksichtigen und mit Ihrem Provider darüber sprechen. Die Telefonanlage der Apfelmus GmbH bedient 78 Endgeräte. Das heißt, dass im schlimmsten Fall 39 zeitgleiche interne Verbindungen zustande kommen können, wenn die Mitarbeiter sich nur intern anrufen. Abhängig vom verwendeten Codec für die Kodierung des Audiosignals ergibt sich ein Netzwerkverkehr von maximal 6.500 kbps[4], was für die heute gängigen 100 Mbps[5] an Netzwerkbandbreite in strukturierten Inhouse-Verkabelungen keine Herausforderung darstellt. Dennoch sollten Sie berücksichtigen, dass anderer Netzwerkverkehr ebenfalls übertragen wird und die verfügbare Bandbreite rasch an Grenzen stoßen kann. Wie kommt der Wert von 6.500 kbps zu Stande? Für die Inhouse-Verbindungen greifen wir in unserem Beispiel auf den a-law[6] Codec zurück, der auch im ISDN-Netz zum Einsatz kommt. Er bietet eine sehr gute Sprachqualität, benötigt dafür aber auch 64 kbps Bandbreite. Jede Verbindung besteht aus einem eingehenden und einem ausgehenden Kanal mit jeweils 64 kbps, die 39 Verbindungen summieren sich auf: 2*39*64 kbps = 4.992 kbps. Zum Bandbreitenbedarf des Codecs kommt noch ein Overhead durch das TCP/IP-Protokoll hinzu, so dass aus den 64 kbps ca. 80 kbps an Netzwerkverkehr werden. Legt man der Rechnung nun die 80 kbps zu Grunde, ergibt das 6.280 kbps, aufgerundet erhält man die genannten 6.500 kbps. Dabei sind die 6.500 kbps natürlich nur ein Worst-Case-Szenario. Trotzdem sollte man immer für genau dieses gerüstet sein![7]

Tipp

Wer sich am Anfang einer solchen Installation nicht sicher ist, ob die Netzwerkbandbreite ausreicht, sollte lieber einen verlustbehafteteren Codec wie GSM benutzen. Der benötigt mit 13-15 kbps nur ein Fünftel der Bandbreite vom a-law Codec und hat eine akzeptable Sprachqualität. Später kann man dann einzelne Bereiche Stück für Stück auf bessere Codecs umstellen und dabei beobachten, ob es Netzwerkprobleme gibt. Sie sollten dabei jedoch die Auslastung der CPU im Auge behalten, da die komprimierenden Codecs deutlich mehr Rechenleistung in Anspruch nehmen.

4.2. Server-Hardware

Die Anforderungen, die ans Netzwerk gestellt werden, treffen im Prinzip auch auf die verwendete Hardware des Rechners zu. Die Notwendigkeit der synchronen Datenübertragung erfordert auch eine entsprechend zeitnahe Abarbeitung der notwendigen Vorgänge durch den verwendeten Rechner. Hierbei gibt die Kodierung und Dekodierung des Sprachsignals den Leistungsbedarf vor, insbesondere dann, wenn die beteiligten Endgeräte unterschiedliche Codecs verwenden und Asterisk die ankommenden und ausgehenden Datenströme umkodieren muss. Aufgrund der großen Bandbreite an verfügbarer Hardware und dem jeweils spezifischen Leistungsvermögen gibt es keine einfache Regel für die Bemessung. Als eine erste Orientierung soll folgende Staffelung dienen:

CPU	Anzahl der Verbindungen	Anzahl der Nebenstellen
1,0 GHz	15	40
1,5 GHz	40	100
2,0 GHz	80	200

[2] TOS - Type Of Service
[3] QOS - Quality Of Service
[4] kbps = kilobits per second
[5] Mbps = Megabits per second, also 1.024 kbps
[6] Eigentlich G.711, der in zwei Varianten auftritt als a-law und u-law.
[7] Auf der Webseite http://www.asteriskguru.com/tools/bandwidth_calculator.php finden Sie ein einfaches Online-Tool zur Bandbreitenberechnung verschiedener Codecs.

Bei der Berechnung der möglichen Anzahl von Nebenstellen gehen wir davon aus, dass nicht alle Nutzer zur gleichen Zeit telefonieren, sondern im Mittel ein Drittel bis die Hälfte. Bitte betrachten Sie die angegebenen Werte lediglich als eine grobe Orientierung, die zudem eher konservativ geschätzt wurden, d.h. mit deutlicher Reserve. Fallen wenig Umkodierungen, Konferenzschaltungen und Echo-Unterdrückungen an, dann können deutlich mehr gleichzeitige Verbindungen bewältigt werden. Die genaue Zahl kann oftmals nur durch sinnvolles Ausprobieren ermittelt werden.

Tipp

Unter Linux gibt es mit **top** ein einfaches Tool, um im laufenden Betrieb zu sehen, wie hoch die Last auf dem Rechner ist. Allerdings gibt Ihnen das natürlich immer nur den aktuellen Wert und reicht nicht für eine Langzeitanalyse. Dazu können Sie leistungsstarke Tools wie Nagios http://www.nagios.org verwenden. Wer sich mit "Bordmitteln" weiterhelfen will, kann auch einen cronjob (siehe **man crontab**) mit folgendem Befehl aufsetzen:

```
w | head -n 1 | logger
```

Damit kann man sich (entsprechend in `/etc/cronjob` eingestellt) jede Minute einmal die aktuelle Systemlast in die Datei `/var/log/messages` schreiben lassen und diese dann am Monatsende auswerten.[8]

In unserem Beispiel haben wir 78 Benutzer, bei denen im Mittel nicht mehr als 40 gleichzeitige Verbindungen vorliegen sollten, somit wäre ein Rechner mit einer halbwegs aktuellen CPU[9]

mit 1,5 GHz Taktfrequenz geeignet. Je aktueller CPU und Gesamtsystem sind, desto höher ist in der Regel die Leistung und es können bei gleicher Taktfrequenz deutlich mehr Gespräche gleichzeitig abgearbeitet werden. Ein Dual-CPU System verbessert hier den Wert ebenfalls, da die Rechenleistung für notwendige Aufgaben des Betriebssystems besser verteilt wird. Für die Voicemailboxen benötigt man ca. 0,1 MByte pro aufgezeichneter Minute[10] an Speicherplatz. Stattet man jeden Nutzer mit 30 Minuten aus, wären für unser Beispiel ca. 2 GByte an Plattenplatz ausreichend großzügig bemessen. Da die aufgezeichneten Daten denselben Stellenwert wie Geschäftsdaten haben, sollten sie entsprechend gegen Ausfall und Verlust abgesichert werden. Ein RAID-System[11] mit zwei Festplatten im Mirroring-Modus (Level 1) sollte hierbei ausreichend Absicherung gewährleisten. Für unser Beispiel würden zwei 18 GByte SCSI-Festplatten im RAID 1-Verbund ausreichend sein. Die Anforderungen an den verfügbaren Hauptspeicher sind sehr moderat. Mit 512 MByte ist das System ausreichend bemessen und ab 1 GByte können selbst große Gruppen problemlos versorgt werden. Alle zusätzlichen Maßnahmen, wie die Auswahl professioneller Hardware für den Einsatz in Servern, die Ausstattung mit redundanten Netzteilen und mit hochwertigen Komponenten erhöhen die Ausfallsicherheit und Betriebstreue des Systems und damit in Folge die Akzeptanz durch die Nutzer - die es eher gelassen nehmen, wenn der E-Mail-Server mal 10 Minuten offline ist, aber das Telefon muss immer funktionieren.

5. Die Grundkonfiguration

Als Erstes wird die Telefonanlage so konfiguriert, dass alle Teilnehmer intern miteinander telefonieren können und jeder eine persönliche Voicemailbox hat. Danach kann die Anlage mit einzelnen Bausteinen erweitert werden.

5.1. sip.conf für fast 100 Teilnehmer

Der Aufbau einer `/etc/asterisk/sip.conf` für 2 Teilnehmer unterscheidet sich nicht vom Aufbau für 100 Teilnehmer. In der `sip.conf` müssen alle SIP-Telefone eingetragen werden. Folgend finden Sie die gekürzte Fassung der Konfigurationsdatei `sip.conf` für die Apfelmus GmbH. Kommentare werden hier immer mit `;` eingeleitet. Diese Konfigurationsdatei ist analog zum Beispiel im Kapitel 1 aufgebaut. Bei der Auswahl der Rufnummern wurden an die Teilnehmer vornehmlich Nummern vergeben, die nicht auf die Ziffer 0 enden. Der Grund hierfür

[8] Es sollte jedem Leser klar sein, dass dieser Cronjob ein wirklich nur ganz einfaches Mittel zur Analyse ist.
[9] Zum Beispiel Pentium 4, AMD Athlon
[10] Dies gilt für das Format wav49 oder GSM, wenn Sie hingegen das unkomprimierte Format WAV verwenden, steigt der Platzbedarf um ein Vielfaches.
[11] Bitte beachten Sie, dass ein Software-RAID die CPU entsprechend zusätzlich belastet, was in unserem Fall eher unerwünscht ist, deshalb ist eine vernünftige Lösung in Hardware angeraten.

ist, dass die Endziffer 0 an späterer Stelle für die Zuteilung von Gruppenrufnummern dienen soll. Lediglich beim Hausmeister wurde eine Ausnahme gemacht, da für ihn keine Gruppenrufnummer vorgesehen ist.

```
[general]
port = 5060
bindaddr = 0.0.0.0
context = sonstige

; Hausmeister
;
[150]  ; normales Telefon im Buero
context=hausmeister
secret=1234
callerid="Hausmeister" <150>
type=friend
host=dynamic

[151]  ; Mobiltelefon
context=hausmeister
secret=1234
callerid="Hausmeister" <150>
type=friend
host=dynamic

; Interne IT Abteilung
;
[161]  ; Mitarbeiter 1
context=it
secret=1234
type=friend
host=dynamic

[162]  ; Mitarbeiter 2
context=it
secret=1234
type=friend
host=dynamic

[165]  ; Mobiles Telefon
context=it
secret=1234
type=friend
host=dynamic

; Geschaeftsfuehrung
;
[201]  ; Geschaeftsfuehrer 1
context=geschaeftsfuehrung
secret=1234
type=friend
host=dynamic

[202]  ; Geschaeftsfuehrer 2
context=geschaeftsfuehrung
secret=1234
type=friend
host=dynamic

; Sekretariat
;
[231]  ; Assistentin 1
context=sekretariat
secret=1234
type=friend
host=dynamic

[232]  ; Assistentin 2
context=sekretariat
secret=1234
```

```
type=friend
host=dynamic

[233]  ; Assistentin 3
context=sekretariat
secret=1234
type=friend
host=dynamic

; Verkauf National
;
[301]  ; Verkaeufer 1
context=verkauf-national
secret=1234
type=friend
host=dynamic

[302]  ; Verkaeufer 2
context=verkauf-national
secret=1234
type=friend
host=dynamic

[303]  ; Verkaeufer 3
context=verkauf-national
secret=1234
type=friend
host=dynamic

[304]  ; Verkaeufer 4
context=verkauf-national
secret=1234
type=friend
host=dynamic

[305]  ; Verkaeufer 5
context=verkauf-national
secret=1234
type=friend
host=dynamic

[306]  ; Verkaeufer 6
context=verkauf-national
secret=1234
type=friend
host=dynamic

; Verkauf Ausland
;
[351]  ; Verkaeufer 1
context=verkauf-ausland
secret=1234
type=friend
host=dynamic

[352]  ; Verkaeufer 2
context=verkauf-ausland
secret=1234
type=friend
host=dynamic

[353]  ; Verkaeufer 3
context=verkauf-ausland
secret=1234
type=friend
host=dynamic

[354]  ; Verkaeufer 4
context=verkauf-ausland
secret=1234
```

```
type=friend
host=dynamic

; Versand
;
[401]  ; Mitarbeiter 1
context=versand
secret=1234
type=friend
host=dynamic

; die weiteren 9 Eintraege fuer den
; Versand sind hier ausgelassen.

; Produktion
;
[501]  ; Mitarbeiter 1
context=produktion
secret=1234
type=friend
host=dynamic

; die weiteren 49 Eintraege fuer die
; Produktion sind hier ausgelassen.
```

5.1.1. CallerID

In der `sip.conf` kommt zweimal die Variable `callerid` vor. Mit dieser Variable wird festgelegt, was beim Angerufenen auf dem Display angezeigt wird. Im konkreten Beispiel:

```
callerid = "Hausmeister" <150>
```

Dies sagt aus, dass sich beide Telefone (150 und 151) mit der Caller-ID 150 und dem Text "Hausmeister" zu erkennen geben. Dadurch ist sichergestellt, dass Rückrufe immer nur auf die 150 gehen.

5.2. Der Wählplan (Dialplan)

Im ersten Kapitel haben wir schon einen einfachen Dialplan für zwei Telefone erstellt. Diesmal haben wir jedoch deutlich mehr Teilnehmer und obwohl es im Grunde lediglich Fleißarbeit bedeutet, ist die Pflege einer Konfiguration mit einigen Hundert Teilnehmern zeitaufwändig. Weiterhin fördert eine schlechte Übersichtlichkeit das Einschleichen von Fehlern. Aus diesen Gründen beschäftigen wir uns am Anfang dieses Abschnitts mit ein paar Asterisk-Funktionen, die uns das Leben einfacher und die `extensions.conf` übersichtlicher machen.

5.2.1. Platzhalter - Pattern Matching

Asterisk bietet die Möglichkeit mit Platzhaltern[12] zu arbeiten. Durch sinnvolles Gruppieren von Nummernblöcken und Zuordnungen können mit Platzhaltern ganze Nummernbereiche mit einzelnen Konfigurationszeilen verwaltet werden. Im Prinzip enthält eine Konfigurationszeile einen variablen Nummernbereich, der mit Hilfe von "Pattern Matching" die definierte Regel auf eine Vielzahl von Nummern anwendet. So kann man anstatt der folgenden 10 Zeilen:

```
exten => 2000,1,Dial(SIP/2000,20,j)
exten => 2001,1,Dial(SIP/2000,20,j)
exten => 2002,1,Dial(SIP/2000,20,j)
exten => 2003,1,Dial(SIP/2000,20,j)
exten => 2004,1,Dial(SIP/2000,20,j)
exten => 2005,1,Dial(SIP/2000,20,j)
exten => 2006,1,Dial(SIP/2000,20,j)
exten => 2007,1,Dial(SIP/2000,20,j)
exten => 2008,1,Dial(SIP/2000,20,j)
exten => 2009,1,Dial(SIP/2000,20,j)
```

auch nur eine Zeile schreiben:

[12] Unix- und Linux-Fans werden das Konzept des "Pattern Matching" von den "regulären Ausdrücken" in Programmen wie **grep** und **sed** kennen.

```
exten => _200X,1,Dial(SIP/2000,20,j)
```

Das `X` steht dann für alle Zahlen von 0 bis 9.[13] Wichtig dabei ist, dass der Suchbegriff (das Pattern) mit einem _ (Underscore) anfängt, sonst würde Asterisk nur auf `200X` (also eine 200 und den Buchstaben X) reagieren, da ja Nebenstellen nicht nur aus Ziffern bestehen können/müssen. Das Definieren von Platzhaltern mit Pattern Matching[14] beinhaltet, dass man bestimmte Zeichenketten oder Zahlenfolgen durch entsprechend vordefinierte Platzhalter ersetzt.

In Asterisk kann man folgende Patterns benutzen:

Pattern	**Beschreibung**
`X`	Alle Zahlen von 0 bis 9
`Z`	Alle Zahlen von 1 bis 9
`N`	Alle Zahlen von 2 bis 9[a]
`[nm]`	Die Zahlen n und m
`[n-m]`	Alle Zahlen von n bis m
`.`	Eine oder mehrere beliebige Zahlen und Buchstaben

[a]Der Grund für das Pattern `N` liegt in der Rufnummernplanung von Amerika. Dort beginnt die Vorwahl nicht wie in Deutschland mit einer 0, sondern mit einer 1. Beispiel: 1-555-12345678

Warnung

Ein Pattern muss immer mit einem _ (Underscore) anfangen!

Für Asterisk ist sowohl ein `2XXX` als auch ein `_2XXX` sinnvoll und wird unterschiedlich ausgewertet.

5.2.1.1. Beispiele für Patterns

Die folgenden Varianten dienen als Beispiele für die Möglichkeiten:

Pattern	**Beschreibung**
_XXX	Alle 3-stelligen Zahlen. Wobei auch `007` eine 3-stellige Zahl ist.
_XXX[13579]	Alle 4-stelligen Zahlen, die ungerade sind.
_[1-5]X	Alle 2-stelligen Zahlen von 10 bis 59.
_0.	Eine beliebige Zeichenkette, die mit einer 0 anfängt.
_.	Alles. Mit Vorsicht zu gebrauchen!

5.2.2. Die Variable ${EXTEN}

Obwohl wir eigentlich erst später über Variablen sprechen, möchte ich eine sehr einfache und intuitiv zu benutzende Variable schon hier vorstellen. Es handelt sich um `${EXTEN}`. In dieser Variable ist die gewählte Nummer gespeichert. Ich kann also in der `extensions.conf` anstatt:

```
exten => 2000,1,Dial(SIP/2000)
```

auch einfach

```
exten => 2000,1,Dial(SIP/${EXTEN})
```

[13]Das Beispiel macht natürlich wenig Sinn. Warum sollte der Anrufer die 2009 wählen, um zur 2000 verbunden zu werden? Zur Lösung dieses Problems kommen wir gleich.
[14]Es gibt keinen wirklich passenden deutschen Begriff zum Fachbegriff "Pattern Matching". Eine mögliche Übersetzung könnte "Musterabgleich" oder "Mustervergleich" lauten.

schreiben. Bei einer Zeile ist das natürlich noch wenig sinnvoll, aber wenn man diese Funktionalität mit Pattern Matching kombiniert, dann kann man sehr viel Zeit und Aufwand sparen und bekommt zusätzlich auch noch eine viel übersichtlichere Konfiguration.

Um somit alle SIP-Telefone mit den Durchwahlen 2000 bis 2999 in der `extensions.conf` anwählbar zu machen, reicht folgende Zeile:

```
exten => _2XXX,1,Dial(SIP/${EXTEN})
```

5.2.3. Include

Innerhalb der `extensions.conf` können Bereiche mit einem `include => Contextname` eingefügt werden. So kann man eine einmal erstellte Definition in mehreren Contexten wiederverwenden. Beispiel:

```
[telefone-im-ersten-stock]
include => 2000er
include => anrufbeantworter

[telefone-im-zweiten-stock]
include => 2000er
include => anrufbeantworter

[telefone-im-dritten-stock]
include => 2000er

[2000er]
exten => _2XXX,1,Dial(SIP/${EXTEN})

[anrufbeantworter]
exten => 3000,1,VoicemailMain(${CALLERID(num)})
```

Der Vorteil dieses Weges ist, dass wenn man z.B. die Rufnummer des Anrufbeantworters ändern will, es reicht, dieses genau in einem Bereich zu machen.

Tipp

Die Variable `${CALLERID(num)}` übergibt die Nummer des Anrufenden. So kann die Applikation `VoiceMailMain()` mit dieser Variable aufgerufen werden und fragt dann nicht mehr nach der Mailbox, sondern nur noch nach dem Passwort der entsprechenden Mailbox.

5.2.4. Die extensions.conf für die Apfelmus GmbH

Der von uns entwickelte Rufnummernplan[15] lässt sich nun wie folgt in eine übersichtliche `extensions.conf` übertragen:

```
[sonstige]

[hausmeister]
include => interne-gespraeche
include => voicemailsystem-komfort

[it]
include => interne-gespraeche
include => voicemailsystem-komfort
;
; Aus Debugging Gruenden ist es fuer
; die IT Abteilung teilweise nuetzlich
; auf alle Voicemailboxen zugreifen
; zu koennen.
;
include => voicemailsystem-normal

[geschaeftsfuehrer]
include => interne-gespraeche
```

[15] Mit Ausnahme der Notrufnummern, die wir später behandeln, da wir jetzt noch keine Verbindung zur Außenwelt haben.

```
include => voicemailsystem-komfort

[sekretariat]
include => interne-gespraeche
include => voicemailsystem-komfort

[verkauf-national]
include => interne-gespraeche
include => voicemailsystem-komfort

[verkauf-ausland]
include => interne-gespraeche
include => voicemailsystem-komfort

[versand]
include => interne-gespraeche
include => voicemailsystem-komfort

[produktion]
include => interne-gespraeche
include => voicemailsystem-komfort

[interne-gespraeche]
exten => _[1-5]XX,1,Dial(SIP/${EXTEN},60,j)
exten => _[1-5]XX,2,Voicemail(u${EXTEN})
exten => _[1-5]XX,102,Voicemail(b${EXTEN})

[voicemailsystem-komfort]
;
; Der User muss nicht die Nummer der
; Voicemailbox eingeben.
;
exten => 800,1,VoicemailMain(${CALLERID(num)})

[voicemailsystem-normal]
exten => 801,1,VoicemailMain()
```

Der Context `[interne-gespraeche]` definiert, dass alle Anrufe an die Nummern 100 bis 599 mit dem Programm Dial() auch mit dieser Nebenstelle verbunden werden. Wer die 800 anruft, bekommt die Voicemailbox für sein eigenes Telefon. Nur die IT-Abteilung kann die 801 anrufen und wird dann vom System erst nach der gewünschten Nebenstelle (Extension) gefragt. Da die IT-Abteilung auch den internen Support der Telefonanlage zur Verfügung stellt, benötigt sie diese Funktion zur Störungsermittlung (Debugging).

5.2.4.1. Schwarze Löcher im Rufnummernplan

Genau genommen müsste dieser Dialplan noch etwas komplexer sein, da es ja laut Rufnummernplan Bereiche gibt (z.B. 270 bis 299), die gar nicht mit Telefonen belegt sind. Da wir dies nicht beachten, kann es zu Missverständnissen kommen. Ein Anrufer kann eine nicht vergebene Rufnummer anrufen und dort auf dem Anrufbeantworter eine Nachricht hinterlassen. Diese Nachricht würde aber nie abgehört werden. Um dies zu vermeiden, müsste man korrekterweise den Context `[interne-gespraeche]` wie folgt gestalten:

```
[interne-gespraeche]
exten => _1[5-6]X,1,Dial(SIP/${EXTEN},60,j)
exten => _1[5-6]X,2,Voicemail(u${EXTEN})
exten => _1[5-6]X,102,Voicemail(b${EXTEN})

exten => _2[0-6]X,1,Dial(SIP/${EXTEN},60,j)
exten => _2[0-6]X,2,Voicemail(u${EXTEN})
exten => _2[0-6]X,102,Voicemail(b${EXTEN})

exten => _[358]XX,1,Dial(SIP/${EXTEN},60,j)
exten => _[358]XX,2,Voicemail(u${EXTEN})
exten => _[358]XX,102,Voicemail(b${EXTEN})

exten => _4[0-4]XX,1,Dial(SIP/${EXTEN},60,j)
exten => _4[0-4]XX,2,Voicemail(u${EXTEN})
exten => _4[0-4]XX,102,Voicemail(b${EXTEN})
```

Wir verzichten bei der Apfelmus GmbH der Einfachheit halber auf diese ausführlichere Variante und benutzen nur den folgenden Context:

```
[interne-gespraeche]
exten => _[1-5]XX,1,Dial(SIP/${EXTEN},60,j)
exten => _[1-5]XX,2,Voicemail(u${EXTEN})
exten => _[1-5]XX,102,Voicemail(b${EXTEN})
```

5.2.5. voicemail.conf

Die `voicemail.conf` bleibt wie im ersten Kapitel beschrieben. Es werden einfach nur ein paar mehr Einträge eingefügt:

```
[general]
format = gsm
serveremail = voicemail@apfelmus-gmbh.de
maxmessage = 600

[local]
150 => 999999,Hans Hausmeister,hausmeister@apfelmus-gmbh.de
200 => 999999,Ernst Wichtig,ernst.wichtig@apfelmus-gmbh.de
201 => 999999,Hans Toll,hans.toll@apfelmus-gmbh.de

; Ich verzichte hier darauf, die restlichen Eintraege aufzulisten.
```

6. Wie geht es weiter?

Die hier besprochene Konfiguration der Apfelmus GmbH dient als Planungsbeispiel für alle Anlagen dieser Größenordnung. Einige Beispiele in diesem Buch können Sie mit dieser Konfiguration durchspielen.

7. FAQ

7.1.1. Welche Kosten kommen auf die Apfelmus GmbH zu?

Das hängt ganz von den Entscheidern in dieser Firma ab. Folgende Faktoren spielen dabei eine Rolle:

- Soll der Asterisk-Server ausfallsicher sein? Reicht ein Cold-Standby oder soll es ein Hot-Standby sein?
- Welche Server-Hardware wird gekauft? Für eine solche Telefonanlage reicht ein relativ normaler Rechner. Es muss also nicht das allerneuste und damit auch teuerste System gekauft werden.
- Welche Telefone werden eingesetzt? Hierbei kann man von 30 bis weit über 300 Euro pro Telefon ausgeben.

7.1.2. Lohnt sich Asterisk für die Apfelmus GmbH oder handelt es sich vielmehr um eine Arbeitsbeschaffungsmaßnahme für die IT-Abteilung?

Das hängt stark von den Angeboten der klassischen Telefonanlagenherstellern und von den Bedürfnissen der Apfelmus GmbH ab. Oft bieten klassische Telefonanlagenhersteller extreme Sonderrabatte, wenn sie erfahren, dass als Alternative das kostenlose Asterisk in Erwägung gezogen wird. Allerdings ist man dann meistens in einem Knebelvertrag und kann die nächsten 5 Jahre nur bei diesem Hersteller einkaufen.

Einer der großen Vorteile von Asterisk ist die Wahl des Supports und des Consultings. Man kann entweder alles selber machen oder einen externen Consultant einkaufen. Sollte dieser nicht den Ansprüchen genügen, kann man ihn gegen einen anderen austauschen (das ist so mit einer klassischen Telefonanlage wegen der Vertragsbindung nicht möglich).

Ob sich Asterisk lohnt, muss jeder für sich ausrechnen. Allerdings spricht in den meisten Fällen sehr viel dafür.

Kapitel 4. Programmieren im Dialplan

Die Erstellung von intelligenten Funktionen oder Programmen kann in Asterisk extern über ein AGI Skript oder intern in der `extensions.conf` erfolgen. In diesem Kapitel beschäftigen wir uns mit dieser internen Variante.

In der Konfigurationsdatei `extensions.conf` wird der so genannte Dialplan definiert. Dabei ähnelt der Dialplan häufig einem BASIC Programm. Der Admin kann in einer einfachen Skriptsprache Programmabläufe und damit auch das Routing von Telefonaten erstellen. Ohne diesen Dialplan kann weder raus- noch reintelefoniert werden!

1. Context

Der Dialplan wird in verschiedene Abschnitte unterteilt. Diese Abschnitte heissen im Asterisk Jargon Contexte. Am Anfang eines Dialplanes muss es immer einen `[general]` Context für allgemeine Konfigurationen geben. Danachfolgende Contexte können beliebig genannt werden. Die Contexte bilden das Verbindungsstrück zwischen der Definition eines Telefones (z.B. SIP oder ISDN) und dem Dialplan. Für ein Telefon wird immer ein Context definiert. Beispiel aus einer sip.conf:

```
[2000]
type=friend
context=interne-telefone
secret=1234
host=dynamic
```

Das SIP-Telefon mit der Nummer 2000 ruft in dieser Konfiguration immer den Context `interne-telefone` auf. Wenn also ein Benutzer mit dem Telefon 2000 eine bestimmte Nummer wählt, dann sucht Asterisk im Context `interne-telefone` nach der entsprechenden Extension (also der dazu passenden Regel). Ist diese Extension nicht vorhanden, passiert auch nichts.

1.1. Syntax

Ein Context selbst wird eingeleitet als Text in einer eckigen Klammer. "Text" ist hierbei ein sinnvoller Name, der den Context benennt und als spätere Referenz für denselben verwendet wird. Alle Zeilen nach einer solchen Einleitung bis zum nächsten Context werden als Bestandteil (Regeln, Anweisungen) dieses Contextes behandelt:

```
[general]

[interne-telefone]
Regeln, Anweisungen, ...

[apfelmus]
Regeln, Anweisungen, ...
```

2. Extension

Die einzelnen Dialplan-Programme werden Asterisk intern Extensions genannt. Eine Extension wird nicht kompiliert, sondern bei jedem Durchlauf von Asterisk interpretiert. Das Einlesen erfolgt einmalig automatisch während des Startens des Asterisk Daemons.[1] Das Neu-Einlesen des Dialplanes kann aber auch im laufenden Betrieb im Command Line Interface durch den Befehl **reload now** bzw. durch den Befehl **extensions reload** forciert werden.

2.1. Syntax

Eine Extension besteht immer aus folgenden Teilen:

- Name oder Nummer der Extension

[1] Eine Ausnahme stellt hier die Asterisk RealTime Architecture (ARA) da. In einem ARA-System wird der Dialplan in einer Datenbank (z.B. MySQL) abgespeichert und dort von Asterisk bei jedem Anruf neu eingelesen (also nicht nur einmal beim Starten von Asterisk). So können Dialpläne auch im laufenden Betrieb ständig geändert werden. Nährere Informationen zu ARA finden Sie unter *http://www.voip-info.org/wiki/view/Asterisk+RealTime*

- Priorität (also der Programmzeilenzähler)
- Applikation. Das ist die Anweisung, die Asterisk ausführen soll.

```
exten => Name,Prioritaet,Applikation()
```

2.2. Priorität

Eine typische Extension besteht aus mehreren Schritten. Damit Asterisk diese Schritte in der richtigen Reihenfolge ausführen kann, braucht es eine Art Zähler. Das erinnert ein wenig an frühe BASIC-Programme, die auch am Anfang einer jeden Zeile einen solchen Zähler hatten. Dieser Zähler heißt bei Asterisk Priorität. Prioritäten werden der Reihenfolge nach abgearbeitet (es wird immer +1 gezählt). Wenn die nächste logische Priorität (Lücken sind nicht zulässig!) nicht definiert ist bricht Asterisk ab.

2.2.1. Ein hello-world Beispiel

Die folgende Extension wird immer ausgelöst, wenn ein Telefon mit dem Context `apfelmus` die Nummer 8888 anruft. Asterisk nimmt dann ab, spielt den Sprachbaustein hello-world ab und legt auf.

```
[apfelmus]
exten => 8888,1,Answer()
exten => 8888,2,Playback(hello-world)
exten => 8888,3,Hangup()
```

2.2.2. n-Priorität

Seit der Asterisk Version 1.2.0 ist es möglich, Prioritäten nicht nur streng mit Zahlen, sondern auch mit dem Platzhalter `n` zu belegen. Der n-Zähler fungiert hierbei als ein automatischer Programmzähler. Jedesmal, wenn die Programmsteuerung auf die n-Priorität stößt, addiert sie 1 zum letzten Wert der Priorität. Dies ist dann hilfreich, falls Sie viele aufeinander folgende Regeln definiert haben und Sie eine Regel einfügen möchten, denn dann müssen Sie nicht mehr die Zähler der nachfolgenden Regeln neu nummerieren. Wenn eine normale Extension wie folgt aussieht:

```
exten => 1234,1,Answer()
exten => 1234,2,Wait(2)
exten => 1234,3,Play(carried-away-by-monkeys)
exten => 1234,4,Wait(2)
exten => 1234,5,Hangup()
```

kann man die gleiche Extension auch mit der n-Priorität definieren:

```
exten => 1234,1,Answer()
exten => 1234,n,Wait(2)
exten => 1234,n,Play(carried-away-by-monkeys)
exten => 1234,n,Wait(2)
exten => 1234,n,Hangup()
```

Dies kann nicht nur an der zweiten Priorität, sondern an einer beliebigen Stelle passieren:

```
exten => 1234,1,Answer()
exten => 1234,2,Wait(2)
exten => 1234,3,Play(carried-away-by-monkeys)
exten => 1234,n,Wait(2)
exten => 1234,n,Hangup()
```

Die meisten Beispiele in diesem Buch verwenden die traditionelle Art (also ohne die `n`-Priorität), weil es Applikationen (z.B. `Dial()`) gibt, die eine bestimmte Anzahl von Prioritäten (nämlich 101) hochzählen, falls ein bestimmtes Ereignis eintritt. Dies ist mit der n-Priorität nicht mehr einfach abbildbar.

2.2.3. Prioritäten mit Label

Es gibt aber auch noch die Möglichkeit, für `n`-Prioritäten Label zu vergeben. Damit sind z.B. solche Konstrukte möglich:

```
...
exten => s,n,GotoIf($[${ACCEPT} = 1 ] ?yes:no)
exten => s,n(yes),SetVar(MACRO_RESULT=CONTINUE)
exten => s,n(no),System(/bin/rm ${ARG1})
```

2.3. Regular Expressions

Mit unserem bisherigen Wissen, müssen wir pro möglicher Rufnummer immer eine eigene Extension schreiben. Dies würde schon nach kurzer Zeit sehr lange und fehleranfällige Dialpläne nach sich ziehen. Sollen z.B. die Rufnummern 100 bis 109 jeweils immer den hello-world Sprachbaustein abspielen, so würde die extensions.conf wie folgt aussehen:

```
[general]

[apfelmus]
exten => 100,1,Answer()
exten => 100,2,Playback(hello-world)
exten => 100,3,Hangup()

exten => 101,1,Answer()
exten => 101,2,Playback(hello-world)
exten => 101,3,Hangup()

exten => 102,1,Answer()
exten => 102,2,Playback(hello-world)
exten => 102,3,Hangup()

exten => 103,1,Answer()
exten => 103,2,Playback(hello-world)
exten => 103,3,Hangup()

exten => 104,1,Answer()
exten => 104,2,Playback(hello-world)
exten => 104,3,Hangup()

exten => 105,1,Answer()
exten => 105,2,Playback(hello-world)
exten => 105,3,Hangup()

exten => 106,1,Answer()
exten => 106,2,Playback(hello-world)
exten => 106,3,Hangup()

exten => 107,1,Answer()
exten => 107,2,Playback(hello-world)
exten => 107,3,Hangup()

exten => 108,1,Answer()
exten => 108,2,Playback(hello-world)
exten => 108,3,Hangup()

exten => 109,1,Answer()
exten => 109,2,Playback(hello-world)
exten => 109,3,Hangup()
```

Unter Verwendung einer regular Expression

Tipp

Definition Regular Expression:

"Reguläre Ausdrücke (Abk. RegExp oder Regex, engl. regular expressions) dienen der Beschreibung von (Unter-)Mengen von Zeichenketten mit Hilfe syntaktischer Regeln. Sie finden vor allem in der Softwareentwicklung Verwendung; für fast alle Programmiersprachen existieren Implementierungen." (zitiert aus *http://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck*)

sieht der gleiche Dialplan gleich viel handlicher aus:

```
[general]

[apfelmus]
exten => _10X,1,Answer()
exten => _10X,2,Playback(hello-world)
exten => _10X,3,Hangup()
```

2.3.1. Syntax

Eine regular Expression wird immer mit einem Unterstrich (_) vor der Expression eingeleitet:

```
exten => _Regular Expression,Prioritaet,Applikation
```

Eine regular Expression kann in Asterisk aus den folgenden Elementen[2] bestehen:

- `X`

  Beliebige Zahl von 0 bis 9.

- `[A-B]`

  Beliebige Zahl von A bis B. Beispiel für alle Zahlen von 1 bis 5:

  ```
  exten => _[1-5],1,NoOp(Test)
  ```

- `[AB]`

  Die Zahlen A und B. Beispiel für die Zahlen 54 und 57:

  ```
  exten => _5[47],1,NoOp(Test)
  ```

- `.`

  Beliebige Anzahl von Zeichen (mindestens aber eines). Alle Rufnummern die mit einer 0 beginnen können mit folgender Extension behandelt werden:

  ```
  exten => _0.,1,NoOp(Test)
  ```

3. Grundlegende Applikationen

Um die Programmierbeispiele in diesem Kapitel halbwegs sinnvoll zu gestalten, benötigen wir folgende Applikationen:

- `Answer()`

  Die `Answer()`-Applikation dient dazu einen Verbindungsversuch zu akzeptieren. Wenn ein Channel klingelt, dann kann `Answer()` den virtuellen Hörer abnehmen.

- `Hangup()`

  `Hangup()` ist das Gegenstück zu `Answer()`. Die Verbindung wird getrennt, der virtuelle Hörer aufgelegt.

- `Playback(Soundfile)`

  Mit `Playback()` kann man Sounddateien abspielen. Diese finden sich (wenn kein anderes Verzeichnis angegeben worden ist) im Verzeichnis `/var/lib/asterisk/sounds/`. Die Dateiendung wird dabei nicht angegeben (Asterisk sucht sich den optimalen Codec selbstständig raus).

[2] Es gibt noch weitere Elemente die im deutschen Sprachraum aber im allgemeinen wenig Sinn machen. Aus diesem Grund werden sie hier nicht aufgeführt.

- `Wait(Zahl)`

  Mit `Wait()` kann man eine Pause abrufen. Die Zahl in der Klammer gibt die Anzahl der zu wartenden Sekunden an.

- `NoOp(String)`

  Die Applikation `NoOp()` macht nichts. NoOP steht für No-Operation. Sie ist aber ein praktisches Tool, um Dialpläne zu debuggen. Der Inhalt des übergebenen Strings wird auf dem CLI (Console) ausgegeben. Im CLI muss dafür aber der Verbose Level auf mindestens 4 eingestellt sein (einfach im CLI **set verbose 4** eingeben).

4. Variablen

Variablen sind Platzhalter für konkrete Werte. Diese konkreten Werte sind abhängig von der Definition der Variablen, also dem Typ, und können bei Asterisk Zahlen, Buchstaben und Buchstabenfolgen sein. Variablen dienen dazu das kompilierte Programm flexibler zu gestalten und für unterschiedliche oder wechselnde Einsatzzwecke anzupassen. Die Verwendung von Variablen ermöglicht erst den individuellen Einsatz eines Programms, ohne die erneute Übersetzung des Programmcodes in ein ausführbares Programm.

Tipp

Wer noch nie programmiert hat oder mit Variablen in Berührung gekommen ist, sollte sich an dieser Stelle kurz bei *http://de.wikipedia.org/wiki/Variable_%28Programmierung%29* in die Materie einlesen.

Variablen haben bei Asterisk unterschiedliche Reichweiten. Es gibt lokale (im Asterisk Jargon Channel-Variablen genannt) Variablen, die Werte nur für den jeweiligen und aktiven Channel (also das aktuelle Gespräch) setzen und globale Variablen, die Werte für alle Channels setzen. Die bereits von Asterisk vorgesehenen Variablen haben wir ja in Form der Parameter in den Konfigurationsdateien schon ausführlich kennen gelernt. Neben diesen gibt es auch die Möglichkeit eigene Variablen zu definieren und diese in den Konfigurationsdateien zu verwenden.

4.1. Variablen in einer Extension auslesen

Der Wert einer Variable kann immer in der Syntax `${VARIABLENNAME}` ausgelesen werden. Es gibt Variablen die vom Asterisk System automatisch gesetzt werden. So wird z.B. die angerufene Nummer immer in der Variable `EXTEN` abgespeichert. In Kombination von regular Expressions und Einsatz von Variablen kann man somit einen langen Dialplan häufig stark komprimieren.

Beispiel vorher:

```
exten => 100,1,Dial(SIP/100)
exten => 101,1,Dial(SIP/101)
exten => 102,1,Dial(SIP/102)
exten => 103,1,Dial(SIP/103)
exten => 104,1,Dial(SIP/104)
exten => 105,1,Dial(SIP/105)
exten => 106,1,Dial(SIP/106)
exten => 107,1,Dial(SIP/107)
exten => 108,1,Dial(SIP/108)
exten => 109,1,Dial(SIP/109)
```

Beispiel nachher:

```
exten => _10X,1,Dial(SIP/${EXTEN})
```

4.2. Allgemeines

Variablennamen müssen nicht wie in unseren Beispielen groß geschrieben werden und sind auch nicht an Groß- und Kleinschreibung gebunden. Die Großschreibung von Variablen führt aber meistens zu besser lesbarem Code, da Sie bereits an der Schreibweise erkennen können, dass es sich um eine Variable handelt. Umgekehrt bedeutet das aber auch, dass Groß- und Kleinschreibung nicht für die Unterscheidung von Variablen verwendet werden kann (z.B. EXTEN, Exten, ExTen referenzieren alle nur eine Variable).

4.2.1. Strings

Strings (als Variablen, die keine Zahlen, sondern Text enthalten) sollten immer in Anführungszeichen gesetzt werden. Asterisk akzeptiert sie aber auch ohne diese Markierung. So sind die folgenden zwei Zeilen im Ergebnis gleich:

```
exten => 1234,1,Set(OBST=Apfel)
exten => 1234,2,Set(OBST="Apfel")
```

Sollte der String ein Komma oder ein Leerzeichen enthalten, dann *müssen* Sie Anführungszeichen verwenden, da diese sonst als Kommandos und nicht als Strings interpretiert werden:

```
exten => 1234,1,Set(OBSTSORTEN="Apfel, Birne, usw.")
```

4.2.2. Quoting

Bei einer Variablen, die Zeichen oder Zeichenfolgen (Wörter, Text) als Werte enthält, kann es vorkommen, dass man bestimmte Zeichen verwenden möchte, die bereits für andere Funktionen reserviert sind. Wollen Sie zum Beispiel ein Underscore-Zeichen als Bestandteil des Variablenwertes verwenden, müssen Sie es besonders kennzeichnen, bzw. maskieren. Diese Kennzeichnung nennt man beim Programmieren escapen. Folgende Zeichen müssen auf jeden Fall maskiert ("escaped") werden:

```
[ ] $ " \
```

Das Quoting in der `extensions.conf` erfolgt mit einem vorangestellten `\` (Backslash).

Beispiel:

```
exten => 1234,1,Set(BETRAG="10,00 US \$")
```

4.2.3. Integer

Bestehen Variablen aus einer Ganzzahl (Integer) so kann diese maximal 18 Stellen betragen. Bei Zahlen, die größer sind, tritt ein Fehler auf, der im Logfile protokolliert wird.

> **Tipp**
>
> Wer mit größeren oder Kommazahlen (Real) arbeiten will, kann dies mit einem AGI-Skript realisieren (siehe AGI Kapitel).

4.3. Globale Variablen in der extensions.conf definieren

Eine globale Variable kann am Anfang der `extensions.conf` gesetzt werden. Dies muss im besonderen Context `[globals]` erfolgen (er folgt dem Context `[general]`).

Beispiel:

```
[general]


[globals]
KLINGELZEIT=90

[from-intern]
exten => _XXX,1,Dial(SIP/${EXTEN},${KLINGELZEIT})
exten => _XXX,2,Voicemail(u${EXTEN})
exten => _XXX,102,Voicemail(u${EXTEN})
```

4.4. Variablen mit Set() definieren

Die Applikation `Set()` wird benutzt um eine Variable in einer Extension zu setzen.

4.4.1. Syntax

```
Set(Variablenname=Wert[|Variablenname2=Wert2][|Option])
```

Als Option kann ein *g* angegeben werden. Mit diesem *g* kann mit `Set()` eine globale Variable gesetzt werden. Ohne diese Option geht Asterisk immer von lokalen Channel-Variablen aus.

Beispiel:

```
; Eine globale Variable setzen:
exten => 10,1,Set(KLINGELZEIT=90|g)

; Eine Channel Variable setzen:
exten => 10,2,Set(LIEBLINGSOBSTSORTE="Apfel")

; Zwei Channel Variablen auf einmal setzen:
exten => 10,3,Set(VAR1=10|VAR2=23)

; Die Variablen auf dem CLI ausgeben
exten => 10,4,NoOp(KLINGELZEIT = ${KLINGELZEIT})
exten => 10,5,NoOp(LIEBLINGSOBSTSORTE = ${LIEBLINGSOBSTSORTE})
exten => 10,6,NoOp(VAR1 = ${VAR1})
exten => 10,7,NoOp(VAR2 = ${VAR2})
```

4.5. Vererbung von Channel Variablen

Wird im Lauf eines Gespräches ein weiterer Channel aufgebaut, so hat dieser natürlich wieder eigene Channel-Variablen.

4.5.1. Einstufige Vererbung

Will man eine Channel-Variable übergeben, aber keine globale Variable dafür benutzen, so kann man der Channel-Variable ein _ (Underscore) voransetzen. Diese Channel-Variable wird dann an den nächsten Channel vererbt. Dabei wird der Underscore entfernt. Die Vererbung kann also nur einmal erfolgen.

Beispiel:

```
exten => 1234,2,Set(_KUCHENSORTE="Marmorkuchen")
```

4.5.2. Mehrstufige Vererbung

Will man eine Channel-Variable beliebig oft vererben, so muss man zwei _ (Underscore)-Zeichen vor die Variable setzten. Die zwei Underscore-Zeichen werden dann immer mitvererbt.

Warnung

Asterisk unterscheidet die Namen von Variablen, die mit einem Underscore anfangen, nicht von Variablen, die keinen Underscore haben. So wird im folgenden Beispiel aus der vererbbaren Channel-Variable `KUCHENSORTE` eine normale (nicht vererbbare) Channel-Variable:

```
exten => 1234,1,Set(__KUCHENSORTE="Marmorkuchen")
exten => 1234,2,Set(KUCHENSORTE="Marmorkuchen")
```

Beispiel:

```
exten => 1234,2,Set(__KUCHENSORTE="Nusskuchen")
```

Bei einem Abruf einer vererbten Channel-Variable ist es egal, ob man die Underscores voranstellt oder nicht. Die beiden folgenden Zeilen geben zwei mal den gleichen Wert im CLI aus:

```
exten => 1234,1,NoOp(${__KUCHENSORTE})
exten => 1234,2,NoOp(${KUCHENSORTE})
```

4.6. Feste Channel Variablen

Die folgende Liste enthält die wichtigsten fest einprogrammierten Channel-Variablen, die nicht in der `extensions.conf` überschrieben, aber ausgelesen werden können:[3]Die für das jeweilige Thema interessanten Variablen werden aber im Buch immer in jedem Kapitel extra aufgeführt.[4]

`${ANSWEREDTIME}`
: Die Gesamtzeit (in Sekunden) seitdem das Gespräch zustande gekommen ist.

`${BLINTRANSFER}`
: Der Name des Channels auf der anderen Seite eines Blind-Transfers.

`${CALLERID(all|name|num)}`
: CallerID des Anrufers (wenn diese übermittelt wurde).

  `number` Nummer des Anrufers

  `name` Name des Anrufers

  `all` Name und Nummer im Format *`Name <Nummer>`*, z.B. `Fritz Meier <2000>`

`${CHANNEL}`
: Name des aktuellen Channels.

`${CONTEXT}`
: Name des aktuellen Contextes.

`${EPOCH}`
: Aktueller Unix Style Epoch.

`${EXTEN}`
: Aktuell gewählte Extension.

`${ENV(VARIABLENNAME)}`
: Umgebungsvariable `VARIABLENNAME`

`${HANGUPCAUSE}`
: Grund für das Beenden (hangup) eines Gespräches.

`${INVALID_EXTEN}`
: Wird in der `i`-Extension benutzt und enthält die gewählte Extension.

`${LANGUAGE}`
: Aktuell eingestellte Sprache (z.B. `de` für Deutsch).

`${PRIORITY}`
: Aktuelle Priorität innerhalb der Extension.

`${TRANSFER_CONTEXT}`
: Context eines weiterverbundenen Gespräches.

`${UNIQUEID}`
: Im System einzigartige (unique) ID für das aktuelle Gespräch.

[3]Eine komplette Liste aller vordefinierten Variablen finden Sie in der Datei `doc/README.variables` (1.2) / `doc/channelvariables.txt` (1.4)
[4]Ein klassisches Henne-Ei-Problem.

4.7. Variablenmanipulation

Variablen an sich eröffnen ihren Nutzen erst dann, wenn man sie zur Laufzeit verändern kann. Mit Hilfe dieser Variabilität lassen sich komplexe Verhaltensweisen steuern und vorsehen.

4.7.1. Substring

In der Regel bezeichnen Strings eine Aneinanderreihung einzelner Zeichen (character). Die Größe eines Strings bestimmt sich durch die Anzahl der identifizierbaren einzelnen Zeichen. Zum Beispiel besteht der String „Apfelbaum“ aus 9 Zeichen. Ein String kann grundsätzlich in so genannte Teilstrings zerlegt werden, also Strings, die, wenn man sie wieder aneinanderreiht den ursprünglichen String ergeben. Im genannten Beispiel könnten wir den String „Apfelbaum“ in die Teilstrings „Apfel“ und „baum“ zerlegen aber auch „Apf“ und „elbaum“ sind korrekte Teilstrings. Ein String hat theoretisch keine begrenzte Größe, er kann beliebig viele Zeichen enthalten, der gesamte Text dieses Buches könnte als ein String aufgefasst werden. Natürlich wäre das dann nicht mehr sehr handlich, dennoch, ist der Vorgang des Zerlegens eines Strings in Teilstrings eine Standardprozedur beim Programmieren von Anwendungen. In vielen Programmiersprachen heißt die hier beschriebene Funktionalität *Substring*. Mit einer Substring-Funktion kann man Teile eines Strings ausschneiden und das Ergebnis in einer anderen Variablen abspeichern. In Asterisk gibt es keine eigenständige Routine für diese Substring-Funktion, sondern hierfür wird ein : (Doppelpunkt) nach dem Variablennamen eingesetzt. Vom Inhalt der Variablen, also dem String, wird dann eine vorgegebene Anzahl von Zeichen (Länge) als Teilstring abgetrennt.

4.7.1.1. Syntax

```
${VARIABLENNAME[:Anfang[:Laenge]]}
```

4.7.1.2. Beispiele

Bei einer Telefonanlage wird traditionell eine führende 0 (Null) gewählt, um eine Amtsleitung zu bekommen. Die zu wählende Telefonnummer darf diese 0 aber nicht enthalten. Wird die Nummer `0030 227 32320` gewählt, so kann man mit folgender Zeile die eigentliche Rufnummer in die Variable `${RUFNUMMER}` abspeichern.[5]

```
exten => _0X.,1,Set(RUFNUMMER=${EXTEN:1})
```

Wenn die Angabe der Länge fehlt, wird automatisch der Rest des Strings genommen.

Wenn wir die letzten 5 Ziffern der gewählten Nummer benötigen, dann erreichen wir das mit einer negativen Zahl. Bei der oben gewählten Nummer würde die folgende Zeile den Wert 32320 in der Variablen `${DURCHWAHL}` abspeichern.

```
exten => _0X.,1,Set(DURCHWAHL=${EXTEN:-5})
```

Möchten wir die Vorwahl in einer separaten Variable abspeichern, so erreichen wir dies durch:

```
exten => _0X.,1,Set(VORWAHL=${EXTEN:1:3})
```

Diese Zeile geht davon aus, dass die Vorwahl 3-stellig ist und speichert sie in der Variable `${VORWAHL}`.[6]

Nehmen wir an, wir möchten bei Telefonaten zur Nummer `00012024562121` nach Amerika die einzelnen Bestandteile herausfiltern:[7]

```
exten => _0001X.,1,Set(INTERNATIONALEVORWAHL=${EXTEN:3:1})
exten => _0001X.,2,Set(ORTSVORWAHL=${EXTEN:4:3})
exten => _0001X.,3,Set(RUFNUMMER=${EXTEN:7})
exten => _0001X.,4,Set(DURCHWAHL=${EXTEN:-4}
```

[5] Für alle neugierigen Leser: Das ist die Telefonnummer des Parlamentsarchives des Deutschen Bundestages in Berlin.
[6] Leider macht diese Variante in Deutschland wenig Sinn, da es unterschiedlich lange Vorwahlen gibt. Mehr Informationen zur deutschen Vorwahl finden Sie unter *http://de.wikipedia.org/wiki/Telefonvorwahl_%28Deutschland%29*
[7] Um Ihnen das Gespräch nach Amerika zu sparen: Dies ist die Nummer der Besucherzentrale des Weißen Hauses in Washington DC. ;-)

5. Besondere Extensions

Da sämtliche Programmierlogik über Extensions programmiert werden muss, benötigen wir noch eine Reihe von besonderen und vom System vorbelegte Extensions.

5.1. Die h-Extension

Das `h` steht bei dieser Standard-Extension für Hangup (Auflegen). Sie wird aufgerufen, sobald ein Gespräch beendet wurde. Dabei muss berücksichtig werden, dass dann logischerweise der Wert der Variable `${EXTEN}` nicht mehr die ursprüngliche Extension, sondern den Wert `h` enthält.

5.1.1. Beispiel

Sollen in der globalen Variable GESPRAECHE immer die Anzahl der aktuell geführten Gespräche gespeichert werden, so muss diese beim Aufbau eines Gesprächs um 1 erhöht und beim Abbau eines Gesprächs (also beim Auflegen) wieder um 1 runtergezählt werden. Der folgende Dialplan zeigt die Grundidee:

```
[global]
GESPRAECHE=0

[from-intern]
exten => _X.,1,Set(GESPRAECHE=$[${GESPRAECHE} + 1]|g)
exten => _X.,2,Dial(SIP/${EXTEN})

exten => h,1,Set(GESPRAECHE=$[${GESPRAECHE} - 1]|g)
```

5.2. Die i-Extension

Um einen Context "wasserdicht" zu machen, wird die `i`-Extension benutzt. Das `i` steht dabei für invalid (ungültig) und behandelt alle im entsprechenden Context nicht definierten Zielrufnummern. Auch hier wird in der Variable `EXTEN` natürlich nicht mehr die angewählte Nummer abgebildet. Diese ist mit der Variable `INVALID_EXTEN` abrufbar.

5.2.1. Beispiel

In der Apfelmus GmbH können aus der Abteilung B nur die Rufnummern 100 bis 199 angerufen werden. Alle anderen Gespräche führen zur Ansage des that-is-not-rec-phn-num Bausteines.[8]

```
[abteilung-b]
exten => _1XX,1,Dial(${EXTEN})

exten => i,1,NoOp(Undefinierte Nummer ${INVALID_EXTEN} wurde gewaehlt.)
exten => i,2,Answer()
exten => i,3,Playback(that-is-not-rec-phn-num)
exten => i,4,Hangup()
```

5.3. Die o- und a-Extension

Wurde in der Konfigurationsdatei `voicemail.conf` der Eintrag `operator=yes` gesetzt, so kann innerhalb der Voicemailbox durch Drücken der `0` (Null) die `o`-Extension aufgerufen werden (`o` für Operator).

Durch Drücken der Stern-Taste (`*`) kommt man hingegen in die `a`-Extension (abort, Abbruch).

5.4. Die t- und T-Extension

Bei den `t`- und `T`-Extensions handelt es sich jeweils um Timeout-Extensions, also Zeitüberschreitungen.

[8] Der Sprachbaustein in der Datei `that-is-not-rec-phn-num.gsm` im spricht den folgenden Text: "That is not a recognized phone number".

5.4.1. t-Extension

Erfolgt nach einer bestimmten Zeit in einem IVR-Menü keine Eingabe, so wird die `t`-Extension aufgerufen.

Beispiel:

```
[hauptmenue]
exten => 10,1,Answer()
exten => 10,n,Background(marryme)             ; "Heiraten? 1 für ja, 2 für nein"

exten => 1,1,Playback(thank-you-cooperation)  ; 1 => "Danke"
exten => 1,n,Hangup()

exten => 2,1,Playback(hangup-try-again)       ; 2 => "Nochmal probieren"
exten => 2,n,Hangup()

exten => t,1,Hangup()                         ; keine Eingabe => auflegen
```

5.4.2. T-Extension

Die `T`-Extension wird nach einem Absolute Timeout aufgerufen. Dieser kann mit `Set(TIMEOUT(absolute) = <Sekunden>)` gesetzt werden. Der Timeout wird immer dann gestartet, wenn der Zähler neu gesetzt wird (es wird also nicht automatisch ab Anfang des Gespräches gezählt). Mit `Set(TIMEOUT(absolute) = 0)` wird dieser Timeout wieder deaktiviert.

Beispiel:

```
exten => 20,1,Answer()
exten => 20,2,Set(TIMEOUT(absolute) = 120)
exten => 20,3,Playback(hello-world)
exten => 20,4,Wait(1)
exten => 20,5,Goto(3)

exten => T,1,Wait(1)
exten => T,2,Playback(thank-you-for-calling)
exten => T,3,Wait(1)
exten => T,4,Hangup()
```

5.5. Die s-Extension

Der erste Parameter einer Regel (Extension) ist immer der Name bzw. die Nummer. Was passiert aber, falls der Anruf auf einer klassischen analogen Leitung eintrifft und Asterisk gar nicht wissen kann, an wen der Anruf gerichtet ist? Dies ist dann der Fall, wenn Sie Asterisk mittels eines analogen Anschlusses an das Festnetz anschließen und die Rufnummerweitergabe aus dem Festnetz an den analogen Teilnehmeranschluss nicht erfolgt. Für dieses und alle anderern Szenarien, in denen der eingehende Anruf ohne eine Zielrufnummer ankommt, gibt es die `s`-Extension.

> **Warnung**
>
> Wenn Sie ein ATA-Device, also einen Analog to VoIP Adapter benutzen, so benötigen Sie dafür keine `s`-Extension. Die Zielrufnummer müssen Sie hierbei im Adapter (meist über ein Webinterface) konfigurieren.

Beispiel:

```
exten => s,1,Answer()
exten => s,2,Wait(1)
exten => s,3,Play(carried-away-by-monkeys)
exten => s,4,Wait(1)
exten => s,5,Hangup()
```

6. Applikationen im Dialplan

Im Folgenden werden die Applikationen beschrieben, die im Dialplan (also in der `/etc/asterisk/extensions.conf`) verwendet werden können. Beachten Sie bitte, dass nur solche Applikationen in Ihrer Installation verfügbar sind, die in der `/etc/asterisk/modules.conf` im Abschnitt `[modules]` durch `autoload=yes` oder explizites `load => app_`*`applikationsname`*`.so` geladen wurden. Welche in Ihrer Asterisk-Version vorhanden sind, erfahren Sie, indem Sie im Asterisk-CLI **`show applications`** oder **`show application`** ***`Applikationsname`*** eingeben.

Der Begriff „Applikationen“ (oder „Befehle“) darf nicht mit „Funktionen“ verwechselt werden, die ggf. innerhalb eines Befehlsaufrufs stehen. „Applikationen“ (applications) ist vielleicht irreführend, aber der gängige Ausdruck. Einige Applikationen älterer Asterisk-Versionen wurden mittlerweile durch Funktionen ersetzt und werden hier nicht mehr erwähnt (da sie nicht mehr verwendet werden sollen).

> Ein Hinweis am Rande: Die Konfigurationsdateien von Asterisk verwenden leider ein recht schwammiges Format (als INI bekannt), für das nie eine offizielle *Grammatik* veröffentlicht wurde. Der Parser geht auch nicht den üblichen Weg von lexikalischer Analyse, Tokenizing und syntaktischer Analyse. Aus diesem Grund steigt übrigens der Asterisk-Ableger *OpenPBX* auf die von Mac OS X bekannten aber auch auf anderen Plattformen verfügbaren „Property Lists“ (`.plist`) als Format für Konfigurationsdateien um.
>
> Hier sei nur gesagt, dass es aufgrund mangelnder Spezifikation nicht immer klar ist, an welcher Stelle z.B. Leerzeichen erlaubt sind oder wo Anführungszeichen erwartet werden. Meist werden mehrere Schreibweisen erkannt. Im Zweifelsfall hilft nur Ausprobieren, wenn eine Schreibweise in Ihrer Asterisk-Version nicht funktioniert. Sollten Sie Fehler entdecken, wird um Rückmeldung gebeten.

Es ist oft möglich, Parameter auszulassen. Wenn ausgelassene Parameter nicht am Ende stehen, müssen Sie trotzdem ein Komma angeben, um anzuzeigen, dass der Parameter leer ist (also der Default-Wert verwendet wird), z.B. so:

```
exten => s,1,Dial(IAX2/User:Passwort@example.com/123,,flags)
```

Im Allgemeinen kann gesagt werden, dass Fehler beim Ausführen einer Applikation durch Rückgabe des Wertes -1 ausgedrückt werden. Bitte beachten Sie, dass, abhängig von der eingesetzten Asterisk-Version die Parameter durch `,` (Komma) oder `|` (Pipe) getrennt werden.

Wer Asterisk schon länger kennt, wird sich vielleicht wundern, warum hier die eine oder andere Applikation nicht aufgeführt ist. Das hat den einfachen Grund, dass bereits in Asterisk 1.2 etliche Applikationen als „deprecated“ (= veraltet, soll nicht mehr verwendet werden) gekennzeichnet sind und in 1.4 nicht mehr existieren. Solche Befehle werden hier nicht mehr beschrieben, Sie finden die entsprechenden Funktionen in Abschnitt 7, „Funktionen im Dialplan“.

In den Beispielen wird oft die willkürlich gewählte Extension `123` und die Priorität `1` verwendet, was in der Praxis natürlich nicht immer sinnvoll ist.

Um leichtere Verwendung als Nachschlagewerk zu ermö glichen, sind die Applikationen in alphabetischer Reihenfolge aufgeführt. Da sie sich aber auch sinnvoll gruppieren lassen, finden Sie hier einen entsprechenden Index:

Anruf-Verwaltung (abheben, durchstellen, auflegen, ...)

Abschnitt 6.80, „`Ringing()`“ - Klingeln signalisieren

Flusskontrolle und Timeouts

Abschnitt 6.29, „`EndWhile()`“ - Ende einer While-Schleife
Abschnitt 6.31, „`ExecIf()`“ - Bedingtes Ausführen
Abschnitt 6.37, „`Gosub()`“ - Zu einer Unterroutine springen
Abschnitt 6.38, „`GosubIf()`“ - Bedingtes `Gosub()`
Abschnitt 6.39, „`Goto()`“ - Zu einer Priorität, Extension oder anderem Kontext springen
Abschnitt 6.40, „`GotoIf()`“ - Bedingtes `Goto()`
Abschnitt 6.41, „`GotoIfTime()`“ - Bedingtes `Gosub()` je nach Datum/Uhrzeit
Abschnitt 6.47, „`Macro()`“ - Macro aufrufen
Abschnitt 6.70, „`Random()`“ - Zufällig im Dial-Plan springen
Abschnitt 6.77, „`ResponseTimeout()`“ - Maximale Antwortzeit festlegen
Abschnitt 6.79, „`Return()`“ - Aus einem `Gosub()` oder `GosubIf()` zurückkehren
??? - Eine Adresse vom Gosub-Stack entfernen und zurückgeben ohne dahin zurückzukehren
Abschnitt 6.117, „`While()`“ - Anfang einer While-Schleife

Anrufer-Kennung (Caller-ID, Name, ...)

Abschnitt 6.13, „`CallingPres()`“ - Anzeige der Caller-ID verändern
Abschnitt 6.45, „`LookupBlacklist()`“ - Caller-ID in Blacklist nachschlagen
Abschnitt 6.46, „`LookupCIDName()`“ - Caller-ID-Name in Datenbank nachschlagen
Abschnitt 6.67, „`PrivacyManager()`“ - Eingabe einer Telefonnummer verlangen (falls ohne Caller-ID)
Abschnitt 6.92, „`SetCallerPres()`“ - Caller-ID setzen, unanhängig vom Kanal
Abschnitt 6.95, „`SoftHangup()`“ - Auflegen verlangen
Abschnitt 6.118, „`Zapateller()`“ - Werbeanrufe blockieren

Gesprächsprotokolle (CDRs)

??? - Daten ans CDR-User-Feld anhängen
Abschnitt 6.35, „`ForkCDR()`“ - Den CDR in zwei getrennte Einträge aufspalten
Abschnitt 6.58, „`NoCDR()`“ - CDR für diesen Anruf deaktivieren
Abschnitt 6.76, „`ResetCDR()`“ - CDR-Eintrag zurücksetzen
??? - Die CDR-Kontonummer setzen
Abschnitt 6.91, „`SetAMAFlags()`“ - AMA-Flags setzen
??? - Das CDR-User-Feld setzen

Voicemail

Abschnitt 6.24, „`Directory()`“ - Dial-by-name-Telefonbuch anbieten
Abschnitt 6.48, „`MailboxExists()`“ - Prüft, ob Mailbox existiert
Abschnitt 6.110, „`VoiceMail()`“ - Anrufbeantworter
Abschnitt 6.111, „`VoiceMailMain()`“ - Anrufbeantworter abhören
Abschnitt 6.109, „`VMAuthenticate()`“ - User anhand der `voicemail.conf` authentifizieren

Konferenzen

Abschnitt 6.49, „`MeetMe()`“ - MeetMe-Konferenz
Abschnitt 6.50, „`MeetMeAdmin()`“ - MeetMe-Konferenz verwalten
Abschnitt 6.51, „`MeetMeCount()`“ - Anzahl der Teilnehmer einer MeetMe-Konferenz

Variablen verändern

Abschnitt 6.44, „`ImportVar()`“ - Variable von einem Kanal importieren
Abschnitt 6.71, „`Read()`“ - Eine Variable (Ziffern) vom User einlesen
Abschnitt 6.90, „`Set()`“ - Eine Kanal-Variable setzen

Musik-/Sprachausgabe

Abschnitt 6.10, „`Background()`“ - Im Hintergrund eine Sound-Datei spielen
Abschnitt 6.11, „`BackgroundDetect()`“ - `Background()` mit Spracherkennung
Abschnitt 6.18, „`ControlPlayback()`“ - `Playback()` mit Vor-/Zurückspulen und Abbrechen
Abschnitt 6.28, „`Echo()`“ - Audio-Eingabe widergeben
Abschnitt 6.33, „`Festival()`“ - Text mit dem Festival-Synthesizer sprechen
??? - Text mit dem Festival-Lite-Synthesizer sprechen
Abschnitt 6.52, „`Milliwatt()`“ - Einen konstanten 1000-Hz-Ton mit 0 dbm erzeugen
Abschnitt 6.55, „`MP3Player()`“ - MP3-Datei oder -Stream spielen
Abschnitt 6.56, „`MusicOnHold()`“ - Wartemusik spielen
Abschnitt 6.64, „`Playback()`“ - Eine Sound-Datei spielen
Abschnitt 6.65, „`Playtones()`“ - Töne spielen
Abschnitt 6.68, „`Progress()`“ - Audio vor dem Abheben spielen
Abschnitt 6.81, „`SayAlpha()`“ - Buchstabieren (alphanumerisch)
Abschnitt 6.82, „`SayDigits()`“ - Ziffern ansagen
Abschnitt 6.83, „`SayNumber()`“ - Nummer ansagen
Abschnitt 6.84, „`SayPhonetic()`“ - Buchstabieren (mit `sounds/phonetic/`*`Zeichen`*`_p.gsm`)
Abschnitt 6.85, „`SayUnixTime()`“ - Datum/Uhrzeit ansagen
Abschnitt 6.97, „`StopPlaytones()`“ - `Playtones()` beenden

Aufzeichnen

Abschnitt 6.5, „`AgentMonitorOutgoing()`“ - Ausgehende Anrufe eines Agenten aufzeichnen
??? - Die ALSA-Konsole überwachen
Abschnitt 6.14, „`ChangeMonitor()`“ - Die Datei für `Monitor()` verändern
Abschnitt 6.16, „`ChanSpy()`“ - Ein Gespräch mithören
Abschnitt 6.22, „`Dial()`“ - Mit der Option `w` oder `W` im Gespräch aufzeichnen
Abschnitt 6.23, „`Dictate()`“ - Diktat aufnehmen und widergeben
Abschnitt 6.53, „`MixMonitor()`“ - Ähnlich `Monitor()`, aber nur eine Datei
Abschnitt 6.54, „`Monitor()`“ - Ein Gespräch mitschneiden
Abschnitt 6.74, „`Record()`“ - Eingehendes Audio mitschneiden
Abschnitt 6.96, „`StopMonitor()`“ - `Monitor()` beenden
Abschnitt 6.119, „`ZapBarge()`“ - Einen ZAP-Kanal belauschen
Abschnitt 6.121, „`ZapScan()`“ - Zwecks Mithören durch ZAP-Kanäle schalten

Datenbank-Zugriffe

Abschnitt 6.19, „`DBdel()`“ - Einen Datenbankeintrag löschen
Abschnitt 6.20, „`DBdeltree()`“ - Einen Datenbank-Zweig löschen
??? - MySQL-Abfragen

Allgemeines

Abschnitt 6.9, „`Authenticate()`“ - Einen User authentifizieren
??? - Ausgabe z.B. auf einem Intercom
Abschnitt 6.86, „`SendDTMF()`“ - DTMF-Töne senden
Abschnitt 6.87, „`SendImage()`“ - Ein Bild senden
Abschnitt 6.88, „`SendText()`“ - Einen Text senden
Abschnitt 6.89, „`SendURL()`“ - Eine URL senden
Abschnitt 6.103, „`Transfer()`“ - Anruf weiterleiten
Abschnitt 6.109, „`VMAuthenticate()`“ - User anhand der `voicemail.conf` authentifizieren
Abschnitt 6.112, „`Wait()`“ - Eine bestimmte Zeit warten
Abschnitt 6.113, „`WaitExten()`“ - Auf Eingabe einer Extension warten
Abschnitt 6.114, „`WaitForRing()`“ - Auf Klingeln warten
Abschnitt 6.116, „`WaitMusicOnHold()`“ - Warten mit Musik

Externe Skripte

Abschnitt 6.6, „`AGI()`“ - Eine AGI-Anwendung ausführen
Abschnitt 6.21, „`DeadAGI()`“ - `AGI()` auf einem aufgelegten Kanal

Abschnitt 6.27, „EAGI()“ - Siehe AGI()
??? - Einen externen IVR-Generator ausführen
Abschnitt 6.32, „FastAGI()“ - AGI() auf einem entfernten Rechner
Abschnitt 6.47, „Macro()“ - Ein Makro ausführen
Abschnitt 6.59, „NoOp()“ - Nichts tun. Schreibt Debugging-Informationen
??? - res_perl is the mod_perl of Apache, only for Asterisk
??? - res_php is the mod_php of Apache, only for Asterisk
Abschnitt 6.71, „Read()“ - Eine Variable (Ziffern) vom User einlesen
Abschnitt 6.105, „TXTCIDName()“ - Anrufernummer per DNS (txt!) nachschlagen (Deprecated in favor of TXTCID)
Abschnitt 6.102, „System()“ - Einen Shell-Befehl ausführen
Abschnitt 6.104, „TrySystem()“ - Wie System(), gibt aber immer 0 zurück
Abschnitt 6.107, „UserEvent()“ - Dem Manager-Interface ein Event schicken

SIP

Abschnitt 6.94, „SIPdtmfMode()“ - DTMF-Modus während SIP-Verbindung ändern
Abschnitt 6.93, „SIPAddHeader()“ - Einem ausgehenden Anruf einen SIP-Header hinzufügen

ZAP

Abschnitt 6.34, „Flash()“ - Einen Switchhook-Flash (auch „Link“) auf einem ZAP-Trunk senden
Abschnitt 6.119, „ZapBarge()“ - Einen ZAP-Kanal belauschen
Abschnitt 6.120, „ZapRAS()“ - RAS (Remote Access Server) auf einem ZAP-ISDN-Kanal bereitstellen
Abschnitt 6.121, „ZapScan()“ - Zwecks Mithören durch ZAP-Kanäle schalten
??? - Ziffern „out of band“ auf einem ZAP-PRI-Kanal senden

Warteschlangen, Call-Center

Abschnitt 6.1, „AddQueueMember()“ - Interface dynamisch in Warteschleife einreihen
Abschnitt 6.3, „AgentCallbackLogin()“ - Call-Center-Agenten einloggen (mit Rückruf)
Abschnitt 6.4, „AgentLogin()“ - Call-Center-Agenten einloggen
Abschnitt 6.5, „AgentMonitorOutgoing()“ - Ausgehende Anrufe eines Agenten aufzeichen (nur mit AgentCallbackLogin())
Abschnitt 6.61, „ParkAndAnnounce()“ - Anruf parken und ankündigen
Abschnitt 6.62, „ParkedCall()“ - Einen geparkten Anruf annehmen
Abschnitt 6.63, „PauseQueueMember()“ - Einen Agenten pausieren
Abschnitt 6.69, „Queue()“ - Eingehenden Anruf in Warteschleife einreihen
Abschnitt 6.75, „RemoveQueueMember()“ - Interface aus Warteschleife entfernen
Abschnitt 6.106, „UnpauseQueueMember()“ - Einen pausierenden Agenten wieder aktivieren

ADSI

Abschnitt 6.2, „ADSIProg()“ - Ein ADSI-Skript in ein Telefon laden
Abschnitt 6.36, „GetCPEID()“ - ADSI-CPE-ID abfragen

Verschiedenes

Abschnitt 6.7, „AlarmReceiver()“ - Einen Contact-ID-Alarm-Empfänger von Ademco emulieren

6.1. AddQueueMember()

Reiht Interfaces dynamisch in eine Anruf-Warteschleife ein.

```
AddQueueMember(Warteschleife[,Interface[,Malus[|,Optionen]]])
```

Reiht das angegebene Interface dynamisch als Agent in eine vorhandene Warteschleife (Queue) ein, die in queues.conf definiert wurde. Die Malus-Punkte beeinflussen ggf. die Position bei der Abarbeitung. Agenten mit niedrigerem Malus werden vor Einträgen mit höherem Malus aufgerufen.

Falls das angegebene Interface bereits in die Warteschlange eingereiht ist und eine n+101 Priorität existiert (n ist die aktuelle Priorität), wird die Abarbeitung bei dieser Priorität fortgesetzt, andernfalls wird ein Fehler (d.h. -1) zurückgeliefert. (Je nach Asterisk-Version muss für das Springen zu n+101 die Option `j` (jump) angegeben werden.)

Wird `AddQueueMember()` ohne den Interface-Parameter aufgerufen, findet das Interface, das vom Teilnehmer zu diesem Zeitpunkt benutzt wird, Verwendung.

In manchen Asterisk-Versionen kann man statt der Pipe-Zeichen (`|`) auch Kommas verwenden.

Die Applikation setzt auch die Kanal-Variable `${AQMSTATUS}` auf `ADDED` (hinzugefügt), `MEMBERALREADY` (ist bereits Mitglied) oder `NOSUCHQUEUE` (Warteschlange nicht vorhanden).

```
; SIP/3000 zur supportschlange hinzufügen:
exten => 123,1,AddQueueMember(supportschlange,SIP/3000)

; das aktuelle Interface mit Malus 2 hinzufügen:
exten => 123,1,AddQueueMember(supportschlange,,2)
```

Siehe auch. Abschnitt 6.69, „`Queue()`“, Abschnitt 6.75, „`RemoveQueueMember()`“, `queues.conf`

6.2. `ADSIProg()`

Lädt ein ADSI-Skript in ein ADSI-fähiges Telefon

```
ADSIProg([Script])
```

Programmiert ein ADSI-Telefon (ADSI: Analog Display Services Interface) mit einem vorhandenen Skript. Wird kein Skript angegeben, wird das Default-Skript `asterisk.adsi` verwendet. Die Pfadangabe zum `Script` versteht sich relativ zum Asterisk-Konfigurationsverzeichnis, i.d.R. `/etc/asterisk/`, es kann aber ebenso ein vollständiger (absoluter) Pfad zu der Skript-Datei angegeben werden.

Benutzen Sie die `GetCPEID()`-Anwendung, um die CPE-ID und weitere Informationen über das ADSI-fähige Telefon zu erfahren.

```
; das ADSI-Telefon mit dem Skript telcordia-1.adsi programmieren:
exten => 123,1,ADSIProg(telcordia-1.adsi)
```

Siehe auch. Abschnitt 6.36, „`GetCPEID()`“, `adsi.conf`

6.3. `AgentCallbackLogin()`

Ermöglicht Login eines Call-Center-Agenten mit Rückruf

```
AgentCallbackLogin([Agenten-Nr][,Optionen[,Extension@Context]])
```

Ermöglicht einem Anruf-Agenten, identifiziert durch die `Agenten-Nummer`, sich in das Anruf-Warteschlangen-System einzuloggen, um bei einem eingehenden Anruf für ihn zurückgerufen zu werden.

Bei einem eingehenden Anruf für diesen Agenten wird Asterisk die spezifizierte zugehörige `Extension` (gegebenenfalls mit dem optional angegebenem Kontext `Context`) aufrufen.

Der `Optionen`-Parameter kann den Wert `s` enthalten, der ein Silent Login (stille Anmeldung) bewirkt.

```
; still als Agent 33 (wie in agents.conf definiert) einloggen. Asterisk
; wird SIP/300 anrufen, sobald für diesen Agenten ein Anruf eingeht:
exten => 123,1,AgentCallbackLogin(33,s,${CALLERID(num)})

; wenn wir davon ausgehen, dass die Agentennummer mit der Anrufer-
; Extension übereinstimmt:
exten => 123,1,AgentCallbackLogin(${CALLERID(num)},s,${CALLERID(num)})
```

Auf http://www.voip-info.org/wiki/index.php?page=Asterisk+cmd+AgentCallbackLogin sind viele Beispiele zu finden.

Siehe auch. Abschnitt 6.4, „`AgentLogin()`“

6.4. `AgentLogin()`

Erlaubt es einem Anruf-Agenten, sich in das System einzuloggen/sich am System anzumelden

```
AgentLogin([Agenten-Nr][,Optionen])
```

Meldet den gegenwärtigen Anrufer als Call Agent (optional durch `Agenten-Nr` zu identifizieren) im Anruf-Warteschlangen-System an. Solange er als Agent angemeldet ist, kann er Anrufe entgegennehmen und hört bei neuen eingehenden Anrufen einen Piepton auf der Leitung. Der Agent kann den Anruf durch Betätigen der Stern-Taste (`*`) beenden.

Der `Optionen`-Parameter kann den Wert `s` enthalten, der ein Silent Login (stille Anmeldung) bewirkt.

```
; still als Agent 33 (wie in agents.conf definiert) einloggen
exten => 123,1,AgentLogin(33,s)
```

Siehe auch. Abschnitt 6.3, „`AgentCallbackLogin()`“

6.5. `AgentMonitorOutgoing()`

Speichert die ausgehenden Anrufe eines Agenten

```
AgentMonitorOutgoing([Optionen])
```

Zeichnet alle ausgehenden Anrufe eines Agenten auf.

Diese Anwendung versucht, die ID eines Agenten, der einen ausgehenden Anruf tätigt, herauszufinden, indem er die ID des Anrufers (Caller ID) des gegenwärtigen Interfaces mit einer von der `AgentCallbackLogin()`-Anwendung gesetzten globalen Variablen vergleicht. Daher wird dringend empfohlen, diese Anwendung nur in Verbindung mit der `AgentCallbackLogin()`-Anwendung (und zwar nach dieser!) zu verwenden. Um die Anrufe aufzuzeichnen, werden anstelle der `Monitor()`-Anwendung die Überwachungsfunktionen des `chan_agent`-Moduls benutzt. Deshalb muss das Aufzeichnen der Anrufe in der `agents.conf`-Datei korrekt konfiguriert sein.

Standardmäßig werden aufgezeichnete Anrufe im Verzeichnis `/var/spool/asterisk/monitor/` abgelegt. Diese Einstellung kann durch Anpassung des Parameters `savecallsin` in der Datei `agents.conf` überschrieben werden.

Können die ID des Anrufers (Caller-ID) und/oder die ID des Agenten (Agent-ID) nicht ermittelt werden, wird diese Anwendung zur Priorität n+101 springen, falls diese existiert.

Falls keine der folgenden Optionen anderes bewirkt, liefert die Anwendung 0 zurück.

Der `Optionen`-Parameter kann aus einem oder mehreren der folgenden Optionen zusammengesetzt sein.

`d` Zwingt die Anwendung zur Rückgabe von -1, falls ein Fehler vorliegt und keine Priorität n+101 existiert.

`c` Ändert den Call Detail Record dahingehend, dass *`Agent/AgentenNr`* als Kanal, von dem der Anruf ausgeht, gespeichert wird, damit festgestellt werden kann, von welchem Agenten der Anruf ausging.

`n` Unterdrückt Warnmeldungen im Falle, dass Caller-ID bzw. Agent-ID unbekannt sind. Diese Option ist hilfreich, wenn ein gemeinsamer Kontext für Agenten- und Nicht-Agenten-Anrufe erwünscht ist.

```
; ausgehende Anrufe dieses Agenten aufzeichnen, und den CDR
; entsprechend anpassen:
exten => 123,1,AgentMonitorOutgoing(c)
```

Siehe auch. Abschnitt 6.3, „`AgentCallbackLogin()`“, `agents.conf`, Abschnitt 6.54, „`Monitor()`“, Abschnitt 6.16, „`ChanSpy()`“

6.6. AGI()

Ruft eine AGI-konforme Anwendung auf

```
AGI(Programm[,Argumente])
```

(Analog auch EAGI(), FastAGI(), DeadAGI())

Führt ein Astersik-Gateway-Interface-kompatibles Programm Programm auf dem aktuellen Kanal aus. Solche externen Programme (die in fast jeder beliebigen Sprache vorliegen können - z.B. Perl, PHP, ...), können den Telefonkanal steuern, Audiodateien abspielen, DTMF-Ziffern auslesen und so weiter. Asterisk kommuniziert dabei mit dem AGI-Program über STDIN und STDOUT. Die spezifierten Argumente werden an das AGI-Programm durchgereicht.

Das Programm muss im zugrundeliegenden Dateisystem als executable (ausführbar) gekennzeichnet sein. Der Pfad zum Programm ist relativ zum Asterisk-AGI-Verzeichnis, standardmäßig /var/lib/asterisk/agi-bin/.

Soll ein AGI-Programm laufen, ohne dass ein Kanal offen ist (wie in einer h-Extension, aufgelegt), muss statt dieser die DeadAGI()-Anwendung verwendet werden. Falls AGI über ein Netzwerk laufen soll, benutzen Sie FastAGI().

Benötigen Sie aus Ihrem AGI-Programm heraus Zugriff auf den eingehenden Audiostream, benutzen Sie EAGI() (Extended AGI) an Stelle von AGI(). Das eingehende Audio-Signal kann dann unter File Descriptor 3[9] eingelesen werden.

Liefert -1 mit dem Auflegen oder falls ein Auflegen durch das Programm verlangt wurde, oder 0, falls nicht aufgelegt werden soll.

```
; mein AGI-Skript aufrufen:
exten => 123,1,AGI(agi-skript)
exten => 123,2,EAGI(eagi-skript)
```

Siehe auch. Abschnitt 6.21, „DeadAGI()“, Abschnitt 6.32, „FastAGI()“

6.7. AlarmReceiver()

Bietet Unterstützung für den Empfang von Alarm-Reporten von einem Einbrecher- oder Feueralarm-Bedienfeld

```
AlarmReceiver()
```

Emuliert einen Alarm-Empfänger und ermöglicht Asterisk, spezielle Daten von Einbrecher- und/oder Feueralarm-Bedienfeldern zu empfangen und zu dekodieren. Zum jetzigen Zeitpunkt wird ausschließlich das Ademco Contact ID Format unterstützt.

Nachdem AlarmReceiver() aufgerufen worden ist, wird ein Handshake mit dem Alarm-Bedienfeld durchgeführt, Events empfangen und validiert und anschließend gespeichert, bis das Bedienfeld die Verbindung beendet. Nachdem das Bedienfeld die Verbindung beendet hat, wird die in der eventsmd-Einstellung der Datei alarmreceiver.conf spezifizierte Befehlskette von der Anwendung ausgeführt, und die Events auf die Standardeingabe der Anwendung umgeleitet. Die Datei alarmreceiver.conf enthält ebenfalls Einstellungen zum DTMF-Timing und für die Lautstärke der Meldetöne.

Für die zuverlässige Funktion dieser Anwendung kann keine Garantie übernommen werden. Verlassen Sie sich also nicht ausschließlich auf diese Anwendung, wenn Sie sie nicht bereits ausgiebig getestet haben!

Diese Applikation gibt immer 0 zurück.

```
; Asterisk soll Meldungen eines Feueralarm-Knopfes empfangen:
exten => s,1,AlarmReceiver()
```

Siehe auch. alarmreceiver.conf

[9] zu Erinnerung: 0: stdin, 1: stdout, 2:stderr. 3 ist ein frei definierter File descriptor

6.8. `Answer()`

Antwortet auf einen klingelnden Kanal

```
Answer([Verzögerung])
```

Veranlasst Asterisk zum Beantworten eines Kanals, wenn er gerade klingelt. Falls der aktuelle Kanal nicht klingelt, tut diese Anwendung nichts.

Es ist generell empfehlenswert, `Answer()` vor dem Aufrufen irgendwelcher anderen Anwendungen auf dem Kanal aufzurufen, außer es besteht ein guter Grund, das nicht zu tun. Die meisten Anwendungen verlangen, dass der Kanal beantwortet wird, bevor sie aufgerufen werden, und werden andernfalls eventuell nicht in der Lage sein, korrekt ihre Arbeit zu verrichten.

Die optionale `Verzögerung` weist Asterisk an, soviele Millisekunden vor dem Antworten zu warten.

Gibt bei Erfolg 0 zurück.

```
exten => 123,1,Answer()
exten => 123,2,Wait(1)
exten => 123,3,Playback(hallo)
```

Siehe auch. Abschnitt 6.42, „`Hangup()`“

6.9. `Authenticate()`

Verlangt zur Fortsetzung eines Vorgangs die Eingabe eines Passworts vom Anrufer

```
Authenticate(Passwort[,Optionen])
```

Verlangt, dass der Anrufer ein gegebenes Passwort `Passwort` korrekt eingibt, um dann mit der Ausführung der nächsten Priorität im Wählplan (dialplan) fortzufahren. `Authenticate()` gibt dem Anrufer drei Versuche, das Passwort korrekt einzugeben. Andernfalls wird die Verbindung beendet.

Falls `Passwort` mit dem `/`-Zeichen beginnt, wird es als Datei interpretiert, die eine Liste gültiger Passwörter (genau eins pro Zeile) enthält. Passwörter können auch in der Asterisk-Datenbank (AstDB) gespeichert sein, siehe dazu die Option `d` weiter unten.

Folgende Optionen sind möglich (auch in Kombination):

`a` (accountcode) Setzt den Namen des CDR-Felds `accountcode` und die Variable für den Kanal, `ACCOUNTCODE`, auf das eingegebene Passwort.

`d` (database) Interpretiert `Passwort` als Schlüssel der Asterisk-Datenbank. Wenn ein Datenbankschlüssel benutzt wird, kann der mit dem Schlüssel assoziierte Wert beliebig sein.

`r` (remove) Entfernt den Datenbankschlüssel nach erfolgreicher Verbindung und Eintreten in die Datenbank (nur möglich mit Option `d`)

`j` (jump) Bei falschem Passwort statt aufzulegen zur Priorität n+101 springen, falls diese existiert.

Wichtig

Beachten Sie bei der Option `d`, dass - völlig unverständlicherweise! - als Parameter */passwoerter/1234* verlangt wird, wobei `passwoerter` die Familie ist und `1234` das Passwort; der unter diesem Schlüssel gespeicherte Wert ist irrelevant. Der gesunde Menschenverstand würde einen Datenbank-Eintrag wie /passwoerter/art => 1234 erwarten.

Liefert 0 zurück, falls der Benutzer nach maximal drei Versuchen ein gültiges Passwort eingegeben hat, sonst -1 (auch nach Auflegen).

```
; Passwort 1234 verlangen:
exten => 123,1,Answer()
exten => 123,2,Authenticate(1234)
exten => 123,3,Playback(pin-nummer-akzeptiert)
exten => 123,4,SayDigits(${ACCOUNTCODE})
```

Siehe auch. Abschnitt 6.109, „`VMAuthenticate()`“

6.10. `Background()`

Spielt eine Audiodatei, während Touch-Tone-Ziffern (DTMF-Töne) akzeptiert werden

```
Background(Datei1[&Datei2...][,Optionen[,Sprache]])
```

Spielt die angebene(n) Audiodatei(en), während auf den Start der Eingabe einer Extension durch den Benutzer gewartet wird. Sobald der Benutzer mit der Eingabe einer Extension begonnen hat, wird die Wiedergabe beendet. Der Dateiname sollte ohne Dateiendung angegeben werden, da Asterisk dann automatisch das Dateiformat mit den geringsten Übersetzungskosten auswählt.

`Optionen` ist einer der folgenden Werte (nicht kombinierbar):

`skip` Die Wiedergabe der Nachricht wird übersprungen, falls sich der Kanal gegenwärtig nicht im `up`-Status befindet (z.B., weil er bisher nicht beantwortet wurde). Ist `skip` angegeben, wird die Anwendung unverzüglich beendet, falls der Kanal aufgelegt ist.

`noanswer` Der Kanal kann erst beantwortet werden, nachdem die angegebene Audiodatei abgespielt wurde. Ist diese Option nicht angegeben, wird der Kanal automatisch beantwortet, bevor die Audiodatei abgespielt wird. Es gilt zu beachten, dass nicht alle Kanäle erlauben, vor ihrer Beantwortung eine Nachricht abzuspielen.

Der Parameter `Sprache` kann verwendet werden, um eine bestimmte Sprache zu definieren, in der Nachrichten abgespielt werden sollen, falls diese nicht der für diesen Kanal aktuell eingestellten Sprache entspricht.

Liefert -1 zurück wenn aufgelegt wurde, oder falls der angegebene Dateiname nicht existiert, sonst 0.

```
exten => 123,1,Answer()
exten => 123,2,Background(bitte-extension-eingeben);
```

Siehe auch. Abschnitt 6.64, „`Playback()`“, Abschnitt 6.11, „`BackgroundDetect()`“, der `show translation` Befehl

6.11. `BackgroundDetect()`

Spielt eine Audiodatei im Hintergrund und erkennt Sprechen

```
BackgroundDetect(SoundDatei[,Stille[,Min[,Max]]])
```

Ähnlich `Background()`, versucht jedoch Sprechen zu erkennen.

Während der Wiedergabe der Datei wird Eingang von Audio in Empfangsrichtung überwacht. Wird eine Geräuschphase von mehr als `Min` Millisekunden Dauer und weniger als `Max` Millisekunden registriert, und darauf folgend eine geräuschlose Phase von wenigstens `Stille` Millisekunden, wird die Audio-Wiedergabe abgebrochen und die Ausführung an die Extension `talk` delegiert, falls diese existiert ist.

Sind `Stille`, `Min` und `Max` nicht spezifiziert, werden jeweils die Standardwerte 1000 ms, 100 ms und unendlich angenommen.

Wenn aufgelegt wird, liefert die Anwendung -1 zurück, nach abgeschossener Wiedergabe ohne vorhandene Ausstiegsbedingungen 0.

```
exten => 123,1,BackgroundDetect(sinfonie)
exten => 123,2,Playback(vm-sorry)
exten => talk,1,Playback(ja-bitte)
```

Siehe auch. Abschnitt 6.64, „`Playback()`“, Abschnitt 6.10, „`Background()`“

6.12. `Busy()`

Zeigt dem Kanal besetzt an

```
Busy([Timeout])
```

Fordert den Kanal auf, ein Besetzt-Zeichen zu signalisieren und wartet dann, bis der Benutzer auflegt oder auf den Ablauf der optional anzugebenden zeitlichen Frist (`Timeout`, in Sekunden).

Diese Anwendung signalisiert einen Besetzt-Zustand nur dem überbrückten Kanal (bridged channel). Jeder spezielle Kanaltyp hat seine eigene Art, dem Anrufer besetzt mitzuteilen. Sie können `Playtones(busy)` verwenden, um dem Anrufer ein Besetztzeichen vorzuspielen.

Gibt immer -1 zurück.

```
exten => 123,1,Playback(vm-sorry)
exten => 123,2,Playtones(busy)
exten => 123,3,Busy()
```

Siehe auch. Abschnitt 6.17, „`Congestion()`“, Abschnitt 6.68, „`Progress()`“, Abschnitt 6.65, „`Playtones()`“

6.13. `CallingPres()`

Ändert die Präsentation der Anrufer-ID (Caller ID)

```
CallingPres(presentation)
```

Ändert die Präsentations-Parameter für die Anrufer ID auf einer Q.931 PRI-Verbindung. Diese Parameter sollten gesetzt sein, bevor ein ausgehender Anruf getätigt wird. Der Präsentations-Parameter `presentation` bestimmt zum ersten, ob der angerufene Teilnehmer die Caller ID-Information angezeigt bekommt oder nicht, und zum zweiten, ob die Caller ID-Information von verlässlicher Quelle auf verifiziert geprüft wurde oder nicht (engl.: screening).

Diese Anwendung wurde durch `SetCallerPres()` ersetzt, welche einfacher handzuhaben und weniger abhängig von Zaptel ist.

Diese Anwendung generiert aus der Kombination von Anrufpräsentations-Einstellungen und Screening-Einstellungen eine Zahl. Die Werte selbst sind im ITU Q.931-Standard wie in den folgenden Tabellen definiert.

Screening wird durch die Bits 1 und 2 gesteuert:

Bit1	Bit2	Erläuterung
0	0	Die Caller ID Information liefert der Teilnehmer, sie ist nicht verifiziert
0	1	Die Caller ID Information liefert der Teilnehmer, sie ist erfolgreich verifiziert
1	0	Die Caller ID Information liefert der Teilnehmer, die Verifizierung war nicht erfolgreich
1	1	Die Caller ID Information liefert das Netzwerk

Präsentation wird durch die Bits 6 und 7 gesteuert:

Bit6	Bit7	Erläuterung
0	0	Anzeige der Caller ID Information ist erlaubt
0	1	Anzeige der Caller ID Information ist nicht erlaubt
1	0	Die Nummer ist aufgrund von Zwischenstationen nicht verfügbar
1	1	Reserviert

Die Bits 3, 4, 5 und 8 sollten alle auf Null (0) gesetzt werden. Beachten Sie, dass die Bits in der Folge vom höchstwertigsten zum niederwertigsten nummeriert sind in der Form 87654321.

```
; presentation setzen auf:
; presentation erlaubt                (00000000)
; Netzwerk angegeben                  (00000011)
; ------------------                  ----------
; Ergebnis: 3 (bitweises UND)         (00000011)
exten => 123,1,CallingPres(3)
exten => 123,2,Dial(Zap/g1/1234567)
; presentation setzen auf:
; presentation nicht erlaubt          (00100000)
; vom User, verifiziert               (00000001)
; ------------------                  ----------
; Ergebnis: 33 (bitweises UND)        (00100001)
exten => 124,1,CallingPres(33)
exten => 124,2,Dial(Zap/g1/1234568)
```

Siehe auch. Abschnitt 6.92, „`SetCallerPres()`“, ???

6.14. `ChangeMonitor()`

Ändert den Dateinamen der Protokolldatei für einen Kanal

```
ChangeMonitor(Basis-Dateiname)
```

Ändert den Namen der bei der Überwachung eines Kanals mit `Monitor()` aufgezeichneten Datei. Diese Anwendungen hat keinerlei Auswirkungen, wenn der fragliche Kanal nicht überwacht wird.

```
; Kanal mitschneiden mit Datei-Basisname 'mitschnitt'
exten => 123,1,Monitor(mitschnitt)
; Datei-Basisnamen auf 'mitschnitt2' ändern
exten => 123,2,ChangeMonitor(mitschnitt2)
```

Siehe auch. Abschnitt 6.54, „`Monitor()`“, Abschnitt 6.96, „`StopMonitor()`“

6.15. `ChanIsAvail()`

Ermittelt, ob ein bestimmter Kanal momentan verfügbar ist

```
ChanIsAvail(Technology1/Resource1[&Technology2/Resource2...][,Optionen])
```

Prüft, ob irgendeiner der angeforderten Kanäle verfügbar ist. Ist keiner der angeforderten Kanäle verfügbar, wird die neue Priorität auf n+101 gesetzt (wobei `n` die aktuell gültige Priorität ist), es sei denn, diese Priorität existiert nicht oder ein Fahler tritt auf. In diesem Fall wird `ChanIsAvail()` den Wert -1 zurückliefern.

Ist einer der angeforderten Kanäle verfügbar, wird die nächste Priorität auf `n+1` gesetzt, und `ChanIsAvail()` liefert den Wert 0 zurück.

`ChanIsAvail()` setzt die folgenden Kanalvariablen:

`${AVAILCHAN}` — Der Name des verfügbaren Kanals, inklusive der Session-Nummer des Testanrufs.

`${AVAILORIGCHAN}` — Der kanonische Kanalname, der benutzt wurde um den Kanal zu erstellen, d.h. der Kanalname ohne eine Session-Nummer.

`${AVAILSTATUS}` — Der Status-Code des Kanals:

- `AST_DEVICE_UNKNOWN` (0) — Unbekannt. Der Kanal ist gültig, aber in unbekanntem Zustand.
- `AST_DEVICE_NOT_INUSE` (1) — Kanal wird nicht verwendet.
- `AST_DEVICE_IN_USE` (2) — Der Kanal ist in Gebrauch.

`AST_DEVICE_BUSY` (3)	Besetzt.
`AST_DEVICE_INVALID` (4)	Ungültig. Unbekannter Kanal.
`AST_DEVICE_UNAVAILABLE` (5)	Nicht verfügbar. Kanal ist nicht registriert.
`AST_DEVICE_RINGING` (6)	Kanal klingelt.

Ist die Option `s` (für state) angegeben, wird Asterisk den Kanal als nicht verfügbar betrachten, wann immer er in Gebrauch ist, sogar falls er weiteren anderen Anruf aufnehmen kann. Mit der Option `j` wird, falls der Kanal nicht verfügbar ist, zur Priorität n+101 gesprungen.

Diese Anwendung arbeitet auf MGCP-Kanälen nicht ordnungsgemäß.

```
; Zap/1 und Zap/2 auf Verfügbarkeit prüfen:
exten => 123,1,ChanIsAvail(Zap/1&Zap/2)
; wenn mindestens einer verfügbar ist, geht's bei Priorität 2 weiter:
; In Priorität 2 wählen wir eine Nummer auf dem freien Kanal:
exten => 123,2,NoOp(${AVAILORIGCHAN})
exten => 123,3,Dial(${AVAILORIGCHAN}/123456)
; Wenn wir bei Prio. 102 landen ist weder Zap/1 noch Zap/2 verfügbar
exten => 123,102,Playback(alle-kanaele-belegt)
```

6.16. `ChanSpy()`

Einen Kanal belauschen

```
ChanSpy([Kanalpräfix[,Optionen]])
```

Erlaubt es, das Gespräch auf beliebigen Kanälen mitzuhören. (Ist also nicht wie `ZapBarge()`/`ZapScan()` an Zap-Kanäle gebunden.) Für das Verständnis einiger Optionen ist zu beachten, dass diese Applikation einzelne Kanäle abhört, aber nicht Gespräche im eigentlichen Sinn (min. 2 Teilnehmer), obwohl Sie natürlich das ein- und ausgehende Audio auf dem abgehörten Kanal hören können.

Wenn `Kanalpräfix` angegeben ist, stehen nur Kanäle, die mit diesem String beginnen, zur Auswahl.

`Optionen` (auch in Kombination):

`b`	(bridged) Auswahl nur auf verbundene Kanäle beschränken.
`g(grp)`	(group) Auswahl auf Kanäle beschränken, auf denen `grp` in der durch Doppelpunkt (`:`) getrennten Liste `${SPYGROUP}` (Kanalvariable) enthalten ist.
`q`	(quiet) Keine Piep-Töne beim Umschalten von Kanälen spielen und Kanalname nicht ankündigen.
`r[(Name)]`	(record) Die Sitzung in einer Datei im Verzeichnis `/var/spool/asterisk/monitor/` aufzeichnen. Default-Basis-Dateiname (also ohne Endung) ist `chanspy`.
`v[(Wert)]`	(volume) Die anfängliche Lautstärkeeinstellung von `-4` (leise) bis `4` (laut) verändern.
`w`	(whisper) Flüster-Modus aktivieren. Damit kann der lauschende Kanal mit dem belauschten sprechen, wobei dies nur der belauschte Kanal hört, nicht dessen Gesprächspartner. (Diese Option ist seit Asterisk 1.4 verfügbar.)
`W`	(private whisper) Privater Flüster-Modus. Wie `w`, aber der lauschende Kanal kann den belauschten nicht hören. (sinnvoll?)

Während des Abhörens können folgende Tastenbefehle gegeben werden:

`#`	Schaltet durch die Lautstärkeeinstellung (-4 bis 4)
`*`	Schaltet zu einem anderen Kanal

`...#` Eine Reihe von Ziffern, abgeschlossen durch `#`, wird an `Kanalpräfix` angehängt. Wenn Sie z.B. `ChanSpy(Agent)` ausführen und `1234#` eingeben, werden Sie auf dem Kanal `Agent/1234` lauschen.

```
; Agenten belauschen:
exten => 123,1,Chanspy(Agent)
exten => 123,n,Hangup()

; Beispiel für die Option g:
; auf Anrufen auf 0... SPYGROUP 10005 setzen:
exten => _0.,1,Set(SPYGROUP=10005)
;...
; Kanäle der SPYGROUP 10005 belauschen:
exten => 123,1,ChanSpy(,g(10005))
exten => 123,n,Hangup()
```

Siehe auch. Abschnitt 6.119, „`ZapBarge()`“, Abschnitt 6.121, „`ZapScan()`“, Abschnitt 6.54, „`Monitor()`“

6.17. `Congestion()`

Zeigt Stau (Überlastung) auf dem Kanal an

```
Congestion([Timeout])
```

Zeigt auf dem Kanal Stau an und wartet dann, bis der Teilnehmer auflegt, oder bis die optional spezifizierbare Frist (`Timeout`, in Sekunden) abgelaufen ist.

Diese Anwendung signalisiert lediglich Stau, sie spielt aber dem Teilnehmer keinen Stau-Ton vor. Um einem Anrufer einen Stau-Ton vorzuspielen, kann `Playtones(congestion)` verwendet werden.

Liefert stets -1 zurück.

```
; bei Caller ID is 0123-123456 immer Stau-Signal spielen:
exten => 123,1,GotoIf($[${CALLERID(num)} = 0123123456]?10)
exten => 123,n,Playtones(congestion)
exten => 123,n,Congestion(5)
exten => 123,n,Hangup()
exten => 123,10,Dial(Zap/1)
```

Siehe auch. Abschnitt 6.12, „`Busy()`“, Abschnitt 6.68, „`Progress()`“, Abschnitt 6.65, „`Playtones()`“

6.18. `ControlPlayback()`

Spielt eine Datei ab und kann diese auch schnell vor- und zurückspulen

```
ControlPlayback(Datei[,skipms[,ffchar[,rewchar[,stopchar[,pausechar]]]]])
```

Spielt die angegebene Datei ab und ermöglicht es dem Anrufer, sich, durch Betätigen der Tasten `ffchar` und `rewchar`, in dieser Datei vorwärts und rückwärts zu bewegen. Als Standardeinstellung können die Tasten `#` und * zum Vor- und Zurückspulen der Datei verwendet werden. Ist `stopchar` spezifiziert, hält die Anwendung die Wiedergabe an, wenn die stopchar-Taste gedrückt wird. Existiert die Datei nicht, geht die Anwendung zu Priorität n+101 über, falls vorhanden.

Die `skipms`-Option gibt an, wie weit mit jedem Drücken von `ffchar` oder `rewchar` in der Datei vor- bzw. zurückgesprungen wird.

Die `pausechar-Option` kann ebenfalls spezifiziert werden, sie erlaubt es, die Wiedergabe der Datei vorübergehend anzuhalten (Pause). Erneutes Drücken von `pausechar` setzt die Wiedergabe der Datei dann fort.

Gibt -1 zurück, falls der Kanal während der Wiedergabe aufgelegt wurde.

```
; dem Anrufer "sinfonie" vorspielen mit Steuerungsmöglichkeit:
exten => 123,1,ControlPlayback(sinfonie,5000,#,*,5,0)
```

Siehe auch. Abschnitt 6.64, „`Playback()`“, Abschnitt 6.10, „`Background()`“

6.19. DBdel()

Löscht einen Schlüssel aus der AstDB

```
DBdel(Familie/Schlüssel)
```

Löscht den durch `key` spezifizierten Schlüssel aus der Schlüsselfamilie `Familie` aus der AstDB.

Liefert stets 0 zurück.

```
exten => 123,1,Set(DB(test/name)=Peter)    ; in AstDB speichern
exten => 123,2,Set(name=${DB(test/name)})  ; abfragen
exten => 123,3,DBdel(test/name)            ; löschen
```

Siehe auch. Abschnitt 6.20, „`DBdeltree()`“, Abschnitt 7.11, „`DB()`“

6.20. DBdeltree()

Löscht eine Familie oder einen Schlüsselbaum aus der AstDB

```
DBdeltree(Familie[/Schlüsselzweig])
```

Löscht den angegebenen Schlüsselzweig aus der AstDB.

Liefert immer 0 zurück.

```
; Einträge in der AstDB anlegen:
exten => 123,1,Set(DB(farben/eins)=rot)
exten => 123,n,Set(DB(farben/zwei)=blau)
; now delete the key family named test
exten => 123,n,DBdeltree(farben)
```

Siehe auch. Abschnitt 6.19, „`DBdel()`“, Abschnitt 7.11, „`DB()`“

6.21. DeadAGI()

Führt ein AGI-konformes Skript auf einem toten (aufgelegten) Kanal aus

```
DeadAGI(Programm,Argumente)
```

Führt ein AGI-konformes Skript auf einem toten (aufgelegten) Kanal aus. Mit Hilfe von AGI kann Asterisk externe Programme in nahezu jeder beliebigen Sprache ausführen (starten), um einen Telefonkanal zu steuern, Audio abzuspielen, DTMF-Ziffern auszulesen usw., indem es nach dem AGI-Protokoll auf `STDIN` und `STDOUT` mit diesen kommuniziert. Die übergebenen `Argumente` werden an das AGI-Programm durchgereicht.

Diese Anwendung wurde speziell für tote Kanäle entwickelt, weil das Standard AGI-Interface nicht funktioniert, nachdem der Kanal aufgelegt wurde. Dass der Kanal „tot“ ist, ist allerdings keine Voraussetzung für den Start eines DeadAGI-Skripts!

Benutzten sie den Befehl `show agi` auf der Kommandozeilen-Schnittstelle, um eine Liste aller verfügbaren AGI-Befehle zu erhalten.

Liefert -1 zurück, falls die Anwendung ein Auflegen angefordert hat, oder 0 beim Austritt ohne Auflegen.

```
exten => h,1,DeadAGI(agi-test)
```

Wichtig

Der Kanal wird solange als aktiv angesehen wird, bis das Skript beendet wird, was z.B. für CDRs von Bedeutung sein kann.

Beachten Sie auch, dass auch DeadAGI-Skripte beim Auflegen des Kanals ein `SIGHUP`-Signal erhalten, was ggf. ignoriert werden muss:

Perl	`$SIG{HUP} = "IGNORE";`
PHP	`pcntl_signal(SIGHUP, SIG_IGN);`
	(PHP muss mit `--enable-pcntl` kompiliert worden sein, was per Default nicht der Fall ist!)
Ruby	`trap('SIGHUP','IGNORE')`

Es ist auch wichtig, dass Sie nach dem Auflegen nicht mehr mit Asterisk kommunizieren, sonst erhält das Skript ein `SIGPIPE`, was das Skript beendet, sofern Sie es nicht ignorieren.

Siehe auch. Abschnitt 6.6, „`AGI()`“, Abschnitt 6.32, „`FastAGI()`“

6.22. `Dial()`

Verbindet Kanäle

```
Dial(Tech/Resource,Timeout,Optionen,URL)
```

```
Dial(Tech1/Resource1[&Tech2/Resource2[&...]],Timeout,Optionen,URL)
```

```
Dial(Tech/User:Passwort@Host/Extension,Timeout,Optionen)
```

Ermöglicht es, zwei Kanäle miteinander zu verbinden.[10] `Dial()` ist eine der wichtigsten Anwendungen in Asterisk, lesen Sie diesen Ausschnitt deshalb bitte aufmerksam und eventuell mehrmals durch.

Jeder gültige Kanaltyp (wie z.B. SIP, IAX2, H.323, MGCP, Local oder Zap) wird von `Dial()` akzeptiert, aber die Parameter, die jedem Kanal übergeben werden müssen, hängen von denjenigen Informationen ab, die der entsprechende Kanaltyp benötigt, um seine Arbeit zu verrichten. So wird zum Beispiel ein SIP-Kanal eine Netzwerkadresse und einen Benutzer benötigen, zu dem die Verbindung hergestellt werden soll, während ein ZAP-Kanal irgendeine Art von Telefonnummer verlangt.

Wenn ein Netzwerk-basierter Kanaltyp spezifiziert wird, können Zielhost (Name oder IP-Adresse), Benutzername, Passwort und Fernsteuerungs-Extension als Teil der Optionen an `Dial()` übergeben werden, oder man kann auf den Namen eines Kanaleintrags in der entsprechenden Konfigurationsdatei (`.conf`-Datei) verweisen - alle notwendigen Informationen müssen dann in dieser Datei vorhanden sein. Benutzername und Passwort können durch den Namen der Kanalkonfigurationsdatei, umgeben von eckigen Klammern (`[]`) ersetzt werden. Die Angabe des Hostnamens ist optional.

Ein Beispiel für einen gültigen Aufruf:

```
exten => s,1,Dial(SIP/peter:geheim@meier.tld)
```

Das gleiche würde auch der folgende Aufruf bewirken:

```
exten => s,1,Dial(SIP/ein_SIP_friend)
```

Letzteres funktioniert nur, wenn in `sip.conf` ein entsprechender Kanal definiert ist:

```
[ein_SIP_friend]
fromuser=peter
password=geheim
host=meier.tld
```

Oft ist eine Extension wie folgt an die Adressinformation angehängt:

```
exten => s,1,Dial(IAX2/benutzer:passwort@host.tld/500)
```

Dadurch wird das entfernte Ende aufgefordert, den Anruf mit Extension 500 zu verbinden, und zwar in dem Kontext, in der der Kanal eingegangen ist. Diese Extension wird von `Dial()` nicht benötigt, weil die in der Kanalkonfiguration des entfernten Endes vorliegenden Daten verwendet werden oder der Remoteserver den Anruf zur `s`-Extension in dem Kontext, in dem der Anruf eingegangen ist, weiterleitet.

[10] z.B. IAX, SIP, H.323, Skinny, PRI, FXO, FXS, Local ...

Schlussendlich kontrolliert das entfernte Ende, was mit dem Anruf geschehen soll - Sie können lediglich eine spezielle Behandlung anfragen.

Wenn kein `Timeout` spezifiziert ist, wird der Kanal unendlich klingeln. Dieses Verhalten muss nicht immer schlecht sein, es besteht keine Veranlassung, ihn immer zu setzen - man muss sich nur vergegenwärtigen, dass unendlich evtl. eine sehr lange Zeit bedeuten kann.[11] `Timeout` wird in Sekunden angegeben. Der Klingel-Timeout folgt immer der Adressinformation, wie folgt:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500,20)
```

Mit `Dial()` können auch mehrere Channels parallel angerufen werden. Dabei gilt das Prinzip „Wer zuerst kommt, mahlt zuerst“. Welcher Angerufene als erstes abhebt, bekommt das Gespräch:

```
exten => s,1,Dial(SIP/2000&SIP/2001&SIP/2303)
```

Ein großer Teil der Mächtigkeit der `Dial()`-Anwendung liegt in den Flags. Diese werden nach der Adress- und Timeout-Information angegeben, und zwar so:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500,60,flags)
```

Wichtig! Es gilt zu beachten, dass, wenn Sie flags hinzufügen möchten, aber kein Timeout spezifiert ist, dennoch der Ort für den Timeout nicht fehlen darf. D. h., ein zusätzliches Komma muss an der Stelle eingefügt werden, an der normalerweise der Wert für den Timeout stehen würde, also so:

```
exten => s,1,Dial(IAX2/user:pass@otherend.com/500,,flags)
```

Die gültigen Flags, die mit der `Dial()`-Anwendung benutzt werden können, sind:

`d`
: Erlaubt es dem Teilnehmer, während des Wartens auf die Beantwortung eines Anrufs eine einziffrige Extension zu wählen. Der Anruf wird dann zu dieser Extension umgeleitet (d.h. der Anruf wird zugunsten dieser Extension beendet) (entweder im aktuellen Kontext oder, falls er spezifiziert wurde, im Kontext `${EXITCONTEXT}`).

`t`
: Erlaubt es dem angerufenen Teilnehmer, einen verbundenen Anruf durch Drücken der #-Taste zu übertragen. Es gilt zu beachten, dass Neueinladungen nicht möglich sind, wenn diese Option benutzt wird, weil Asterisk den Anruf überwachen muss, um zu erkennen, wenn die angerufene Seite die #-Taste drückt.

`T`
: Erlaubt es dem Anrufer, einen verbundenen Anruf durch Drücken der #-Taste zu übertragen. Auch hier beachten Sie bitte, dass Neueinladungen bei Benutzung dieser Option nicht möglich sind, weil Asterisk den Anruf überwachen muss, um zu erkennen, wenn die angerufene Seite die #-Taste drückt.

`w`
: Ermöglicht es dem angerufenen Teilnehmer den Anruf durch Eintippen der `automon`-Sequenz (wie in `features.conf` konfiguriert) aufzunehmen (Audio to disk). Falls die Variable `TOUCH_MONITOR` gesetzt ist, wird ihr Wert als Parameter der `Monitor()`-Anwendung weitergeleitet, wenn die Aufnahme startet. Ist sie nicht gesetzt, wird `Monitor()` die Vorgabe `WAV,,m` übergeben.

`W`
: Ermöglicht es dem Anrufer den Anruf durch Eintippen der `automon`-Sequenz (wie in `features.conf` konfiguriert) aufzunehmen (Audio to disk).

`f`
: Durch dieses Flag muss die Caller ID als die Extension der Leitung gesetzt werden, die den ausgehenden Anruf erstellt oder umleitet. Das wird so gemacht, weil einige PSTN-Diensteanbieter es nicht erlauben, die Caller ID auf irgendeinen anderen als den Ihnen zugewiesenen Wert zu setzen. Zum Beispiel würden Sie, wenn Sie einen PRI hatten, das Flag `f` verwenden um irgendeine lokal im SIP-Telefon gespeicherte Caller ID zu überscheiben.

[11] :-)

`o`
: Benutzt die auf der eingehenden Strecke des Anrufs empfangene Caller ID als die Caller ID auf der ausgehenden Richtung des Anrufs. Das ist nützlich, falls ein Anruf akzeptiert und dann zu einem anderen Ziel weitergeleitet wird, wobei die Caller ID der eingehenden Richtung des Anrufs weitergeben wird, statt diese zunächst mit den lokalen Caller ID-Einstellungen zu überschreiben.

`r`
: Signalisiert dem Anrufer Klingeln, ohne dass Audio weitergeleitet wird, bevor der Anruf beantwortet wird. Dieses Flag wird für gewöhnlich nicht benötigt um Klingeln zu signalisiern, da Asterisk Klingeln signalisiert, wenn ein Kanal angerufen wird.

`m[`*`Klasse`*`]`
: Beliefert den Anruf mit Musik, bis der Anruf beantwortet wird. Optional können Sie auch die Music-on-Hold-Klasse (z.B. Musikrichtung) angeben.

`M(`*`x`*`[^`*`arg`*`])`
: Führt auf die Verbindung eines Anrufs hin das Makro `x` aus, und leitet dabei optional Parameter, getrennt durch `^`, weiter. Das Makro kann auch die `MACRO_RESULT` Kanalvariable auf einen der folgenden Werte setzen:

`ABORT`	Legt bei beiden Enden des Anrufs auf.
`CONGESTION`	Wirkt, als wäre Stau auf der Leitung.
`BUSY`	Wirkt, als wäre die Leitung besetzt (geht zur Priorität n+101).
`CONTINUE`	Legt die Seite des Angerufenen auf und fährt im Wählplan fort.

`GOTO:<`*`Context`*`>^<`*`Extension`*`>^<`*`Priorität`*`>`
: Springt zu der angegebenen Position im Wählplan

`h`
: Erlaubt es dem angerufenen Teilnehmer, die Leitung durch Drücken von `*` aufzulegen.

`H`
: Erlaubt es dem anrufenden Teilnehmer, die Leitung durch Drücken von `*` aufzulegen.

`C`
: Setzt das Call Detail Record (Anrufprotokoll) zurück. Da die CDR-Zeit zum Zeitpunkt der Beantwortung des Anrufs gesetzt wird, ist es sinnvoll, CDR ebenfalls zurückzusetzen, damit dem Teilnehmer nicht die Zeit vor dem Aufruf der Call()-Anwendung in Rechnung gestellt wird.

`P[(`*`x`*`)]`
: Aktiviert den Geheimhaltungs-Modus, wobei optional `x` als Familie/Schlüssel-Wert in der AstDB spezifiziert werden kann. Dieser Modus ist beispielsweise nützlich, wenn Anrufe aus einer Blacklist (Anrufe von Nummern aus der Blacklist werden explizit abgelehnt) oder Whitelist (Anrufe von den gelisteten Nummern werden explizit akzeptiert) angenommen werden. Siehe auch `LookupBlacklist()`.

`g`
: Geht weiter im Kontext, wenn der Zielkanal aufgelegt wird.

`G(`*`context^extension^priority`*`)`
: Übermittelt beide Seiten zum spezifizierten Ziel, wenn der Anruf beantwortet wird.

`A(`*`x`*`)`
: Spielt dem angerufenen Teilnehmer eine Ankündigung vor. `x` ist der Dateiname der Audiodatei, die als Ankündigung abzuspielen ist.

`D([called][:calling])`
: Sendet DTMF-Ziffern, nachdem der Anruf beantwortet wurde, aber bevor der Anruf überbrückt wird. Der Parameter `called` wird an die angerufene Seite weitergegeben, und der Parameter `calling` an die anrufende Seite. Beide Parameter können individuell eingesetzt werden.

`L(x[:y][:z])`
: Begrenzt die Dauer des Anrufs auf `x` Millisekunden, wobei `y` Millisekunden vor Ablauf der festgesetzten Dauer und danach wiederholt alle `z` Millisekunden, bis zum Anlauf der Frist, eine Warnung signalisiert wird. Der `x`-Parameter muss angegeben werden, `y` und `z` sind optional. Die folgenden speziellen Variablen können ebenfalls gesetzt werden, und stellen zusätzliche Kontrollmöglichkeiten zur Verfügung:

  `LIMIT_PLAYAUDIO_CALLER=yes|no`
  : Gibt an, ob dem Anrufer Audiodateien vorgespielt werden sollen

  `LIMIT_PLAYAUDIO_CALLEE=yes|no`
  : Gibt an, ob dem Angerufenen Audiodateien vorgespielt werden sollen

  `LIMIT_TIMEOUT_FILE=filename`
  : Gibt an, welche Datei abgespielt werden soll, nachdem die Zeit abgelaufen ist

  `LIMIT_CONNECT_FILE=filename`
  : Gibt an, welche Datei zu Beginn des Anrufs abgespielt werden soll

  `LIMIT_WARNING_FILE=filename`
  : Gibt an, welche Datei gespielt werden soll, wenn der Parameter `y` angegeben ist

`n`
: Verhindert, dass zu Priorität n+101 übergegangen wird (`n`: aktuelle Priorität), falls alle Kanäle als besetzt erachtet werden.

  Ein Anruf kann auch geparkt werden, statt übermittelt zu werden (was mit `t` oder `T`-Flag der Fall ist). Anrufe werden gewöhnlich geparkt, indem man sie der Extension 700 übermittelt werden, aber dieses Verhalten ist in `features.conf` konfigurierbar.

  Mit dem Enden der `Dial()`-Anwendung werden die folgenden Variablen gesetzt:

  `DIALEDTIME` — Die gesamte Zeit, die von der Ausführung der `Dial()`-Anwendung an bis zu ihrer Beendigung verstrichen ist.

  `ANWEREDTIME` — Die gesamte Zeit, die während des Anrufs vergangen ist.

  `DIALSTATUS` — Der Status des Anrufs, ausgedrückt durch einen der folgenden Werte:

`CHANUNAVAIL`	Der Kanal ist nicht verfügbar.
`CONGESTION`	Der Kanal hat ein Stau-Signal zurückgeliefert, was gewöhnlich die Unfähigkeit der Fertigstellung der Verbindung kennzeichnet.
`NOANSWER`	Der Kanal hat in der durch die Klingel-Timeout Option gesetzten Frist nicht geantwortet.
`BUSY`	Der angerufene Kanal ist momentan belegt.
`ANSWER`	Der Kanal hat den Anruf beantwortet.
`CANCEL`	Der Anruf wurde abgebrochen.

```
; eine Nummer auf Zap-Kanal 2 wählen, max. 10 Sekunden klingeln:
exten => 123,1,Dial(Zap/2/1234567,10,tTm)
; sonst weiter im Dialplan:
exten => 123,n,Playback(tut-uns-leid)
exten => 123,n,Hangup()
```

```
; über IAX die Extension 500 auf dem Host host.tld wählen:
exten => 123,1,Dial(IAX/username:password@host.tld/500)
```

Siehe auch. Abschnitt 6.78, „`RetryDial()`“

6.23. `Dictate()`

Virtuelles Diktiergerät

```
Dictate([Verzeichnis[,Dateiname]])
```

Startet ein virtuelles Diktiergerät. Mit den Parametern lassen sich das Verzeichnis für die Aufzeichnungen (Default: `/var/spool/asterisk/dictate/`) und der Dateiname (numerisch!) bestimmen. Aufgezeichnet wird im Format `raw`.

Der User kann das Diktiergerät mit folgenden Tasten steuern:

`0` Hilfe

`1` Schaltet zwischen Aufnahme- und Abspielmodus hin und her.

`*` Pause / weiter

`#` Datei wählen / neuen Dateinamen eingeben (z.B. `1234#`)

Im Abspielmodus:

`2` Schaltet die Abspielgeschwindigkeit um (1x, 2x, 3x, 4x)

`7` Ein Stück zurück springen

`8` Ein Stück weiter springen

Im Aufnahmemodus:

`8` Aufnahme löschen und neu anfangen

```
; Diktiergerät starten:
exten => 123,1,Dictate()
```

Um jedem User sein eigenes Diktiergerät zu geben, könnte man z.B. als Pfad `/var/spool/asterisk/dictate/${EXTEN}` angeben.

Siehe auch. Abschnitt 6.74, „`Record()`“

6.24. `Directory()`

Stellt ein Verzeichnis anwählbarer (Voicemail-)Extensions bereit (internes Telefonbuch, Dial-by-name, siehe Abschnitt 5, „Telefonbuch (Dial-by-Name)“).

```
Directory(VM-Kontext[,Wähl-Kontext[,Optionen]])
```

Liefert Benutzern ein Verzeichnis mit Extensionen, aus dem anhand des Namens ausgewählt werden kann. Die Liste von Namen und Extensionen findet sich in `voicemail.conf`. Der `VM-Kontext` muss angegeben werden, er bestimmt den zu benutzenden Kontext aus `voicemail.conf`.

Der `Wähl-Kontext` bestimmt den zum Anrufen der Benutzer zu verwendenden Kontext. Falls er nicht angegeben ist, wird `VM-Kontext` angenommen. Momentan ist die einzige mögliche Option, die im `Optionen`-Parameter Verwendung finden kann, `f` (wie first name), welche das Verzeichnis veranlasst, die Eingabe anhand des Vornamens statt des Nachnamens abzugleichen.

Gibt der Benutzer `0` (Null) ein, und es existiert eine Extension o (kleiner Buchstabe o) im aktuellen Kontext, geht die Steuerung des Anrufs an diese Extension über. Durch Drücken der Taste * wird analog dazu zur a-Extension übergegangen. Diese Verhalten ähnelt dem von `Voicemail()`.

Gibt 0 zurück, es sei denn der Teilnehmer legt auf.

```
exten => *,1,Directory(default,incoming)
exten => #,1,Directory(default,incoming,f)
```

Siehe auch. `voicemail.conf`

6.25. `DISA()`

Direct Inward System Access erlaubt auswärtigen Anrufern Anrufe mit internem System-Wählton

```
DISA(Passwort[,Kontext[,CallerID[,Mailbox[@VM-Kontext]]]])
```

```
DISA(Passwort-Datei[,CallerID[,Mailbox[@VM-Kontext]]])
```

Erlaubt Anrufern von außen einen internen System-Wählton zu erhalten und entsprechend Anrufe zu tätigen, als würden diese von innerhalb des Switches aus getätigt. Dem Teilnehmer wird ein Wählton zur Verfügung gestellt, nach dem er einen Zugangskode eingeben muss gefolgt von der Rautetaste (#). Ist der Zugangskode korrekt, erhält er einen System-Wählton auf dem er einen Anruf tätigen kann.

Achtung

Diese Art von Zugang kann ein ernstes Sicherheitsrisiko sein und sollte sorgfältig überlegt werden, damit die Sicherheit Ihres Systems nicht gefährdet ist!

Der `Passwort`-Parameter ist ein numerischer Zugangskode, den der Benutzer eingeben muss um Anrufe tätigen zu können. Mit dieser Syntax werden alle Benutzer dieser Extension das gleiche Passwort benutzen. Möchten Sie Benutzern erlauben, `DISA()` ohne Passwort zu verwenden, übergeben Sie die Zeichenkette `no-password` an der Stelle des Passwortes.

Der `Kontext`-Parameter spezifiziert den Kontext, in dem sich ein Benutzer einwählen wird. Wird kein Kontext spezifiziert, nimmt die `DISA()`-Anwendung den Kontext `disa` an.

Die `CallerID` ist die Mailbox-Nummer (und optional Voicemail-Kontext `VM-Kontext`) einer Voicemail-Box. Der Anrufer wird einen sogenannten „stuttered dial-tone“ (das ist ein regelmäßig kurz unterbrochenen, „stotternder“ Wählton) hören, falls neue Nachrichten in der angegebenen Voicemail-Box vorliegen.

Außerdem können Sie eine alternative Syntax benutzen und den Namen einer global verfügbaren Passwortdatei statt der Parameter `Passwort` und `Kontext` übergeben. In jeder Zeile kann diese Datei entweder einen Zugangskode oder eine Kombination aus Zugangskode und Kontext, getrennt durch das Zeichen `|`, enthalten. Ist kein Kontext mit angegeben, wird standardmäßig der Kontext `disa` verwendet.

Ist das Einloggen des Teilnehmers erfolgreich, wird die Anwendung die gewählte Nummer in dem spezifizierten Kontext interpretieren.

```
; Anrufern von außen erlauben, 0800er-Nummern zu wählen, sofern sie
; das Passwort (1234) wissen. Dabei ihre Caller-ID so setzen, dass
; es aussieht, als riefen sie von innerhalb der Firma an:
[incoming]
exten => 123,1,DISA(1234,disa,Apfelmus GmbH <(0261) 1234567>)
[disa]
exten => _0800XXXXXXX,1,Dial(Zap/4/${EXTEN})
```

6.26. `DumpChan()`

Gibt Informationen über den anrufenden Kanal auf der Konsole aus

```
DumpChan([min_verbose_level])
```

Zeigt Informationen über den anrufen Kanal an, als auch eine Auflistung aller Kanalvariablen. Falls `min_verbose_level` angegeben ist, werden Ausgaben nur im Falle eines aktuell gleich hoch oder höher gesetzten Verbosity-Levels angezeigt.

Gibt stets 0 zurück.

```
exten => s,1,Answer()
exten => s,2,DumpChan()
exten => s,3,Background(enter-ext-of-person)
```

6.27. `EAGI()`

Siehe. Abschnitt 6.6, „`AGI()`“.

6.28. `Echo()`

Wiederholt inwärtiges Audio für den Anrufer

```
Echo()
```

Wiederholt vom Kanal gelesenes Audio wieder auf diesem Kanal aus. Diese Anwendung findet häufig zum Testen der Verzögerung und Sprachqualität einer VoIP-Verbindung Verwendung. Der Anrufer kann zum beenden die #-Taste drücken.

Gibt 0 zurück, falls der Benutzer durch die #-Taste beendet, oder -1, falls der Benutzer auflegt.

```
exten => 123,1,Echo()
exten => 123,2,Playback(vm-goodbye)
```

Siehe auch. Abschnitt 6.52, „`Milliwatt()`“

6.29. `EndWhile()`

Beendet eine While-Schleife

```
EndWhile()
```

Kehrt zur vorangehend aufgerufenen `While()`-Anwendung zurück. Für eine vollständige Beschreibung zur Benutzung einer While-Schleife siehe Abschnitt 6.117, „`While()`“.

```
exten => 123,1,Set(COUNT=1)
exten => 123,2,While($[ ${COUNT} < 5 ])
exten => 123,3,SayNumber(${COUNT})
exten => 123,4,EndWhile()
```

Siehe auch. Abschnitt 6.117, „`While()`“, Abschnitt 6.40, „`GotoIf()`“

6.30. `Exec()`

Führt eine Asterisk-Anwendung dynamisch aus

```
Exec(Applikationsname(Argumente))
```

Erlaubt den Aufruf einer beliebigen Anwendung, selbst wenn er nicht fest im Wählplan einprogrammiert ist. Liefert den Wert zurück, den diese Asterisk-Anwendung zurückliefert, oder -2, wenn die aufgerufene Anwendung nicht gefunden werden kann. Die Parameter `arguments` werden an die aufgerufene Anwendung durchgereicht.

Diese Anwendung erlaubt den dynamischen Aufruf von Anwendungen, die aus einer Datenbank oder externen Quelle gelesen werden.

```
exten => 123,1,Set(app=SayDigits(12345))
exten => 123,2,Exec(${app})
```

Siehe auch. Abschnitt 6.31, „`ExecIf()`“, Abschnitt 6.102, „`System()`“

6.31. `ExecIf()`

Führt eine Asterisk-Anwendung unter bestimmten Bedingungen aus

```
ExecIf(Ausdruck,Applikation,Argumente)
```

Ist `Ausdruck` wahr, wird die angegebene `Applikation` mit den Parametern `Argumente` ausgeführt, und deren Ergebnis zurückgeliefert. Siehe `doc/README.variables` (1.2) / `doc/channelvariables.txt` (1.4) für weiterführende Informationen über Standard-Asterisk-Ausdrücke.

Ist der `Ausdruck` nicht wahr (also false), wird die Ausführung mit der nächsten Priorität forgesetzt.

```
exten => 123,1,ExecIf($[ ${CALLERID(num)} = 101],SayDigits,12345)
exten => 123,2,SayDigits(6789)
```

Siehe auch. Abschnitt 6.30, „`Exec()`“, ???

6.32. `FastAGI()`

Führt ein AGI-konformes Skript über eine Netzwerkverbindung aus

```
FastAGI(agi://hostname[:Port][/script],Argumente)
```

Führt ein AGI-konformes Skript über das Netzwerk aus. Diese Anwendung ist `AGI()` sehr ähnlich, außer dass sie ein FastAGI-Skript über eine Netzwerkverbindung ausführt. Der wichtigste Verwendungszweck ist, rechenintensive AGI-Skripte von Remote-Servern ausführen zu lassen und die Startzeiten von AGI-Skripten zu reduzieren (ein FastAGI Skript läuft bereits bevor Asterisk sich mit ihm verbindet, ähnlich FastCGI bei Web-Servern).

`FastAGI()` versucht sich direkt zu einem laufenden FastAGI-Programm zu verbinden, welches auf einem spezifizierten `Port` des Servers `hostname` bereits auf Verbindungen wartet. Als Standardport wird 4573 benutzt, falls kein anderer angegeben ist. Ist `script` angegeben, wird es als `agi_network_script`-Variable an das FastAGI-Programm übergeben. Die in `Argumente` angegebenen Parameter werden an das Programm weitergegeben.

Im Verzeichnis `agi/fastagi-test` befindet sich ein Beispiel-FastAGI-Skript. Verwenden Sie es als Ansatzpunkt für Ihre eigenen FastAGI-Programme.

Liefert -1, falls die Anwendung ein Auflegen verlangt, oder 0 bei Beendigung ohne Auflegen.

```
; mit dem Beispiel-FastAGI-Programm "fastagi-test" verbinden, das
; bereits auf dem lokalen Rechner laufen muss:
exten => 123,1,Answer()
exten => 123,n,FastAGI(agi://localhost/fastagi-test)

; mit dem FastAGI-Script "test" auf einem Host namens "testbox" auf
; Port 9000 verbinden und als Parameter "123" übergeben:
exten => 124,1,Answer()
exten => 124,n,FastAGI(agi://testbox:9000/test,123)
```

Siehe auch. Abschnitt 6.6, „`AGI()`“, Abschnitt 6.21, „`DeadAGI()`“

6.33. `Festival()`

Verwendet das Sprachsynthese-System (Text-to-speech) Festival um dem Anrufer Text vorzulesen

```
Festival(Text[,Tasten])
```

Verbindet sich zum lokal laufenden Festival-Server (muss installiert sein), sendet ihm den angegebenen `Text` und spielt dem Teilnehmer die resultierende Audiodatei vor. Wenn `Tasten` angegeben wurden und der Anrufer während des Abspielens eine davon drückt, wird der Abspielvorgang beendet und die entsprechende Taste zurückgegeben. Falls als `Tasten` der Wert `any` angegeben ist, wird jede Taste erkannt (und an die entsprechende Extension weitergeleitet).

Sie müssen den Festival-Server *vor* Asterisk starten und die `Answer()`-Anwendung zum Beantworten des Kanals verwenden, bevor sie `Festival()` aufrufen.

```
exten => 123,1,Answer()
exten => 123,n,Festival('Hallo Welt',#)
```

Alternativ zu der Applikation `Festival()` kann man auch mit dem `System()`-Befehl Festivals Programm **text2wave** aufrufen und die resultierende Audio-Datei mit `Background()` oder `Playback()` abspielen, etwa so (nur als Beispiel, Verzeichnisse beachten!):

```
exten => 123,1,Answer()
exten => 123,n,System(echo 'Hallo Welt' | text2wave -o sound.wav -otype wav -)
exten => 123,n,Background(sound)
```

Siehe auch. `contrib/README.festival`

6.34. `Flash()`

Flasht einen Zap-Kanal

```
Flash()
```

Sendet ein Flash auf einen Zap-Kanal. Dies ist nichts weiter als ein Hack für Leute, die Übermittlungen und andere Aktionen, die einen Flash durch ein AGI-Skript benötigen, durchführen möchten.

> Ein Flash (auch Switch-Hook-Flash oder Link genannt) entspricht dem ganz kurzen Drücken der Gabel für 80 - 500 Millisekunden (je nach Anbieter) an einem analogen Telefon (manche Apparate haben dafür eine Taste), wodurch z.B. Rückfrage-Gespräche o.ä. eingeleitet werden, ohne den aktuellen Anruf aufzulegen.

Liefert bei Erfolg 0 zurück, oder -1, falls der Kanal kein Zap-Kanal ist.

```
exten => 123,1,Flash()
```

Falls Ihnen auf Ihrer Leitung Link-Transfer bereitgestellt wird (typischerweise ein Zusatzdienst), könnten Sie so einen Anruf auf eine andere externe Nummer umleiten:

```
[macro-link-umleitung]
exten => s,1,Playback(transfer)
exten => s,n,Flash()
exten => s,n,Wait(1)
exten => s,n,SendDTMF(${ARG1})
exten => s,n,Wait(1)
exten => s,n,Hangup()

[externe-apparate]
; eingehende Anrufe auf 6001 auf die externe Nummer 0261123456 umleiten:
exten => 6001,1,Macro(link-umleitung,0261123456)
```

Es kann dazu nötig sein, die Flash-Dauer (Einstellung `flash`) in der `zapata.conf` auf die Vorgabe Ihres Anbieters zu setzen, z.B. `flash=200`.

6.35. `ForkCDR()`

Erstellt ein zusätzliches CDR aus dem aktuellen Anruf.

```
ForkCDR()
```

Erstellt für den Rest des aktuellen Anrufs einen zusätzlichen Call Detail Record, um die in Rechnung zu stellende Gesprächszeit von der Gesamtdauer des Anrufs zu unterscheiden.

```
exten => 123,n,ForkCDR()
```

Siehe auch. ???, Abschnitt 6.58, „`NoCDR()`“, Abschnitt 6.76, „`ResetCDR()`“, ???

6.36. `GetCPEID()`

Holt die CPE-ID eines ADSI-fähigen Telefons

```
GetCPEID()
```

Holt die CPE-ID und weitere Informationen und zeigt sie auf der Asterisk-Konsole an. Diese Informationen werden häufig benötigt, um die Datei `zapata.conf` für Operationen bei ADSI-fähigen Telefonen einzurichten.

Liefert zur beim Auflegen -1 zurück.

```
exten => 123,1,GetCPEID()
```

Siehe auch. Abschnitt 6.2, „`ADSIProg()`“, `adsi.conf`, `zapata.conf`

6.37. `Gosub()`

Springt zu einer bestimmten Priorität, Extension und den Kontext (mit der Möglichkeit der Rückkehr).

```
Gosub([[Kontext,]Extension,]Priorität)
```

```
Gosub(benannte_Priorität)
```

Springt wie `Goto()` im Dialplan, erlaubt es dem Unterprogramm aber, mit `Return()` zurückzukehren.

Gibt 0 zurück, oder -1, wenn das Sprungziel ungültig ist.

```
exten => 123,1,Gosub(cid-setzen)
exten => 123,n,Dial(Zap/g0/${EXTEN})
exten => 123,n,Congestion()

exten => 123,10(cid-setzen),Set(CALLERID(all)=Apfelmus <012345678>)
exten => 123,n,Return()
```

Siehe auch. Abschnitt 6.38, „`GosubIf()`“, Abschnitt 6.39, „`Goto()`“, Abschnitt 6.40, „`GotoIf()`“, Abschnitt 6.79, „`Return()`“, Abschnitt 6.47, „`Macro()`“

6.38. `GosubIf()`

Springt bedingt zu der angegebenen Priorität (mit der Möglichkeit der Rückkehr).

```
GosubIf(Bedingung?PrioWahr:PrioFalsch)
```

Springt wie `GotoIf()` unter einer bestimmten Bedingung im Dialplan, erlaubt es dem Unterprogramm aber, mit `Return()` zurückzukehren.

Gibt 0 zurück, oder -1, wenn das Sprungziel ungültig ist.

```
exten => telcid,1,Set(CALLERID(all)=Apfelmus <0123456780>)
exten => telcid,n,Return()
exten => faxcid,1,Set(CALLERID(all)=Apfelmus <0123456785>)
exten => faxcid,n,Return()

exten => _0.,1,GosubIf($[${CHANNEL:4:2} = 43]?faxcid,1:telcid,1)
exten => _0.,n,Dial(${TRUNK}/${EXTEN:1},,T)
```

Siehe auch. Abschnitt 6.37, „`Gosub()`“, Abschnitt 6.39, „`Goto()`“, Abschnitt 6.40, „`GotoIf()`“, Abschnitt 6.79, „`Return()`“, Abschnitt 6.47, „`Macro()`“

6.39. `Goto()`

Springt zu einer bestimmten Priorität, Extension und den Kontext.

```
Goto([[Kontext,]Extension,]Priorität)
```

```
Goto(benannte_Priorität)
```

Übergibt die Kontrolle des aktuellen Kanals ohne Möglichkeit der Rückkehr an die spezifizierte Priorität, und setzt optional Extension und Kontext des Ziels.

Optional können Sie die Anwendung dazu benutzen, zu der durch den Parameter `benannte_Priorität` angegebenen Priorität (also Priorität mit Label) zu gelangen. Benannte Prioritäten funktionieren ausschließlich in der aktuellen Extension.

Liefert immer 0 zurück, auch falls der gegebene Kontext, die geg. Extension oder Priorität nicht gültig sind.

```
exten => 123,1,Answer()
exten => 123,2,Set(COUNT=1)
exten => 123,3,SayNumber(${COUNT})
exten => 123,4,Set(COUNT=$[ ${COUNT} + 1 ])
exten => 123,5,Goto(3)

; das gleiche mit einer benannten Priorität:
exten => 124,1,Answer()
exten => 124,2,Set(COUNT=1)
exten => 124,3(wdh),SayNumber(${COUNT})
exten => 124,4,Set(COUNT=$[ ${COUNT} + 1 ])
exten => 124,5,Goto(wdh)
```

Siehe auch. Abschnitt 6.40, „`GotoIf()`“, Abschnitt 6.41, „`GotoIfTime()`“, Abschnitt 6.37, „`Gosub()`“, Abschnitt 6.38, „`GosubIf()`“, Abschnitt 6.47, „`Macro()`“

6.40. `GotoIf()`

Springt bedingt zu der angegebenen Priorität

```
GotoIf(Bedingung?PrioWahr:PrioFalsch)
```

Schickt den Anruf ohne Möglichkeit der Rückkehr an die Priorität `PrioWahr`, falls die `Bedingung` wahr ist oder zu `PrioFalsch`, falls sie falsch ist. Entweder `PrioWahr` oder `PrioFalsch` können ausgelassen werden (dann wird ggf. mit der nächsten Priorität weitergemacht), aber nicht beide (und nicht der Doppelpunkt!).

Eine Priorität ist in hier eines der Folgenden:

- eine Priorität, z.B. `10`
- eine Extension und eine Priorität, z.B. `123,10`
- Kontext, Extension und Priorität, z.B. `incoming,123,10`
- eine benannte Priorität innerhalb der gleichen Extension, z.B. `ok`

```
exten => 123,1,GotoIf($[ ${test} = 5 ]?ok:nein)
exten => 123,10(ok),Playback(tt-monkeys)
exten => 123,20(nein),Playback(tt-weasels)
```

Siehe auch. Abschnitt 6.39, „`Goto()`“, Abschnitt 6.41, „`GotoIfTime()`“, Abschnitt 6.37, „`Gosub()`“, Abschnitt 6.38, „`GosubIf()`“, Abschnitt 6.47, „`Macro()`“

6.41. `GotoIfTime()`

Verzweigt bedingt, abhängig von Zeit und Tag

```
GotoIfTime(Zeit,Wochentag,Monatstag,Monat?[[Kontext,]Extension,]Prio)
```

Verzweigt zu der angegebenen Priorität, falls die aktuelle Zeit in das angegebene Muster fällt. Jedes Element kann auch als * (für immer) oder als Spanne (mit -) definiert werden.

Die Parameter zu dieser Anwendung sind:

`Uhrzeit-Spanne` Zeitintervall, im 24-Stunden-Format mit Minuten, z.B. `9:00-17:00`

`Wochentag`	Wochentage (`mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`), z.B. `mon-fri`
`Monatstag`	Tag des Monats (1-31), z.B. `1-15`
`Monat`	Monate (`jan`, `feb`, `mar`, `apr`, `mai`, `jun`, `jul`, `aug`, `sep`, `oct`, `nov`, `dec`), z.B. `apr-oct`

```
; Während der Öffnungszeiten zum Kontext incoming-geoeffnet springen.
; Wir haben montags-freitags von 9 bis 18 Uhr geöffnet:
exten => s,1,GotoIfTime(09:00-17:59,mon-fri,*,*?incoming-geoeffnet,s,1)
; außerdem samstags von 9 bis 12:
exten => s,n,GotoIfTime(09:00-11:59,sat,*,*?incoming-geoeffnet,s,1)
; Außerhalb der Öffnungszeiten zum Kontext incoming-geschlossen:
exten => s,n,Goto(incoming-geschlossen,s,1)
```

Siehe auch. Abschnitt 6.40, „`GotoIf()`“, Abschnitt 7.27, „`IFTIME()`“

6.42. `Hangup()`

Legt den aktuellen Kanal auf.

```
Hangup()
```

Legt den aktuellen Kanal bedingungslos auf.

Liefert stets -1 zurück.

```
exten => 123,1,Answer()
exten => 123,n,Playback(vm-goodbye)
exten => 123,n,Hangup()
```

Siehe auch. Abschnitt 6.8, „`Answer()`“

6.43. `IAX2Provision()`

Versorgt eine anrufende IAXy-Einheit

```
IAX2Provision([Template])
```

Stellt einer anrufende IAXy-Einheit die Vorlage `Template` bereit. Ist keine Vorlage angegeben, wird die Standard-Vorlage benutzt. IAXy-Versorgungsvorlagen sind in der Konfigurationsdatei `iaxprov.conf` definiert.

Liefert 0 bei Erfolg oder -1 bei einem Fehler.

```
exten => 123,1,IAX2Provision(default)
```

6.44. `ImportVar()`

Setzt eine Variable basierend auf einer Kanalvariable eines anderen Kanals

```
ImportVar(NeueVariable=Kanal,Variable)
```

Setzt die Variable `NeueVariable` auf den Wert von `Variable` des anegegebenen Kanals. Falls `NeueVariable` mit dem Zeichen _ beginnt, wird einfache Vererbung angenommen. Falls es `newvar` mit __ beginnt, wird unbeschränkte Vererbung angenommen.

```
; Caller-ID vom Kanal Zap/1 importieren:
exten => 123,1,Answer()
exten => 123,n,ImportVar(cid=Zap/1,CALLERID)
```

Siehe auch. Abschnitt 6.90, „`Set()`“

6.45. `LookupBlacklist()`

Schlägt die Caller-ID-Nummer in der lokalen Blacklist-Datenbank in der AstDB nach

```
LookupBlacklist([Optionen])
```

Sucht die CallerID-Nummer (/-Name) des aktiven Kanals in der AstDB (Familie `blacklist`). Wenn die Option j (jump) angegeben ist, die Nummer gefunden wird und eine Priorität n+101 existiert, geht die Ausführung bei dieser Priorität weiter. Wird auf dem Kanal keine CallerID empfangen, tut die Anwendung nichts.

Die Applikation setzt auch die Kanal-Variable `LOOKUPBLSTATUS` auf `FOUND` (gefunden) oder `NOTFOUND` (nicht gefunden).

Um der Blacklist Einträge hinzuzufügen, geben Sie im CLI `database put blacklist "Nummer" "1"`, bzw. `database del blacklist "Nummer"` zum Löschen oder `database show blacklist` für eine Auflistung ein.

```
; Nummern aus der Schwarzen Liste in eine Endlosschleife schicken,
; andernfalls die Nummer in der Variablen ${PETER} wählen:
exten => 123,1,Answer()
exten => 123,2,LookupBlacklist(j)
exten => 123,3,Dial(${PETER},30)
exten => 123,103,Playback(tt-allbusy)
exten => 123,104,Wait(10)
exten => 123,105,Goto(103)
```

Die Applikation `LookupBlacklist()` ist offenbar ab Asterisk 1.4 deprecated und kann folgendermaßen ersetzt werden:

```
exten => 123,1,Macro(blacklist,${CALLERID(num)})
exten => 123,n,Dial(IAX2/benutzer:passwort@beispiel.de/500)

[macro-blacklist]
; Aufruf: Macro(blacklist,${CALLERID(num)})
exten => s,1,GotoIf(${DB_EXISTS(blacklist/${ARG1})}?black)
exten => s,10(black),NoOp(Nummer ${ARG1} ist auf der Blacklist)
exten => s,n,Busy(5)
exten => s,n,Hangup()
```

6.46. `LookupCIDName()`

Sucht einen Caller ID Namen in der AstDB

```
LookupCIDName()
```

Schlägt die CallerID-Nummer des aktiven Kanals in der AstDB (Familie `cidname`) nach und setzt den CallerID-Namen - wenn vorhanden - auf den gespeicherten Wert. Diese Anwendung tut nichts, wenn auf dem Kanal keine CallerID empfangen wird. `LookupCIDName()` kann nützlich sein, wenn Sie nur eine CallerID-Nummer aber keinen -Namen empfangen, oder wenn Sie den CallerID-Namen für einige eingehende Anrufe ändern möchten.

Liefert stets 0 zurück.

Um der Liste Einträge hinzuzufügen, geben Sie im CLI `database put cidname "Nummer" "Name"`, bzw. `database del cidname "Nummer"` zum Löschen oder `database show cidname` für eine Auflistung ein.

```
exten => 123,1,Answer()
exten => 123,n,LookupCIDName()
```

6.47. `Macro()`

Ruft ein zuvor definiertes Makro auf

```
Macro(macroname[,Arg1[,Arg2[,...]]])
```

Führt ein im Kontext `macro-macroname` definiertes Makro aus, in dem es zur `s`-Extension dieses Kontextes springt und nach der Ausführung des Macros zurückkehrt.

Die angerufene Extension, Kontext und Priorität sind innerhalb des Macros in `${MACRO_EXTEN}`, `${MACRO_CONTEXT}` und `${MACRO_PRIORITY}` verfügbar. Die Parameter `Arg1`, `Arg2` ... werden im Makrokontext zu `${ARG1}`, `${ARG2}` usw.

`Macro()` liefert -1 zurück, falls irgendein Schritt des Makros -1 zurückliefert, sonst 0. Falls bei Beendigung des Ablaufs `${MACRO_OFFSET}` gesetzt ist, versucht die Anwendung bei Priorität n+1+`MACRO_OFFSET` fortzufahren, falls ein solcher Schritt existiert, sonst bei n+1.

Falls die `Goto()`-Anwendung aus dem Makro heraus aufgerufen wird, wird das Makro enden und die Kontrolle an das Ziel von `Goto()` gehen.

```
; ein Macro definieren, das vom übergebenen Wert runterzählt:
[macro-countdown]
exten => s,1,Set(COUNT=${ARG1})
exten => s,n,While($[ ${COUNT} > 0])
exten => s,n,SayNumber(${COUNT})
exten => s,n,Set(COUNT=$[ ${COUNT} - 1 ])
exten => s,n,EndWhile()
; das Macro aufrufen:
[default]
exten => 123,1,Macro(countdown,3)
exten => 124,1,Macro(countdown,5)
```

Siehe auch. Abschnitt 6.39, „`Goto()`“, Abschnitt 6.37, „`Gosub()`“

6.48. `MailboxExists()`

Verzweigt bedingt, falls die angegebene Voicemail-Box existiert

```
MailboxExists(Mailbox[@Kontext][,Optionen])
```

Prüft, ob die durch `Mailbox` angegebene Voicemail-Box existiert. Sie können zusätzlich einen Voicemail-Kontext `Kontext` übergeben, wenn die Mailbox nicht zum Standard-Voicemail-Kontext gehört.

Setzt die Kanal-Variable `VMBOXEXISTSSTATUS` auf `SUCCESS` (Erfolg, Mailbox vorhanden) oder `FAILED` (fehlgeschlagen, Mailbox nicht vorhanden).

Mit der Option `j` wird bei vorhandener Mailbox zur Priorität n+101 gesprungen.

```
exten => 123,1,Answer()
exten => 123,n,MailboxExists(123@default)
exten => 123,n,Goto(box-${VMBOXEXISTSSTATUS})
exten => 123,10(box-SUCCESS),Voicemail(u123)
exten => 123,20(box-FAILED),Playback(tut-uns-leid)
```

Siehe auch. Abschnitt 7.56, „`VMCOUNT()`“

6.49. `MeetMe()`

Nimmt den Anrufer in eine MeetMe-Konferenz auf

```
MeetMe([KonferenzNr][,Optionen[,PIN]])
```

Verbindet den Anrufer auf dem aktuellen Kanal zu einer MeetMe-Konferenz, angegeben durch den Parameter `KonferenzNr`. Ohne Angabe der Konferenznummer wird der Teilnehmer zur Eingabe derselben aufgefordert.

Wird als `PIN`-Parameter die PIN-Nummer der Konferenz (statisch in `meetme.conf` oder dynamisch vom Ersteller festgelegt) übergeben, so wir der Anrufer direkt in die Konferenz geleitet. Ansonsten muss der Anrufer die PIN eingeben um beitreten zu können.

Wichtig

Damit MeetMe-Konferenzen funktionieren, muss ein passendes Zaptel-Interface installiert sein. Alternativ kann auch der `ztdummy`-Treiber für Timing-Zwecke verwendet werden.

MeetMe-Konferenzen benutzen offenbar intern immer den uLaw-Codec, je mehr Teilnehmer in einer Konferenz andere Codecs wie GSM verwenden, desto mehr Rechenleistung wird für die Umkodierung benötigt.

Die `Optionen` können sich aus folgenden Zeichen zusammensetzen:

`a` Aktiviert den Administrator-Modus.

`A` Markiert den eintretenden User als besonderen Teilnehmer (siehe `w` und `x`).

`b` Startet das in `${MEETME_AGI_BACKGROUND}` spezifizierte AGI-Skript, standardmäßig `conf-background.agi`. (Funktioniert nur dann korrekt, wenn alle Kanäle in einer Konferenz Zap-Kanäle sind.) Das Skript erhält alle DTMF-Tanstendrücke, wodurch die Kombination mit Optionen, die auch das Abfangen von Tasten zur Folge haben (`p`) nicht möglich ist.

`c` Kündigt beim Betreten die Zahl der Teilnehmer an.

`d` Legt dynamisch einen neuen Konferenzraum an.

`D` Legt dynamisch einen neuen Konferenzraum an, wobei die Festlegung der PIN verlangt wird. (Wenn der User keine PIN will, muss er # drücken.)

`e` Wählt eine leere Konferenz aus.

`E` Wählt eine leere Konferenz aus, die keine PIN verlangt.

`i` Kündigt Betreten oder Verlassen eines Teilnehmers an (nur mit Zap-Kanälen).

`m` Aktiviert den Monitor-Modus (nur hören, nicht sprechen).

`M` Aktiviert Wartemusik (Music on Hold), wenn nur ein Teilnehmer in der Konferenz ist.

`p` Ermöglicht es dem Teilnehmer, die Konferenz durch Drücken von # zu verlassen.

`P` Fragt immer nach der PIN, auch wenn sie hier im Befehl angegeben ist.

`q` Aktiviert den ruhigen Modus (spielt keine Eintritts-/Austrittsklänge ab).

`r` Zeichnet eine Konferenz als auf. Datei: `${MEETME_RECORDINGFILE}`, Format: `${MEETME_RECORDINGFORMAT}`. Der Standard-Dateiname lautet `meetme-conf-rec-${`*KonferenzNr*`}-${`*UniqueID*`}`, wobei `UniqueID` eine einmalige ID ist. Das Standardformat ist `wav`. (Funktioniert nur mit Zap-Kanälen)

`s` Wechselt beim Drücken von * ins (Teilnehmer- oder Administrator-)Menü.

`t` Aktiviert den Nur-Sprechen-Modus (nur reden, nicht hören).

`T` Aktiviert die Sprecher-Erkennung (wird zum Manager-Inferace gesendet und in der MeetMe-Liste im CLI angezeigt).

`v` Aktiviert den Video-Modus (bisher ohne Funktion).

`w` Wartet, bis ein gekennzeichneter Teilnehmer (siehe `A`) die Konferenz betritt. Bis dahin hören alle Teilnehmer Wartemusik.

`x` Schließt die Konferenz, wenn sie vom letzten gekennzeichneten Teilnehmer (siehe `A`) verlassen wird.

`X` Ermöglicht es dem Teilnehmer, die Konferenz durch Eingabe einer einziffrigen Extension im Kontext `${MEETME_EXIT_CONTEXT}` zu verlassen, oder durch Eingabe der Nummer einer Extension des aktuellen Kontexts, falls diese Variable nicht definiert ist.

Die Option `X` funktioniert nicht zusammen mit `p` oder `s`.

Verwenden Sie `e` (oder `E`) zusammen mit `d` (oder `D`), um dynamisch einen neuen Konferenzraum zu eröffnen, wobei Asterisk automatisch die `KonferenzNr` festlegt. Sie müssen dann natürlich den anderen Teilnehmern die Nummer mitteilen oder entsprechende Dialplan-Logik einsetzen.

Anmerkung

Mit `d` oder `D` werden *dynamisch* Konferenzräume angelegt, in der `meetme.conf` können sie *statisch* definiert werden.

```
exten => 123,1,Answer()
; den Anrufer in Konferenzraum 333 schicken (mit der PIN 1234):
exten => 123,n,MeetMe(333,DpM,1234)
```

Siehe auch. Abschnitt 6.50, „`MeetMeAdmin()`“, Abschnitt 6.51, „`MeetMeCount()`“

Befehle im CLI. Diese Befehle stehen im CLI zur Verfügung, um MeetMe-Konferenzen zu verwalten (Dabei ist die `TeilnehmerNr` die in der Liste angezeigte Nummer des Teilnehmers):

`MeetMe`
Listet alle Konferenzen auf.

`MeetMe list` *`KonferenzNr`*
Listet die Teilnehmer einer Konferenz auf.

`MeetMe kick` *`KonferenzNr TeilnehmerNr`*
Wirft einen Teilnehmer aus der Konferenz.

`MeetMe kick` *`KonferenzNr`*
Wirft alle Teilnehmer aus der Konferenz.

`MeetMe lock` *`KonferenzNr`*
Sperrt eine Konferenz, so dass keine Teilnehmer mehr aufgenommen werden.

`MeetMe unlock` *`KonferenzNr`*
Hebt eine Konferenz-Sperre (s.o.) wieder auf.

`MeetMe mute` *`KonferenzNr TeilnehmerNr`*
Schaltet einen Teilnehmer stumm.

`MeetMe unmute` *`KonferenzNr TeilnehmerNr`*
Hebt die Stummschaltung eines Teilnehmers (s.o.) wieder auf.

6.50. `MeetMeAdmin()`

Administration von MeetMe-Konferenzen

```
MeetMeAdmin(KonferenzNr,Befehl[,TeilnehmerNr])
```

Führt den angegebenen MeetMe-Administrationsbefehl in der angegebenen Konferenz aus. Der Befehl kann einer der folgenden sein (Die `TeilnehmerNr` wird nur für den Befehl `k` benötigt.):

`K` Wirft alle Teilnehmer aus der Konferenz

`k` Wirft den Teilnehmer mit der angegebenen `TeilnehmerNr` aus der Konferenz

`e` Wirft den letzten Teilnehmer, der sich zu der Konferenz verbunden hat, raus

`L` Sperrt die Konferenz (d.h. keine neuen Teilnehmer)

`l` Entsperrt die Konferenz

`M` Schaltet die Konferenz stumm

`m` Hebt die Stummschaltung (durch `M`) auf

`N` Schaltet alle Teilnehmer außer dem Administrator stumm

`n` Hebt die Stummschaltung (durch `N`) auf

```
; Konferenz 333 stummschalten:
exten => 123,1,MeetMeAdmin(333,M)

; Teilnehmer 3 aus der Konferenz 333 kicken:
exten => 123,1,MeetMeAdmin(333,k,3)
```

Siehe auch. Abschnitt 6.49, „`MeetMe()`“, Abschnitt 6.51, „`MeetMeCount()`“

6.51. `MeetMeCount()`

Zählt die Teilnehmer einer MeetMe-Konferenz

```
MeetMeCount(KonferenzNr[,Variablenname])
```

Spielt die Anzahl der Teilnehmer in der MeetMe-Konferenz ab. Falls eine Variable angegeben ist, wird der Abspielvorgang übersprungen und die Anzahl in diese Variable geschrieben.

Liefert bei Erfolg 0 zurück, bei einem Fehler -1

Zähl die Teilnehmer in Konferenz 501, und weist diese Zahl `${COUNT}` zu

```
; die Teilnehmerzahl der Konferenz 333 in ${anzahl} speichern:
exten => 333,1,MeetMeCount(123,anzahl)
```

Siehe auch. Abschnitt 6.49, „`MeetMe()`“, Abschnitt 6.50, „`MeetMeAdmin()`“

6.52. `Milliwatt()`

Erzeugt einen 1000-Hz-Ton

```
Milliwatt()
```

Erzeugt einen konstanten 1000-Hz-Ton bei 0 dbm (u-law). Diese Anwendung wird häufig verwendet, um die Audioeigenschaften eines bestimmten Kanals zu testen. Allerdings ist die Bezeichnung etwas irreführend, sinnvoller schiene „Kilohertz()“.

```
; einen 1000-Hz-Ton erzeugen:
exten => 123,1,Milliwatt()
```

Siehe auch. Abschnitt 6.28, „`Echo()`“

6.53. `MixMonitor()`

```
MixMonitor(Basisname.Format[|Optionen[|Befehl]])
```

...

Beachten Sie auch die Hinweise bei `Monitor()`.

Siehe auch. Abschnitt 6.54, „`Monitor()`“

6.54. `Monitor()`

Schneidet das Gespräch auf dem aktuellen Kanal mit

```
Monitor([Format[,Basisname[,Optionen]]])
```

Startet die Audio-Aufzeichnung des aktuellen Kanals. Die auf dem Kanal eingehenden und ausgehenden Sprachpakete werden in Dateien aufgezeichnet, bis der Kanal aufgelegt oder die Überwachung durch die `StopMonitor()`-Anwendung beendet wird.

`Format` gibt das Dateiformat (= Dateiendung) an. Ohne Angabe wird `wav` verwendet.

`Basisname` gibt den Basis-Dateinamen an (also ohne Dateiendung). Ohne Angabe wird der Name aus dem Kanalnamen und einer Nummer zusammengesetzt, z.B. `IAX2[foo@bar]-3`. Das eingehende Audio wird in *`Basisname`*`-in.`*`Format`*, das ausgehende in *`Basisname`*`-out.`*`Format`* im Verzeichnis `/var/spool/asterisk/monitor/` gespeichert.

Eine der beiden Optionen kann angegeben werden:

`m` Nach dem Beenden der Aufnahme werden die Dateien für In und Out in eine gemischt und die ursprünglichen Dateien gelöscht. Dazu muss das Programm **soxmix** aus dem Paket **sox** installiert sein[12]. Falls die Variable `${MONITOR_EXEC}` definiert ist, wird statt **soxmix** die angegebene Anwendung ausgeführt, und die ursprünglichen Dateien für beide Richtungen nicht automatisch gelöscht[13]. **soxmix** (bzw. **${MONITOR_EXEC}**) werden drei Parameter übergeben, die beiden Aufzeichnungsdateien und der Dateiname für die zu erstellende gemischte Datei, welcher dem Basisnamen ohne `-in`/`-out` entspricht. Ist `${MONITOR_EXEC_ARGS}` gesetzt, werd der Inhalt als zusätzliche Argumente an `${MONITOR_EXEC}` übergeben.

> ### Wichtig
>
> Bei der Verwendung von **soxmix** ist zu beachten, dass **soxmix** ohne explizite Angabe der Dateitypen diese aus den Endungen erkennt. `gsm` und `wav` bereiten z.B. keine Probleme, aber für die Formate `alaw` und `ulaw` werden als Endungen `al` bzw. `ul` erwartet. Lesen Sie also ggf. die Anleitung von sox (/soxmix) und verwenden Sie `${MONITOR_EXEC_ARGS}` oder benutzen Sie ein kleines Wrapper-Skript als `${MONITOR_EXEC}`, das die Datei-Parameter liest und soxmix mit Angabe der Typen aufruft.

> ### Anmerkung
>
> Wenn Sie eine kombinierte Aufzeichnung wollen, ist meist `MixMonitor()` die bessere Alternative, da es die Kanäle direkt während der Aufnahme mischt und dadurch Lastspitzen am Ende der Aufzeichnung vermeidet.

`b` Startet die Aufnahme erst, nachdem ein Anruf zu einem anderen Kanal verbunden wurde, also nachdem z.B. durch `Dial()` tatsächlich ein Gespräch zustande kommt.

Gibt 0 bei Erfolg zurück oder -1 bei einem Fehler (Überwachungsdateien konnten nicht geöffnet werden, Kanal wir bereits aufgezeichnet, ...)

```
; das Gespräch aufzeichnen und hinterher die Audio-Kanäle mixen:
exten => 123,1,Answer()
exten => 123,n,Monitor(gsm,,mb)
exten => 123,n,SayDigits(123456789)
exten => 123,n,Hangup()

; wie oben aber mit einem eigenen Wrapper, der soxmix aufruft:
exten => 123,1,Answer()
exten => 123,n,Set(MONITOR_EXEC=/pfad/zu/mein-soxmix-wrapper.sh)
```

[12] http://sox.sourceforge.net/, siehe Beschreibung in Abschnitt 2, „`musiconhold.conf`“, mindestens Version 12.17.7, Ihre installierte Version erfahren Sie mit **soxmix -help**

[13] abhängig von der Asterisk-Version; ältere Versionen löschen nicht automatisch

```
exten => 123,n,Monitor(gsm,,mb)
exten => 123,n,SayDigits(123456789)
exten => 123,n,Hangup()
```

Wichtig

Stellen Sie vor dem Mitschneiden von Gesprächen sicher, dass die rechtlichen Voraussetzungen erfüllt sind. Meist müssen beide Teilnehmer von der Aufzeichnung informiert sein.[14]

Anmerkung

Einige Anwender, die viele (50-500) Gespräche gleichzeitig aufnehmen, berichten von *stark* verbesserter Performance, wenn man auf eine RAM-Disk aufzeichnet (weniger Such-Operationen) und die Dateien erst nach Gesprächsende auf die Festplatte (lokal oder gemountet) kopiert.

Siehe auch. Abschnitt 6.14, „`ChangeMonitor()`“, Abschnitt 6.96, „`StopMonitor()`“, Abschnitt 6.53, „`MixMonitor()`“, Abschnitt 6.74, „`Record()`“

6.55. `MP3Player()`

Gibt eine MP3 Datei oder einen MP3-Stream wieder

```
MP3Player(location)
```

Verwendet das Programm `mpg123`[15], um dem Anrufer `location` vorzuspielen. Die angegebene `location` kann entweder ein Dateiname oder eine URL sein. Der Anrufer kann die Wiedergabe durch Betätigung einer beliebigen Taste stoppen.

Die richtige Version von `mpg123` muss installiert sein, damit diese Anwendung ordnungsgemäß funktioniert. Asterisk arbeitet momentan mit der Version 0.59r ideal.

Anmerkung

Das häufig als Alternative installierte und auf `mpg123` verlinkte Programm **mpg321** funktioniert *nicht* einwandfrei.

Liefert -1, falls aufgelegt wird, sonst 0.

```
exten => 123,1,Answer()
exten => 123,2,MP3Player(test.mp3)

exten => 123,1,Answer()
exten => 123,2,MP3Player(http://server.tld/test.mp3)
```

6.56. `MusicOnHold()`

Spielt endlos Wartemusik

```
MusicOnHold(class)
```

Spielt die durch class spezifizierte Wartemusik, wie in `musiconhold.conf` konfiguriert. Wird `class` nicht angegeben, wird die Standardmusik-Klasse für dieses Kanal verwendet. Um die Standardmusik-Klasse für einen Kanal einzustellen, können Sie die Funktion `MUSICCLASS()` verwenden.

Liefert -1, falls aufgelegt wird, ansonsten wird sie endlos ausgeführt.

```
; transfer telemarketers to this extension to keep them busy
exten => 123,1,Answer()
```

[14] siehe auch http://www.voip-info.org/wiki/view/Monitor+Recording+Legal+Issues
[15] http://mpg123.org/, http://sourceforge.net/projects/mpg123/, für MacOS X siehe auch http://sourceforge.net/projects/mosx-mpg123/

```
exten => 123,2,Playback(tt-allbusy)
exten => 123,3,MusicOnHold(default)
```

Siehe auch. Abschnitt 6.116, „`WaitMusicOnHold()`“, Abschnitt 7.34, „`MUSICCLASS()`“

6.57. `NBScat()`

Wiedergabe eines lokalen NBS-Streams

```
NBScat()
```

Verwendet das `nbscat8k`-Programm um den lokalen NBS (Network Broadcast Sound)-Stream zu hören. (Schauen Sie sich das `nbs`-Modul in Digiums CVS-Server an, um weitere Informationen zu erhalten.) Der Anrufer kann durch Drücken einer beliebigen Taste aussteigen.

Die Anwendung liefert -1, falls aufgelegt wird, ansonsten wird sie endlos ausgeführt.

```
exten => 123,1,Answer()
exten => 123,2,NBScat()
```

6.58. `NoCDR()`

Unterdrückt Anrufprotokolle (Call Detail Records) für den aktuellen Anruf

```
NoCDR()
```

Für den aktuellen Anruf ist die Erstellung von CDRs nicht möglich.

```
; don't log calls to 555-1212
exten => 5551212,1,Answer()
exten => 5551212,2,NoCDR()
exten => 5551212,3,Dial(Zap/4/5551212)
```

Siehe auch. ???, Abschnitt 6.35, „`ForkCDR()`“, ???

6.59. `NoOp()`

Tut nichts (no operation)

```
NoOp(text)
```

Diese Anwendung tut nichts - sie ist lediglich als Platzhalter gedacht. Als Nebeneffekt evaluiert diese Anwendung Text und gibt ihn als Ergebnis auf dem Asterisk-Kommandozeileninterface aus - eine zum Debugging nützliche Eigenschaft.

Sie müssen den Text nicht in Anführungszeichen betten. Stehen Anführungszeichen innerhalb der Klammern, werden diese auf der Konsole ausgegeben.

```
exten => 123,1,NoOp(CallerID is ${CALLERID})
```

6.60. `Park()`

Parkt den aktuellen Anruf

```
Park(exten)
```

Parkt den aktuellen Anruf (typischerweise in Kombination mit einer beaufsichtigten Übertragung, um die Parkplatznummer zu bestimmen). Diese Anwendung ist intern stets registriert und muss nicht explizit im Wählplan eingefügt werden, jedoch sollten sie den `parkedcalls`-Kontext einbinden. Die Parkkonfiguration wird in `features.conf` vorgenommen.

```
; park the caller in parking space 701
include => parkedcalls
```

```
exten => 123,1,Answer()
exten => 123,2,Park(701)
```

Siehe auch. Abschnitt 6.61, „ParkAndAnnounce()“, Abschnitt 6.62, „ParkedCall()“

6.61. ParkAndAnnounce()

Parkt den aktuellen Anruf und macht ihn über den spezifizierten Kanal bekannt

```
ParkAndAnnounce(template,timeout,channel[,return_context])
```

Parkt den aktuellen Anruf in dem dafür vorgesehenen Bereich (auf dem Parkplatz) und meldet ihn über den spezifizierten Kanal. template bezeichnet eine Liste von durch Kommas voneinander getrennten Dateien, die anzukündigen sind; das Wort PARKED wird durch die Parkplatznummer des Anrufs ersetzt. Der Parameter timeout bestimmt die Zeit, in Sekunden, nach deren Ablauf der Anruf zum return_context zurückkehrt. Der Parameter channel bezeichnet den Kanal, der anzurufen ist und die Ankündigung zu machen. Console/dsp ruft die Konsole auf. Der return_context-Parameter ist ein Label im GoTo()-Stil, um den Anruf nach der Beendigung zurückzubringen. Standardeinstellung hierfür ist n+1 (mit n: aktuelle Priorität) im Kontext return_context.

```
include => parkedcalls
exten => 123,1,Answer()
exten => 123,2,ParkAndAnnounce(vm-youhave:a:pbx-transfer:at:vm-extension:
PARKED,120,Console/dsp)
exten => 123,3,Playback(vm-nobodyavail)
exten => 123,4,Playback(vm-goodbye)
exten => 123,5,Hangup()
```

Siehe auch. Abschnitt 6.60, „Park()“, Abschnitt 6.62, „ParkedCall()“

6.62. ParkedCall()

Beantwortet einen geparkten Anruf

```
ParkedCall(exten)
```

Verbindet den Anrufer mit dem, auf dem durch exten identifizierten Parkplatz, geparkten Anruf. Diese Anwendung ist intern ständig registriert und muss daher dem Wählplan nicht explizit hinzugefügt werden. Sie sollten jedoch den parkedcalls-Kontext einbeziehen.

```
; pick up the call parked in parking space 701
exten => 123,1,Answer()
exten => 123,2,ParkedCall(701)
```

Siehe auch. Abschnitt 6.60, „Park()“, Abschnitt 6.61, „ParkAndAnnounce()“

6.63. PauseQueueMember()

Blockiert ein Warteschlangenmitglied temporär, so dass es keine Anrufe empfangen kann

```
PauseQueueMember([queuename],interface)
```

Blockiert Anrufe für ein Warteschlangenmitglied. Das spezifizierte Interface wird in der gegebenen Warteschlange zeitweilig angehalten. Dadurch wird verhindert, dass irgendwelche Anrufe aus der Warteschlange zu diesem Interface gesendet werden, bis es nicht durch wieder freigegeben wurde/bis die Blockade nicht wieder aufgehoben wurde durch die UnpauseQueueMember()-Anwendung oder das Manager-Interface. Wenn kein Warteschlangenname queuename angegeben ist, wird das Interface in jeder Warteschlange, in der es ein Mitglied ist, angehalten. Befindet sich das Interface nicht in der benannten Warteschlange, oder wenn keine Warteschlange angegeben und das Interface nicht Mitglied irgendeiner Warteschlange ist, wird es zu Priorität n+101 übergehen (mit n: aktuelle Priorität), falls diese existiert.

Gibt -1 zurück, falls das Interface nicht gefunden wurde und keine Extension existiert, zu der gesprungen werden kann, sonst 0.

```
exten => 123,1,PauseQueueMember(,SIP/300)
exten => 124,1,UnpauseQueueMember(,SIP/300)
```

Siehe auch. Abschnitt 6.106, „`UnpauseQueueMember()`“

6.64. `Playback()`

Spielt dem Anrufer die spezifizierte Audiodatei vor

```
Playback(filename[,options])
```

Spielt dem Anrufer die durch `filename` spezifizierte Audiodatei (aus dem Verzeichnis `/var/lib/asterisk/sounds/`) vor. Der Dateiname enthält keine Dateiendung, damit Asterisk automatisch die Audiodatei mit dem niedrigsten Konvertierungsaufwand auswählen kann. Keine, eine oder mehr Optionen können hinzugefügt werden. Die Option `skip` bewirkt, dass die Wiedergabe der Nachricht übersprungen wird, wenn sich der Kanal nicht im Up-Status befindet (z.B., falls er noch nicht beantwortet wurde). Ist `skip` angegeben, wird die Anwendung augenblicklich beendet, wenn mit dem Kanal nicht alles in Ordnung ist. Ansonsten wird der Kanal beantwortet, bevor die Audiodatei wiedergegeben wird, es sei denn, `noanswer` ist angegeben. (Nicht alle Kanäle unterstützen die Wiedergabe von Nachrichten, wenn sie noch aufgelegt sind.) Liefert -1 zurück, wenn der Kanal aufgelegt wurde. Wenn die Datei nicht existiert, wird zur Priorität n+101 übergegangen (mit `n`: aktuelle Priorität), falls sie existiert.

```
exten => 123,1,Answer()
exten => 123,2,Playback(tt-weasels)
```

Siehe auch. Abschnitt 6.10, „`Background()`“

6.65. `Playtones()`

Wiedergabe einer Tonliste

```
Playtones(tonelist)
```

Spielt eine Liste von Tönen ab. Die Ausführung wird augenblicklich mit dem nächsten Schritt fortgesetzt, während die Töne weiter abgespielt werden. `tonelist` repräsentiert entweder einen Tonnamen, der in der `indications.conf` Konfigurationsdatei definiert ist, oder eine gegebene Liste von Tonfrequenzen und -dauern. Um eine Beschreibung der Spezifikation einer Tonliste einzusehen, siehe `indications.conf`.

Verwendet die `StopPlaytones()`-Anwendung, um die Wiedergabe der Töne anzuhalten.

```
; play a busy signal for two seconds, and then a congestion tone for two seconds
exten => 123,1,Playtones(busy)
exten => 123,2,Wait(2)
exten => 123,3,StopPlaytones()
exten => 123,4,Playtones(congestion)
exten => 123,5,Wait(2)
exten => 123,6,StopPlaytones()
exten => 123,7,Goto(1)
```

Siehe auch. Abschnitt 6.97, „`StopPlaytones()`“, `indications.conf`, Abschnitt 6.12, „`Busy()`“, Abschnitt 6.17, „`Congestion()`“, Abschnitt 6.68, „`Progress()`“, Abschnitt 6.80, „`Ringing()`“

6.66. `Prefix()`

Stellt die angegebenen Ziffern der aktuellen Extension voran und geht zu der resultierenden Extension über.

```
Prefix(digits)
```

Stellt der aktuellen Extension die durch die Ziffernkette `digits` angegebenen Ziffern voran und fährt mit der Ausführung bei der nächsten Priorität für die neue Extension fort. Zum Beispiel, falls die Priorität 1 der Extension 1212 Prefix(555) ist, wird 555 der 1212 vorangestellt und der nächste Schritt, der ausgeführt wird, wird Priorität 2 von Extension 5551212 sein. Wenn sie zu einer Extension wechseln, die keine Priorität `n+1` besitzt (`n` bezeichnet die aktuelle Priorität), wird Asterisk verfahren, als hätte der Benutzer eine ungültige Extension angewählt.

Liefert immer 0 zurück.

```
exten => 1212,1,Prefix(555)
exten => 5551212,2,SayDigits(${EXTEN})
```

Siehe auch. Abschnitt 6.101, „`Suffix()`“

6.67. `PrivacyManager()`

Verlangt die Eingabe der Telefonnummer eines Teilnehmers, falls keine Caller ID-Informationen empfangen werden

```
PrivacyManager()
```

Wird keine Caller ID empfangen, wird der Kanal beantwortet und der Anrufer oder die Anruferin aufgefordert, seine bzw. ihre Telefonnummer einzugeben. Standardmäßig werden dem Anrufer dabei drei Versuche gewährt. Wenn er nach diesen drei Versuchen nicht mindestens eine zehnstellige Telefonnummer eingegeben hat und es eine Priorität n+101 gibt (wobei `n` die aktuelle Priorität bezeichnet), wird der Kanal so eingestellt, dass er mit diesem Prioritätslevel fortfährt. Andernfalls wird 0 zurückgeliefert. Wird auf dem Kanal eine Caller ID empfangen, tut PrivacyManager() nichts.

Die Konfigurationsdatei `privacy.conf` bestimmt die Funktionsweise der `PrivacyManager()`-Anwendung. Sie enthält die beiden folgenden Zeilen:

`maxretries` Gibt die Anzahl der Versuche an, die einem Anrufer zur Eingabe einer Caller ID gewährt werden (Standard: 3)

`minlength` Gibt die minimal zu erlaubende/minimale erlaubte Anzahl von Ziffern in der eigegebenen Caller ID-Nummer an (Standard: 10)

```
exten => 123,1,Answer()
exten => 123,2,PrivacyManager()
exten => 123,3,Dial(Zap/1)
exten => 123,103,Playback(im-sorry)
exten => 123,104,Playback(vm-goodbye)
```

Siehe auch. Abschnitt 6.118, „`Zapateller()`“

6.68. `Progress()`

Signalisiert Fortschritt

```
Progress()
```

Erbittet, dass dem Teilnehmer angezeigt wird, dass es bandinternen Fortschritt gibt.

Liefert stets 0 zurück.

```
; indicate progress to the calling channel
exten => 123,1,Progress()
```

Siehe auch. Abschnitt 6.12, „`Busy()`“, Abschnitt 6.17, „`Congestion()`“, Abschnitt 6.65, „`Playtones()`“, Abschnitt 6.80, „`Ringing()`“

6.69. `Queue()`

Legt den aktuellen Anruf in der spezifizierten Anrufwarteschlange ab

```
Queue(queuename,options,optionalurl,announceoverride,timeout)
```

Fügt der durch queuename spezifizierten Anrufwarteschlange einen eingehenden Anruf hinzu, wie in `queues.conf` definiert.

Der `options`-Parameter kann sich aus keinem, einem oder mehreren der folgenden Zeichen zusammensetzen:

`t` Erlaubt es dem angerufenen Teilnehmer, den Anruf weiterzuleiten

`T` Erlaubt es dem anrufenden Teilnehmer, den Anruf weiterzuleiten

`d` Spezifiziert einen Anruf mit Datenqualität (Modem, minimale Verzögerung)

`h` Erlaubt es dem Angerufenen, durch Drücken von `*` aufzulegen

`H` Erlaubt es dem Anrufer, durch Drücken von `*` aufzulegen

`n` Verbietet eine Wiederholung des Timeouts; beendet diese Anwendung und geht weiter zum nächsten Schritt

`r` Klingelt, statt Wartemusik zu spielen

Außer dass ein Anruf weitergeleitet werden kann, kann er auch geparkt und dann von einem anderen Teilnehmer aufgenommen werden.

Der Parameter `announceoverride` überschreibt die Standardankündigung, die Warteschlangenagenten vorgespielt wird, bevor sie den angegebenen Anruf beantworten.

Der optinale `URL` wird dem angerufenen Teilnehmer übersendet, falls dies vom Kanal unterstützt wird.

Nach einer angegebenen Zahl von Sekunden, überprüft zwischen jedem `queues.conf` Timeout und Retry-Kreislauf, wird `timeout` die Warteschlange zu einem Ausfall zwingen.

Liefert -1 zurück, falls der ursprüngliche Kanal aufgelegt wird, oder falls der Anruf überbückt wird und einer der Teilnehmer in der Brücke den Anruf beendet. Ist die Warteschlange voll, existiert sie nicht oder hat sie keine Mitglieder, wird 0 zurückgeliefert.

```
; place the caller in the techsupport queue
exten => 123,1,Answer()
exten => 123,2,Queue(techsupport,t)
```

6.70. `Random()`

Verzweigt wahrscheinlichkeitsbasiert bedingt

```
Random([probability]:[[context,]extension,]priority)
```

Springt bedingt zu der spezifizierten Priorität (und optional Extension und Kontext), basierend auf der angegebenen Wahrscheinlichkeit (`probability`). `probability` wird als ganze Zahl zwischen 1 und 100 angegeben. Die Anwendung springt in `priority` Prozent der Zeit zu dem angegebenen Ziel.

```
; test your luck over and over again
exten => 123,1,Random(20:lucky,1)
exten => 123,2,Goto(unlucky,1)

exten => lucky,1,Playback(good)
exten => lucky,2,Goto(123,1)

exten => unlucky,1,Playback(bad)
exten => unlucky,2,Goto(123,1)
```

6.71. `Read()`

Liest DTMF-Ziffern eines Anrufers und verknüpft das Ergebnis mit einer Variablen

```
Read(variable[,filename][,maxdigits][,option][,attempts][,timeout])
```

Liest eine mit # abgeschlossene Reihe von Ziffern so oft wie definiert von einem Teilnehmer, und legt sie in einer gegebenen Variablen ab.

Andere Parameter sind:

`filename` Gibt die Datei an, die vor dem Einlesen der Ziffern abgespielt werden soll

`maxdigits` Setzt die maximal zu akzeptierende Anzahl Ziffern. Ist der Parameter angegeben, beendet die Anwendung das Einlesen, nachdem maxdigits Ziffern eingegeben wurden (ohne dass der Teilnehmer mit der #-Taste bestätigen muss). Standardeinstellung ist 0, was bedeutet, es existiert keine Begrenzung und es wird auf die Eingabe von # gewartet. Das gilt auch für alle anderen Werte unter 0. Der größte akzeptierte Wert ist 255.

`option` Geben sie `skip` an, um umgehend zurückzukehren, wenn der Anschluss nicht beantwortet wird, oder `noanswer`, um Zahlen sogar dann einzulesen, wenn der Anschluss nicht beantwortet wird.

`attempts` Falls größer als 1, werden genau so viele Versuche gemacht in dem Fall, dass keine Daten eingegeben werden.

`timeout` Falls größer als 0, wird dieser Wert die Standardeinstellungen für die Zeitgrenze überschreiben.

Liefert -1 zurück im Falle eines Auflegens oder Fehlers, sonst 0.

```
; read a two-digit number and repeat it back to the caller
exten => 123,1,Read(NUMBER,,2)
exten => 123,2,SayNumber(${NUMBER})
exten => 123,3,Goto(1)
```

Siehe auch. Abschnitt 6.86, „`SendDTMF()`“

6.72. `RealTime()`

Besorgt Informationen vom Echtzeitkonfigurationssteuerprogramm

```
RealTime(family,colmatch,value[,prefix])
```

Benutzt das RealTime Konfigurationssteuerprogramm, um Daten in Kanalvariablen einzulesen. Alle eindeutigen Spaltennamen (aus der spezifizierten Familie `family`) werden als Kanalvariablen gesetzt. Dabei kann ihr Name ein optionales Namenspräfix `prefix` erhalten (zum Beispiel würde ein Präfix `var_` den Spaltennamen `name` in die Variable `${var_name}` überführen).

```
; retrieve all columns from the sipfriends table where the name column
; matches "John", and prefix all the variables with "John_"
exten => 123,1,RealTime(sipfriends,name,John,John_)
; now, let's read the value of the column named "port"
exten => 123,2,SayNumber(${John_port})
```

Siehe auch. Abschnitt 6.73, „`RealTimeUpdate()`“

6.73. `RealTimeUpdate()`

Aktualisiert einen Wert mit Hilfe des Echtzeitkonfigurationssteuerprogramms

```
RealTimeUpdate(family,colmatch,value,newcol,newval)
```

Benutzt das RealTime Konfigurationssteuerprogrammsystem, um einen Wert zu aktualisieren. Die Spalte `newcol` in der Familie `family`, die mit der Spalte `colmatch=value` übereinstimmt, wird auf den Wert `newval` aktualisiert.

```
; this will update the port column in the sipfriends table to a new
; value of 5061, where the name column matches "John"
exten => 123,1,RealTimeUpdate(sipfriends,name,John,port,5061)
```

Siehe auch. Abschnitt 6.72, „`RealTime()`“

6.74. `Record()`

Aufnahme von Audio auf einem Kanal in eine Datei

ab Asterisk 1.2:

```
Record(filename.format,silence[,maxduration][,options])
```

Nimmt Audio auf dem Kanal auf und speichert es in der Datei `filename`. Existiert diese Datei bereits, wird sie überschrieben.

Optionale Parameter sind:

`format` Spezifizert das Dateiformat der Aufnahmedatei. Gültige Formate sind `g723`, `g729`, `gsm`, `h263`, `ulaw`, `alaw`, `vox`, `wav` und `WAV`.

`silence` Gibt die Dauer in Sekunden an, die geräuschlos verstreichen darf, bevor die Aufnahme beendet wird.

`maxduration` Gibt die maximale Aufnahmedauer in Sekunden an. Falls nicht angegeben oder 0, gibt es keine Obergrenze.

`options` Kann beliebige der aufgeführten Buchstaben enthalten:

- `s` Nimmt nicht auf, wenn die Leitung bisher nicht beantwortet wurde.
- `n` Antwortet nicht, aber nimmt nichtsdestotrotz auf, wenn die Leitung bisher nicht beantwortet wurde.
- `a` Hängt die Aufnahme an die bereits bestehende Aufnahme an, statt sie zu überschreiben.
- `t` Benutzt die alternative Abschlusstaste `*` an Stelle der standardmäßig eingestellten #-Taste.

Enthält der Dateiname die Zeichenkette `%d`, wird diese durch eine Nummerierung ersetzt, die bei jedem erneuten Aufnehmen der Datei um eins erhöht wird.

Der Teilnehmer kann die Taste # benutzten, um die Aufnahme abzubrechen und zur nächsten Priorität überzugehen.

Liefert -1 zurück, wenn der Benutzer auflegt.

```
; record the caller's name
exten => 123,1,Playback(pls-rcrd-name-at-tone)
exten => 123,2,Record(/tmp/name:gsm,3,30)
exten => 123,3,Playback(/tmp/name)
```

Beachten Sie auch die Hinweise bei `Monitor()`.

Siehe auch. Abschnitt 6.23, „`Dictate()`“, Abschnitt 6.54, „`Monitor()`“, Abschnitt 6.53, „`MixMonitor()`“

6.75. `RemoveQueueMember()`

Entfernt dynamisch Warteschlangenmitglieder

```
RemoveQueueMember(queuename[,interface])
```

Enfernt das spezifizierte Interface interface dynamisch aus der Anrufwarteschlange `queuename`. Ist `interface` nicht angegeben, entfernt die Anwendung das momentan in Verwendung befindliche Interface aus der Warteschlange.

Ist das Interface nicht in der Warteschlange enthalten und eine Priorität n+101 existiert (`n` bezeichnet dabei die aktuelle Priorität), wird die Anwendung zu dieser Priorität übergehen. Andernfalls wird sie einen Fehler zurückliefern.

Liefert -1, wenn ein Fehler auftritt.

```
; remove SIP/3000 from the techsupport queue
exten => 123,1,RemoveQueueMember(techsupport,SIP/3000)
```

Siehe auch. Abschnitt 6.1, „`AddQueueMember()`“, Abschnitt 6.69, „`Queue()`“, `queues.conf`

6.76. `ResetCDR()`

Setzt das Anrufprotokoll (Call Detail Record) zurück

```
ResetCDR([options])
```

Setzt das Anrufprotokoll (Call Detail Record) zurück. Ist der `w`-Parameter angegeben, wird eine Kopie des akutellen Anrufprotokolls gespeichert, bevor das aktuelle CDR gelöscht wird.

Liefert stets 0 zurück.

```
; write a copy of the current CDR record, and then reset the CDR
exten => 123,1,Answer()
exten => 123,2,Playback(tt-monkeys)
exten => 123,3,ResetCDR(w)
exten => 123,4,Playback(tt-monkeys)
```

Siehe auch. Abschnitt 6.35, „`ForkCDR()`“, Abschnitt 6.58, „`NoCDR()`“

6.77. `ResponseTimeout()`

Legt die maximale Zeitgrenze fest, die auf die Antwort eines Anrufers gewartet wird

```
ResponseTimeout()
```

Setzt die maximal erlaubte Zeitdauer, nachdem eine Reihe von Prioritäten für einen Kanal nicht zu Stande gekommen sind, während dessen ein Anrufer damit begonnen haben könnte, einen Extension einzugeben. Gibt der Anrufer in dieser Zeitdauer keine Extension ein, fällt die Kontrolle an die `t`-Extension, falls sie existiert; falls nicht, wird der Anruf beendet.

Liefert stets 0 zurück.

```
; allow callers three seconds to make a choice, before sending them
; to the 't' extension
exten => s,1,Answer()
exten => s,2,ResponseTimeout(3)
exten => s,3,Background(enter-ext-of-person)

exten => t,1,Playback(im-sorry)
exten => t,1,Playback(goodbye)
```

Siehe auch. ???, ???

6.78. `RetryDial()`

Versucht einen Anruf zu tätigen und wiederholt den Versucht bei einem Fehlschlagen

```
RetryDial(announce,sleep,loops,technology/resource[&technology2/resource2...][,timeout][,options][,URL])
```

Versucht, einen Anruf zu tätigen. Kann kein Kanal erreicht werden, wird die durch announce definierte Datei abgespielt und `sleep` Sekunden gewartet, bis der Versuch wiederholt wird. Wenn die angegebene Zahl Versuche `loops` entspricht, wird der Anruft mit der nächsten Priorität auf dem Wählplan fortgesetzt. Ist `loop` auf 0 gesetzt, wird der Anrufversuch endlos wiederholt.

In der Wartezeit kann eine einziffrige Extension angewählt werden. Existiert diese Extensions im durch `${EXITCONTEXT}` spezifizierten Kontext (falls angegeben) oder im aktuellen, wird der Anruf unmittelbar an diese Extension weitergeleitet.

Alle Parameter nach `loops` werden direkt an die `Dial()`-Anwendung weitergeleitet.

```
; attempt to dial the number three times via IAX, retrying every five seconds
exten => 123,1,RetryDial(priv-trying,5,3,IAX2/VOIP/8885551212,30)
; if the caller presses 9 while waiting, dial the number on the Zap/4 channel
exten => 9,1,RetryDial(priv-trying,5,3,Zap/4/8885551212,30)
```

Siehe auch. Abschnitt 6.22, „`Dial()`“

6.79. `Return()`

Kehrt von einem Unterprogramm zurück.

```
Return()
```

Kehrt von einem Unterprogramm, das mit `Gosub()` oder `GosubIf()` aufgerufen wurde, zurück an die Anweisung, die im Dialplan auf den Sprungbefehl folgt.

```
exten => 123,1,Playback(tt-monkeys)
exten => 123,n,Gosub(mein-unterprogramm,s,1)
exten => 123,n,Playback(tt-monkeys)
exten => 123,n,Hangup()

[mein-unterprogramm]
exten => s,1,Playback(tt-weasels)
exten => s,n,Return()
```

Siehe auch. Abschnitt 6.37, „`Gosub()`“, Abschnitt 6.38, „`GosubIf()`“

6.80. `Ringing()`

Signalisiert Klingelton

```
Ringing()
```

Verlangt, dass dem Teilnehmer Klingelton vom Kanal signalisiert wird. Wie genau diese Signalisierung erfolgt, hängt vom Kanaltreiber ab.

Beachten Sie, dass diese Anwendung den Anrufer nicht mit tatsächlichem Klingeln versieht. Benutzen Sie hierzu die Anwendung `Playtones()`.

Liefert stets 0 zurück.

```
; indicate that the phone is ringing, even though it isn't
exten => 123,1,Ringing()
exten => 123,2,Wait(5)
exten => 123,3,Playback(tt-somethingwrong)
```

Siehe auch. Abschnitt 6.12, „`Busy()`“, Abschnitt 6.17, „`Congestion()`“, Abschnitt 6.68, „`Progress()`“, Abschnitt 6.80, „`Ringing()`“, Abschnitt 6.65, „`Playtones()`“

6.81. `SayAlpha()`

Spricht eine Zeichenkette

```
SayAlpha(string)
```

Buchstabiert die übergebene Zeichenkette `string` unter Verwendung der aktuellen Spracheinstellung für den Kanal. Die `SetLanguage()`-Anwendung wird verwendet, um die aktuell eingestellte Sprache zu ändern.

```
exten => 123,1,SayAlpha(ABC123XYZ)
```

Siehe auch. Abschnitt 6.82, „`SayDigits()`“, ???, Abschnitt 6.83, „`SayNumber()`“, Abschnitt 6.84, „`SayPhonetic()`“

6.82. `SayDigits()`

Spricht die übergebenen Ziffern

```
SayDigits(digits)
```

Spricht die übergebenen Ziffern unter Verwendung der aktuellen Spracheinstellung für den Kanal. Die `SetLanguage()`-Anwendung wird verwendet, um die aktuell eingestellte Sprache zu ändern.

```
exten => 123,1,SayDigits(1234)
```

Siehe auch. Abschnitt 6.81, „`SayAlpha()`“, ???, Abschnitt 6.83, „`SayNumber()`“, Abschnitt 6.84, „`SayPhonetic()`“

6.83. `SayNumber()`

Spricht die angegebene Zahl

```
SayNumber(digits[,gender])
```

Spricht die angegebene Zahl unter Verwendung der aktuellen Spracheinstellung für den Kanal. Die `SetLanguage()`-Anwendung wird verwendet, um die aktuell eingestellte Sprache zu ändern.

Momentan werden folgenden Sprachen syntaktisch unterstützt:

`da` dänisch

`de` deutsch

`en` englisch

`es` spanisch

`fr` französisch

`it` italienisch

`nl` niederländisch

`no` norwegisch

`pl` polnisch

`pt` portugiesisch

`se` schwedisch

`tw` taiwanesisch

Sind der der aktuellen Sprache verschiedene Genera bekannt, kann der `gender`-Parameter übergeben werden und das Genus der zu sprechenden zahl zu ändern. Folgende `gender`-Parameter können verwendet werden:

Benutzen Sie in europäischen Sprachen wie Portugiesisch, Französisch, Spanisch und Deutsch die `gender`-Parameter `f` für Femininum, `m` für Maskulinum und `n` für Neutrum.

Benutzen Sie in nordischen Sprachen wie Dänisch, Schwedisch oder Norwegisch die `gender`-Parameter `c` für Menschlich (commune) und `n` für Neutrum.

Benutzen Sie für Aufzählungen im deutschen Plural den `gender`-Parameter `p`.

Damit diese Anwendung mit anderen Sprachen als Englisch funktioniert, müssen Sie die entsprechenden Audiodateien für die jeweilige gewünschte Sprache besitzen.

```
; say the number in English
exten => 123,1,SetLanguage(en)
exten => 123,2,SayNumber(1234)
```

Siehe auch. Abschnitt 6.81, „SayAlpha()“, Abschnitt 6.82, „SayDigits()“, ???, Abschnitt 6.84, „SayPhonetic()“

6.84. SayPhonetic()

Buchstabiert die angegebene Zeichenkette phonetisch.

```
SayPhonetic(string)
```

Buchstabiert die angegebene Zeichenkette string unter Verwendung des phonetischen NATO-Alphabets.

```
exten => 123,1,SayPhonetic(asterisk)
```

Siehe auch. Abschnitt 6.81, „SayAlpha()“, Abschnitt 6.82, „SayDigits()“, Abschnitt 6.83, „SayNumber()“

6.85. SayUnixTime()

Zeitansage in einem spezifischen Format

```
SayUnixTime([unixtime][,[timezone][,format]])
```

Sagt die angegebene Zeit unter Berücksichtigung der angegebenen Zeitzone und des vorgegebenen Formats. Die Parameter sind:

`unixtime` Die Zeit, angegeben in Sekunden seit dem 1. Januar 1970. Kann negative Werte annehmen. Standardwert ist die aktuelle Zeit.

`timezone` Die Zeitzone. Eine Liste findet sich in `/usr/share/zoneinfo/`. Standardwert ist die Geräteeinstellung.

`format` Das Format, in der die Zeit gesprochen werden soll. Eine Liste von Formaten findet sich in `voicemail.conf`. Als Standardformat wird `ABdY 'digits/at' IMp` verwendet.

Liefert 0 zurück, oder -1 wenn aufgelegt wird.

```
exten => 123,1,SayUnixTime(,,IMp)
```

6.86. SendDTMF()

Sendet beliebige DTMF-Ziffern zu dem Kanal

```
SendDTMF(digits[,timeout_ms])
```

Sendet die angegebenen DTMF-Ziffern auf einem Kanal. Gültige DTMF-Symbole sind 0-9, *, # und A-D. Sie können auch den Buchstaben w als eine Ziffer verwendet, er steht für eine Wartezeit von 500 Millisekunden. Der Parameter `timeout_ms` bezeichnet die Zeitdauer in Millisekunden zwischen Ziffern. Falls nicht angegeben, wird ein Wert von 250 Millisekunden für `timeout_ms` angenommen.

Liefert bei Erfolg 0 zurück, oder -1 wenn aufgelegt wird.

```
exten => 123,1,SendDTMF(3212333w222w366w3212333322321,250
```

Siehe auch. Abschnitt 6.71, „Read()“

6.87. SendImage()

Sendet eine Bilddatei

```
SendImage(filename)
```

Sendet ein Bild auf einem Kanal, falls die Übermittlung von Bildern unterstützt wird. Unterstützt der Kanal keine Bildübertragung, und es existiert eine Priorität n+101 (wobei n die aktuelle Priorität bezeichnet), wird die Ausführung mit diesem Schritt fortgesetzt. Andernfalls wird die Ausführung auf dem nächsten Prioritätslevel fortgesetzt.

Liefert 0 zurück, wenn das Bild fehlerfrei übertragen wurde oder der Kanal eine Bildübertragung nicht unterstützt, sonst -1.

```
exten => 123,1,SendImage(logo.jpg)
```

Siehe auch. Abschnitt 6.88, „`SendText()`“, Abschnitt 6.89, „`SendURL()`“

6.88. `SendText()`

Sendet Text zu dem Kanal

```
SendText(text)
```

Überträgt den Text text auf einem Kanal, falls die Übertragung von Text unterstützt wird. Unterstützt der Kanal keine Textübertragung, und es existiert eine Priorität n+101 (wobei n die aktuelle Priorität bezeichnet), wird die Ausführung mit diesem Schritt fortgesetzt. Andernfalls wird die Ausführung auf dem nächsten Prioritätslevel fortgesetzt.

Liefert 0 zurück, wenn der Text fehlerfrei übertragen wurde oder der Kanal eine Textübertragung nicht unterstützt, sonst -1.

```
exten => 123,1,SendText(Welcome to Asterisk)
```

Siehe auch. Abschnitt 6.87, „`SendImage()`“, Abschnitt 6.89, „`SendURL()`“

6.89. `SendURL()`

Sendet dem Kanal eine URL

```
SendURL(URL[,option])
```

Der Client soll zu der spezifizierten `URL` gehen. Unterstützt der Client keine HTML-Übertragung, und es existiert eine Priorität n+101 (wobei n die aktuelle Priorität bezeichnet), wird die Ausführung mit diesem Schritt fortgesetzt. Andernfalls wird die Ausführung auf dem nächsten Prioritätslevel fortgesetzt.

Liefert 0 zurück, wenn der URL fehlerfrei übertragen wurde oder der Kanal eine HTML-Übertragung nicht unterstützt, sonst -1.

Ist die Option wait angegeben, wird die Ausführung erst fortgesetzt, wenn eine Bestätigung für den komplettierten Ladevorgang der URL eingegangen ist bzw. -1 zurückgeliefert, falls die Gegenstelle (peer ?) nicht in der Lage ist, die URL zu laden.

```
exten => 123,1,SendURL(www.asterisk.org,wait)
```

Siehe auch. Abschnitt 6.87, „`SendImage()`“, Abschnitt 6.88, „`SendText()`“

6.90. `Set()`

Setzt eine Variable auf den angegebenen Wert

```
Set(Variable=Wert[,Optionen])
```

Setzt die `Variable` auf den angegebenen `Wert`. Beginnt der Name der Variablen mit dem Zeichen _, wird einfache Vererbung angenommen, beginnt er mit __, wird Endlosvererbung angenommen. Variablen sind normalerweise nur innerhalb eines Kanals gültig und werden beim Auflegen gelöscht. Mit der Option g setzt man in Asterisk 1.2 eine Variable global; in 1.4 macht man das mit Hilfe der Funktion `GLOBAL()`.

```
; die Variable test auf "123" setzen:
exten => 123,1,Set(test=123)
```

```
exten => 123,n,SayDigits(${test})

; die globale Variable test2 auf "456" setzen:
exten => 123,n,Set(test2=456,g)         ; Asterisk 1.2
exten => 123,n,Set(GLOBAL(test2)=456)   ; Asterisk 1.4
```

Anmerkung

Ob globale Variablen auch nach einem Reload von Asterisk noch gültig sind, wird durch die Einstellung `clearglobalvars` in der `extensions.conf` bestimmt.

`Set()` wird auch verwendet, um in Funktionen zu schreiben (siehe Abschnitt 7, „Funktionen im Dialplan“).

```
exten => 123,1,Set(CALLERID(name)=Apfelmus)  ; CALLERID(name) setzen
exten => 123,n,Set(CALLERID(name)=)          ; CALLERID(name) leeren

exten => 123,n,Set(DB(mein/test)=ok)         ; in AstDB schreiben
exten => 123,n,Set(var=${DB(mein/test)})     ; aus AstDB lesen
```

Siehe auch. Abschnitt 6.44, „`ImportVar()`“, `doc/README.variables` (1.2) / `doc/channelvariables.txt` (1.4), Abschnitt 7.20, „`GLOBAL()`“

6.91. `SetAMAFlags()`

Setzt AMA-Flags im Anrufprotokoll

```
SetAMAFlags(Flags)
```

Setzt, zu Abrechnungszwecken, die AMA-Flags im Anrufprotokoll (Call Detail Record), wobei die AMA-Einstellungen in den Kanalkonfigurationsdateien überschrieben werden. Gültige Werte sind `default`, `omit`, `billing` und `documentation`.

Liefert stets 0 zurück.

```
exten => 123,1,SetAMAFlags(billing)
```

Siehe auch. ???, ???, ???

6.92. `SetCallerPres()`

Setzt Caller ID Darstellungsflags (presentation flags)

```
SetCallerPres(presentation)
```

Setzt die Caller ID Darstellungsflags auf einer Q931 PRI-Verbindung.

Gültige Darstellungsflags sind:

`allowed_not_screened`	Anzeige erlaubt und nicht überprüft
`allowed_passed_screen`	Anzeige erlaubt und Überprüfung erfolgreich
`allowed_passed_screen`	Anzeige erlaubt und Überprüfung nicht erfolgreich
`allowed`	Anzeige erlaubt, Netzwerknummer
`prohib_not_screened`	Anzeige verboten und nicht überprüft
`prohib_passed_screen`	Anzeige verboten und Überprüfung erfolgreich
`prohib_failed_screen`	Anzeige verboten und Überprüfung nicht erfolgreich
`prohib`	Anzeige verboten, Netzwerknummer

`unavailable`	Nummer nicht verfügbar

Liefert stets 0 zurück.

```
exten => 123,1,SetCallerPres(allowed_not_screened)
exten => 123,2,Dial(Zap/4/1234567)
```

Evtl. müssen Sie `usecallingpres=yes` in der `zapata.conf` setzen.

Siehe auch. ???

6.93. `SIPAddHeader()`

Fügt dem ausgehenden SIP-Anruf einen Header hinzu.

```
SIPAddHeader(Header: Wert)
```

Fügt einem mit der `Dial()`-Anwendung initiierten SIP-Anruf einen Header hinzu. Ein nicht standardisierter SIP-Header sollte mit einem `X-` beginnen, wie etwa `X-Asterisk-Accountcode:`. Verwenden Sie diese Anwendung mit Vorsicht, denn das Hinzufügen eines falschen Headers kann Probleme verursachen. Mit dieser Applikation können keine SIP-Header verändert werden.

Liefert stets 0 zurück.

```
exten => 123,1,SIPAddHeader(X-Asterisk-Account: ${CDR(accountcode)})
exten => 123,2,Dial(SIP/123)
```

Siehe auch. Abschnitt 7.47, „`SIP_HEADER()`“

6.94. `SIPdtmfMode()`

Ändert das DTMF-Verfahren für einen SIP-Anruf

```
SIPdtmfMode(Methode)
```

Ändert den DTMF-Modus für einen Anruf, der von einem SIP-Kanal ausgeht (nicht für einen Anruf auf einen SIP-Kanal). `Methode` kann die Werte `inband` (RTP), `info` oder `rfc2833` annehmen.

```
exten => 123,n,SIPdtmfMode(rfc2833)
```

Siehe auch. RFC 2833[16], RFC 2976[17]

6.95. `SoftHangup()`

Legt den angegebenen Kanal auf.

```
SoftHangup(Technologie/Ressource,[Optionen])
```

Legt den angegebenen Kanal auf. Liefert stets 0 zurück. Der `Optionen`-Parameter kann den Buchstaben a enthalten, was bewirkt, dass alle Kanäle auf dem angegebenen Gerät (statt nur der Resource) aufgelegt werden.

```
; alle Kanäle auflegen, die Zap/4 benutzen:
exten => 123,1,SoftHangup(Zap/4,a)
exten => 123,2,Wait(2)
exten => 123,3,Dial(Zap/4/1234567)
```

Siehe auch. Abschnitt 6.42, „`Hangup()`“

[16] http://www.ietf.org/rfc/rfc2833.txt
[17] http://www.ietf.org/rfc/rfc2976.txt

6.96. `StopMonitor()`

Beendet die Überwachung eines Kanals

```
StopMonitor()
```

Beendet die Überwachung (Aufnahme) eines Kanals. Diese Anwendung hat keinerlei Auswirkungen, wenn der Kanal momentan nicht überwacht wird.

```
exten => 123,1,Answer()
exten => 123,2,Monitor(wav,monitor_test,mb)
exten => 123,3,SayDigits(12345678901234567890)
exten => 123,4,StopMonitor()
```

Siehe auch. Abschnitt 6.54, „`Monitor()`“

6.97. `StopPlaytones()`

Beendet die Wiedergabe einer Liste von Tönen

```
StopPlaytones()
```

Beendet die Wiedergabe der momentan laufenden Tonliste.

```
exten => 123,1,Playtones(busy)
exten => 123,2,Wait(2)
exten => 123,3,StopPlaytones()
exten => 123,4,Playtones(congestion)
exten => 123,5,Wait(2)
exten => 123,6,StopPlaytones()
exten => 123,7,Goto(1)
```

Siehe auch. Abschnitt 6.65, „`Playtones()`“, `indications.conf`

6.98. `StripLSD()`

Entfernt die angegebene Zahl von Endziffern (von niederwertigsten Ziffern) von der aktuellen Extension

```
StripLSD(count)
```

Entfernt die `count` niederwertigsten Ziffern (die letzen `count` Ziffern) von der dem Kanal zugeordneten Extension und setzt die Ausführung bei der nächsten Priorität für die resultieren Extension fort. Ist zum Beispiel Priorität 1 der Extension 5551212 `StripLSD(4)`, werden die letzten vier Ziffern von 5551212 abgelöst und der nächste Schritt, Priorität 2 von Extension 555, ausgeführt. Wird zu einer Extions gewechselt, die keine Priorität `n+1` besitzt (mit `n`: aktuelle Priorität), wird das PBX sie behandeln, als hätte der Teilnehmer eine ungültige Extension angewählt.

Liefert stets 0 zurück.

Diese Anwendung ist veraltet und wurde durch den Teilkettenausdruck `${EXTEN:X:Y}` ersetzt.

```
exten => 5551212,1,StripLSD(4)
exten => 555,2,SayDigits(${EXTEN})

; a better way of doing the same thing
exten => 5551234,1,SayDigits(${EXTEN::3})
```

Siehe auch. Abschnitt 6.99, „`StripMSD()`“, `doc/README.variables` (1.2) / `doc/channelvariables.txt` (1.4)

6.99. `StripMSD()`

Entfernt die angegebene Zahl führender Ziffern (höchstwertiger Ziffern) von der aktuellen Extension

```
StripMSD(count)
```

Entfernt die ersten `count` Ziffern von der dem Kanal zugeordneten Extension und setzt die Ausführung bei der nächsten Priorität für die resultieren Extension fort. Ist zum Beispiel Priorität 1 der Extension 5551212 `StripMSD(3)`, werden die ersten drei Ziffern von 5551212 abgelöst und der nächste Schritt, Priorität 2 von Extension 1212, ausgeführt. Wird zu einer Extions gewechselt, die keine Priorität `n+1` besitzt (mit `n`: aktuelle Priorität), wird das PBX sie behandeln, als hätte der Teilnehmer eine ungültige Extension angewählt.

Liefert stets 0 zurück.

Diese Anwendung ist veraltet und wurde durch den Teilkettenausdruck `${EXTEN:X:Y}` ersetzt.

```
exten => 5551212,1,StripMSD(3)
exten => 1212,2,SayDigits(${EXTEN})
; a better way of doing the same thing
exten => 5551234,1,SayDigits(${EXTEN:3})
```

Siehe auch. Abschnitt 6.98, „`StripLSD()`“, `doc/README.variables` (1.2) / `doc/channelvariables.txt` (1.4)

6.100. `SubString()`

Speichert eine Ziffernteilkette in eine gegebene Variable

```
SubString(variable=string_of_digits,count1,count2)
```

Weist die Teilkette von `string_of_digits` einer gegebenen Variablen zu. Der Parameter `count1` kann dabei sowohl positiv als auch negativ sein. Ist er positiv, werden die ersten `count1` Ziffern von links ausgelassen, ist er negativ, werden `count1` Ziffern vom Ende des Strings nach links verschoben. Parameter `count2` bestimmt hierbei, wie viele Ziffern an der durch `count1` markierten Stelle weggenommen werden. Falls `count2` negativ ist, wird die dieser Zahl entsprechende Anzahl von Ziffern am Ende weggelassen.

Diese Verwendung dieser Anwendung wird nicht mehr empfohlen. Verwenden Sie an Ihrer Stelle `${EXTEN:X:Y}`.

```
; here are some examples using SubString():
;assign the area code (3 first digits) to variable TEST
exten => 8885551212,1,SubString(TEST=8885551212,0,3)
; assign the last 7 digits to variable TEST
exten => 8885551212,1,SubString(TEST=8885551212,-7,7)
; assign all but the last 4 digits to variable TEST
exten => 8885551212,1,SubString(TEST=8885551212,0,-4)
;
; and here are the preferred alternatives:
;assign the area code (3 first digits) to variable TEST
exten => 8885551212,1,Set(TEST=${EXTEN::3})
; assign the last 7 digits to variable TEST
exten => 8885551212,1,Set(TEST=${EXTEN:-7:7})
; assign all but the last 4 digits to variable TEST
exten => 8885551212,1,Set(TEST=${EXTEN:6}
```

6.101. `Suffix()`

Fügt Ziffern am Ende der aktuellen Extension an

```
Suffix(digits)
```

Hängt die durch digits spezifizierte Ziffernkette an die mit dem Kanal verknüpfte Extension an und fährt mit der Ausführung an der nächsten Priorität dieser neuen Extension fort. Sei etwa `Suffix(1212)` die Priorität 1 von Extension 555, dann wird 1212 an 555 angehangen und der nächste Schritt, der ausgeführt wird, ist Priorität 2 von Extension 5551212. Falls zu einer Extension gewechselt wird, die keine Priorität n+101 besitzt (wobei `n` die aktuelle Priorität bezeichne), wird sich PBX so verhalten, als ob der Teilnehmer eine ungültige Extension angewählt hat.

Liefert stets 0 zurück.

```
exten => 555,1,Suffix(1212)
exten => 5551212,2,SayDigits(${EXTEN})
```

Siehe auch. Abschnitt 6.66, „`Prefix()`“

6.102. `System()`

Führt einen Betriebssystembefehl aus

```
System(command)
```

Führ einen Befehl `command` auf dem zugrundeliegenden Betriebssystem aus. Falls der Befehl ausgeführt wird, jedoch einen Fehler hervorruft, und eine Priorität n+101 existiert (wobei `n` die aktuelle Priorität bezeichne), wird die Ausführung des Wählplans auf dieser Prioritätsstufe fortgesetzt.

Diese Anwendung ist der `TrySystem()`-Anwendung sehr ähnlich, abgesehen davon, dass sie -1 zurückliefert, falls sie den Systembefehl nicht ausführen kann, wohingegen die `TrySystem()`-Anwendung stets 0 zurückliefert.

```
exten => 123,1,System(echo hello > /tmp/hello.txt)
```

Siehe auch. Abschnitt 6.104, „`TrySystem()`“

6.103. `Transfer()`

Übermittelt den Anrufer zu einer entfernten Extension (Remote Extension)

```
Transfer(exten)
```

Verlangt die Übermittlung des fernen Anrufers zu einer gegebenen Extension. Falls eine solche Übermittlung nicht unterstützt wird oder erfolgreich verläuft, und eine Priorität n+101 existiert (wobei `n` die aktuelle Priorität bezeichne), wird diese Priorität als Nächste herangezogen.

```
; transfer calls from extension 123 to extension 130
exten => 123,1,Transfer(130)
```

6.104. `TrySystem()`

Versucht die Ausführung eines Betriebssystembefehls

```
TrySystem(command)
```

Versucht den Befehl command auf dem zugrundeliegenden Betriebssystem auszuführen. Falls der Befehl ausgeführt wird, jedoch einen Fehler hervorruft, und eine Priorität n+101 existiert (wobei `n` die aktuelle Priorität bezeichne), wird die Ausführung des Wählplans auf dieser Prioritätsstufe fortgesetzt.

Diese Anwendung entspricht weitestgehend der `System()`-Anwendung, abgesehen von der Tatsache, dass sie stets 0 zurückliefert, wohingegen die `System()`-Anwendung -1 zurückliefert, falls sie den Systembefehl nicht ausführen kann.

```
exten => 123,1,TrySystem(echo hello > /tmp/hello.txt)
```

Siehe auch. Abschnitt 6.102, „`System()`“

6.105. `TXTCIDName()`

Schlägt den Namen eines Anrufers in einer DNS TXT-Aufzeichnung nach

```
TXTCIDName(CallerID)
```

Schlägt den Namen eines Anrufers mittels DNS nach und setzt die Variable `${TXTCIDNAME}`. `TXTCIDNAME` wird daraufhin entweder leer sein oder den in der TXT-Aufzeichnung des DNS gefundenen Wert liefern. Diese Anwendung liest/sucht die Nummer mit Hilfe der in `enum.conf` gelisteten ENUM-Quellen.

```
exten => 123,1,TXTCIDName(8662331454)
exten => 123,2,SayAlpha(${TXTCIDNAME})
exten => 123,3,Playback(vm-goodbye)
```

6.106. `UnpauseQueueMember()`

Hebt die Blockierung für ein Warteschlangenmitglied auf

```
UnpauseQueueMember([queuename],interface)
```

Hebt die Blockierung für ein Mitglied einer Warteschlange auf, d.h. es kann Anrufe wiederaufnehmen. Dies ist das Gegenstück zu `PauseQueueMember()`. Die Arbeitsweise ist die gleiche, außer dass das gegebene Interface nicht blockiert, sondern seine Blockierung aufgehoben wird.

```
exten => 123,1,PauseQueueMember(,SIP/300)
exten => 124,1,UnpauseQueueMember(,SIP/300)
```

Siehe auch. Abschnitt 6.63, „`PauseQueueMember()`“

6.107. `UserEvent()`

Sendet ein beliebiges Ereignis an das Managerinterface

```
UserEvent(eventname[,body])
```

Sendet ein beliebiges Ereignis an das Managerinterface, mit einem optionalen Körper `body`, der zusätzliche Argumente repräsentiert. Das Ereignis steht in folgendem Format:

```
Event: UserEvent <specified event name>
Channel: <channel name>
Uniqueid: <call uniqueid>
[body]
```

Ist der Körper (`body`) nicht spezifiziert, können nur Event, Kanal und Uniqueid-Felder berücksichtigt werden.

Liefert stets 0 zurück.

```
exten => 123,1,UserEvent(BossCalled,${CALLERIDNAME} has called the boss!)
exten => 123,2,Dial(${BOSS})
```

Siehe auch. `manager.conf`, Asterisk Manager interface

6.108. `Verbose()`

Sendet beliebigen Text zum ausführlichen Ausgang (Verbose Output)

```
Verbose([level,]message)
```

Sendet die angegebene Nachricht `message` an den Verbose-Ausgang. Die Stufe `level` muss als ganzzahliger Wert angegeben sein. Ist `level` nicht angegeben, wird 0 angenommen.

Liefert stets 0 zurück.

```
exten => 123,1,Verbose(Somebody called extension 123)
exten => 123,2,Playback(extension)
exten => 123,3,SayDigits(${EXTEN})
```

6.109. `VMAuthenticate()`

Authentifiziert den Anrufer mittels Voicemail-Passwörtern

```
VMAuthenticate([mailbox][@context])
```

Verhält sich wie die `Authenticate()`-Anwendung, mit der Ausnahme, dass die Passwörter aus `voicemail.conf` gelesen werden.

Ist eine Mailbox `mailbox` stezifiziert, wird nur das Passwort dieser Mailbox als gültig betrachtet. Wird `mailbox` nicht angegeben, wird die Kanalvariable `${AUTH_MAILBOX}` mit der authentifizierten Mailbox belegt.

```
; authenticate off of any mailbox password, and tell us the matching
; mailbox number
exten => 123,1,VMAuthenticate()
exten => 123,2,SayDigits(${AUTH_MAILBOX})
```

Siehe auch. Abschnitt 6.9, „`Authenticate()`“, `voicemail.conf`

6.110. `VoiceMail()`

Hinterlässt eine Voicemail-Nachricht in der angegebenen Mailbox

```
VoiceMail([s|u|b]mailbox[@context][&mailbox[@context]][...])
```

Hinterlässt Voicemail für eine spezifizierte Mailbox `mailbox` (muss in `voicemail.conf` konfiguriert sein).

Steht der Mailbox ein `s` voran, werden die Anweisungen zum Hinterlassen der Nachricht übersprungen. Bei vorausgehendem `u` wird die Nicht-verfügbar-Nachricht abgespielt, falls sie vorhanden ist. Bei `b` wird die Besetzt-Botschaft abgespielt (Datei `busy` statt `unavail`).

Drückt der Teilnehmer während der Eingabeaufforderung `0` (Null), geht der Anruf zur `o` (kleiner Buchstabe o)-Extension im aktuellen Kontext über.

Drückt der Teilnehmer während der Eingabeaufforderung `*`, geht der Anruf zur `a`-Extension im aktuellen Kontext über. Dies ist gebräuchlich, um den Anrufer an einen persönlichen Assistenten zu übersenden/verweisen.

Falls die angeforderte Mailbox nicht existiert, und es eine Priorität n+101 gibt (mit `n`: aktuelle Priorität), wird die Priorität als nächstes herangezogen.

Sind mehrere Mailboxen angegeben, wird die Nicht verfügbar- bzw. Besetzt-Nachricht der ersten angegebenen Mailbox genommen.

Liefert im Falle eines Fehlers oder falls die Mailbox nicht gefunden wird, oder falls der Teilnehmer auflegt, -1 zurück, sonst 0.

```
; send caller to unavailable voicemail for mailbox 123
exten => 123,1,VoiceMail(u123)
```

```
; bei Anrufen auf der Durchwahl 123:
exten => 123,1,Answer                   ; abheben
exten => 123,2,Playback(meine-ansage)   ; Ansage abspielen
exten => 123,3,Voicemail(123)           ; Nachricht aufnehmen
exten => 123,4,Hangup                   ; auflegen
```

Siehe auch. Abschnitt 6.111, „`VoiceMailMain()`“, `voicemail.conf`

6.111. `VoiceMailMain()`

Betritt das Voicemail-System

```
VoiceMailMain([[s|p]mailbox][@context])
```

Tritt zur Überprüfung von (auf ??) Voicemail in das Voicemail-Hauptsystem ein. Die Übergabe des `mailbox`-Parameters hält das Voicemail-System von der Abfrage des Teilnehmers nach der Mailbox-Nummer ab.

Steht der Mailbox der Buchstabe `s` voran, wird die Überprüfung eines Passworts übersprungen. Bei der Mailbox vorangehendem Buchstaben `p` wird die bereitgestellte/angegebene Mailbox dem Eintrag des Benutzers vorangestellt und die daraus resultierende Zeichenkette als Mailboxnummer verwendet. Das ist nützlich bei der Bereitstellung virtueller Voicemail-Boxen (Virtual Hosting). Ist ein Kontext `context` spezifiziert, werden nur Anmeldungen in diesem Voicemail-Kontext berücksichtigt.

Liefert -1 zurück, wenn der Benutzer auflegt, sonst 0.

```
; go to voicemail menu for mailbox 123 in the default voicemail context
exten => 123,1,VoiceMailMain(123@default)
```

Siehe auch. Abschnitt 6.110, „`VoiceMail()`“, `voicemail.conf`

6.112. `Wait()`

Wartet für eine in Sekunden vorgegebene Dauer

```
Wait(seconds)
```

Wartet die angegebene Anzahl Sekunden, und liefert dann 0 zurück. Es können auch Sekundenbruchteile übergeben werden (z.B. 1.5 für 1,5 Sekunden).

```
; wait 1.5 seconds before playing the prompt
exten => s,1,Answer()
exten => s,2,Wait(1.5)
exten => s,3,Background(enter-ext-of-person)
```

6.113. `WaitExten()`

Wartet auf die Eingabe einer Extension

```
WaitExten([seconds])
```

Wartet die angegebene Anzahl Sekunden auf die Eingabe einer neuen Extension durch den Teilnehmer, und liefert dann 0 zurück. Es können auch Sekundenbruchteile übergeben werden (z.B. 1.5 für 1,5 Sekunden). Falls keine Zeitdauer spezifiziert wird, wird die Standardfrist (Default Extension Timeout) angewendet.

```
; wait 15 seconds for the user to dial an extension
exten => s,1,Answer()
exten => s,2,Playback(enter-ext-of-person)
exten => s,3,WaitExten(15)
```

6.114. `WaitForRing()`

Wartet die in Sekunden vorgegebene Dauer auf ein Klingelzeichen

```
WaitForRing(timeout)
```

Wartet nach dem Abschluss des nächsten Klingelns mindestens `timeout` Sekunden.

Liefert bei Erfolg 0, falls aufgelegt wurde -1 zurück.

```
; wait five seconds for a ring, and then send some DTMF digits
exten => 123,1,Answer()
exten => 123,2,WaitForRing(5)
exten => 123,3,SendDTMF(1234)
```

6.115. `WaitForSilence()`

Wartet auf eine festgelegte Dauer von Geräuchlösigkeit/Wartet auf eine festgelegt lange Ruhephase

```
WaitForSilence(wait[,repeat])
```

Wartet `repeat` mal auf `wait` Millisekunden Ruhe. Wird `repeat` weggelassen, wartet die Anwendung auf das Einmalige Auftreten von `wait` Millisekunden Ruhe.

```
; wait for three instances of 300 ms of silence
exten => 123,WaitForSilence(300,3)
```

6.116. `WaitMusicOnHold()`

Wartet für eine in Sekunden vorgegebene Dauer, wobei Wartemusik abgespielt wird

```
WaitMusicOnHold(delay)
```

Spielt die in Sekunden angegebene Zeit Wartemusik. Ist keine Wartemusik vorhanden, wird die Verzögerung dennoch ausgeführt, jedoch tonlos.

Liefert nach Abschluss 0, oder -1 falls aufgelegt wurde.

```
; allow caller to hear Music on Hold for five minutes
exten => 123,1,Answer()
exten => 123,2,WaitMusicOnHold(300)
exten => 123,3,Hangup()
```

Siehe auch. ???, `musiconhold.conf`

6.117. `While()`

Beginnt eine Whileschleife

```
While(expr)
```

Beginnt eine `while`-Schleife. Die Ausführung kehrt zu diesem Punkt zurück, wenn `Endwhile()` aufgerufen wird oder der Ausdruck `expr` nicht länger wahr ist. Wird eine Bedingung angetroffen, die ein Verlassen der Schleife bewirkt, wird sie noch nach `EndWhile()` fortgesetzt.

```
exten => 123,1,Set(COUNT=1)
exten => 123,2,While($[ ${COUNT} < 5 ])
exten => 123,3,SayNumber(${COUNT})
exten => 123,4,EndWhile()
```

Siehe auch. Abschnitt 6.29, „`EndWhile()`“, Abschnitt 6.40, „`GotoIf()`“

6.118. `Zapateller()`

Verwendet einen speziellen Informationston, um automatisierte Werbeanrufe abzublocken

```
Zapateller(options)
```

Generiert einen speziellen Informationston (SIT, special information tone, "Kein Anschluss unter dieser Nummer"), um Telehändler und andere maschinengewählte Anrufe davon abzuhalten, Sie zu belästigen.

Der `options`-Parameter stellt eine Liste von Optionen dar, die durch `|`-Zeichen voneinander getrennt sind. Die folgenden Optionen sind verfügbar:

`answer` Bewirkt, dass die Leitung beantwortet wird bevor der Ton abgespielt wird

`nocallerid` Bewirkt, dass Zapateller den Ton nur abspielt, wenn keine Caller ID-Information verfügbar ist

```
[incoming]
; im Kontext incoming answer the line, and play the SIT tone if there is no Caller ID information
exten => s,1,Zapateller(answer|nocallerid)
```

Siehe auch. Abschnitt 6.67, „`PrivacyManager()`“

6.119. `ZapBarge()`

Tritt in einen Zap-Kanal ein und überwacht ihn

```
ZapBarge([channel])
```

Tritt in den angegebenen Zap-Kanal `channel` ein, oder fragt zunächst per Eingabeaufforderung einen ab, falls keiner angegeben wurde. Die anderen Teilnehmer auf dem Kanal können Sie dabei nicht hören werden und werden auch sonst keine Anzeichen wahrnehmen, dass ihr Anruf überwacht wird.

Ist der Kanal (`channel`) nicht angegeben, werden Sie zunächst zur Eingabe einer Kanalnummer aufgefordert. Geben Sie z.B. `4#` für Zap/4 ein.

Liefert -1, wenn der Anrufer auflegt. Ist unabhängig von dem Zustand, dass der Kanal überwacht wird.

```
exten => 123,1,ZapBarge(Zap/2)
exten => 123,2,Hangup()
```

Siehe auch. Abschnitt 6.121, „`ZapScan()`“, Abschnitt 6.16, „`ChanSpy()`“

6.120. `ZapRAS()`

Führt den entfernten Zaptel ISDN-Zugriffsserver (Zaptel ISDN Remote Access Server) aus.

```
ZapRAS(args)
```

Startet über `pppd` einen ISDN RAS Server auf dem aktuellen Kanal. Der Kanal muss ein freier Kanal sein (z.B. PRI Source), und ein Zaptel-Kanal sein, um diese Funktion durchführen zu können. (Eine Modem-Emulation ist nicht enthalten.)

Um Zaptel zu kennen, muss `pppd` entsprechend präpariert sein. `args` ist eine Liste von durch das |-Symbol getrennten Argumenten.

Liefert stets -1 zurück.

Diese Anwendungen ist nur für die Benutzung mit ISDN-Leitungen ausgelegt, und Ihr Kernel muss mit einem Patch für die Unterstützung von `ZapRAS()` versehen sein. Ihr Kernel muss ebenso ppp-Unterstützung bereitstellen.

```
exten => 123,1,Answer()
exten => 123,1,ZapRas(debug|64000|noauth|netmask|255.255.255.0|10.0.0.1:10.0.0.2)
```

6.121. `ZapScan()`

Durchsucht Zap-Kanäle um Anrufe zu überwachen/Hört Zap-Kanäle ab, um Anrufe zu überwachen

```
ZapScan([group])
```

Erlaubt einem Call Center Manager, Zap Kanäle auf bequeme Art zu überwachen. Benutzen Sie `#` um den nächsten Kanal auszuwählen, oder `*` zum beenden. Sie können die Abfrage auf eine bestimmte Kanalgruppe begrenzen, indem Sie den `group`-Parameter verwenden.

```
exten => 123,1,ZapScan()
```

Siehe auch. Abschnitt 6.119, „`ZapBarge()`“, Abschnitt 6.16, „`ChanSpy()`“

7. Funktionen im Dialplan

Seit Asterisk 1.2 gibt es im Dialplan nicht nur Applikationen, sondern auch Funktionen. Anders als Applikationen können Funktionen nicht direkt aufgerufen werden. Stattdessen geben sie einen Wert zurück oder - was dem klassichen Verständnis einer Funktion widerspricht - man kann auch, in Verbindung mit der Applikation `Set()` (siehe Abschnitt 6.90, „`Set()`“), schreibend auf sie zugreifen. Funktionsnamen werden immer ganz in Großbuchstaben geschrieben. Funktionen werden erstaunlicherweise (wie Variablen) immer in `${}` eingeklammert, was notwendig ist, da Strings nicht immer in Hochkommata stehen müssen.

Würde man die in Asterisk nicht unbedingt intuitive Trennung zwischen Applikationen, Funktionen und sogar Variablen kritisieren, hätte man damit wahrscheinlich nicht ganz unrecht. Auch die Benennung ist nicht einheitlich: `SIP_HEADER()`, aber `SIPCHANINFO()` - ein Problem vieler Programmiersprachen, das letztlich nichts aussagt, aber die Einprägsamkeit erschwert. Zudem werden die Zeichen `,` (Komma), `&` (Und) und `|` (Pipe) scheinbar willkürlich als Trennzeichen verwendet. Die Idee, in Funktionen zu *schreiben*, ist recht ungewöhnlich und weicht vom üblichen Verständnis einer Funktion ab.

Insgesamt wirkt die Dialplan-Programmierung - trotz der Neuerungen in Version 1.4 - im Vergleich zu „richtigen“ Programmiersprachen sehr starr und wenig erweiterungsfähig. Wen das stört, der sollte sich vielleicht mit der - noch experimentellen - AEL (Asterisk Extension Language) befassen (siehe Datei `extensions.ael`), die die gleichen Funktionen und Applikationen verwendet, aber eine etwas robustere Struktur bietet und oft übersichtlicher ist.

Im Asterisk-CLI[18] können Sie mit **`show functions`** bzw. **`show function`** ***`FUNKTIONSNAME`*** herausfinden, welche Funktionen in Ihrer Installation verfügbar sind.

7.1. `AGENT()`

```
AGENT(AgentenNr:Feld)
```

Gibt Informationen über einen Agenten, identifiziert duch die `AgentenNr` zurück. Die folgenden Felder können abgefragt werden:

`status`	Der Einlog-Status des Agenten, entweder `LOGGEDIN` oder `LOGGEDOUT`.
`password`	Das Passwort des Agenten.
`name`	Der Name des Agenten.
`mohclass`	Die Wartemusik-Klasse (Music on hold class).
`exten`	Die Rückruf-Extension des Agenten. Wird von der Applikation `AgentCallbackLogin()` verwendet.
`channel`	Der Name des aktiven Kanals des Agenten (`AgentLogin()`)

```
; die Variable foo auf den Namen des Agenten Nr. 42 setzen:
exten => s,1,Set(foo=${AGENT(42:name)})
```

7.2. `ARRAY()`

```
ARRAY(var1[,var2[,...]])
```

(ab Asterisk 1.4 verfügbar)

Setzt mehrere Variablen gleichzeitig. (Der Name ist irreführend, ein Array wird nicht erstellt.) Mit `ARRAY()` kann nur geschrieben, nicht gelesen werden. Denken Sie daran, die nach dieser Funktion folgenden Werte durch Komma zu trennen und dieses Komma durch Backslash (`\`) zu escapen, da sonst von einem zweiten Parameter für `Set()` ausgegangen würde!

```
; var1 auf 1 und var2 auf 2 setzen:
exten => s,1,Set(ARRAY(var1,var2)=1\,2)
```

7.3. `BASE64_DECODE()`

```
BASE64_DECODE(Base64_String)
```

(ab Asterisk 1.4 verfügbar)

[18] Command Line Interface, kann durch **asterisk -r** aufgerufen werden

Dekodiert einen Base64-kodierten String.

```
exten => s,1,Set(foo=${BASE64_DECODE("SGFsbG8gV2VsdA==")})
```

7.4. BASE64_ENCODE()

```
BASE64_ENCODE(String)
```

(ab Asterisk 1.4 verfügbar)

Kodiert einen String mit Base64.

```
exten => s,1,Set(foo=${BASE64_ENCODE("Hallo Welt")})
```

7.5. CALLERID()

```
CALLERID(Feld)
```

Gibt Informationen über den Anrufer zurück oder setzt diese. `Feld` ist eines der folgenden:

`name`	Name des Anrufers, alphanumerischer String. Sie sollten sich auf wenige Zeichen - z.B. 15 - beschränken.
`num`	Nummer, nur Ziffern. (manchmal findet man auch `number`, abhängig von der Asterisk-Version?)
`all`	Name und Nummer mit der Nummer in spitzen Klammern, z.B.: "`Peter Meier <012345>`"
`ani`	ANI[19]-Nummer, für ausgehende Anrufe
`dnid`	DNID[20]-Nummer. Die gewählte Nummer. (manchmal findet man auch `dnis`, abhängig von der Asterisk-Version?)
`rdnis`	RDNIS[21]-Nummer. Die Nummer, die auf die aktuelle Extension umgeleitet hat, z.B. falls die Nummer einer Voicemailbox nicht die gewählte Extension ist.

Ab Asterisk 1.4 muss z.B. statt der Variablen `${CALLERIDNUM}` die Funktion `${CALLERID(num)}` verwendet werden, statt `${RDNIS}` `$(CALLERID(rdnis))` etc. Die Applikation `SetCIDName()` wird durch `Set(CALLERID(name)=`*`Name`*`)` ersetzt, `SetCallerID()` durch `Set(CALLERID(all)=`*`Name <Nummer>`*`)`, etc.

```
; die Variable foo auf die Anruferkennung setzen:
exten => s,1,Set(foo=${CALLERID(all)})

; den Anrufernamen auf "Peter Meier" setzen:
exten => s,1,Set(CALLERID(name)="Peter Meier")
```

7.6. CDR()

```
CDR(Feld)
```

Liest oder setzt CDR[22]-Felder. `Feld` ist eines der folgenden (wenn nicht anders angegeben ist nur Lesen möglich):

`clid`	Caller-ID
`src`	Source, die Caller-ID-Nummer
`dst`	Destination, Ziel-Extension
`dcontext`	Destination context, der Ziel-Kontext

[19] Automatic Number Identification
[20] Dialed/Destination Number Identification Service
[21] Redirected Dialed Number Identification Service
[22] Call Data Record, siehe CDR

`channel`	Der Kanal-Name
`dstchannel`	Destination channel, der Ziel-Kanal, falls zutreffend
`lastapp`	Die letzte ausgeführte Applikation
`lastdata`	Die Argumente der zuletzt ausgeführten Applikation
`start`	Die Start-Zeit des Anrufs
`answer`	Die Zeit, zu der der Anruf beantwortet wurde
`end`	Die End-Zeit des Anrufs
`duration`	Die Dauer des Anrufs (in Sekunden)
`billsec`	Die Dauer des Anrufs seit der Anruf beantwortet wurde, also die zu berechnende Zeit (in Sekunden)
`disposition`	Der Status des Anrufs: `ANSWERED`, `NO ANSWER`, `BUSY` oder `FAILED`
`amaflags`	Die AMA[23]-Flags. `DEFAULT` (System-Default), `BILLING` (zur Abrechnung), `DOCUMENTATION` (zur Dokumentierung) oder `OMIT` (keine Aufzeichnung). (Manchmal findet man statt `BILLING` und `OMIT` auch die Angaben `BILL` und `IGNORE` - abhängig von der Asterisk-Version?)
`accountcode`	Die alphanumerische Nummer des Abrechnungskontos, max. 20 Zeichen (auch Setzen möglich)
`uniqueid`	Die einmalige alphanumerische ID des Kanals (max. 32 Zeichen)
`userfield`	Das User-Feld, in das beliebige Informationen (max. 255 Zeichen) geschrieben werden können (auch Setzen möglich)

```
; foo auf die Dauer des Anrufs setzen:
exten => s,1,Set(foo=${CDR(duration)})

; das User-Feld auf "meine Infos" setzen:
exten => s,1,Set(CDR(userfield)=meine Infos)
```

7.7. `CHANNEL()`

```
CHANNEL(Feld)
```

(ab Asterisk 1.4 verfügbar)

Liest / setzt bestimmte Kanal-Daten. `Feld` ist eines der folgenden (wenn nicht anders angegeben ist nur Lesen möglich):

`audioreadformat`	Das Format eingehender Audio-Daten auf dem Kanal.
`audionativeformat`	Das native Audio-Format des Kanals.
`audiowriteformat`	Das Format ausgehender Audio-Daten auf dem Kanal.
`callgroup`	In Asterisk lassen sich Extensions in Call groups von 0 - 63 einordnet, z.B. als Kundennummer.[24]
`channeltype`	Die „Technologie“ dieses Kanals, also z.B: IAX oder SIP
`language`	Die Sprache für Voice-Prompts (auch Setzen möglich).

[23] Automated Message Accounting
[24] Diese Beschränkung auf 64 Gruppen scheint rein willkürlich zu sein, und dürfte für den ein oder anderen Anwender nicht ausreichen.

`musicclass`	Die Musik-Klasse für Wartemusik, wie in `musiconhold.conf` definiert (auch Setzen möglich).
`state`	Zustand des Kanals (`Down`, `Rsrvd`, `OffHook`, `Dialing`, `Ring`, `Ringing`, `Up`, `Busy`, `Dialing Offhook`, `Pre-ring`, `Unknown`)
`tonezone`	Die „Tone zone“ bestimmt die Ton-Signale (Wählen, Klingeln, Besetzt, ...) für bestimmte Länder. Sie wird in der Konfigurationsdatei des Kanals (z.B. `zaptel.conf`) durch `loadzone` und `defaultzone` festgelegt. Die möglichen Werte sind (wie in der `indications.conf` definiert): `at`, au, be, br, ch, cl, cn, cz, de, dk, ee, es, fi, fr, gr, hu, it, lt, mx, ml, no, nz, pl, pt, ru, se, sg, uk, us, us-old, tw, ve, za
`videonativeformat`	Das native Video-Format dieses Kanals.

Zusätzlich zu diesen Feldern kann der Treiber des Kanals noch weitere bereitstellen, die man in der entsprechenden Dokumentation nachschlagen kann. Felder, die auf dem aktuellen Kanal nicht zur Verfügung stehen, geben einen leeren String zurück.

```
; Typ des Kanals abfragen:
exten => s,1,Set(foo=${CHANNEL(channeltype)})

; Sprache auf Englisch setzen:
exten => s,1,Set(CHANNEL(language)=en)
```

7.8. `CHECKSIPDOMAIN()`

```
CHECKSIPDOMAIN(Domain)
```

Prüft, ob die angegebene Domain (kann auch eine IP-Adresse sein) eine SIP-Domain ist, die Asterisk als lokal behandelt (siehe Datei `sip.conf`). Gibt den Domain-Namen oder einen leeren String zurück.

```
exten => s,1,Set(foo=${CHECKSIPDOMAIN(123.45.67.89)})
```

7.9. `CURL()`

```
CURL(URL[|POST-Daten])
```

Läd eine Web-Seite unter der angegebenen `URL` mit `GET`. Wenn `POST-Daten` angegeben sind, werden diese Daten mit `POST` gesendet. Gibt die Seite als String zurück.

```
; Die Seite unter der URL http://beispiel.de/seite.php?id=1&action=view abrufen:
exten => s,1,Set(foo=${CURL(http://beispiel.de/seite.php?id=1&action=view)})
```

7.10. `CUT()`

```
CUT(Variablenname,Trennzeichen,Feld)
```

(Offenbar müssen in Asterisk 1.2.8 statt der Kommas zwischen den Parametern Pipe-Zeichen (`|`) verwendet werden.)

Zerteilt den String in einer Variablen anhand eines Trennzeichens (Default: `-`) und gibt das so entstandene Einzelteil an der Stelle `Feld` (1-basiert) zurück. `Feld` kann auch eine Spanne sein (z.B. `2-4`) oder mehrere Felder/Spannen, durch `&` getrennt, z.B. `2-4&6`; die entsprechenden Felder werden dann wieder durch das Trennzeichen verbunden. Auch eine Range wie `3-` (alles ab Feld 3) oder `-3` (bis Feld 3) ist möglich.

Ein Komma als Trennzeichen muss durch Backslash escapt werden, z.B. `CUT(var,\,,2)`.

```
exten => s,1,Set(var=1-2-3-4-5)            ; var ist "1-2-3-4-5"
exten => s,2,Set(var=${CUT(var,,1-3&5)})   ; var ist "1-2-3-5"
```

Wichtig

Der Parameter `Variablenname` muss der *Name* einer Variablen, sein, nicht ein String. Die wäre also *falsch*: `CUT(${bar},,3)`

Siehe auch. Abschnitt 7.18, „FIELDQTY()“

7.11. DB()

```
DB(Familie/Schlüssel)
```

Liest / setzt einen Wert in der AstDB. Beim Lesen wird der Wert zurückgegeben oder ein leerer String, wenn der Wert nicht existiert. Dabei wird auch die Variable DB_RESULT gesetzt.

```
; Eintrag open/source setzen und abfragen:
exten => s,1,Set(DB(open/source)=${ja})
exten => s,2,Set(var=${DB(open/source)})
exten => s,3,GotoIf($[[${DB(open/source)} = 1]?opensource:closedsource)
```

Siehe auch. Abschnitt 7.12, „DB_EXISTS()“, Abschnitt 6.19, „DBdel()“, Abschnitt 6.20, „DBdeltree()“

7.12. DB_EXISTS()

```
DB_EXISTS(Familie/Schlüssel)
```

Prüft, ob ein Schlüssel in der AstDB vorhanden ist. Gibt 1 oder 0 zurück. Setzt auch die Variable DB_RESULT auf den unter dem Schlüssel gespeicherten Wert.

```
; abfragen, ob der Eintrag cidnums/0123456789 existiert:
exten => s,1,Set(foo=${DB_EXISTS(cidnums/0123456789)})
```

So kann z.B. die Applikation LookupBlacklist() ersetzt werden. Springt, falls die Anrufernummer in der Blacklist ist, zum Kontext blacklisted, Extension s, Priorität 1:

```
exten => s,1,GotoIf(${DB_EXISTS(blacklist/${CALLERID(num)})}?blacklisted,s,1)
exten => s,2,Dial(IAX2/benutzer:passwort@beispiel.de/500)

[blacklisted]
exten => s,1,NoOp(${CALLERID(num)} ist auf der Blacklist)
exten => s,2,Hangup()
```

Siehe auch. Abschnitt 7.11, „DB()“, Abschnitt 6.19, „DBdel()“, Abschnitt 6.20, „DBdeltree()“

7.13. DUNDILOOKUP()

```
DUNDILOOKUP(Nummer[|DUNDi-Kontext[|Optionen]])
```

Schlägt eine Telefonnummer mit DUNDi (DUNDi) nach. Wird kein DUNDi-Kontext angegeben, wird e164 angenommen. Mit der Option b (bypass) wird der interne DUNDi-Cache umgangen. Gibt die gefundene Verbindung in der Form Technologie/Resource oder einen leeren String zurück.

```
; die Nummer 0123456789 nachschlagen:
exten => s,1,Set(foo=${DUNDILOOKUP(0123456789)})
```

Siehe auch. dundi.conf

7.14. ENUMLOOKUP()

Asterisk 1.2:

```
ENUMLOOKUP(Nummer[,Dienst[,OptionenUndEintragsnr[,Zonen-Suffix]]])
```

Asterisk 1.4:

```
ENUMLOOKUP(Nummer[,Dienst[,Optionen,Eintragsnr[,Zonen-Suffix]]])
```

Schlägt eine Nummer mit ENUM (ENUM) nach. Dienst kann u.A. sip (Default), iax2, h323, tel oder ALL sein. Mit der Option c wird die Anzahl der Einträge zurückgegeben. Die Eintragsnr (Default 1) wählt einen Eintrag

aus der Ergebnisliste aus. Zonen-Suffix (Default: `e164.arpa`) ist die ENUM-Zone. Ausführliche Beschreibung und Beispiele in `doc/README.enum` (1.2) / `doc/enum.txt` (1.4)

```
; in Asterisk 1.2:
exten => s,1,Set(foo=${ENUMLOOKUP(+${CALLERID(num)},sip,1,freenum.org)})

; in Astarisk 1.4:
exten => s,1,Set(foo=${ENUMLOOKUP(+${CALLERID(num)},sip,,1,freenum.org)})
```

Siehe auch. `enum.conf`

7.15. `ENV()`

```
ENV(Variablenname)
```

Liest / setzt eine Umgebungsvariable (Environment). Umgebungsvariablen sind keine Variablen in Asterisk sondern in der Betriebssystem-Umgebung. Sie können in der Shell mit `echo $Variablenname` ausgegeben werden. Umgebungsvariablen unterscheiden zwischen Groß- unf Kleinschreibung und werden fast immer groß geschrieben.

```
; HOME lesen:
exten => s,1,Set(foo=${ENV(HOME)})

; HOME setzen:
exten => s,1,Set(ENV(HOME)=/myAst)
```

7.16. `EVAL()`

```
EVAL(Variable)
```

Wertet eine Variable doppelt aus. Das lässt sich am besten an einem Beispiel erklären: Wenn die Variable `${VAR}` den String "`${VAR2}`" enthält, würde sie ohne `Eval()` zu diesem String ausgewertet. Mit `Eval()` wird auch `${VAR2}` nochmal zu ihrem Inhalt ausgewertet.

```
; wenn VAR den String "${VAR2}" enthält und VAR2 den String "Hallo Welt":
exten => s,1,Set(foo=${EVAL(${VAR})})
; dann ist foo jetzt "Hallo Welt"
```

7.17. `EXISTS()`

```
EXISTS(Variable)
```

Prüft, ob eine Variable definiert ist. Gibt 1 oder 0 zurück.

```
exten => s,1,Set(Var1=test)
exten => s,2,Set(Var2=)
exten => s,3,Set(foo=${EXISTS(${Var1})})   ; foo ist 1
exten => s,4,Set(foo=${EXISTS(${Var2})})   ; foo ist 0
```

7.18. `FIELDQTY()`

```
FIELDQTY(Variablenname,Trennzeichen)
```

Gibt zurück, wieviele Teil-Strings entstehen würden, wenn man den Inhalt der Variablen anhand des Trennzeichens aufteilt.

```
exten => s,1,Set(Var=hallo#du#da#am#telefon)
exten => s,1,Set(Anzahl=${FIELDQTY(Var,#)})    ; Anzahl ist 5
```

Siehe auch. Abschnitt 7.10, „`CUT()`“

7.19. `FILTER()`

```
FILTER(erlaubte Zeichen,String)
```

(ab Asterisk 1.4 verfügbar)

Filtert den String, so dass nur noch die erlaubten Zeichen im Rückgabewert vorkommen.

```
; nur die Zeichen 0123456789 in ${cdrnum} zulassen:
exten => s,1,Set(foo=${FILTER(0123456789,${cdrnum})})
```

7.20. GLOBAL()

```
GLOBAL(Variablenname)
```

(ab Asterisk 1.4 verfügbar)

Wird verwendet, um eine Variable als global zu deklarieren, also gültig über die Lebensdauer eines Kanals hinaus. In Asterisk 1.2 verwendet man stattdessen `Set()` (Abschnitt 6.90, „`Set()`“) mit der Option `g`.

```
; eine globale Variable ${meinevariable} definieren:
exten => s,1,Set(GLOBAL(meinevariable)=Test)
```

Anmerkung

Ob globale Variablen auch nach einem Reload von Asterisk noch gültig sind, wird durch die Einstellung `clearglobalvars` in der `extensions.conf` bestimmt.

7.21. GROUP()

```
GROUP([Kategorie])
```

Liest / setzt die Gruppe des Kanals (Kanäle lassen sich beliebig gruppieren).

```
exten => s,1,Set(GROUP()=ausgehende)                ; Gruppe setzen
exten => s,2,GotoIf($[${GROUP_COUNT()} > 10]?103)   ; zu viele?
exten => s,3,Dial(0123456)                          ; wählen
exten => s,103,SetVar(DIALSTATUS=CHANUNAVAIL)       ; ablehnen
```

Siehe auch. Abschnitt 7.22, „`GROUP_COUNT()`“, Abschnitt 7.23, „`GROUP_LIST()`“, Abschnitt 7.24, „`GROUP_MATCH_COUNT()`“

7.22. GROUP_COUNT()

```
GROUP_COUNT([Gruppe[@Kategorie]])
```

Gibt die Anzahl der Kanäle in der angegebenen Gruppe zurück. Wird keine Gruppe angegeben, gilt die aktuelle Gruppe des Kanals.

```
; Anzahl der Kanäle in Gruppe ausgehende abfragen:
exten => s,1,Set(foo=${GROUP_COUNT(ausgehende)})
```

Siehe auch. Abschnitt 7.21, „`GROUP()`“, Abschnitt 7.23, „`GROUP_LIST()`“, Abschnitt 7.24, „`GROUP_MATCH_COUNT()`“

7.23. GROUP_LIST()

```
GROUP_LIST()
```

Gibt eine durch Leerzeichen getrennt Liste aller Gruppen zurück, die für den aktuellen Kanal gesetzt sind.

```
exten => s,1,Set(foo=${GROUP_LIST()})
```

Siehe auch. Abschnitt 7.21, „`GROUP()`“, Abschnitt 7.22, „`GROUP_COUNT()`“, Abschnitt 7.24, „`GROUP_MATCH_COUNT()`“

7.24. GROUP_MATCH_COUNT()

```
GROUP_MATCH_COUNT(Muster[@Kategorie])
```

Gibt die Anzahl der Kanäle zurück, auf die das angegebene Suchmuster - ein regulärer Ausdruck - passt.

```
; Anzahl der Kanäle in den Gruppen gruppe[1-4] abfragen:
exten => s,1,Set(foo=${GROUP_MATCH_COUNT(gruppe[1-4])})
```

Siehe auch. Abschnitt 7.21, „`GROUP()`“, Abschnitt 7.22, „`GROUP_COUNT()`“, Abschnitt 7.23, „`GROUP_LIST()`“

7.25. `IAXPEER()`

```
IAXPEER(Peer-Name[:Feld])
```

Gibt Daten über einen IAX-Peer zurück. `Peer-Name` kann auch `CURRENTCHANNEL` für den aktuellen Kanal sein. `Feld` ist eines der folgenden:

`ip`	(Default) Die IP-Adresse des Peers
`status`	Der Status (wenn `qualify=yes` gesetzt ist)
`mailbox`	Die eingestellte Mailbox
`context`	Der eingestellte Kontext
`expire`	Die Unix-Time (Epoch), wann die Verbindung das nächste Mal ausläuft.
`dynamic`	Ob die Verbindung dynamisch (`dynamic`) ist (`yes`|`no`)
`callerid_name`	Der eingestellte Caller-ID-Name
`callerid_num`	Die eingestellte Caller-ID-Nummer
`codecs`	Die verfügbaren Codecs
`codec[x]`	Der Codec Nummer `x` (0-basiert) in der Reihenfolge der Präferenz

```
; die IP-Adresse von peer1 abfragen:
exten => s,1,Set(foo=${IAXPEER(peer1:ip)})
```

Siehe auch. Abschnitt 7.46, „`SIPPEER()`“

7.26. `IF()`

```
IF(Ausdruck?trueVal:falseVal)
```

Gibt abhängig von der Bedingung den ersten oder zweiten Wert zurück. Ist die Bedingung wahr, wird der Wert nach `?`, sonst der Wert nach `:` zurückgegeben.

```
; wenn ${Var}=123 ist, dann 5 zurückgeben, sonst 9:
exten => s,1,Set(foo=${IF($[ ${Var} = 123]?5:9)})
```

Siehe auch. Abschnitt 7.27, „`IFTIME()`“, xAbschnitt 6.31, „`ExecIf()`“, Abschnitt 6.40, „`GotoIf()`“, Abschnitt 6.41, „`GotoIfTime()`“

7.27. `IFTIME()`

```
IFTIME(Zeit-Ausdruck?trueVal:falseVal)
```

Gibt - wie `IF()` - abhängig von der Bedingung den ersten oder zweiten Wert zurück, nur dass hier ein spezieller Zeit-Ausdruck angegeben wird.

Der Zeit-Ausdruck ist aufgebaut im Format Uhrzeit|Wochentag|Monatstag|Monat, wobei jeder Teil auch eine Spanne (mit `-`) oder das immer passende Wildcard `*` sein kann. Die Zeit wird im 24-Std.-Format (z.B. `08:00-18:00`), Wochentage und Monatsnamen als 3-buchstabige englische Abkürzung (`mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun` bzw. `jan`, `feb`, `mar`, `apr`, `may`, `jun`, `jul`, `aug`, `sep`, `oct`, `nov`, `dec`) angegeben.

```
; gültig 8-18 Uhr, montags, 1.-15. Dezember:
exten => s,1,Set(foo=${IFTIME(08:00-18:00|mon|1-15|dec?5:0)})

; gültig jeden Samstag und Sonntag:
exten => s,1,Set(foo=${IFTIME(*|sat-sun|*|*?5:0)})
```

Siehe auch. Abschnitt 7.26, „`IF()`“, Abschnitt 6.31, „`ExecIf()`“, Abschnitt 6.40, „`GotoIf()`“, Abschnitt 6.41, „`GotoIfTime()`“

7.28. `ISNULL()`

```
ISNULL(Wert)
```

Gibt 1 zurück wenn der Wert NULL (kein Wert) ist, sonst 0.

```
exten => s,1,Set(foo=${ISNULL(${Var1})})
```

7.29. `KEYPADHASH()`

```
KEYPADHASH(String)
```

Wandelt die Buchstaben A-Z (und a-z) entsprechend der Telefontastatur in Ziffern um. Damit können z.B. Vanity-Numbers wie 0800-AUTOHAUS (0800-28864287) in Nummern umgewandelt werden.

1	2 ABC	3 DEF
4 GHI	5 JKL	6 MNO
7 PQRS	8 TUV	9 WXYZ

```
exten => s,1,Set(foo=${KEYPADHASH(AUTOHAUS)})  ; gibt 28864287 zurück
```

7.30. `LANGUAGE()`

```
LANGUAGE()
```

Liest / setzt die Sprache des Kanals. Die Sprache beeinflusst u.a., welche Audio-Dateien abgespielt werden. Ist z.B. `de` eingestellt und `Playback(tt-weasels)` wird ausgeführt, so wird die Datei `de/tt-weasels` abgespielt (wenn vorhanden). Gleiches gilt für `SayDigits()` etc.

```
; abfragen:
exten => s,1,Set(foo=${LANGUAGE()})

; Deutsch setzen:
exten => s,1,Set(LANGUAGE()=de)
```

7.31. `LEN()`

```
LEN(String)
```

Gibt die Länge (Anzahl der Zeichen) eines Strings zurück.

```
; wenn ${test} = "Hallo Welt" ist
exten => s,1,Set(foo=${LEN(${test})})
; dann wird 10 zurückgegeben
```

7.32. `MATH()`

```
MATH(Zahl1 Operator Zahl2[,Ergebnistyp])
```

Berechnet einfache mathematische Ausdrücke. Mögliche Operatoren sind: `+`, `-`, `/`, `*`, `<`, `>`, `<=`, `>=`, `==`, `%` (modulo). Ergebnistyp: `f`, `float` (Fließkommazahl, Default), `i`, `int` (Ganzzahl), `h`, `hex` (Wert im Hexadezimal-System), `c`, `char` (Byte)

```
; 3*8 als Ganzzahl berechnen:
exten => s,1,Set(i=${MATH(3*8,int)})
```

7.33. MD5()

```
MD5(String)
```

Berechnet den MD5-Hash (Digest, Prüfsumme) eines Strings (hexadezimale Rückgabe).

```
exten => s,1,Set(foo=${MD5(${string})})
```

7.34. MUSICCLASS()

```
MUSICCLASS()
```

Liest / setzt die Music-on-Hold-Klasse (Klasse der Wartemusik).

```
; abfragen:
exten => s,1,Set(foo=${MUSICCLASS()})

; auf "HeavyMetal" setzen:
exten => s,1,Set(MUSICCLASS()=HeavyMetal)
```

7.35. ODBC_SQL()

```
ODBC_SQL(SQL-Query)
```

Führt den angegebenen SQL-Query aus und gibt ggf. das Ergebnis zurück.

```
; abfragen:
exten => s,1,Set(Name=${ODBC_SQL(SELECT name FROM liste WHERE nummer='123')})

; setzen:
exten => s,1,Set(ODBC_SQL(UPDATE liste SET name='Peter' WHERE nummer='123'))
```

7.36. ODBC_USER_DATABASE()

```
ODBC_USER_DATABASE(var1[,var2[,...]])
```

Führt das in `func_odbc.conf` definierte SQL-Query aus und gibt ggf. das Ergebnis zurück. Die in der Abfrage definierten `${VAL1}`, `${VAL2}`, ..., `${ARG1}`, `${ARG2}`, ... werden durch die angegebenen Werte ersetzt.

`func_odbc.conf`:

```
[USER_DATABASE]
dsn=meine_datenbank
read=SELECT name FROM liste WHERE nummer='${ARG1}'
write=UPDATE liste SET name=${VAL1} WHERE nummer='${ARG1}'
```

`extensions.conf`:

```
; abfragen (read):
exten => s,1,Set(Name=${ODBC_USER_DATABASE(${EXTEN})})

; setzen (write):
exten => s,1,Set(ODBC_USER_DATABASE(${CALLERID(name)})=1000)
```

7.37. QUEUEAGENTCOUNT()

in Asterisk 1.2 - für 1.4 siehe `QUEUE_MEMBER_COUNT()`

```
QUEUEAGENTCOUNT(Warteschlange)
```

Gibt die Anzahl der Agenten (im Gegensatz zu den Anrufern) auf der Warteschlange zurück.

```
; Anzahl der Agenten in der supportSchlange:
exten => s,1,Set(foo=${QUEUEAGENTCOUNT(supportSchlange)})
```

7.38. QUEUE_MEMBER_COUNT()

in Asterisk 1.4 - für 1.2 siehe QUEUEAGENTCOUNT()

```
QUEUE_MEMBER_COUNT(Warteschlange)
```

Gibt die Anzahl der Agenten (im Gegensatz zu den Anrufern) auf der Warteschlange zurück.

```
; Anzahl der Agenten in der supportSchlange:
exten => s,1,Set(foo=${QUEUE_MEMBER_COUNT(supportSchlange)})
```

7.39. QUEUE_MEMBER_LIST()

```
QUEUE_MEMBER_LIST(Warteschlange)
```

Gibt eine durch Komma getrennte Liste der Agenten auf einer Warteschleife zurück.

```
; Agenten in der supportSchlange:
exten => s,1,Set(foo=${QUEUE_MEMBER_LIST(supportSchlange)})
; gibt z.B. 5,8,33 zurück
```

7.40. QUOTE()

```
QUOTE(String)
```

Escapt Anführungszeichen innerhalb eines Strings.

```
; wenn ${var} >>Die "Asterisk"-PBX<< ist
exten => s,1,Set(foo=${QUOTE(${var})})
; dann wird >>Die \"Asterisk\"-PBX<< zurückggeben
```

7.41. RAND()

```
RAND(min,max)
```

(ab Asterisk 1.4 verfügbar)

Gibt ein Zahl zwischen min und max (beide einschließlich) zurück. Der Default für min ist 0, der für max ist die größte auf dem System darstellbare Ganzzahl (meist 2147483647).

```
; einen zufälligen Wert zwischen 1 und 10 (einschließlich) wählen:
exten => s,1,Set(zufall=${RAND(1,10)})
```

7.42. REGEX()

```
REGEX("Ausdruck" String)
```

Gibt 1 zurück, wenn ein String auf einen regulären Ausdruck passt, sonst 0. Der reguläre Ausdruck kann auch ^ (passt auf den Anfang) und $ (passt auf das Ende) enthalten. Variablen werden vorher ausgewertet.

Der Parser in Asterisk 1.2 arbeitet ncht sauber und kann durch besondere Zeichen wie $ oder geschweifte Klammern im Ausdruck verwirrt werden. Als Workaround definiert man sich z.B. eine Variable ${dollar} mit dem Inhalt "$" und verwendet diese statt des $-Zeichens.

```
; prüfen ob der Ausdruck "[abc][0-9]" auf den String "b3" passt:
exten => s,1,Set(foo=${REGEX("[abc][0-9]" b3)})   ; gibt 1 zurück

; prüfen ob ${str} mit 0 endet, Asterisk 1.4:
exten => s,1,Set(foo=${REGEX("0$" ${str})})

; in Asterisk 1.2 behilft man sich so:
exten => s,1,Set(foo=${REGEX("0${dollar}" ${str})})
```

7.43. SET()

```
SET(Variablenname=Ausdruck)
```

Kann innerhalb von verschachtelten Ausdrücken verwendet werden, um Variablen auf einen Wert zu setzen. (nicht verwechseln mit der Applikation `Set()`!)

```
; ${a}, ${b}, ${c}, und ${d} auf 8 setzen:
Set(a=${SET(b=${SET(c=${SET(d=8)})})})
```

7.44. SHA1()

```
SHA1(String)
```

Berechnet den SHA1-Hash (Digest, Prüfsumme) eines Strings (hexadezimale Rückgabe).

```
; den SHA1-Hash von "Hallo Welt" berechnen:
exten => s,1,Set(sha1hash=${SHA1(Hallo Welt)})
```

7.45. SIPCHANINFO()

```
SIPCHANINFO(Feld)
```

Gibt Daten des aktuellen SIP-Kanals zurück. Gültige Felder sind:

`peerip`	Die IP-Adresse des Peers
`recvip`	Die ursprüngliche IP-Adresse des Peers
`from`	Die URI aus dem `From:`-Header
`uri`	Die URI aus dem `Contact:`-Header
`useragent`	Der User-Agent (Client-Programm)
`peername`	Der Name des Peers

```
; den Namen des Peers abfragen:
exten => s,1,Set(foo=${SIPCHANINFO(peername)})
```

7.46. SIPPEER()

```
SIPPEER(Peername[:Feld])
```

Gibt Informationen über einen SIP-Peer zurück. Gültige Felder sind:

`ip`	(Default) Die IP-Adresse des Peers
`mailbox`	Die eingestellte Mailbox
`context`	Der eingestellte Kontext
`expire`	Die Unix-Zeit (Epoch) wann die Verbindung das nächste Mal ausläuft.
`dynamic`	Ob `dynamic=yes` gesetzt ist (`yes`|`no`)
`callerid_name`	Der eingestellte Caller-ID-Name
`callerid_num`	Die eingestellte Caller-ID-Nummer
`codecs`	Die verfügbaren Codecs
`status`	Der Status (wenn `qualify=yes` gesetzt ist)

`regexten`	Die Registrierungs-Extension (`regexten`)
`limit`	Begrenzung der Anzahl der Anrufe
`curcalls`	Die Anzahl der derzeitigen Anrufe (nur wenn ein Limit gesetzt ist)
`language`	Die Default-Sprache für diesen Peer
`useragent`	Der User-Agent des Peers
`codec[x]`	Der Codec Nummer `x` (0-basiert) in der Reihenfolge der Präferenz
`accountcode`	Die Abrechnungsnummer im CDR für Gespräche mit diesem Peer

```
; beispiel
```

Siehe auch. Abschnitt 7.25, „`IAXPEER()`“

7.47. `SIP_HEADER()`

```
SIP_HEADER(Headername)
```

Gibt einen Header aus dem SIP-Protokoll zurück. Diese Funktion werden Sie nur benötigen, wenn Sie genaue Kenntnis des SIP-Protokolls haben.

```
; den TO-Header abfragen:
exten => s,1,Set(DN=${SIP_HEADER(TO):5})
exten => s,2,Set(DN=${CUT(DN,@,1)})
```

Siehe auch. Abschnitt 6.93, „`SIPAddHeader()`“

7.48. `SORT()`

```
SORT(Schlüssel1:Wert1[,Schlüssel2:Wert2[,...]])
```

Sortiert eine Liste von Schlüsseln und Werten anhand der Werte (als Fließkommazahlen) und gibt eine durch Komma getrennte Liste der Schlüssel zurück.

```
; Liste sortieren:
exten => s,1,Set(foo=${SORT(vier:4|haelfte:.5|hundert:100|pi:3.14|e:2.71828|minuseins:-1)})
; foo ist jetzt "minuseins,haelfte,e,pi,vier,hundert"
```

7.49. `STAT()`

```
STAT(Flag,Dateiname)
```

(ab Asterisk 1.4 verfügbar)

Gibt Status-Informationen über eine Datei zurück (vgl. die Shell-Befehl **test** und **stat**). Mit `Dateiname` ist ein Datei-Knoten („inode“) gemeint, kann also auch ein Verzeichnis oder eine besondere Datei sein. `Flag` ist eines der folgenden:

`d` Prüft, ob die Datei ein Verzeichnis ist

`e` Prüft, ob die Datei existiert

`f` Prüft, ob die Datei eine reguläre Datei ist (im Gegensatz zu besonderen Dateien: block special file, character special file, symbolic link, named pipe, socket)

`m` Gibt den Datei-Modus zurück (oktal), also die Rechte, z.B. 0754

`s` Gibt die Datei-Größe in Bytes zurück

`A` Gibt die Unix-Time des letzten Zugriffs zurück (access)

`C` Gibt die Unix-Time zurück wann sich der Knoten (inode) zuletzt geändert hat (change)

`M` Gibt die Unix-Time der letzten Veränderung zurück (modification)

```
; letzte Veränderung der Datei /etc/crontab abfragen:
exten => s,1,Set(foo=${STAT(M,/etc/crontab)})
```

7.50. `STRFTIME()`

```
STRFTIME([Unixtime][,[Zeitzone][,Format]])
```

Formatiert eine Unix-Zeit (Epoch), umgerechnet in die angegebene Zeitzone, in einem bestimmten Format. Wenn `Unixtime` nicht angegeben wird, gilt die aktuelle Zeit. Der Default der `Zeitzone` ist die des Rechners. Mögliche Zeitzonen finden Sie in `/usr/share/zoneinfo/`. Die Formatierungsplatzhalter sind die wie in der C-Funktion `strftime()` (siehe **man strftime**), der Default ist `%c`, also die Lokale-abhängige Darstellung von Datum und Zeit.

```
; Datum/Zeit im Format JJJJ-MM-TT HH:MM:SS
exten => s,1,set(zeit=${STRFTIME(${EPOCH},Europe/Berlin,"%Y-%m-%d %H:%M:%S")})
```

7.51. `STRPTIME()`

```
STRPTIME(DatumZeit|Zeitzone|Format)
```

(ab Asterisk 1.4 verfügbar)

Wandelt einen formatierten Datums-/Zeit-String unter Berücksichtigung der Zeitzone in einen Unix-Timestamp um.

```
; Datum/Zeit im Format JJJJ-MM-TT HH:MM:SS
exten => s,1,set(zeit=${STRFTIME(${EPOCH},Europe/Berlin,"%Y-%m-%d %H:%M:%S")})
exten => s,1,Set(timestamp=${STRPTIME(${zeit}|Europe/Berlin|%Y-%m-%d %H:%M:%S)}
```

Siehe auch. Abschnitt 7.50, „`STRFTIME()`“

7.52. `TIMEOUT()`

```
TIMEOUT(Art)
```

Liest / setzt einen Timeout auf einem Kanal. Diese Arten von Timeouts sind möglich:

`absolute` Die insgesamte, maximale Zeit, die ein Anruf dauern darf. Beim Erreichen dieser Dauer wird auf die Extension `T` (absolute timeout) weitergeleitet oder aufgelegt. Ein Wert von 0 bedeutet keine Begrenzung. Mit dem Aufruf dieser Funktion wird die vorangehende Einstellung ungültig und neu gesetzt. Der Timeout beginnt zum Zeitpunkt des Aufrufs dieser Funktion, nicht mit dem Beginn des Anrufs.

`digit` Die maximale Zeit, die zwischen dem Eintippen von Ziffern erlaubt ist. Beim Überschreiten wird die Eingabe des Users als beendet angesehen. Wenn dann die Extension nicht existiert, wird auf die Extension `i` (invalid) weitergeleitet oder aufgelegt. Der Default ist 5 Sekunden.

`response` Die maximale Zeit, die auf eine Eingabe vom User gewartet wird. Wenn der User keine Extension tippt, wird auf die Extension `t` (timeout) weitergeleitet oder aufgelegt. Default: 10 Sekunden.

```
; absoluten Timeout abfragen:
exten => s,1,Set(foo=${TIMEOUT(absolute)})

; Anruf-Dauer auf maximal 60 Sekunden begrenzen:
exten => 123,1,Set(TIMEOUT(absolute)=60)
exten => 123,2,Dial(SIP/${EXTEN})
exten => T,1,Playback(tut-uns-leid)
exten => T,2,Playback(auf-wiederhoeren)
exten => T,3,Hangup()
```

7.53. TXTCIDNAME()

```
TXTCIDNAME(Nummer)
```

Schlägt den Namen des Anrufers mittels DNS (im TXT-Record) nach.

```
exten => s,1,Set(foo=${TXTCIDNAME(012345678)})
```

7.54. URIDECODE()

```
URIDECODE(String)
```

Dekodiert einen URI-kodierten String. Siehe URIENCODE().

```
; "www.beispiel.de/?seite=Hallo%20Welt" dekodieren:
exten => s,1,Set(foo=${URIDECODE("Hallo%20Welt")})
; ergibt "Hallo Welt"
```

7.55. URIENCODE()

```
URIENCODE(String)
```

URI-kodiert einen String, d.h., dass Zeichen, die nicht in einem URI vorkommen dürfen, durch Escape-Sequenzen der Form *%XX* ersetzt werden, wobei XX der hexadezimale Wert des Zeichens ist.

```
; "Hallo Welt" kodieren:
exten => s,1,Set(foo=${URIENCODE("Hallo Welt")})
; ergibt "Hallo%20Welt"
```

7.56. VMCOUNT()

```
VMCOUNT(VM-Box[@Kontext][|Ordner])
```

Gibt die Anzahl der Voicemail-Nachrichten in einer Voice-Mialbox zurück. Der Default-Kontext ist default, der Default-Ordner ist INBOX.

```
; Anzahl der Nachrichten in der Mailbox 456 abfragen:
exten => 123,1,Answer()
exten => 123,n,Set(anzahl=${VMCOUNT(456)})
exten => 123,n,Playback(vm-youhave)          ; "Sie haben"
exten => 123,n,GotoIf($[ ${anzahl} = 0 ]?keine:neue)

exten => 123,10(keine),Playback(vm-no)       ; "keine"
exten => 123,n,Goto(weiter)

exten => 123,20(neue),SayNumber($COUNT)      ; Anzahl
exten => 123,n,Goto(weiter)

exten => 123,30(weiter),Playback(vm-INBOX)   ; "neue"
exten => 123,n,Playback(vm-messages)         ; "Nachrichten"
exten => 123,n,Playback(vm-goodbye)          ; "Auf Wiederhören!"
exten => 123,n,Hangup()
```

Siehe auch Abschnitt 6.48, „MailboxExists()“

Kapitel 5. Channels

Ein Channel ist die Verbindung zwischen zwei Punkten (in unserem Fall sind das meistens menschliche Gesprächsteilnehmer). Es gibt folgende Channel-Arten:

- Agent

  Ein ACD Agent-Channel

- CAPI

  Ein ISDN-Channel

- Console

  Ein Linux-Konsolen-Client-Treiber für Soundkarten, die mit OSS oder ALSA angesprochen werden können.

- H.323

  Ein VoIP-Protokoll

- IAX

  Ein VoIP-Protokoll.

  ### Anmerkung

  Prinzipiel gibt es zwei Versionen von IAX (1 und 2). Wer heute von IAX spricht, meint immer IAX2 (also die Version 2).

- Local

  Ein Loopback in einen anderen Context

- MGCP

  Ein VoIP-Protokoll

- mISDN

  Ein ISDN-Channel

- NBS

  Network Broadcast Sound

- phone

  Linux Telephony Channel

- SIP

  Ein VoIP-Protokoll

- Skinny

  Ein VoIP-Protokoll

- vISDN

  Ein ISDN-Channel

- VOFR

  Voice over frame relay Adtran style

- VPB

  Verbindung von normalen Telefonanschlüssen mit Voicetronix-Karten

- Zap

  Verbindung von normalen Telefonanschlüssen mit Digium-Karten. Wird aber auch häufig für Karten anderer Hersteller benutzt.

In den meisten Beispielen in diesem Buch wird immer von SIP-Verbindungen ausgegangen. Der Grund dafür ist einfach: Zurzeit gibt es sehr viel mehr SIP- als z.B. IAX-fähige VoIP-Telefone. In diesem Kapitel werden diese zwei wichtigen Protokolle im Einzelnen beschrieben. Wer will, kann dieses Kapitel aber überspringen und bei konkreten Fragen zu bestimmten Parametern hier nachschlagen.

1. Peers, Users und Friends

Die Unterscheidung der Begriffe Peer, User und Friend ist in der Asterisk-Dokumentation nicht immer leicht verständlich dargestellt und wird daher auch teilweise falsch auf manchen Webseiten wiedergegeben. Die folgende Tabelle zeigt die jeweiligen Funktionen:

Asterisk	<=	User
Asterisk	=>	Peer
Asterisk	<=>	Friend

Ein Peer kann also nur angerufen werden, ein User nur anrufen und ein Friend kann beides.

Tipp

In der Praxis wird meistens nur *`Friend`* benutzt. Entsprechend muss man sich diese Tabelle nicht einprägen, sondern kann sie im Spezialfall nachschlagen.

2. IAX versus SIP

Fast jeder Asterisk-Administrator muss sich irgendwann einmal die Frage stellen, ob er eher auf SIP oder eher auf IAX setzen soll. Die kurze Antwort: Wenn IAX möglich ist (also, wenn die Telefone es unterstützen), dann sollte IAX benutzt werden. Ansonsten immer SIP. Mark Spencer (der Erfinder von Asterisk) hat zu diesem Thema auf einer Asterisk-Mailingliste im Jahr 2004 eine ausführlichere Antwort geschrieben (die original englische E-Mail finden Sie im Anhang C, *IAX vs. SIP*):

```
Date: Mon, 5 Jul 2004 18:59:52 -0500 (CDT)
From: Mark Spencer <markster@digium.com>

Ich möchte einige Unterschiede zwischen SIP und IAX kurz
zusammenfassen. Vielleicht hilft Dir das bei der
Entscheidungsfindung.

1) IAX arbeitet während des Gesprächs unabhängig von der Anzahl
der Anrufe und des verwendeten Codes effizienter als RTP. Der
Vorteil liegt irgendwo zwischen 2400 Kbit/s für einen
Einzelanruf und der dreifachen Anzahl der Anrufe pro Megabit
bei G.729, wenn die Messung bei aktiviertem Trunk-Modus auf der
MAC-Ebene vorgenommen wird.

2) IAX ist nicht ASCII-, sondern datenelementkodiert. Dies macht
Implementierungen wesentlich leichter und zudem robuster
gegenüber Pufferüberlaufangriffen, da absolut keine Textanalyse
```

```
oder -interpretation erforderlich ist. IAX führt den gesamten
IP-Stapel, IAX-Stapel, TDM-Schnittstelle, Echokompensation und
Erzeugung der Anrufer-ID auf 4k Heap und Stack sowie 64k Flash
aus. Dies veranschaulicht ganz klar die
Implementierungseffizienz des Entwurfs. Die Größe der
IAX-Signalpakete ist drastisch geringer als die bei SIP, was
aber in der Regel nur dann erwähnenswert ist, wenn zahlreiche
Clients sich häufig registrieren. Allgemein gesprochen ist IAX2
bei der Kodierung, der Dekodierung und der Überprüfung der Daten
effizienter. Zudem wäre es für den Autor einer
IAX-Implementierung extrem schwierig, eine Inkompatibilität mit
einer anderen Implementierung herzustellen, da für eine
Interpretation kaum Raum vorhanden ist.

3) IAX weist eine sehr klare Trennung von Schicht 2 und Schicht
3 auf, d. h. sowohl Signalisierung als auch Tondaten haben
definierte Zustände, werden robust und in konsistenter Weise
übertragen, und wenn ein Endpunkt des Anrufs unvermittelt
verschwindet, dann wird der Anruf auch zeitnah beendet und zwar
auch dann, wenn keine weiteren Signale und/oder Audiodaten
empfangen werden. Einen solchen Mechanismus weist SIP nicht auf;
hinzu kommt, dass, was die Signalisierung angeht, die
Zuverlässigkeit sehr niedrig und schwerfällig ist, weswegen
zusätzlich zum Kernstandard RF3261 weitere Standards benötigt
werden.

4) Die einheitlichen Signalisierungs- und Audiopfade von IAX
gestatten die transparente Navigation von NATs , und der
Firewall-Administrator muss lediglich einen einzigen Port
öffnen, um den Einsatz von IAX zu gestatten. Der IAX-Client
muss für einen korrekten Betrieb überhaupt nichts über das
Netzwerk wissen, in dem er sich befindet. Anders gesagt: Es kann
niemals eine durch eine Firewall bedingte Situation auftreten,
in der IAX einen Anruf aufbauen und dann keine Audiodaten
übertragen kann (natürlich vorausgesetzt, es ist genügend
Bandbreite vorhanden).

5) Das authentifizierte Übertragungssystem von IAX gestattet
die Übertragung von Audio- und Rufsteuerdaten über einen
zwischengeschalteten Server auf eine robuste Weise: Wenn zwei
Endpunkte einander aus irgendeinem Grund nicht erkennen können,
wird der Ruf über den Zentralserver gehalten.

6) IAX trennt die Caller-ID vom Authentifizierungsmechanismus
des Benutzers. SIP verfügt hierzu über keine eindeutige Methode,
sofern nicht Remote-Party-IDs verwendet werden.

7) SIP ist ein IETF-Standard. Zwar gibt es eine neue
Dokumentation von Frank Miller, aber IAX ist gegenwärtig noch
kein veröffentlichter Standard.

8) IAX ermöglicht es einem Endpunkt, die Gültigkeit einer
Telefonnummer zu überprüfen, damit er weiß, ob die Nummer
vollständig ist, vollständig sein könnte oder aber zwar
vollständig ist, aber länger sein könnte. SIP bietet hierfür
keine vollständige Unterstützung.

9) IAX sendet DTMF stets außerbandig, d. h. es kann keine
Verwirrung bezüglich der Frage entstehen, welche Methode verwendet
wird.

10) IAX unterstützt die Übertragung von Sprache und Kontext, was
in einer Asterisk-Umgebung durchaus sinnvoll ist. Mehr fällt mir
jetzt im Moment nicht ein.

Mark

PS: Ich nehme mal an, dass SIP trotzdem ein paar Vorteile
aufweisen muss (andernfalls wären seine Entwickler ja Dummköpfe).
```

```
Es bleibt also zu fragen, wie IAX die folgenden Aspekte verwaltet:
1) Bandbreitenanzeige
2) Neue Codecs
3) Erweiterbarkeit
4) Parken von Verbindungen und andere komplexe Szenarien
5) Videotelephonie

Ich habe den Eindruck, dass dies alles in SIP besser geregelt ist.
```

Nachtrag zu dieser E-Mail: IAX ist mittlerweile ein offenes und gut dokumentiertes Protokoll.

3. SIP

SIP hat sich de facto zum Standardprotokoll im VoIP-Umfeld etabliert. Warum?[1]

Wahrscheinlich liegt es an folgenden Punkten:

- Es ist ein offenes Protokoll.
- Die Spezifikation war von Anfang an als *RFC 3261* für jeden Entwickler kostenlos einsehbar.

Vom rein technischen Standpunkt ist das IAX-Protokoll dem SIP-Protokoll in einzelnen Aspekten (siehe E-Mail von Mark Spencer) mindestens leicht überlegen. Bei vielen Installationen würde man mit IAX sogar sehr viel weniger Probleme haben, als mit SIP. Ein Grund hierfür ist das NAT-Problem bei SIP, das bei IAX in dieser Form nicht auftritt und das bei der Verwendung von SIP und einer Firewall dazu führt, dass bei der Firewall sehr viele UDP-Ports freigeschaltet werden müssen. Weiterhin können über einen IAX-Channel mehrere Gespräche gleichzeitig geführt werden. Bei SIP muss für jedes Gespräch ein eigener Channel aufgebaut werden (großer Verwaltungsoverhead).

Anmerkung

Das TCP/IP-Protokoll unterscheidet zwei Transportmöglichkeiten: TCP und UDP. TCP unterstützt eine sichere Übertragung von Daten, da nach der Über\-tragung durch Checksummen überprüft wird, ob evtl. ein Paket fehlt. Dieses kann dann noch mal angefordert werden. TCP wird zum Beispiel beim Abruf von Webseiten benutzt. Bei den meisten Übertragungen von Ton- und Bildinformation benötigt man diesen Overhead allerdings nicht. Falls bei einer Videoübertragung ein Bild ausfällt, dann kann man dieses Bild nicht ein paar Sekunden später einfügen. Es wird also einfach weggelassen - Vergleichbares passiert bei Audiostreams. Für die Übertragung von Multimediadaten wird in der Regel UDP verwendet. Es garantiert zwar keine lückenlose Übertragung der Daten, besitzt dafür aber einen geringeren Verwaltungsoverhead und ist in dieser Hinsicht schneller" als TCP.

3.1. Das SIP-NAT-Problem

Die wenigsten Büro- und Privat-PCs haben eine eigene feste IP-Adresse aus dem öffentlichen Adressbereich des Internets. Diese wäre mit dem jetzigen IPv4-Standard auch nicht für jedes TCP/IP-taugliche Gerät verfügbar, da der IP-Adressraum zu klein ist für alle bereits vorhandenen Geräte.

Anmerkung

Der IPv4-Adressraum verfügt über insgesamt 2 hoch 32 einzelne Adressen und man kann wahrscheinlich nicht herausfinden, ob es wirklich noch möglich wäre, jedem aktuell vorhandenen Gerät eine feste IP-Adresse aus dem öffentlichen Adressraum zu geben. Das Problem ist auch, dass die Adressen nicht wahllos vergeben werden können, sondern nur innerhalb sinnvoller Broadcastdomains und bereits ganze Class-A-Bereiche in den Anfängen des Internets großzügig auch einzelnen Firmen zugeteilt wurden.

Eine Lösung dieses Problems ist es Rechner, bzw. gesamte Rechnernetze, über ein so genanntes NAT-Gateway mit dem Internet zu verbinden. NAT ist das Akronym für "Network Address Translation". Mithilfe von NAT teilt ein einzelner Rechner (NAT-Gateway) seine feste offizielle IP-Adresse mit allen verbundenen Rechnern, die in

[1] Das fragen sich viele! ;-)

der Regel über eine IP-Adresse aus dem privaten, im Internet nicht gerouteten Bereich[2], verfügen. Ein NAT-Gateway nimmt alle Anfragen von Rechnern aus dem privaten Netz (meist Intranet genannt) an und leitet diese dann mit der eigenen offiziellen IP-Adresse ins Internet weiter. Kommen die angeforderten Daten aus dem Internet zurück, leitet das NAT-Gateway die Daten entsprechend ins Intranet[3]. TCP/IP-Datenpakete bestehen in der Regel aus einem Envelope (Umschlag) und dem Content/Payload (Inhalt/Nutzlast). Im Envelope stehen Informationen über Ursprung und Ziel des Contents, also auch die IP-Adresse des Rechners, von dem die Kommunikation gestartet wurde. Diese Daten werden vom NAT-Gateway umgeschrieben. Es merkt sich die ursprüngliche IP-Adresse (für die Rückantwort) und schreibt für den öffentlichen Bereich des Internets seine eigene offizielle IP-Adresse in den Envelope. Kommen die angeforderten Daten zurück, schreibt das NAT-Gateway wieder die ursprüngliche IP-Adresse in den Envelope und leitet die Daten entsprechend an den Rechner zurück, der eigentlich die Kommunikaton gestartet hatte. Dies klappt bei den meisten Protokollen sehr gut. Allerdings benutzt SIP nicht den Umschlag, sondern schreibt die für das Protokoll benötigte IP-Adresse in den Inhalt des Paketes. Normale NAT-Gateways können nur mit IP-Adressen im Envelope umgehen, nicht aber im Content. Der gängige Fehler ist nun folgender: Der Zielrechner verfügt über eine offizielle IP-Adresse und der Quellrechner über eine private Adresse. Der anfordernde Rechner adressiert die Pakete mit der offiziellen IP-Adresse und schreibt seine Absenderadresse direkt in den Content und nicht in den Envelope. Die Pakete werden korrekt geroutet und landen beim Zielrechner. Da die Quell-IP nicht im Envelope gesetzt war, konnte das NAT-Gateway die private nicht in eine offizielle IP-Adresse umschreiben. Daher versucht der Zielrechner, die Daten an die in den Paketen angegebene IP-Adresse zu senden. IP-Adressen aus dem privaten Adressbereich werden jedoch nicht geroutet und der Quellrechner erhält keine Daten vom Zielrechner. In der Regel besitzen die Rechner im Intranet eines Unternehmens IP-Adressen aus dem privaten Adressbereich und müssen für die Kommunikation mit Rechnern im Internet auf die Dienste eines NAT-Gateways zurückgreifen.[4] Für Telefone im Intranet ist diese Adressproblematik ohne Belang, da ja im Intranet alle IP-Adressen geroutet und IP-Pakete korrekt verteilt werden. Verbindungen ins Internet sind jedoch mit SIP standardmäßig über ein NAT-Gateway nicht möglich. Eine genaue Beschreibung des NAT-Problems, das auch für einige andere Anwendungen relevant ist, findet man in der Wikipedia unter http://de.wikipedia.org/wiki/Network_Address_Translation.

4. IAX

IAX[5] ist das Inter-Asterisk-eXchange-Protokoll, also das Asterisk-eigene VoIP-Protokoll, das bevorzugt verwendet werden soll.

4.1. Warum IAX?

Braucht die Welt eigentlich *noch* ein VoIP-Protokoll, wo es doch schon SIP, H.323 etc. und eine Reihe proprietärer Protokolle gibt? Darüber lässt sich natürlich streiten. IAX hat aber einige Vorteile, hier die wichtigsten, neben weiteren technischen Details:

- Geringer Overhead. Im Vergleich zu beispielsweise SIP (und dem darin verwendeten RTP) hat IAX als reines UDP-Protokoll ein deutlich besseres Nutzdaten:Overhead-Verhältnis.
- NAT-/Firewall-tauglich. Trotz solcher Hürden kann IAX problemlos Anrufe initiieren und entgegennehmen etc. Es muss nur ein einziger Firewall-Port freigegeben werden (bei SIP mindestens 3).
- Einfaches Protokoll. Es müssen keinerlei Strings geparst werden, was IAX kaum anfällig für Speicherüberläufe macht.
- Gesprächs-Transfers. Gespräche können sowohl über einen zentralen Server laufen als auch direkt zwischen zwei Teilnehmern.

[2] IP-Adressen aus dem privaten Bereich beginnen beispielsweise mit 10. oder mit 192.168., also zum Beispiel 10.128.1.16 oder 192.168.1.3.
[3] Es gibt mittlerweile viele unterschiedliche Formen von NAT-Gateways, mit zum Teil sehr spezieller Funktionsweise. Hier wird lediglich das grundlegende Funktionsprinzip eines NAT-Gateways und die damit verbundenen Auswirkungen beschrieben.
[4] In seltenen Fällen nutzen Firmen trotz der Verwendung von offiziellen IP-Adressen auch für interne Rechner ein NAT-Gateway. Oft, weil sie zum Beispiel die Größe des Netzes nach außen maskieren möchten und sich generell gewisse Sicherheitsvorteile davon erhoffen.
[5] hier ist immer IAX2, also IAX Version 2 gemeint

4.2. Beispiel für eine IAX-Konfiguration

Am Anfang der `iax.conf` steht immer der Eintrag `[general]`. Ähnlich wie bei der `sip.conf` werden im `[general]`-Abschnitt allgemeine Parameter übergeben. Darunter kommt dann die Definition der einzelnen Channels.

Als Beispiel für die Möglichkeiten des IAX-Protokolls verbinden wir zwei voneinander unabhängige Telefonanlagen mit dem IAX-Protokoll. So können Gespräche von der einen zur anderen Anlage geführt werden und müssen nicht über das Festnetz geroutet werden.

4.2.1. Aufgabenstellung

Es gibt zwei Anlagen mit jeweils zwei Telefonen (1000 und 1001). Die Anlagen heißen `ast1` und `ast2`. Jede Anlage soll in der Lage sein, Gespräche an die andere Anlage mit dem IAX-Protokoll zu übermitteln. Dazu wird eine Vorwahl benutzt. Die Vorwahl `0901` verbindet zur Anlage `ast1` und `0902` zur Anlage `ast2`. Die Anlage `ast1` besitzt in diesem Beispiel die IP-Adresse `192.168.0.101` und die Anlage `ast2` die IP-Adresse `192.168.0.102`.

4.2.2. Konfiguration ast1

Die `iax.conf` enthält folgende Konfiguration:

```
[ast2]
type = friend
host = 192.168.0.102
secret = 1234
context = test-telefone
permit = 0.0.0.0/0.0.0.0
```

Die `extensions.conf` enthält folgende Konfiguration:

```
[via-asterisk2]
exten => 09021000,1,Dial(IAX2/ast2/1000)
exten => 09021001,1,Dial(IAX2/ast2/1001)
;          ^   ^              ^     ^
;          |   |              |     |
; virt.Vorwahl ext      Verbindung  ext
```

4.2.3. Konfiguration ast2

Die `iax.conf` enthält folgende Konfiguration:

```
[ast1]
type = friend
host = 192.168.0.101
secret = 1234
context = test-telefone
permit = 0.0.0.0/0.0.0.0
```

Die `extensions.conf` enthält folgende Konfiguration:

```
[via-asterisk1]
exten => 09011000,1,Dial(IAX2/ast1/1000)
exten => 09011001,1,Dial(IAX2/ast1/1001)
```

4.3. Globale Einstellungen

Die folgenden Parameter können nur im `[general]`-Teil definiert werden, sind also gültig für alle Channels. Sie können hier außerdem auch einige der in Abschnitt 4.4, „Channel-Einstellungen“ beschriebenen Einstellungen verwenden.

4.3.1. `bandwidth`

```
bandwidth = [low|medium|high]
```

Dient als Gruppierung verschiedener Soundcodecs. Auf diese Weise kann man einfacher bestimmte Situationen definieren. Das ist eine komfortable Alternative zu `allow` und `disallow` (Abschnitt 4.3.2, „`allow`“).

`high` Erlaubt alle Codecs: G.723.1, GSM, ulaw (G.711), alaw, G.726, ADPCM, slinear, LPC10, G.729, Speex, iLBC. Sollte nur bei Verbindungen ab 10 Mb/s verwendet werden.

`medium` Verbietet die Codecs slinear, ulaw und alaw.

`low` Verbietet zusätzlich die Codecs G.726 und ADPCM.

Beispiel:

```
bandwidth = low
disallow = lpc10   ; hat schlechte Qualität
```

4.3.2. `allow`

```
allow = [all|Name des Codecs]
```

Bestimmte Soundcodecs können gezielt erlaubt (`allow`) oder verboten (`disallow`) werden. Default ist `allow=all`. Die möglichen Codecs sind: `g723.1`, `gsm`, `ulaw`, `alaw`, `g726`, `slinear`, `plc10`, `adpcm`, `g729`, `speex`, `ilbc`, `h261`, `h263` und der Platzhalter `all`. Siehe auch `bandwidth` (Abschnitt 4.3.1, „`bandwidth`“) für eine komfortable Einstellung. `allow` und `disallow` können mehrmals verwendet werden und sind damit eine Ausnahme.

Beispiel:

```
disallow = all
allow = ulaw
allow = gsm
```

4.3.3. `disallow`

```
disallow = [all|Name des Codecs]
```

Siehe `allow` (Abschnitt 4.3.2, „`allow`“).

4.3.4. `codecpriority`

```
codecpriority = [caller|host|disabled|reqonly]
```

Definiert, welcher Teilnehmer einer eingehenden Verbindung die höhere Priorität bei der Verhandlung des Soundcodecs hat. Default: `host`

`caller` Der Anrufer hat Vorrang vor dem Host.

`host` Der Host hat Vorrang vor dem Anrufer.

`disabled` Codec-Präferenzen werden nicht berücksichtigt.

`reqonly` Codec-Präferenzen werden ignoriert und der Anruf nur dann akzeptiert, wenn der angeforderte Codec verfügbar ist.

Beispiel:

```
codecpriority = caller
```

4.3.5. `authdebug`

```
authdebug = [yes|no]
```

Standardmäßig ist ein minimales Debugging bei der Autorisierung von IAX-Verbindungen eingestellt. Dies kann mit `authdebug=no` deaktiviert werden.

Beispiel:

```
authdebug = no
```

4.3.6. autokill

```
autokill = [yes|Timeout in Millisekunden]
```

Falls `autokill` nicht gesetzt ist, versucht Asterisk einen nicht erreichbaren Host sehr lange zu erreichen. Dies kann bei vielen gleichzeitigen Versuchen (auf mehreren Channels) zu Performanceproblemen führen. Mit `autokill=yes` wird die Verbindung nach 2000 Millisekunden abgebrochen. Alternativ kann man auch die Anzahl der Millisekunden angeben.

Beispiel:

```
autokill = 3500
```

4.3.7. amaflags

```
amaflags = [default|omit|billing|documentation]
```

AMA ist die Abkürzung für „Automatic Message Accounting“ und spezifiziert u.a. Standardmechanismen zur Erzeugung und Übermittlung von Anrufprotokollen (Call Data Records / Call Detail Records / CDRs, im Dt. auch Kommunikationsdatensatz / KDS).

Mit `omit` werden keine Aufzeichnungen gemacht.

Beispiel:

```
amaflags = billing
```

4.3.8. bindaddr

```
bindaddr = [IP-Adresse]
```

Definiert die IP-Adresse, auf der Verbindungsanfragen beantwortet werden. Default: `0.0.0.0` für alle Adressen.

Beispiel:

```
bindaddr = 0.0.0.0
```

4.3.9. bindport

```
bindport = [Port-Nummer]
```

Definiert den Netzwerkport, auf dem Verbindungsanfragen beantwortet werden. Der Default-Port für IAX (IAX2!) ist 4569 (IAX Version 1: 5036).

Wichtig

Diese Einstellung muss immer *vor* dem `bindaddr`-Eintrag erfolgen!

Beispiel:

```
bindport = 4569
```

4.3.10. delayreject

```
delayreject = [yes|no]
```

Diese Einstellung kann auf `yes` gesetzt werden, um evtl. Brute-Force-Attacken (zum Passwortknacken) abzuschwächen. Dann wird nach jedem falschen Einlogversuch 1000 Millisekunden (also 1 Sekunde) gewartet, bis der nächste Versuch akzeptiert wird. Default: `no`

Beispiel:

```
delayreject = yes
```

4.3.11. language

```
language = [en|de|...]
```

Konfiguriert die Sprache für den entsprechenden Channel. Gobale Standardsprache ist Englisch. Die Sprache, die gesetzt ist, wird vom Kanal als Informationselement gesendet. Diese Einstellung wird auch von Anwendungen wie `SayNumber()` gelesen, die für verschiedene Sprachen unterschiedliche Sprachbausteine verwenden.

Beispiel:

```
language = de
```

4.3.12. mailboxdetail

```
mailboxdetail = [yes|no]
```

Wenn `mailboxdetail` auf `yes` gesetzt ist, wird dem Apparat des Mailboxbenutzers die Anzahl der neuen (und die der alten) Nachrichten übermittelt (zur Anzeige auf dem Display o.ä.). Ansonsten wird nur signalisiert, dass es neue Nachrichten gibt (ohne die Anzahl zu nennen).

Diese Einstellung ist bei neueren Asterisk-Versionen nicht mehr verfügbar. Mittlerweile verhält sich Asterisk immer (standardkonform) wie `mailboxdetail=yes`

Beispiel:

```
mailboxdetail = yes
```

4.3.13. tos

```
tos = [...]
```

ToS steht für „Type of Service“; das sind Flags im IP-Header, die von manchen Routern ausgelesen und befolgt werden. Damit kann das Routingverhalten optimiert werden. Die Empfehlung dieser Einstellung für IAX ist `tos=ef`. EF steht für Expedited Forwarding, also etwa Express-Übertragung, was nach geringer Latenz, geringer Verlustrate und wenig Jitter verlangt. Default ist `none` (wegen Rückwärts-Kompatibilität).

Die Vielzahl der möglichen Werte soll hier nicht weiter beschrieben werden, Interessierte seien auf die Spezifizierung von Differentiated Services in RFC 2474 und die IANA-DSCP[6] verwiesen. Hier nur ein ins Deutsch übersetzter Auszug aus der Asterisk beiliegenden Datei `doc/README.tos` (1.2) / `doc/ip-tos.txt` (1.4):

> Die zusässigen Werte für die tos*-Einstellungen sind:
>
> `be` (best effort, die normale, geringste Priorität), `cs1`, `af11`, `af12`, `af13`, `cs2`, `af21`, `af22`, `af23`, `cs3`, `af31`, `af32`, `af33`, `cs4`, `af41`, `af42`, `af42`, `ef` (expedited forwarding), `lowdelay` (geringe Latenz), `throughput` (Durchsatz), `reliability` (Zuverlässigkeit), `mincost` (geringste monetäre Kosten), `none` (wie `be`)
>
> Außerdem kann man auch deren numerische Pendants angeben (z.B. `tos=0x18`).
>
> Die Werte `lowdelay`, `throughput`, `reliability`, `mincost` und `none` sind veraltet und sollen nicht mehr verwendet werden, da sie das ToS-Byte nach dem alten „IP precedence“-Modell aus RFC 791 und 1349 setzen.

```
==========================================
Konfig.-         Parameter     Empfohlene
Datei                          Einstellung
------------------------------------------
```

[6] http://www.iana.org/assignments/dscp-registry

```
sip.conf         tos_sip       cs3
sip.conf         tos_audio     ef
sip.conf         tos_video     af41
-----------------------------------------
iax.conf         tos           ef
-----------------------------------------
iaxprov.conf     tos           ef
=========================================
```

> Für den größtmöglichen Nutzen müssen Sie sicherstellen, dass Ihre Netzwerk-Hardware ToS unterstützt (ggf. aktivieren). Für Cisco-Geräte siehe „Enterprise QoS Solution Reference Network Design Guide“[7] , für Linux-Systeme siehe „Linux Advanced Routing & Traffic Control HowTo“[8].
>
> —doc/ip-tos.txt

Beispiel:

```
tos = ef
```

4.3.14. `adsi`

```
adsi = [yes|no]
```

ADSI (Analog Display Services Interface)[9] sind verschiedene Daten-Dienste für analoge Telefone mit einem Display, z.B. blinkende LED / Anzeige im Display bei wartenen Nachrichten etc.

Falls Sie kompatible Telefone haben, können Sie diese Dienst aktivieren.

Beispiel:

```
adsi = yes
```

4.3.15. `register`

```
register => username[:password]@remote-host
```

Ist unser Asterisk-Server nur über eine dynamische IP-Adresse erreichbar (also kein DNS, sondern nur eine IP-Adresse), so muss er sich bei jedem IP-Adresswechsel bei seiner Gegenstelle neu registieren, sonst weiß die Gegenstelle nicht, an welche IP-Adresse sie Gespräche durchstellen soll. Dazu wird `register =>` verwendet.

Beachten Sie, dass `register`-Angaben nur benutzt werden, wenn das entfernte Ende Sie als Peer und `host=dynamic` eingestellt hat.

Das grundlegende Format einer Registerangabe sieht wie folgt aus:

```
register => username[:password]@remote-host
```

Alternativ können Sie durch die Einbettung des Namens eines geeigneten RSA-Schlüssels[10] in eckige Klammern (`[]`) einen RSA-Schlüssel spezifizieren:[11]

```
register => username:[rsa-key-name]@remote-host
```

Standardmäßig werden Registeranfragen über Port 4569 gesendet. Sie können aber durch Anhängen von *`:Port-nummer`* (also z.B. `:4444`) an den Hostnamen explizit einen anderen angeben.

[7] http://www.cisco.com/application/pdf/en/us/guest/netsol/ns432/c649/ccmigration_09186a008049b062.pdf
[8] http://www.lartc.org/
[9] wird per FSK (Frequency Shift Keying, dt.: Frequenzumtastung) übertragen
[10] Asterisk-RSA-Schlüssel finden sich normalerweise in `/var/lib/asterisk/keys/`. Sie können mittels des **astkey**-Skripts eigene Schlüssel generieren.
[11] Die eckigen Klammern sind hier Pflicht und nicht wie im Rest des Buches als Optionsangabe zu lesen.

4.4. Channel-Einstellungen

Die folgenden Parameter können für die einzelnen Channel definiert werden. Einige können aber auch im Abschnitt `[general]` verwendet werden.

4.4.1. `type`

```
type = [user|peer|friend]
```

Legt fest, ob es sich um eine Verbindung zu einem `user` (z.B. Benutzer-Endgerät, verbindet sich mit uns) oder `peer` (z.B. Gateway in ein anderes Asterisk-Netzwerk, zu dem wir die Verbindung herstellen) handelt. `friend` ist eine Kurzschreibweise um einen `user` und einen `peer` mit den gleichen Angaben zu definieren.

Beispiel:

```
type = peer
```

Warnung

Wenn Sie nicht für alle User-Einträge (`type=user`) IP-basierte Zugangsbeschränkungen (siehe Abschnitt 4.4.11, „`permit`“) definiert haben, müssen Sie in der iax.conf einen Eintrag `[guest]` haben, bei dem Sie weder `auth` noch `secret` angeben. Andernfalls können sich beliebige User verbinden, indem sie ein Passwort erraten.

Beispiel:

```
[guest]
type=user
callerid="IAX-Gast-Benutzer"
```

4.4.2. `accountcode`

```
accountcode = [Abrechnungsnummer]
```

Der angegebene String wird als Dateiname für die Abrechnungsdateien im Verzeichnis `/var/log/asterisk/cdr-csv/` benutzt. Verwenden Sie daher nur Kleinbuchstaben, Ziffern, Bindestrich, Unterstrich.

Beispiel:

```
accountcode = iax-hausmeister
```

4.4.3. `bandwidth`

```
bandwidth = [low|medium|high]
```

`bandwidth` dient als Gruppierung verschiedener Soundcodecs. Auf diese Weise kann man einfacher bestimmte Situationen definieren (siehe Abschnitt 4.3.1, „`bandwidth`“).

Beispiel:

```
bandwidth = low
```

4.4.4. `allow`

```
allow = [all|Name des Codecs]
```

Bestimmte Soundcodecs können gezielt erlaubt (`allow`) oder verboten (`disallow`) werden (siehe Abschnitt 4.3.2, „`allow`“). Dabei ist zu beachten, das evtl. im `[general]`-Bereich erlaubte Codecs erst durch ein `disallow=all` wieder verboten werden müssen.

Diese Parameter sind nicht zu verwechseln mit `permit` und `deny` (siehe Abschnitt 4.4.11, „`permit`“).

Beispiel:

```
disallow = all
allow = ulaw
allow = gsm
```

4.4.5. `disallow`

```
disallow = [all|Name des Codecs]
```

Siehe `allow` (Abschnitt 4.4.4, „`allow`“).

4.4.6. `codecpriority`

```
codecpriority = [caller|host|disabled|reqonly]
```

Definiert, welcher Teilnehmer der Verbindung die höhere Priorität bei der Verhandlung des Soundcodecs hat (siehe Abschnitt 4.3.4, „`codecpriority`“).

Beispiel:

```
codecpriority = caller
```

4.4.7. `amaflags`

```
amaflags = [default|omit|billing|documentation]
```

AMA ist die Abkürzung für „Automatic Message Accounting“ und spezifiziert u.a. Standardmechanismen zur Erzeugung und Übermittlung von Anrufprotokollen (CDRs) (siehe Abschnitt 4.3.7, „`amaflags`“).

Beispiel:

```
amaflags = documentation
```

4.4.8. `callerid`

```
callerid = [Name[ <Nummer>]]
```

Mit `callerid` können Sie die angezeigte Caller-ID für einen Nutzer oder ein Peer setzen. Wenn Sie ein Caller-ID-Feld für einen Benutzer definieren, werden alle Anrufe, die auf diesem Kanal eingehen, mit dieser Caller-ID verknüpft, unabhängig davon, was Ihnen das andere Ende sendet. Das ist aus Sicherheitsgründen zu empfehlen (außer, Sie vertrauen dem anderen Ende), da es sehr leicht möglich ist, seine Caller-ID zu fälschen.

Beispiel:

```
callerid = "Mark Spencer" <(256) 428 6000>
```

4.4.9. `host`

```
host=[Host|dynamic]
```

Als `host` wird entweder der Hostname oder die IP-Adresse des anderen Rechners angegeben. Bei Hosts mit dynamisch vergebenen (also wechselnden) IP-Adressen verwenden Sie den Sonderfall `host=dynamic`, vorzugsweise in Kombination mit der Einstellung `defaultip` (Abschnitt 4.4.10, „`defaultip`“).

Beispiele:

```
host = 192.168.0.201
```

```
host = dynamic
```

4.4.10. `defaultip`

```
defaultip = [IP-Adresse]
```

Die `defaultip`-Einstellung ergänzt `host=dynamic` (siehe Abschnitt 4.4.9, „`host`“). Ist ein Host noch nicht bei Ihrem Server angemeldet, versucht Asterisk, Nachrichten an die hier definierte Standard-IP-Adresse zu schicken.

Beispiel:

```
defaultip = 192.168.0.201
```

4.4.11. `permit`

```
permit = IP-Adresse[/Netzmaske]
```

`permit` und `deny` erlauben und beschränken die IP-Adressen, von denen Verbindungen zum entsprechenden Eintrag möglich sind. Üblicherweise wird zuerst ein `deny=0.0.0.0/0.0.0.0` (alle verbieten) vorangestellt und dann mit `permit` die erlaubte IP-Adresse oder ein Bereich von Adressen freigegeben. Die Reihenfolge von `deny` und `permit` ist entscheidend, und beide können für mehrere Hosts/Bereiche auch mehrfach auftreten. Wenn genau ein Host gemeint ist, können Sie die Netzmaske (wäre `255.255.255.255`) weglassen.

Dieses Beispiel erlaubt nur Verbindungen aus dem Bereich `192.168.0.*` (Netzmaske 24 Bits, Klasse C Netzwerk):

```
deny = 0.0.0.0/0.0.0.0                ; alle verbieten
permit = 192.168.0.102/255.255.255.0  ; 192.168.0.* erlauben
permit = 192.168.5.5                  ; 192.168.5.5 erlauben
```

Alle erlauben bis auf `192.168.*.*`:

```
permit = 0.0.0.0/0.0.0.0              ; alle erlauben
deny = 192.168.0.0/255.255.0.0        ; 192.168.*.* verbieten
```

Dieses Beispiel ist Blödsinn, da zuerst `192.168.0.0` - `192.168.0.127` verboten, dann aber wieder *alle* erlaubt werden, was die erste Regel überschreibt. So wird also nichts verboten:

```
deny = 192.168.0.0/255.255.255.127
permit = 0.0.0.0/0.0.0.0              ; alle erlauben
```

4.4.12. `deny`

```
deny = [IP-Adresse]/[Netzmaske]
```

Siehe `permit` (Abschnitt 4.4.11, „`permit`“).

Beispiel:

```
deny = 0.0.0.0/0.0.0.0
```

4.4.13. `auth`

```
auth = [plaintext|md5|rsa]
```

Zum Identifizieren des anderen Peers/Users stehen 3 Methoden, zur Auswahl: `plaintext`, `md5` und `rsa`.

`plaintext` — Sehr unsicher, da das Passwort im Klartext übertragen wird. Sollte daher nicht verwendet werden. Das Passwort wird mit `secret` (Abschnitt 4.4.14, „`secret`“) angegeben.

`md5` — In diesem Challenge/Response-Verfahren wird eine MD5-Prüfsumme übertragen. Das Passwort wird mit `secret` (Abschnitt 4.4.14, „`secret`“) angegeben.

`rsa` — Das sicherste Verfahren durch öffentliche und private Schlüssel. Diese Schlüssel werden bei `inkeys` (Abschnitt 4.4.15, „`inkeys`“) und `outkeys` (Abschnitt 4.4.16, „`outkey`“) angegeben.

Mit `auth` legen Sie eine durch Komma getrennte Liste von erlaubten Authentifizierungsmethoden fest.

Beispiel:

```
auth = rsa,md5
```

Wenn Sie weder `auth` noch `secret` angeben, bedeutet das, dass keine Authentifizierung notwendig ist.

4.4.14. `secret`

```
secret = [Passwort]
```

Legt das Passwort für die Authentifizierungsmethoden `plaintext` oder `md5` fest (siehe Abschnitt 4.4.13, „`auth`“).

Beispiel:

```
secret = meinpasswort
```

4.4.15. `inkeys`

```
inkeys = [rsa-key:rsa-key:...]
```

Legt für die Authentifizierungsmethode `rsa` (siehe Abschnitt 4.4.13, „`auth`“) den öffentlichen RSA-Schlüssel des zu authentifizierenden Systems fest. Um mehr als einen RSA-Schlüssel mit einer Benutzer-Kanaldefinition zu verknüpfen, trennen Sie die Schlüsselnamen mit einem Doppelpunkt (`:`). Jeder dieser Schlüssel wird dann eine Verbindung für gültig erklären können.

Diese Schlüssel müssen als Dateien so benannt sein: `/var/lib/asterisk/keys/`*`schluesselname`*`.pub`

Beispiel:

```
inkeys = server-koblenz:server-bonn
```

4.4.16. `outkey`

```
outkey = [rsa-key]
```

Sie können die `outkey`-Option verwenden, um sich bei einem Peer mit Hilfe eines RSA-Schlüssels zu authentifizieren. Für ausgehende Authentifizierung kann nur ein RSA-Schlüssel verwendet werden. Der `outkey` wird nicht verteilt, er ist Ihr privater Schlüssel und sollte 3DES-verschlüsselt sein.

Diese Schlüssel müssen als Dateien so benannt sein: `/var/lib/asterisk/keys/`*`schluesselname`*`.key`

Beispiel:

```
outkey = privater-schluessel
```

4.4.17. `mailbox`

```
mailbox = [Mailboxname[@Mailbox-Context]]
```

Wenn Sie in einer Kanaldefinition einen Peer mit einer Mailbox verknüpfen, wird Voicemail eine MWI (Message Waiting Indication)-Nachricht an die Knoten am Ende dieses Kanals schicken. Befindet sich die Mailboxnummer in einem anderen Voicemail-Kontext als `default`, können Sie sie als *`mailbox@context`* angeben. Um mehrere Mailboxen mit einem einzigen Peer zu verknüpfen, müssen Sie den `mailbox`-Befehl mehrmals verwenden.

Beispiel:

```
mailbox = 1000
```

4.4.18. `language`

```
language = [en|de|...]
```

Konfiguriert die Sprache für den entsprechenden Channel. Die Standardsprache ist Englisch. Diese Einstellung wird von verschiedenen Anwendungen wie z.B. der Voicemailbox oder auch der Applikation `SayNumber()` verwendet, um verschiedene Sprachbausteine zu benutzen.

Beispiel:

```
language = de
```

4.4.19. `context`

```
context = [Kontext]
```

Der Kontext dem Benutzer dieser Verbindung zugeordnet werden.

Beispiel:

```
context = default
```

4.4.20. `regcontext`

```
regcontext = [Kontext]
```

Sie können einen Kontext definieren, der automatisch beim ersten Registrieren eines Peers ausgeführt wird. Ist mit `regexten` keine Extension angegeben, so wird als auszuführende Extension der Name des Peers benutzt. Dabei ist zu beachten, dass Asterisk als erstes automatisch ein `NoOp` mit der Priorität 1 ausführt. Als erste Priorität danach ist also die 2 zu benutzen. Mehrere Kontexte können nacheinander aufgeführt werden; als Trennungszeichen wird ein `&` benutzt.

Beispiel:

```
regcontext = from-iax
```

4.4.21. `regexten`

```
regexten = [Nummer]
```

Wird in Verbindung mit `regcontext` verwendet, um die Extension zu spezifizieren, die in dem konfigurierten Kontext ausgeführt werden soll. Falls `regexten` nicht explizit konfiguriert ist, wird der Name des Peers als Extension verwendet.

Beispiel:

```
regexten = 3000
```

4.4.22. `jitterbuffer`

```
jitterbuffer = [yes|no]
```

Schaltet den Jitter-Buffer an oder aus (dies gilt bei Verbindungen, bei denen Asterisk als Endgerät fungiert). Der Jitter-Buffer gleicht unterschiedliche Laufzeiten der IP-Pakete aus.

Der Jitter-Buffer hat evtl. Probleme bei eingeschaltetem `trunk` (siehe Abschnitt 4.4.26, „`trunk`“).

Beispiel:

```
jitterbuffer = yes
```

4.4.23. `forcejitterbuffer`

```
forcejitterbuffer = [yes|no]
```

Normalerweise ist es Aufgabe der Endgeräte einen Jitter-Buffer [12] zu realisieren. Sollte dies aus irgendeinem Grund nicht oder nur schlecht funktionieren (manche Geräte haben einen schlechten Jitter-Buffer), kann man diesen Buffer auch auf Asterisk als VoIP-Brücke aktivieren.

Beispiel:

```
forcejitterbuffer = yes
```

4.4.24. `maxjitterbuffer`

```
maxjitterbuffer = [Laenge in Millisekunden]
```

Mit diesem Parameter wird die maximale Größe des Jitter-Buffers eingestellt. Sie sollten diesen Wert i.d.R. nicht höher als 500 einstellen, sonst kann es zu Tonaussetzern kommen.

Beispiel:

```
maxjitterbuffer = 400
```

4.4.25. `resyncthreshold`

```
resyncthreshold=[Wert in Millisekunden]
```

Der Resynchronisierungs-Grenzwert wird benutzt, um den Jitter-Buffer, im Falle der Erkennung einer signifikanten Änderung über wenige Frames, neu zu takten/neu abzustimmen unter der Annahme, dass der Veränderung eine Verwechslung von Zeitstempeln zugrunde liegt. Der Resynchronisierungs-Grenzwert ist definiert als die gemessene Schwankung plus dem Wert von `resyncthreshold`, der in Millisekunden angegeben wird.

Beispiel:

```
resyncthreshold = 800
```

4.4.26. `trunk`

```
trunk = [yes|no]
```

Asterisk kann bei IAX-Verbindungen zu einer bestimmten IP-Adresse (z.B. beim Verbinden von zwei Asterisk-Servern) alle Channels über einen Trunk versehen. Dadurch wird der Overhead von mehreren Verbindungen zu einer Adresse reduziert. Der Trunk-Modus funktioniert nur mit geeigneter Hardware (Digium Zaptel) oder entsprechender Emulationssoftware für timing-Zwecke[13].

Bei der Version 1.2.x arbeitet der Jitter-Buffer (siehe Abschnitt 4.4.22, „`jitterbuffer`“) nicht gut bei eingeschaltetem Trunk-Modus.

Beispiel:

```
trunk = yes
```

4.4.27. `trunkfreq`

```
trunkfreq = [Wiederholrate in Millisekunden]
```

Mit `trunkfreq` wird in Millisekunden angegeben, wie oft Trunk-Nachrichten gesendet werden. Gilt nur, wenn `trunk=yes` gesetzt wurde.

Beispiel:

```
trunkfreq = 20
```

[12] Jitter bezieht sich auf die variierende Latenz zwischen den Datenpaketen. Bei zu großen Schwankungen kann es zu Tonausfällen kommen.
[13] z.B. das bei Asterisk enthaltene Kernel-Modul `ztdummy` oder `zaprtc` von Junghanns (http://www.junghanns.net/downloads/)

4.4.28. `qualify`

```
qualify = [yes|no|Zeit in Millisekunden]
```

Mit `qualify=yes` werden in festen Zeitabständen Ping-Nachrichten an die entfernten Peers gesendet, um herauszufinden, ob sie verfügbar sind und welche Latenz zwischen den Antworten liegt. Die Peers antworten mit Pong-Nachrichten. Ein Peer wird als nicht erreichbar angesehen, wenn innerhalb von 2000 ms noch keine Antwort vorliegt (um diese Voreinstellung zu ändern, setzen Sie `qualify` auf die entsprechende Anzahl an Millisekunden, die auf eine Anwort gewartet werden soll). Einige Endgeräte (/Software) kommen nicht mit diesen Nachrichten zurecht.

Beispiel:

```
qualify = yes
```

4.4.29. `qualifysmoothing`

```
qualifysmoothing = [yes|no]
```

Stellt ein, ob für die Verfügbarkeitsprüfung durch `qualify` immer der Mittelwert der letzten beiden Pong-Nachrichten verwendet werden soll. Hosts mit schlechter Verbindung könnten sonst fälschlicherweise als nicht verfügbar eingestuft werden.

Beispiel:

```
qualifysmoothing = yes
```

Kapitel 6. Voicemailsystem

Versionsgeschichte
Version Rechtsschreiblektorat 23.12.2006

1. Einleitung

In diesem Kapitel beschäftigen wir uns mit den Möglichkeiten des eingebauten Voicemailsystems von Asterisk. Dies ist nicht zu verwechseln mit einem Interactive Voice Response System (IVR), dem Thema von Kapitel 8, *Interactive Voice Response Systeme (IVR)*.

Da jede Telefonanlage einen Anrufbeantworter oder Neudeutsch ein Voicemailsystem[1] benötigt, haben die Asterisk-Entwickler ein solches System als Grundausstattung eingebaut.[2].

2. Beispielanwendungen

Die folgenden Beispiele geben einen schnellen Überblick über typische Konstellationen. Anschließend (in Abschnitt 3, „Applikationen“) wird auf die Besonderheiten und Möglichkeiten von Voicemailsystemen unter Asterisk eingegangen.

2.1. Voicemailbox der Familie Meier

Die Familie Meier stellt im Zuge der Modernisierung des Haushaltes auf eine Asterisk-Telefonanlage um. Dies beinhaltet eine moderne Voicemailbox für alle Familienmitglieder.

2.1.1. Aufgabenstellung

Folgende Personen müssen mit einer Voicemailbox ausgerüstet werden:

Name	Extension	Bemerkung
Horst Meier	200	Normale Voicemailbox
Eva Meier	201	Normale Voicemailbox
Peter Meier	202	Normale Voicemailbox und Benachrichtigung per E-Mail, die Sprachnachricht wird der E-Mail beigefügt
Lisa Meier	203	Normale Voicemailbox, Benachrichtung per E-Mail (Sprachnachricht wird der E-Mail beigefügt) und danach automatische Löschung der Voicemail im System

2.1.2. Lösung

Die `voicemail.conf` sieht in diesem Fall wie folgt aus:

```
[general]
; Die Sprachnachrichten werden im GSM Format gespeichert,
; um Platz zu sparen.
format = gsm
attach = yes

[default]
; Die Syntax fuer die Eintraege lautet:
; MailboxNr => Passwort,Name,E-Mail,Pager,Optionen
; (wobei die MailboxNr oft der Extension entspricht)
200 => 1234,Horst Meier
201 => 1234,Eva Meier
```

[1] In diesem Buch wird das Wort Voicemailsystem oder kurz Voicemail benutzt anstelle von Anrufbeantworter. Ein „Anrufbeantworter“ ist nach allgemeinem Verständnis eher ein schwarzer Kasten mit einem Kassettenrekorder neben dem Telefon. Ein Voicemailsystem bietet viel mehr Möglichkeiten als ein Anrufbeantworter im herkömmlichen Sinne.
[2] Das Voicemailsystem wurde übrigens von den Asterisk-Entwicklern scherzhaft *Comedian Voicemail* genannt und meldet sich im englischen Original auch so.

```
202 => 1234,Peter Meier,peter@die-familie-meier.de
203 => 1234,Lisa Meier,lisa@die-familie-meier.de,,delete=yes
```

Und in der `extensions.conf` wird auf die Voicemailbox wie folgt verwiesen:

```
[familie-meier]
; Wenn nach 80 Sekunden niemand abnimmt, geht das Gespraech
; auf die Voicemailbox, das gleiche bei "besetzt".
exten => 20[0-3],1,Dial(SIP/${EXTEN},80,j)
exten => 20[0-3],2,VoiceMail(u${EXTEN})
exten => 20[0-3],102,VoiceMail(b${EXTEN})

; auf der Extension 250 kann jeder Teilnehmer mit seinem
; Telefon seine persoenliche Voicemailbox abrufen.
exten => 250,1,VoiceMailMain(${CALLERID(num)})
```

2.2. Die Apfelmus GmbH

Die in ??? vorgestellte Apfelmus GmbH benötigt ein umfangreicheres Voicemailsystem. Wir bedienen uns dieses Szenarios, um die Möglichkeiten des Voicemailsystems näher zu erläutern.

2.2.1. Aufgabenstellung

Folgende Einstellungen sollen für jede Voicemailbox als Grundeinstellung vorgegeben werden:

- Sprachnachrichten werden im WAV-Format abgespeichert.
- Jede Voicemailbox kann maximal 200 Nachrichten abspeichern.
- Eine Sprachnachricht darf maximal 5 Minuten lang sein.
- Sprachnachrichten werden auf dem System gespeichert und dem Nutzer als Attachment per E-Mail zugestellt.

Einzelne Abteilungen haben darüber hinaus noch zusätzliche Wünsche und Bedürfnisse für ihre Voicemailboxen:

Voicemailbox	**Funktion**	**Bemerkung**
150	Hausmeister	• Eine Benachrichtigung wird nur per Pager zugestellt und nicht per E-Mail. • Anrufer dürfen die Sprachnachricht vor dem Abspeichern noch einmal anhören und ggfs. ändern. • Nach dem Abhören der Nachricht kann der Anrufer aus dem Menü zurückgerufen werden.
160 - 169	IT	• In der IT-Abteilung bekommt jeder Mitarbeiter eine eigene Voicemailbox. • Es wird nur dann auf die Voicemailboxen umgeleitet, wenn kein Mitarbeiter ans Telefon geht. Bei besetztem Anschluss kann hingegen keine Nachricht hinterlassen werden.[a]
802	Verkauf Inland	• Es wird keine E-Mail verschickt.[b] • Bei der Abfrage der Voicemailbox muss kein Passwort eingegeben werden.
803	Verkauf Ausland	• Es wird keine E-Mail verschickt. • Bei der Abfrage der Voicemailbox muss kein Passwort eingegeben werden.
201	Geschäftsführer 1	• Anrufer dürfen die Sprachnachricht vor dem Abspeichern noch einmal anhören und ggfs. ändern. • Nach dem Abhören der Nachricht kann der Anrufer aus dem Menü zurückgerufen werden.

Voicemailbox	Funktion	Bemerkung
202	Geschäftsführer 2	• Anrufer dürfen die Sprachnachricht vor dem Abspeichern noch einmal anhören und ggfs. ändern. • Nach dem Abhören der Nachricht kann der Anrufer aus dem Menü zurückgerufen werden.
804	Sekretariat	• Anrufer dürfen die Sprachnachricht vor dem Abspeichern noch einmal anhören und ggfs. ändern.

[a]Dies ist eine Vorsichtsmaßnahme, die einen Denial-Of-Service im Falle eines großen IT-Problems verhindern soll. Wenn alle Mitarbeiter eine Nachricht auf dem Voicemailsystem hinterlassen würden, dann käme die IT-Abteilung mit dem Abhören dieser Nachrichten nicht mehr nach.
[b]In der Verkäufergruppe sind mehrere Personen, es soll daher nicht eine E-Mail an eine bestimmte Einzelperson geschickt werden. Man könnte aber die Funktion vieler SIP-Telefone nutzen, mit einem Blinklicht anzuzeigen, ob eine Nachricht in der Voicemailbox vorhanden ist.

2.2.2. Lösung

Die `voicemail.conf` der Apfelmus GmbH sieht wie folgt aus:

```
[general]
; Die Sprachnachrichten werden im qualitativ hochwertigen
; WAV Format gespeichert.
format = wav

; Die E-Mails mit den Sprachnachrichten fuer die Nutzer erhalten
; als Absender: voicemailsystem@apfelmus-gmbh.de
serveremail = voicemailsystem@apfelmus-gmbh.de

; Es duerfen maximal 200 Nachrichten pro Mailbox
; gespeichert werden.
maxmsg = 200

; Die maximale Laenge einer Sprachnachricht ist 5 Minuten
maxmessage = 300

; Text fuer die Benachrichtigungs-E-Mail.
; Muss in einer Zeile stehen!
emailbody = Guten Tag ${VM_NAME},\n\nSie haben eine neue Nachricht von ${VM_CALLERID} in Ihrer Voicemailb

; Text fuer die Benachrichtigung per Pager.
; Muss in einer Zeile stehen!
pagerbody = Neue Voicemail von ${VM_CALLERID} um ${VM_DATE}.

; Voice-Nachrichten anhaengen:
attach = yes

[default]
; Die Syntax fuer die Konfigurationszeilen ist:
; MailboxNr => Passwort,Name,E-Mail,Pager,Optionen
150 => 1234,Hans Hausmeister,,pager.hausmeister@apfelmus-gmbh.de,review=yes|callback=interne-gespraeche
802 => 1234,Verkauf Inland
803 => 1234,Verkauf Ausland
201 => 1234,Hans Wichtig,geschaeftsfuehrer1@apfelmus-gmbh.de,,review=yes|callback=interne-gespraeche
202 => 1234,Uwe Wichtig,geschaeftsfuehrer2@apfelmus-gmbh.de,,review=yes|callback=interne-gespraeche
804 => 1234,Sekretariat,sekr@apfelmus-gmbh.de,,,review=yes
```

Und in der `extensions.conf` wird auf die Voicemailbox wie folgt verwiesen:

```
[hausmeister]
include => interne-gespraeche
include => voicemailsystem-hausmeister

[it]
include => interne-gespraeche
include => voicemailsystem-komfort
include => voicemailsystem-normal
```

```
[geschaeftsfuehrer]
include => interne-gespraeche
include => voicemailsystem-komfort

[sekretariat]
include => interne-gespraeche
include => voicemailsystem-komfort

[verkauf-national]
include => interne-gespraeche
include => voicemailsystem-verkauf-national

[verkauf-ausland]
include => interne-gespraeche
include => voicemailsystem-verkauf-ausland

[versand]
include => interne-gespraeche
include => voicemailsystem-komfort

[produktion]
include => interne-gespraeche
include => voicemailsystem-komfort

[sonstige]

[interne-gespraeche]
; Wenn der Hausmeister nach 60 Sekunden nicht
; ans Telefon geht, wird das Telefonat auf die
; Voicemailbox 150 geleitet.
exten => _15X,1,Dial(SIP/${EXTEN},60)
exten => _15X,2,VoiceMail(u150)
exten => _15X,102,VoiceMail(b150)

; Die IT hat normale Voicemailboxen.
exten => _16X,1,Dial(SIP/${EXTEN},60)
exten => _16X,2,VoiceMail(u${EXTEN})

; Die beiden Geschaeftsfuehrer haben eine
; eigene Voicemailbox.
exten => _20[1-2],1,Dial(SIP/${EXTEN},120)
exten => _20[1-2],2,VoiceMail(u${EXTEN})
exten => _20[1-2],102,VoiceMail(b${EXTEN})

; Das Sekretariat hat eine Gruppenmailbox.
exten => _2[3-6]X,1,Dial(SIP/${EXTEN},120)
exten => _2[3-6]X,2,VoiceMail(u804)
exten => _2[3-6]X,102,VoiceMail(b804)

; Der Verkauf National hat eine Gruppenmailbox
exten => _3[0-4]X,1,Dial(SIP/${EXTEN},80)
exten => _3[0-4]X,2,VoiceMail(u802)
exten => _3[0-4]X,102,VoiceMail(b802)

; Der Verkauf Ausland hat eine Gruppenmailbox
exten => _3[5-9]X,1,Dial(SIP/${EXTEN},80)
exten => _3[5-9]X,2,VoiceMail(u803)
exten => _3[5-9]X,102,VoiceMail(b803)

; Alle anderen erhalten keine Voicemailbox.
exten => _[4-5]XX,1,Dial(SIP/${EXTEN},120)


[voicemailsystem-komfort]
exten => 800,1,VoiceMailMain(${CALLERID(num)})

[voicemailsystem-normal]
exten => 801,1,VoiceMailMain()

[voicemailsystem-hausmeister]
```

```
exten => 800,1,VoiceMailMain(150)

[voicemailsystem-verkauf-national]
exten => 800,1,VoiceMailMain(s802)

[voicemailsystem-verkauf-ausland]
exten => 800,1,VoiceMailMain(s803)
```

2.2.3. Besonderheiten

Die `extensions.conf` der Apfelmus GmbH sieht jetzt schon etwas aufwendiger aus. Das liegt daran, dass wir verschiedene Voicemailboxarten verwenden. Es gibt einmal die normale Voicemailbox und dann noch die Gruppen-Voicemailboxen für den Verkauf und das Sekretariat. Weiterhin sollen die Mitarbeiter des Verkaufs ihre Voicemailboxen ohne Passwort-Eingabe abrufen können. Das Ganze soll natürlich für den einzelnen Mitarbeiter möglichst komfortabel abgebildet werden. Aus diesem Grund bleibt für alle User des Systems die `800` die Rufnummer der Voicemailbox, unter der sie dieselbige abrufen können.

3. Applikationen

Es gibt zwei Voicemail-Applikationen, die in der `extensions.conf` aufgerufen werden können:

`VoiceMail()` — Dieses Programm leitet einen Anrufer an das Voicemailsystem weiter, und dort wird er aufgefordert, eine Nachricht zu hinterlassen.

`VoiceMailMain()` — VoiceMailMain ist das Abfrageprogramm für die Nutzer der Telefonanlage. Mit diesem Programm können Nutzer selbst Ansagen aufnehmen und Nachrichten abhören.

3.1. `VoiceMail()`

Funktion: Der Anrufer kann eine Nachricht auf dem Voicemailsystem hinterlassen.

Die Applikation `VoiceMail()` wird immer aus der Datei `extensions.conf` aufgerufen. Beispiel:

```
exten => 2000,2,VoiceMail(u2000)
```

3.1.1. Syntax

```
VoiceMail([u|b|s]mailbox[@context])
```

`[u|b|s]`

- `u` veranlasst, dass die „unavailable“ (nicht verfügbar) Nachricht abgespielt wird. Der Dateiname im Filesystem ist `/var/lib/asterisk/sounds/vm-isunavail.gsm`[3]
- `b` veranlasst, dass die „busy“ (besetzt) Nachricht abgespielt wird. Der Dateiname im Filesystem ist `/var/lib/asterisk/sounds/vm-rec-busy.gsm`.
- `s` unterdrückt das Abspielen einer Nachricht (Sounddatei) und startet unmittelbar die Aufnahme.

`mailbox` — Das ist der Name der Mailbox (Ziffern). Er muss nicht identisch mit der entsprechenden Extension sein. Dies ist aber aus praktischen Gründen sinnvoll, da die Konfiguration dadurch übersichtlicher wird.

`@context` — Eine Mailbox kann in einem bestimmten Context ausgeführt werden. Ist kein Context angegeben, so wird der `[default]`-Context verwendet.

Wenn der Anrufer während der Ansage die `0` (Null) drückt, dann springt die Applikation zu der Extension `o` (Buchstabe o kleingeschrieben) in diesem Context.
Wenn der Anrufer während der Ansage den `*` (Stern) drückt, dann springt die Applikation zu der Extension `a` (Buchstabe a kleingeschrieben) in diesem Context.

[3] Falls Sie ein fertiges Asterisk-Paket verwenden, kann der Pfad z.B. auch `/usr/share/asterisk/sounds/` sein.

Falls keine Mailbox in der Datei `voicemail.conf` definiert ist, aber ein Eintrag mit der Priorität n+101 existiert, dann wird dieser angesprungen und ausgeführt.

3.2. `VoiceMailMain()`

Funktion: Stellt Nutzern den Zugang zum Voicemailsystem her.

Die Funktion `VoiceMailMain()` wird immer aus der Datei `extensions.conf` aufgerufen. Beispiel:

```
exten => 300,1,VoiceMailMain()
```

3.2.1. Syntax

```
VoiceMailMain([s|p]mailbox[@context])
```

`[s|p]`
- `s` Wird ein `s` vor den Mailboxnamen gestellt, dann entfällt die Passwortabfrage.
- `p` Wird ein `p` vor den Mailboxnamen gestellt, dann wird der User gebeten, eine Mailbox anzugeben. Aus dem Inhalt von *`mailbox`* und dieser Eingabe wird dann der neue Mailboxname. Damit lässt sich eine mandantenfähige Mailboxstruktur realisieren.

`mailbox` Der Name (also Nummer) der Mailbox. Wenn kein Name vorgegeben wurde, fragt das System danach.

`@context` Definiert den Context, der innerhalb der `voicemail.conf` angesprochen wird.

3.2.2. Menü

Eine Beschreibung des Sprachmenüs von `VoiceMailMain()` ist schwierig, da es von den eingesetzten Prompts (Sprachbausteinen) abhängt. Die prinzipiellen Funktionen sind hier aufgelistet. In Klammern finden sich die englischen Beschreibungen.

1. Nachrichten abspielen *(Read voicemail messages)*
   - 3 Erweiterte Optionen *(Advanced options)*
     1. Antwort *(Reply)*
     2. Rückruf *(Call back)*
     3. Zusatzinformationen *(Envelope)*
     4. Anruf tätigen *(Outgoing call)*
   - 4 Vorherige Nachricht *(Play previous message)*
   - 5 Wiederholung der aktuellen Nachricht *(Repeat current message)*
   - 6 Nächste Nachricht *(Play next message)*
   - 7 Lösche aktuelle Nachricht *(Delete current message)*
   - 8 Nachricht an eine andere Mailbox weiterleiten *(Forward message to another mailbox)*
   - 9 Nachricht in einem Ordner speichern *(Save message in a folder)*
   - * Hilfe; während des Abspielens einer Nachricht: Zurückspulen *(Help; during msg playback: Rewind)*
   - # Beenden; während des Abspielens einer Nachricht: Weiter *(Exit; during msg playback: Skip forward)*
2. Ordner wechseln *(Change folders)*
0. Mailbox-Optionen *(Mailbox options)*
   - 1 Aufnahme der „nicht erreichbar“-Ansage *(Record your unavailable message)*
   - 2 Aufnahme der „im Gespräch“-Ansage *(Record your busy message)*
   - 3 Aufnahme des Namens *(Record your name)*
   - 4 Aufnahme einer temporären Ansage *(Record your temporary message)*

Jeweils nach der Aufnahme:

1. Akzeptieren *(Accept)*
2. Nochmals abspielen *(Review)*
3. Nochmals aufnehmen *(Re-record)*

* Hilfe *(Help)*

\# Beenden *(Exit)*

4. `voicemail.conf`

In der Datei `voicemail.conf` wird das Voicemailsystem von Asterisk konfiguriert. Es besteht aus 3 Bereichen:

- `[general]`
- `[zonemessages]`
- Freie Contexte

4.1. `[general]`

Im Bereich `[general]` werden allgemeine Konfigurationen des Voicemailsystems vorgenommen. Es stehen folgende Optionen zur Verfügung:

```
attach = [yes|no]
```

Legt fest, ob an die Benachrichtigungs-E-Mails an die Nutzer auch die Sprachnachricht als Sounddatei angehängt wird. Default: `yes`. Normalerweise wird das erste unter `format` eingestellte Format verwendet.

Beispiel:

```
attach = no
```

```
callback = [Context]
```

Legt einen Context fest, der für Rückrufe aus dem System benutzt wird. Ist dieser Wert nicht gesetzt, was per Default der Fall ist, dann kann der User keine Rückrufe aus dem Voicemailsystem machen.

Beispiel:

```
callback = intern
```

```
charset = [Zeichensatz]
```

Gibt den Zeichensatz (Charset) vor, in dem die E-Mails vom Voicemailsystem kodiert werden.

Beispiel:

```
charset = ISO-8859-1
```

```
delete = [yes|no]
```

Definiert, ob eine Sprachnachricht nach dem Versenden per E-Mail automatisch gelöscht werden soll. Dies spart Festplattenplatz auf dem Server, falls man Sprachnachrichten nur per E-Mail empfangen will.

Beispiel:

```
delete = yes
```

`directoryintro = [Dateiname]`

Definiert eine Sprachdatei innerhalb des Default-Verzeichnisses `/var/lib/asterisk/sounds/`, die anstatt der normalen Datei für das Dial-by-Name-System abgespielt wird (siehe ???).

Beispiel:

```
directoryintro = intro-telefonbuch
```

`emailsubject = [Text der Betreffzeile]`

Definiert den Betreff (Subject) einer Benachrichtigungs-E-Mail. Für Informationen über Variablen siehe `emailbody`.

Beispiel:

```
emailsubject = Neue Sprachnachricht von ${VM_CALLERID}
```

`pbxskip = [yes|no]`

Standardmäßig schreibt Asterisk am Anfang der Betreffzeile der E-Mail-Benachrichtigung über eine Sprachnachricht den Text `[PBX]`: Dies kann mit `yes` unterdrückt werden.

Beispiel:

```
pbxskip = yes
```

`emailbody = [Text der Email]`

Definiert den E-Mail-Body[4] einer Benachrichtigungs-E-Mail (maximal 512 Zeichen).

Im Betreff und im Body einer solchen E-Mail kann auf folgende Variablen zugegriffen werden:

`${VM_NAME}`	Name des Mailboxinhabers
`${VM_DUR}`	Länge der Nachricht
`${VM_MSGNUM}`	Nummer der Nachricht
`${VM_MAILBOX}`	Name der Mailbox
`${VM_CALLERID}`	Telefonnummer und Name des Anrufers
`${VM_CIDNUM}`	Telefonnummer des Anrufers
`${VM_CIDNAME}`	Name des Anrufers
`${VM_DATE}`	Datum und Uhrzeit des Anrufes
`${VM_MESSAGEFILE}`	Name der Sounddatei, in der die Nachricht abgespeichert ist

Beispiel:

```
emailbody = Hallo ${VM_NAME},\n\nSie haben eine neue Nachricht von ${VM_CALLERID} in
Mailbox ${VM_MAILBOX}.
```
[5]

`serveremail = [Absenderadresse]`

E-Mail-Adresse, die als Absender der E-Mail vom Voicemailsystem an den User eingestellt wird.

Beispiel:

```
serveremail = voicemail@apfelmus-gmbh.de
```

[4] Textteil einer E-Mail

[5] Es muss alles in eine Zeile geschrieben werden. Durch `\n` wird in der E-Mail ein Zeilenumbruch eingefügt.

`fromstring = [Absendername]`

Definiert den Absender (`From`-Header) in E-Mails vom Voicemailsystem.

Beispiel:

```
fromstring = AB
```

`mailcmd = [Shell-Befehl]`

Definiert das Programm (mit absolutem Pfad und ggf. Parametern), das zum Versenden von E-Mails benutzt werden soll. Default: `/usr/sbin/sendmail -t`

Beispiele:

```
mailcmd = /usr/sbin/sendmail -t
```

```
mailcmd = /usr/exim/bin/exim -t
```

`externnotify = [Shell-Befehl]`

Definiert ein Programm (mit absolutem Pfad), das von Asterisk aufgerufen wird, wenn eine neue Sprachnachricht für einen User vorliegt.

Beispiel:

```
externnotify = /usr/bin/local/mach-was.sh
```

`externpass = [Shell-Befehl]`

Definiert ein Programm (mit absolutem Pfad), das von Asterisk aufgerufen wird, wenn ein User sein Passwort geändert hat.

Beispiel:

```
externpass = /usr/bin/local/mach-was.sh
```

`forcegreetings = [yes|no]`

Definiert, ob ein User vom System dazu gezwungen wird, beim ersten Benutzen des Systems eine Begrüßungsnachricht aufzunehmen. Default: `no`

Beispiel:

```
forcegreetings = no
```

`forcename = [yes|no]`

Definiert, ob der User vom System dazu gezwungen wird, bei der ersten Nutzung seinen Namen aufzunehmen. Default: `no`

Beispiel:

```
forcename = yes
```

`format = [gsm|wav|wav49]`

Listet die Codecs auf, mit denen eine Sprachnachricht abgespeichert werden soll. Werden mehrere Codecs aufgeführt, werden auch mehrere Dateien gespeichert. Das kann im Extremfall zu Platzproblemen auf der Festplatte führen. Der Vorteil liegt darin, dass bei einer Abfrage die Nachricht nicht noch einmal neu kodiert werden muss, falls der Abfragende mit einem anderen Codec abfragt, als mit dem, in dem die Nachricht bereits gespeichert wurde.

Bei aktiviertem `attach` wird das erste Format dieser Liste für E-Mail-Anhänge verwendet.

Beispiele:

```
format = gsm|wav
```

Jede Sprachnachricht wird im GSM- und im WAV-Format abgespeichert.

```
format = wav
```

Jede Sprachnachricht wird nur im WAV-Format abgespeichert.

Achtung

Wird diese Einstellung im laufenden Betrieb verändert, müssen vorher vom System angelegte Sounddateien manuell gelöscht werden.

```
searchcontexts = [yes|no]
```

Standardmäßig werden Voicemailboxen nur im angegebenen Context gesucht. Mit diesem Schalter kann man das Suchverhalten auf alle Contexte erweitern. Default: `no`

Beispiel:

```
searchcontexts = no
```

```
maxmsg = [Anzahl Nachrichten]
```

Definiert die maximale Anzahl an Sprachnachrichten in einer Voicemailbox. Default: `100`. Danach können keine neuen Nachrichten mehr aufgenommen werden, und der Anrufer hört die Nachricht `/var/lib/asterisk/sounds/vm-mailboxfull.gsm`[6].

Beispiel:

```
maxmsg = 50
```

```
maxmessage = [Länge in Sekunden]
```

Definiert die maximale Länge einer Sprachnachricht in Sekunden. Default: unendlich

Beispiel:

```
maxmessage = 120
```

```
minmessage = [Länge in Sekunden]
```

Definiert die minimale Länge einer Sprachnachricht in Sekunden. Default: `0`

Beispiel:

```
minmessage = 5
```

```
maxgreet = [Länge in Sekunden]
```

Definiert die maximale Länge einer Begrüßungsnachricht in Sekunden. Default: unendlich

Beispiel:

```
maxgreet=240
```

[6] Pfad abhängig von der Installation, z.B. auch `/usr/share/asterisk/sounds/`

`maxsilence = [Länge in Sekunden]`

Definiert die Anzahl der Sekunden an Stille, die das System wartet, bevor es davon ausgeht, dass die Nachricht zu Ende gesprochen wurde.

Beispiel:

```
maxsilence=10
```

`silencethreshold = [Schwellenwert]`

Definiert, was für Asterisk bei der Einstellung `maxsilence` Stille (also Silence) ist. Je niedriger dieser Wert ist, desto höher ist die Empfindlichkeit. Default: `128`[7]

Beispiel:

```
silencethreshold = 50
```

`maxlogins = [Anzahl]`

Definiert die maximale Anzahl von fehlerhaften Einlogversuchen (falsches Passwort durch den User), bevor Asterisk auflegt. Default: `3`

Beispiel:

```
maxlogins = 3
```

`skipms = [Millisekunden]`

Setzt die Anzahl an Millisekunden, um die während des Abhörens einer Sprachnachricht mit den Tasten für weiter und zurück in der Nachricht gesprungen wird. Default: `3000`

Beispiel:

```
skipms = 6000
```

`usedirectory = [yes|no]`

Erlaubt es dem User von Voicemailboxen, die Einträge eines Dial-by-Name-Systems zur Weiterleitung von Nachrichten zu benutzen. Default: `no`

Beispiel:

```
usedirectory = yes
```

`saycid = [yes|no]`

Definiert, ob die Telefonnummer des Anrufers beim Abrufen der Nachrichten angesagt werden soll. Default: `no`

Beispiel:

```
saycid = yes
```

`cidinternalcontexts = [context,context,...]`

Legt die Contexte fest (durch Kommata getrennt), die beim Abrufen einer Voicemail-Nachricht als intern gelten, für die also z.B. nur die Durchwahl (Extension) als Caller-ID angesagt wird. Default: keine

[7] Es lässt sich selbst im Source-Code von Asterisk keine Information finden, in welchem Rahmen sich dieser Wert sinnvoll bewegen kann (es handelt sich um einen Int). Ggf. ist also Ausprobieren erforderlich.

```
pagerfromstring = [Absendername]
```

Analog zu `fromstring` für Nachrichten an Pager.

Beispiel:

```
pagerfromstring = Apfelmus GmbH
```

```
pagersubject = [Betreff]
```

Analog zu `emailsubject` für Nachrichten an Pager.

Beispiel:

```
pagersubject = Neue Voicemail
```

```
pagerbody = [Absendername]
```

Analog zu `emailbody` für Nachrichten an Pager.

Beispiel:

```
pagerbody = Neue Voicemail in Mailbox ${VM_MAILBOX} von ${VM_CALLERID}.
```

Neben den hier aufgeführten Parametern gibt es noch weitere, die genannten sind jedoch die wichtigsten für gängige Konfigurationen. Eine Auflistung aller verfügbaren Parameter findet sich in der Datei `voicemail.conf`. In der Grundinstallation sind dort alle Parameter mit einer kurzen Beschreibung enthalten.

4.2. `[zonemessages]`

Da es vorkommt, dass sich die User eines Voicemailsystems in unterschiedlichen Zeitzonen befinden, kann im Bereich `[zonemessages]` für jede Zeitzone ein unterschiedliches Format definiert werden, mit dem dem User die Uhrzeit angesagt wird.[8]

4.2.1. Syntax

```
zonenname = timezone | format
```

`zonenname`

Unter diesem Namen kann man später auf das entsprechende Format zugreifen. Verwenden Sie nur Kleinbuchstaben, Ziffern, Bindestrich (`-`) und Unterstrich (_).

`timezone`

In einem normalen Debian-Linux-System finden Sie im Verzeichnis `/usr/share/zoneinfo/` eine Auflistung von verschiedenen Regionen. Jede Region wird mit einem Unterverzeichnis dargestellt. In diesem Unterverzeichnis stehen Städtenamen. Ein `timezone`-Eintrag besteht aus der Region und dem Städtenamen (getrennt durch ein `/`).

Beispiele:

```
Europe/Berlin
```

```
Australia/Sydney
```

`format`

Definiert das Format, in dem die Zeit angesagt wird. Es stehen folgende Variablen zur Verfügung:

[8] Beachten Sie, dass Sie eine wie hier beschrieben definierte Zeitzone noch für die User aktivieren müssen. Siehe Abschnitt 4.4, „Mailbox-Definition“

`A` Wochentag (Montag bis Sonntag)

`a` siehe `A`

`B` Monat (Januar bis Dezember)

`b` siehe `B`

`h` siehe `B`

`d` Tag des Monats (1. bis 31.)

`e` siehe `d`

`Y` Jahr (z.B. 2006)

`I` Stunde im amerikanischen 12-Stunden-Format

`l` siehe `I`

`H` Stunde im europäischen 24-Stunden-Format. Dabei wird beim englischen Sprachmenü einstelligen Uhrzeiten eine 0 (englische Aussprache „oh“) vorangestellt.

`k` Stunde im europäischen 24-Stunden-Format

`M` Minute

`P` AM oder PM

`p` siehe `P`

`Q` „heute“, „gestern“ oder `ABdY` (Wochentag, Monat, Tag, Jahr)

`q` „ “ (also nichts, wenn es heute ist), „gestern“ oder `ABdY` (Wochentag, Monat, Tag, Jahr)

`R` Stunde:Minute:Sekunde

Daneben gibt es noch die Sonderfälle `${`*`VAR`*`}` (Inhalt dieser Variablen wird eingefügt) und `'`*`sprachdatei`*`'`. Dabei ist zu beachten, dass '*`sprachdatei`*' ohne „`.gsm`“ ist. Dieser Sprachbaustein aus dem Verzeichnis `sounds/` wird dann abgespielt. Die einfachen Anführungszeichen sind zwingend.

Beispiel:

```
[zonemessages]
deutschland = Europe/Berlin | 'vm-received' Q 'digits/at' kM
england = Europe/London | 'vm-received' Q 'digits/at' R
military = Zulu | 'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'
```

4.3. Freie Contexte

Genau wie in der `extensions.conf` gibt es auch in der `voicemail.conf` die Möglichkeit, einen Context zu definieren. Das ist praktisch, um zum Beispiel verschiedene Firmenabteilungen separat zu behandeln. Dadurch können auch unterschiedliche Adress- oder Telefonbücher definiert werden (siehe ???).

4.3.1. Der Default-Context

Wenn kein anderer Context eingerichtet wurde, muss zumindestens der `[default]`-Context existieren. Für die meisten aller Installationen reicht dieser aus.

4.4. Mailbox-Definition

In einem Context können beliebig viele Mailboxen definiert werden.

4.4.1. Syntax

```
mailbox => passwort,name[,e-mail[,pager-e-mail[,optionen]]]
```

Beispiel:

```
202 => 1234,Peter Meier,peter@familie-meier.de,,attachfmt=wav|delete=yes
```

`mailbox`
: Name der Mailbox (Ziffern)

`passwort`
: Passwort für die Mailbox (Ziffern oder Buchstaben). Ziffern sind in der Regel praktischer, da sie auch auf normalen Telefonen vom User eingegeben werden können.

`e-mail`
: E-Mail-Adresse, an die Sprachnachrichten gesendet werden sollen

`pager-e-mail`
: E-Mail Adresse des Pagers, an die Benachrichtigungen (eine Textzeile) geschickt werden sollen[9]

`optionen`
: Unterschiedliche Optionen, die durch ein Pipe-Zeichen (`|`) voneinander getrennt aneinandergereiht werden. Sie können hier die unter `[general]` (siehe Abschnitt 4.1, „`[general]`“) global festgelegten Einstellungen pro Mailbox verändern. Bitte beachten Sie, dass zwischen Parametername, dem Gleich-Zeichen (`=`) und dem Wert keine Leerzeichen sein dürfen. Dasselbe gilt für das Pipe-Zeichen (`|`).

  Die wichtigsten Optionen in diesem Zusammenhang sind:

  `tz=[Zeitzone]`

  Setzt die gewünschte Zeitzone aus dem vorher definierten `[zonemessages]` Context (siehe Abschnitt 4.2, „`[zonemessages]`“).

  Beispiel:

  ```
  tz=deutschland
  ```

  `attach=[yes|no]`

  Legt fest, ob die Sounddatei der Sprachnachricht der E-Mail beigefügt werden soll.

  `attachfmt=[gsm|wav|wav49]`

  Legt ggf. ein anderes als das unter `[general]` mittels `format` eingestellte Format fest, das für E-Mail-Anhänge verwendet wird.

  `saycid=[yes|no]`

  Definiert, ob die Telefonnummer des Anrufers angesagt werden soll.

  `sayduration=[yes|no]`

  Definiert, ob die Länge der Nachricht angesagt werden soll. Default: `yes`

  `saydurationm=[ab Dauer in Minuten]`

  Legt die Mindestlänge einer Nachricht in Minuten fest, ab der die Länge angesagt wird. Default: `0`

[9] Bitte beachten Sie, dass diese Nachrichten ebenfalls E-Mails sind. Sollten Sie SMS-Nachrichten verschicken wollen, suchen Sie im Internet nach „sms gateway“ oder „asterisk sms gateway“.

`dialout=[context]`

Definiert den Context, der zum Rauswählen benutzt werden soll. Ohne diese Angabe kann man aus dem Voicemailsystem heraus keine Telefonate führen.

Beispiel:

```
dialout=intern
```

`sendvoicemail=[yes|no]`

Bestimmt, ob Teilnehmer Sprachnachrichten an andere Teilnehmer des Voicemailsystems schicken können.

`callback=[context]`

Definiert den Context, der für Rückrufe benutzt wird. Ist er festgelegt, kann man den Anrufer, der eine Sprachnachricht hinterlassen hat, einfach aus dem Voicemailboxsystem zurückrufen.

`review=[yes|no]`

Legt fest, ob der Anrufer die von ihm hinterlassene Sprachnachricht vor dem Speichern erneut anhören kann. Default: `no`

`operator=[yes|no]`

Definiert, ob der Anrufer vor, während oder nach der Aufnahme seiner Sprachnachricht mit der Eingabe von `0` (also Null auf dem Telefon drücken) zum Operator durchgestellt wird. Operator ist immer die Extension `o` (kleiner Buchstabe o) im aktuellen Context. Default: `no`

`envelope=[yes|no]`

Legt fest, ob allgemeine Informationen vor der Sprachnachricht angesagt werden sollen. Wenn dieser Parameter auf `no` gesetzt ist, dann werden dadurch bestimmte andere Einstellungen (wie zum Beispiel `tz`) überstimmt. Default: `yes`

`delete=[yes|no]`

Definiert, ob Sprachnachrichten nach dem Versenden per E-Mail automatisch gelöscht werden sollen. Default: `no`

`nextaftercmd=[yes|no]`

Definiert, ob das System nach dem Löschen oder Abspeichern einer Sprachnachricht automatisch zur nächsten Nachricht springt. Default: `no`

`forcename=[yes|no]`

Bestimmt, ob das System den User beim ersten Abruf des Systems dazu zwingt, seinen Namen aufnehmen zu lassen. Default: `no`

`forcegreetings=[yes|no]`

Definiert, ob das System den User bei der ersten Nutzung des Voicemailsystems dazu zwingt, eine persönliche Ansage für seine Voicemailbox aufzunehmen. Default: `no`

`hidefromdir=[yes|no]`

Definiert, ob dieser Eintrag aus dem Dial-by-Name-Verzeichnis herausgenommen (versteckt) werden soll. Default: `no`

5. Telefonbuch (Dial-by-Name)

Auch wenn es eigentlich nicht unmittelbar mit der VoiceMail-Funktion von Asterisk zu tun hat, gehört das Dial-by-Name-Directory[10] sinngemäß in dieses Kapitel. Es wurde von den Asterisk-Entwicklern als Teil des Voicemailsystems programmiert und greift auf Daten der Datei `voicemail.conf` zu.

Wenn wir für das Beispiel in Abschnitt 2.1, „Voicemailbox der Familie Meier“ die folgende Konfiguration zu Grunde legen:

```
[general]
format = gsm

[default]
; Ext. => Passw.,Vorname Nachname,E-Mail,P.,Optionen
;                ^^^^^^^^^^^^^^^^
200 => 1234,Horst Meier
201 => 1234,Eva Meier
202 => 1234,Peter Meier,peter@die-familie-meier.de
203 => 1234,Lisa Meier,lisa@die-familie-meier.de,,delete=yes
```

Dann können wir daraus recht einfach ein Dial-by-Name-Directory generieren. Dies geschieht durch Aufruf des Programmes `Directory()` in der `extensions.conf`:

```
exten => 800,1,Directory(default,from-intern)
```

5.1. Syntax

```
Directory(vm-context[,dial-context[,options]])
```

`vm-context`
: Das Directory wird immer aus einem bestimmten Context generiert (alle Einträge aus diesem und nur diesem Context werden zur Auswahl gestellt). In den meisten Fällen kann man den `default`-Context nehmen. In großen Firmen kann man für bestimmte Abteilungen eigene Contexte definieren und somit Unter-Directories erstellen.

`dial-context`
: Das Directory wird benutzt, um eine bestimmte Person anzurufen, entsprechend muss auch ein Context definiert werden, aus dem (in der `extensions.conf`) der Anruf geführt werden soll. Ist *`dial-context`* nicht definiert, wird automatisch der Context von *`vm-context`* benutzt.

  Drückt der User des Systems die Taste `0` (Null), so wird er in diesem Context auf die Extension `o` (kleiner Buchstabe o) geleitet. Bei Eingabe des `*` (Stern) wird er zur Extension `a` (kleiner Buchstabe a) geleitet.

`options`
: Mit der Option `f` kann man das Directory so einstellen, dass nach Vornamen und nicht nach Nachnamen gesucht wird.

Siehe auch Abschnitt 6.24, „`Directory()`“

5.2. Funktionsweise

Der Anrufer wird vom Directory aufgefordert, die ersten 3 Buchstaben des Nachnamens (beim Aufruf mit der Option `f`, des Vornamens) mit dem Tastenfeld des Telefons einzugeben. Das Programm sucht dann den entsprechenden Eintrag aus dem Directory und verbindet den Anrufer auf Wunsch.

[10] Anstelle des deutschen Wortes Verzeichnis verwenden wir der Einfachheit halber das englische Wort Directory.

6. `voicemail.conf` als Passwortspeicher

Ebenso nützlich wie als internes Rufnummernverzeichnis kann die `voicemail.conf` auch als Speicherort für Passwörter sein. Sie werden vielleicht eine Situation haben, in der sich Ihre internen Benutzer aus welchem Grund auch immer am System authentifizieren sollen - ein konkreter Fall wären Call-Center-Agenten, die sich anmelden.

Es bietet sich in solchen Fällen manchmal an, für die betreffenden User einen Eintrag in der `voicemail.conf` anzulegen, obwohl diese vielleicht gar keine Mailbox brauchen. Und keine Angst: auf den so angelegten Mailboxen wird niemand versehentlich eine Nachricht hinterlassen können, solange Sie nicht im Dialplan mit `VoiceMail()` (oder `VoiceMailMain()`) darauf verweisen.

Sie könnten beispielsweise die VM-Boxen

```
1001 => 1234,Hans Glueck,,,hidefromdir=yes
1002 => 1234,Peter Pech,,,hidefromdir=yes
```

im Abschnitt `[call-center-agenten]` anlegen und die User an geeigneter Stelle im Dialplan mit VMAuthenticate (Abschnitt 6.109, „`VMAuthenticate()`“) authentifizieren:

```
exten => 988,n,Read(agentennr,agent-user,10,,3)
exten => 988,n,VMAuthenticate(${agentennr}@call-center-agenten)
```

Betrachten Sie das aber nur als Denkanstoß bzw. Beschreibung einer von vielen Möglichkeiten zur Authentifizierung. Speicherung der Passwörter in der AstDB oder einer anderen Datenbank wäre diesem Ansatz sicher vorzuziehen.

Kapitel 7. Asterisk-Datenbank (AstDB)

Versionsgeschichte
Version Rechtsschreiblektorat 23.12.2006

1. Einleitung

Das Problem bei der Benutzung von Variablen im Dialplan ist, dass der Wert dieser und überhaupt aller zur Laufzeit definierten Variablen bei einem Systemabsturz oder einem Neustart von Asterisk gelöscht bzw. auf ihre Anfangswerte zurückgesetzt werden. Dadurch sind bestimmte Einsatzszenarien gar nicht denkbar. Wenn man beispielweise eine Call-Forwarding-Funktion[1] oder ein Calling-Card-System implementieren möchte, dann sollten diese natürlich z.B. das Restguthaben in einer Datenbank speichern, damit diese Daten bei einem Neustart des Systems wieder korrekt zur Verfügung stehen.

1.1. Performance

Bezüglich der Asterisk-Datenbank wird immer wieder die Frage nach der Performance derselben gestellt. Das lässt sich nicht so pauschal beantworten. Falls Sie nur kleine Datenbestände (wie unser Wahlwiederholungsbeispiel) benötigen, so ist die Asterisk-Datenbank sicherlich eine sinnvolle Wahl. Bei größeren und komplexen Datenbeständen sollten Sie aber überlegen, ob das Verwenden einer externen SQL-Datenbank geeigneter ist. Allerdings ist diese Diskussion bei dem überwiegenden Teil aller Anwendungen rein theoretischer Natur, da es sich bei der Asterisk-Datenbank um eine Berkley DB handelt und diese ausreichend performant ist. Tatsächlich ist die Berkley DB, wenn es sich um reine "Key => Value"-Datenpaare handelt, mit die schnellste ihrer Zunft. Sie sollten sich daher diese Frage erst stellen, wenn die Geschwindigkeit der Datenbank nachweislich zu Problemen führt oder Sie eine größere Installation mit umfangreicher Funktionalität aufsetzen wollen.

2. Asterisk-Datenbank

Asterisk bringt standardmäßig eine Datenbank mit, die die Berkeley DB (BDB)[2] als Datenbank-Engine einsetzt. In dieser Datenbank-Engine wird immer einem Schlüssel (key) ein Wert (value) zugeordnet, wobei Schlüssel in Familien (family) zusammengefasst werden.

Warnung

Bis einschließlich Asterisk Version 1.2 wurden die folgenden Befehle eingesetzt:

- `DBput(family/key=value)`

  zum Speichern eines Werts in der Datenbank.

- `DBget(var=family/key)`

  um einen Wert aus der Datenbank auszulesen. Wird der aufgerufene Key in der Datenbank nicht gefunden, erhöht sich die Priorität um 101.

Diese Befehle werden ab der Version 1.4 nicht mehr unterstützt. Aus diesem Grund wird hier nur die neue Variante über die Funktion `DB()` besprochen. Diese Variante funktioniert auch in der Version 1.2.

Auf diese Datenbank kann mit der Funktion `DB()` zugegriffen werden (siehe Abschnitt 7.11, „`DB()`“).

[1] CallForwarding-Funktionalität: Jeder Teilnehmer kann durch Wahl einer bestimmten Nummer alle Gespräche an einen anderen Apparat weiterleiten lassen. Durch Wahl einer anderen Nummer wird diese Funktion wieder deaktiviert.

[2] Die Berkeley-Datenbank (Berkeley DB) ist eine hochperformante, eingebettete Datenbank-Bibliothek mit Programmierschnittstellen zu C, C++, Java, Perl, Python, Tcl und vielen weiteren Programmiersprachen. Die DB speichert beliebige Schlüssel- oder Datenpaare und unterstützt mehrere Datenelemente für einen einzelnen Schlüssel. Die DB ermöglicht tausende von simultanen Threads zum Manipulieren der Datenbanken, die bis zu 256 Terabyte groß sein können, und läuft auf einer großen Anzahl von Systemen, unter anderem auf den meisten UNIX-artigen und Windows-Systemen und auch auf Echtzeitbetriebssystemen. [Zitiert aus *http://de.wikipedia.org/wiki/Berkeley_DB*]

2.1. Werte in die Datenbank schreiben

Die Funktion `DB()` kann innerhalb der Applikation `Set()` aufgerufen werden. Soll in der Family `obst` der Eintrag `apfel` den Wert `20` bekommen, so geschieht das mit folgendem Aufruf:

```
exten => 1234,1,Set(DB(obst/apfel)=20)
```

2.2. Werte aus der Datenbank lesen

Die Werte in der Datenbank können mit der Funktion `DB()` in der Form `${DB(family/key)}` abgerufen werden. Um den Inhalt des Eintrags `apfel` in der Family `obst` auf dem Command Line Interface auszugeben, kann man folgenden Aufruf ausführen:

```
exten => 1234,1,NoOp(obst/apfel hat den Wert ${DB(obst/apfel)})
```

Soll der Inhalt dieses Datenbankfeldes in der Variablen `apfelmenge` gespeichert werden, so kann dies mit `Set()` erfolgen:

```
exten => 1234,1,Set(apfelmenge=${DB(obst/apfel)})
```

2.3. Werte aus der Datenbank löschen

Die beiden Applikationen `DBdel()` und `DBdeltree()` können zum Löschen von Datenbankeinträgen genutzt werden.

2.3.1. DBdel()

`DBdel()` löscht einen einzelnen Eintrag in der Datenbank. Soll der Key `apfel` in der Family `obst` gelöscht werden, so geschieht das wie folgt:

```
exten => 1234,1,DBdel(obst/apfel)
```

Siehe auch: Abschnitt 6.19, „`DBdel()`“

2.3.2. DBdeltree()

Soll eine ganze Family gelöscht werden, so verwendet man `DBdeltree()`. Die Family `obst` kann mit

```
exten => 1234,1,DBdeltree(obst)
```

gelöscht werden.

Siehe auch: Abschnitt 6.20, „`DBdeltree()`“

3. Datenbankzugriff vom CLI

Vom Command Line Interface (CLI) kann der Administrator ebenfalls auf die Datenbank zugreifen.

Tipp

Sollte man den entsprechenden Befehl oder die genaue Syntax vergessen haben, so kann man jederzeit `help database` im CLI eingeben:

```
big-island*CLI> help database
             database del  Removes database key/value
         database deltree  Removes database keytree/values
             database get  Gets database value
             database put  Adds/updates database value
            database show  Shows database contents
         database showkey  Shows database contents
```

Wichtig

Denken Sie bei den folgenden Beschreibungen daran, dass Sie, wenn die Family, der Schlüssel oder der Wert Leerzeichen enthält, Anführungszeichen (") um den String schreiben müssen! Also z.B. `database put test eintrag "Hallo Welt"`.

3.1. Werte in die Datenbank schreiben

Mit `database put` *`family key value`* können Werte in die Datenbank geschrieben werden:

```
big-island*CLI> database put obst apfel 20
Updated database successfully
```

3.2. Werte aus der Datenbank lesen

Mit `database get` *`family key`* lesen Sie Werte aus der Datenbank aus:

```
big-island*CLI> database get obst apfel
Value: 20d*CLI>
```

3.3. Werte aus der Datenbank löschen

`database del` *`family key`* und `database deltree` *`family`* löschen Werte aus der Datenbank.

3.3.1. database del

Um den Key `apfel` aus der Family `obst` zu löschen, kann man `database del` aufrufen:

```
big-island*CLI> database del obst apfel
Database entry removed.
```

3.3.2. database deltree

Mit `database deltree` kann eine ganze Family aus der Datenbank gelöscht werden:

```
big-island*CLI> database deltree obst
Database entries removed.
```

3.4. Datenbankinhalt anzeigen

Mit den Befehlen `database show` und `database showkey` kann der Administrator den Inhalt der Datenbank auf dem CLI ausgeben lassen. Ein Beispiel:

```
big-island*CLI> database put einkaufsliste eier 2
Updated database successfully
big-island*CLI> database put einkaufsliste butter 250
Updated database successfully
big-island*CLI> database put einkaufsliste zucker 500
Updated database successfully
big-island*CLI> database show
/einkaufsliste/butter                             : 250
/einkaufsliste/eier                               : 2
/einkaufsliste/zucker                             : 500
big-island*CLI> database showkey butter
/einkaufsliste/butter                             : 250
big-island*CLI> database deltree einkaufsliste
Database entries removed.
```

4. Datenbankzugriff von der Shell

Über den Befehl **asterisk -rx '*command*'** können natürlich alle CLI-Kommandos auch von einem Shell-Skript ausgeführt werden. Das folgende Beispiel zeigt, wie man von der Shell aus die Datenbank erst mit Werten füllt, diese dann ausliest und zum Schluss wieder löscht:

```
big-island:~# asterisk -rx 'database put test var1 23'
Updated database successfully
big-island:~# asterisk -rx 'database put test var2 42'
Updated database successfully
big-island:~# asterisk -rx 'database show test'
/test/var1                                       : 23
/test/var2                                       : 42
big-island:~# asterisk -rx 'database get test var2'
Value: 42
big-island:~# asterisk -rx 'database deltree test'
Database entries removed.
```

Wichtig

Denken Sie wieder daran, dass Sie, wenn die Family, der Schlüssel oder der Wert Leerzeichen enthält, Anführungszeichen (") um den String schreiben müssen! Also z.B.

```
big-island:~# asterisk -rx 'database put test eintrag "Hallo Welt"'
```

5. Backup der Datenbank

Die Asterisk-Datenbank wird standardmäßig im Verzeichnis `/var/lib/asterisk/astdb/` gespeichert. Sobald Asterisk gestoppt ist ist, kann man das Verzeichnis sichern.

Ein Backup im laufenden Betrieb kann auch von der Shell durch folgenden Befehl

```
asterisk -rx "database show" > /tmp/backup-asterisk-database.txt
```

durchgeführt werden. Allerdings ist das Restore dann etwas aufwendiger.

6. Anwendungsbeispiel CallForwarding

Wer bei Abwesenheit alle Gespräche an seine Durchwahl z.B. an sein Handy weiterleiten will, benötigt eine CallForwarding-Funktionalität. Diese kann entweder vom Endgerät (also dem SIP-Telefon) oder zentral von der Telefonanlage realisiert werden. Letztere Variante ist in der Praxis vorzuziehen, da man dadurch unabhängig von den Endgeräten ist (diese also im Zweifelsfall auch austauschen kann) und zusätzlich die gesamte Kontrolle auf dem Server behält. Außerdem ist im Falle eines Stromausfalls die Konfiguration nicht verloren und wird beim nächsten Start der Telefonanlage automatisch wiederhergestellt.

6.1. Einfaches CallForwarding

Bei der Apfelmus GmbH soll jeder Mitarbeiter die Möglichkeit bekommen, Gespräche zu seinem Telefon an eine beliebige andere Nummer weiterzuleiten. Um diese Weiterleitung zu aktivieren, muss die interne Rufnummer 44 gefolgt von der Zielrufnummer angerufen werden. Zum Deaktivieren der Weiterleitung muss wieder die 44 (aber diesmal ohne eine weitere Nummer) angerufen werden. Dies wird mit folgendem Wählplan realisiert:

```
[from-intern]
; Call Forwarding für einen einzelnen Anschluss
;
; aktivieren
exten => _44X.,1,Answer()
exten => _44X.,2,Set(DB(CF/${CALLERID(number)})=${EXTEN:2})
exten => _44X.,3,SayDigits(${EXTEN:2})
exten => _44X.,4,NoOp(Weiterleitung fuer ${CALLERID(number)} auf ${EXTEN:2} aktiviert.)
exten => _44X.,5,Hangup()
```

```
; deaktivieren
exten => 44,1,Answer()
exten => 44,2,DBdel(CF/${CALLERID(number)})
exten => 44,3,Playback(auth-thankyou)
exten => 44,4,NoOp(Weiterleitung fuer ${CALLERID(number)} deaktiviert.)
exten => 44,5,Hangup()

[from-extern]
exten => _X.,1,NoOp(Anruf von ${CALLERID(number)} fuer ${EXTEN})
exten => _X.,2,GotoIf($[foo${DB(CF/${EXTEN})} != foo]?100:20)
exten => _X.,20,Dial(SIP/${EXTEN})
exten => _X.,100,NoOp(Anruf fuer ${EXTEN} wird verbunden zu ${DB(CF/${EXTEN})})
exten => _X.,101,Dial(local/${DB(CF/${EXTEN})})
```

6.2. Komplexes CallForwarding

Diesmal soll in der Apfelmus GmbH jeder Mitarbeiter ein CallForwarding aktivieren können, allerdings soll es ein weiteres CallForwarding für die gesamte Firma geben, damit bei einem Betriebsausflug alle Gespräche an eine andere Niederlassung geleitet werden können. Diese Weiterleitung wird über die Funktionsnummer 55 aktiviert (gefolgt von der Zielrufnummer). Die große Weiterleitung für die gesamte Anlage hat dabei eine höhere Priorität als die einzelnen Regeln. Eine Realisierung kann wie folgt aussehen:

```
[from-intern]
; CallForwarding für einen einzelnen Anschluss
;
; aktivieren
exten => _44X.,1,Answer()
exten => _44X.,2,Set(DB(CF/${CALLERID(number)})=${EXTEN:2})
exten => _44X.,3,SayDigits(${EXTEN:2})
exten => _44X.,4,NoOp(Weiterleitung fuer ${CALLERID(number)} auf ${EXTEN:2} aktiviert.)
exten => _44X.,5,Hangup()

; deaktivieren
exten => 44,1,Answer()
exten => 44,2,DBdel(CF/${CALLERID(number)})
exten => 44,3,Playback(auth-thankyou)
exten => 44,4,NoOp(Weiterleitung fuer ${CALLERID(number)} deaktiviert.)
exten => 44,5,Hangup()

; CallForwarding für die gesamte Telefonanlage
;
; aktivieren
exten => _55X.,1,Answer()
exten => _55X.,2,Set(DB(CF/anlage)=${EXTEN:2})
exten => _55X.,3,SayDigits(${EXTEN:2})
exten => _55X.,4,NoOp(Weiterleitung der Anlage auf ${EXTEN:2} aktiviert.)
exten => _55X.,5,Hangup()

; deaktivieren
exten => 55,1,Answer()
exten => 55,2,DBdel(CF/anlage)
exten => 55,3,Playback(auth-thankyou)
exten => 55,4,NoOp(Weiterleitung der Anlage deaktiviert.)
exten => 55,5,Hangup()

[from-extern]
exten => _X.,1,NoOp(Anruf von ${CALLERID(number)} fuer ${EXTEN})
exten => _X.,4,GotoIf($[foo${DB(CF/anlage)} != foo]?200:5)
exten => _X.,5,GotoIf($[foo${DB(CF/${EXTEN})} != foo]?100:6)
exten => _X.,6,Dial(SIP/${EXTEN})
exten => _X.,100,NoOp(Anruf fuer ${EXTEN} wird verbunden zu ${DB(CF/${EXTEN})})
exten => _X.,101,Dial(local/${DB(CF/${EXTEN})})
exten => _X.,200,NoOp(Anruf fuer ${EXTEN} wird verbunden zu ${DB(CF/anlage)})
exten => _X.,201,Dial(local/${DB(CF/anlage)})
```

7. Anwendungsbeispiel CallingCard

In der Apfelmus GmbH sollen private Gespräche über eine CallingCard abgerechnet werden. Diese wird vom Hausmeister über sein Telefon virtuell aufgeladen (durch Anwahl der internen Servicenummer 88 gefolgt von der 3-stelligen Durchwahl und vom gewünschten Geldbetrag in Euro).[3] Die Gespräche werden zu einem Pauschalpreis von 1 Euro pro Gespräch abgerechnet. Diese privaten Gespräche werden über die interne Servicenummer 99 gefolgt von der Zielrufnummer geführt. Dabei ist zu beachten, dass bei einem Besetzt (busy) kein Geldbetrag vom Konto abgebucht wird. Die interne Rufnummer 98 kann dazu benutzt werden, den aktuellen Betrag auf dem CallingCard-Konto abzurufen.

```
[from-hausmeister]
; Aufladen der virtuellen CallingCard
;
exten => _88XXX.,1,Answer()
exten => _88XXX.,2,Set(DB(CallingCard/${EXTEN:2:3})=${EXTEN:5})
exten => _88XXX.,3,SayNumber(${EXTEN:5})
exten => _88XXX.,4,NoOp(CallingCard fuer ${EXTEN:2:3} mit ${EXTEN:5} aufgeladen.)
exten => _88XXX.,5,Hangup()

[from-intern]
; private Gespraeche
;
exten => _99.,1,GotoIf($[${DB(CallingCard/${CALLERID(number)} > 0]?2:200)
exten => _99.,2,Set(DB(CallingCard/${CALLERID(number)})=$[DB(CallingCard/${CALLERID(number)}) - 1])
exten => _99.,3,Dial(local/${EXTEN:2})
exten => _99.,104,Set(DB(CallingCard/${CALLERID(number)})=$[DB(CallingCard/${CALLERID(number)}) + 1])

exten => _99.,200,NoOp(CallingCard Konto ${CALLERID(number) ist leer.)
exten => _99.,201,Answer()
exten => _99.,202,SayNumber(0)
exten => _99.,203,Hangup()

; Mit der 98 kann der aktuelle Kontostand abgefragt werden.
;
exten => 98,1,Answer()
exten => 98,2,SayNumber(${DB(CallingCard/${CALLERID(number)})})
exten => 98,3,Hangup()
```

[3] Um das Beispiel möglichst einfach zu halten, wird der Fall, dass auf ein bereits bestehendes CallingCard-Konto ein weiterer Betrag eingezahlt wird (also eine Addition vollzogen werden muss) nicht besprochen. Ebenfalls nicht behandelt wird die Möglichkeit, dass ein Mitarbeiter von einem anderen Telefon ein privates Gespräch führen könnte.

Kapitel 8. Interactive Voice Response Systeme (IVR)

Versionsgeschichte
Version Rechtsschreiblektorat 23.12.2006

Auf Interactive Voice Response Systeme (IVR) trifft man heute bei vielen Einrichtungen. IVRs sind automatisierte Dialogsysteme, die einem Anrufer bestimmte Informationsangebote oder andere Leistungen in Dialogform anbieten. Das grundlegende Prinzip ist bei fast allen Systemen gleich, dem Anrufer wird ein Menü vorgelesen, aus dem er dann durch bestimmte Aktionen (meist das Drücken einer Taste auf dem Telefon) einzelne auswählen kann. So kann man über solche Systeme die aktuellen Börsenkurse oder auch Abfahrtzeiten von Zügen der Deutschen Bahn abfragen. Es gibt sogar Systeme, die Buchungen von Veranstaltungstickets ermöglichen. Die Dialogsysteme unterscheiden sich hierbei in dem Grad der Automation. Voll automatisierte Systeme generieren die Sprachansagen (Text2Speech) und verfügen über eine maschinelle Spracherkennung, um die Benutzereingaben zu erkennen. Solche Systeme bieten einen sehr hohen Komfort, sind jedoch technisch derart anspruchsvoll, dass sie selten zum Einsatz kommen. Die einfachste Variante ist zugleich die verbreitetste. Vorher aufgenommene Textnachrichten (Soundfiles) werden abgespielt und die Nutzereingaben erfolgen durch Drücken der Telefontasten, deren charakteristische Frequenz einfach und robust erkannt werden kann. Diese Form des Dialogsystems bietet Asterisk standardmäßig an, mit ein wenig Aufwand können sogar die Ansagen per Sprachsynthesemodule erzeugt werden. Die Meinungen über IVRs gehen zum Teil deutlich auseinander, die einen empfinden sie als nützliche Hilfe, andere wiederum ärgern sich darüber. Oft bieten sie dabei immer wieder Stoff für reichlich Lacher. Meistens liegt das an ungeschickten Menüführungen oder bei Systemen mit einer maschinellen Spracherkennung an einer schlechten Erkennungsrate. Eine gut funktionierende IVR kann dem Kunden eine echte Hilfe sein, aber eine schlecht umgesetzte kann einen Kunden auch abschrecken. Deshalb sollte man dieses Mittel mit Bedacht einsetzen und sich Zeit für eine sorgfältig geplante und sauber integrierte Lösung nehmen. Häufiges Überprüfen der Zahl der vorzeitigen Abbrüche durch Auflegen sollte dabei zu den Routineaufgaben gehören. Und -- IVRs sind weder ein Selbstzweck noch ein Allheilmittel -- denken Sie an Ihre Kunden!

1. Eine einfache IVR

Bei den Standardsprachbausteinen gibt es eine Datei `marryme.gsm`, die den Text "Will you marry me? Press 1 for yes or 2 for no."[1] enthält. Um hiermit eine Heiratsantrags-IVR zu realisieren, reicht folgender Dialplan.[2]

```
exten => 10,1,Answer()
exten => 10,2,Background(marryme)
exten => 10,3,Hangup()

exten => 1,1,Playback(thank-you-cooperation)
exten => 1,2,Hangup()

exten => 2,1,Playback(sorry)
exten => 2,1,Hangup()
```

Wird die 10 angerufen, hebt Asterisk ab und spielt das Soundfile `marryme.gsm` ab. Während der Ansage kann jederzeit vom Benutzer eine Eingabe gemacht werden. Diese Eingabe wird als weitere Extension gewertet und abgerufen. Wer also die 1 drückt, bekommt die Ansage "Thank you for your cooperation." abgespielt. Danach legt Asterisk auf.

1.1. Unterschied zwischen Playback() und Background()

Mit der Applikation Playback() können Soundfiles nur abgespielt werden, jegliche Eingabe während des Abspielens wird nicht beachtet. Background() hingegen spielt die Datei ab und wartet während des Abspielens auf eine Eingabe. Diese wird dann als Extension interpretiert.

[1] Übersetzung: Möchtest Du mich heiraten? Drücke 1 für ja und 2 für nein.
[2] Es sollte auch dem größten Geek unter den Lesern klar sein, dass eine solche IVR nur als Beispiel und niemals als reale Möglichkeit eines Heiratsantrages in Betracht gezogen werden sollte! ;-)

1.2. Unterschied zwischen 10 und 1000

Damit Asterisk zwischen den Eingaben 2, 22 und 2200 unterschieden kann, wartet Background() nach jeder Eingabe eine bestimmte Zeit. Ist diese Zeit (*TIMEOUT*) abgelaufen, so gilt die Eingabe als beendet.

Tipp

Mit *TIMEOUT* können noch andere Timeouts definiert werden, Informationen dazu bekommen Sie im CLI mit der Eingabe **show function TIMEOUT**.

Den *TIMEOUT* kann man in Sekunden angeben und wie folgt im Dialplan setzen:

```
exten => 10,1,Set(TIMEOUT(digit)=3)
```

1.3. Falscheingaben (die i-Extension)

Eine Falscheingabe (also eine nicht im Dialplan definierte Eingabe) kann mit der `i`-Extension (i wie invalid) abgefangen werden. Das einfache Beispiel würde folgendermaßen aussehen:

```
exten => 10,1,Answer()
exten => 10,2,Background(marryme)
exten => 10,3,Hangup()

exten => 1,1,Playback(thank-you-cooperation)
exten => 1,2,Hangup()

exten => 2,1,Playback(sorry)
exten => 2,1,Hangup()

; Bei allen anderen Eingaben wird diese
; i-Extension aufgerufen.
exten => i,1,Background(sorry)
exten => i,2,Hangup()
```

1.4. Pausen

Die einfachste Möglichkeit, kurze Pausen für die Eingabe zu realisieren, ist das Abspielen von Soundfiles ohne Inhalt. Im Verzeichnis `/var/lib/asterisk/sounds/silence/` gibt es 1 bis 9 Sekunden lange Dateien, die nur Stille abspielen. Soll in unserem Heiratsantragsbeispiel noch 5 Sekunden auf eine Antwort gewartet werden, so kann man das wie folgt erreichen:

```
exten => 10,1,Answer()
exten => 10,2,Background(marryme)
exten => 10,3,Background(silence/5)
exten => 10,4,Hangup()

exten => 1,1,Playback(thank-you-cooperation)
exten => 1,2,Hangup()

exten => 2,1,Playback(sorry)
exten => 2,1,Hangup()

exten => i,1,Background(marryme)
exten => i,2,Hangup()
```

2. Mehrstufige IVR-Systeme

Das Problem bei mehrstufigen IVRs ist, dass der Benutzer je nach Menüstruktur mehrmals eine einstellige Zahl (evtl. sogar die gleiche) drücken soll, aber jeweils ein anderes Ergebnis erhält oder eine andere Aktion erfolgt. Da innerhalb eines Contextes eine Zahl aber nur einmal vergeben werden kann, bliebe der Anrufer immer auf dieser Menüebene hängen. Sollen mehrere Menüs hintereinander geschaltet werden und in jedem Menü gleiche Eingaben andere Ergebnisse bringen, so muss man diese Untermenüs in andere Contexte (in unserem Beispiel `[kantine]`)

legen. Der Sprung zwischen den einzelnen Positionen erfolgt mit Goto(). Nehmen wir an, Sie haben folgende Soundfiles im Verzeichnis `/var/lib/asterisk/sounds/` abgespeichert:

- `grundmenue.gsm`

  Bitte drücken Sie die 1 für den Verkauf, die 2 für den Hausmeister und die 3 für die Kantine.

- `kantine.gsm`

  Drücken Sie die 1 für den Essensplan dieser Woche und die 2 für den Essensplan der nächsten Woche.

- `essensplan-diese-woche.gsm`

  Montag gibt es Nudeln mit Tomatensoße, Dienstag gibt es Jägerschnitzel, ...

- `essensplan-naechste-woche.gsm`

  Montag gibt es Eintopf, Dienstag gibt es Jägerschnitzel, ...

Wenn der Verkauf unter der Extension 100 und der Hausmeister unter der Extension 150 erreichbar ist, sieht der Dialplan für diese IVR folgendermaßen aus:

```
[beispiel-ivr]
; Das Menue wird staendig wiederholt.
;
exten => 30,1,Answer()
exten => 30,2,Background(grundmenue)
exten => 30,3,Background(silence/3)
exten => 30,4,Goto(2)

exten => 1,1,Dial(SIP/100)

exten => 2,1,Dial(SIP/150)

; Goto() springt in einen anderen
; Context ([kantine])
;
exten => 3,1,Goto(kantine,100,1)

exten => i,1,Goto(30,2)

[kantine]
exten => 100,1,Background(kantine)
exten => 100,2,Background(silence/3)
exten => 100,3,Goto(1)

exten => 1,1,Playback(essensplan-diese-woche)
exten => 1,2,Wait(2)
exten => 1,3,Goto(1)

exten => 2,1,Playback(essensplan-naechste-woche)
exten => 2,2,Wait(2)
exten => 2,3,Goto(1)

; Bei einer Falscheingabe wird
; wieder in das Start-Menue
; gesprungen.
;
exten => i,1,Goto(beispiel-ivr,30,2)
```

3. Text2Speech (TTS)

Text2Speech beschreibt die Umwandlung von geschriebenem Text in gesprochenen Text (Sprachsynthese) auf Computersystemen. Auf unserem Asterisk-System bedeutet das, dass ein Programm anhand einer Textdatei (meist in ASCII) eine entsprechende Audiodatei (Soundfile) erzeugt. Diese kann wie jede andere Multimediadatei abgespielt

werden und man hört den Text dann gesprochen. Wie bei vielen Software-Projekten wird die englische Sprache meist besser unterstützt als die deutsche.

Darüber hinaus gibt es große Qualitätsunterschiede zwischen den offenen und freien (meist GPL) Engines und entsprechenden kommerziellen Lösungen.

Tipp

Einen kostenlosen Test einer qualitativ sehr guten Engine kann man online bei IBM unter http://www.ibm.com/software/pervasive/tech/demos/tts.shtml machen.

Das Speech Synthesis System Festival (http://www.cstr.ed.ac.uk/projects/festival/) eignet sich gerade noch so für englische Text, aber spätestens bei deutschen Texten lässt die Qualität sehr zu wünschen übrig. Als guten Kompromiss kann ich die Software der amerikanischen Firma Cepstral (http://www.cepstral.com/) empfehlen. Es gibt dort eine kostenlose Probeversion und eine recht günstige Vollversion.[3]Die hier vorgestellte Lösung baut auf der Cepstral Engine auf.[4]

3.1. Installation Cepstral Text-to-Speech

Auf der Webseite http://www.cepstral.com/downloads/ kann man die deutsche Sprach-Engine herunterladen (evtl. auf den Link "Additional Voices" klicken). Die entsprechende Datei (hier beispielhaft immer als `Cepstral_Matthias_i386-linux_4.1.2.tar.gz` bezeichnet) wird wie folgt installiert:

```
tar xvzf Cepstral_Matthias_i386-linux_4.1.2.tar.gz
cd Cepstral_Matthias_i386-linux_4.1.2
./install
```

3.2. Beispiele und Tests

Falls bei der Installation nicht anders angegeben, wird die Engine als `/opt/swift/bin/swift` installiert. Testen kann man dies mit folgender Eingabe auf der Linux-Kommandozeile:

```
/opt/swift/bin/swift -o /tmp/test.wav -p audio/sampling-rate=8000,audio/channels=1 "Dies ist ein Test."
```

Das Ergebnis kann man sich entweder mit einem Audioplayer oder mit Asterisk anhören. Dazu einfach in der `extensions.conf` folgenden Eintrag hinzufügen:

```
exten => 1234,1,Answer()
exten => 1234,2,Playback(/tmp/test)
exten => 1234,3,Hangup()
```

Um einen beliebigen Text ausgeben zu lassen, benutzen wir die System()-Applikation im Dialplan. Hier ein Beispiel:

```
exten => 1222,1,Answer()
exten => 1222,2,System(rm -rf /tmp/test.wav)
exten => 1222,3,System(/opt/swift/bin/swift -o /tmp/test.wav -p audio/sampling-rate=8000,audio/channels=1
exten => 1222,4,Playback(/tmp/test)
exten => 1222,5,Hangup()
```

3.3. Pausen in Texten

Cepstral benutzt SSML (Speech Synthesis Markup Language) innerhalb der Engine. So kann eine Pause im obigen Beispiel wie folgt eingefügt werden:

```
exten => 1222,1,Answer()
exten => 1222,2,System(rm -rf /tmp/test.wav)
exten => 1222,3,System(/opt/swift/bin/swift -o /tmp/test.wav -p audio/sampling-rate=8000,audio/channels=1
exten => 1222,4,Playback(/tmp/test.wav)
exten => 1222,5,Hangup()
```

[3]Die Engine der deutschen Stimme kostet zur Drucklegung USD 29,99 und ist online bestellbar.

Auch hier gibt es eine Online-Testversion unter http://www.cepstral.com/demos/.

[4]Wer sich ein wenig mit Festival beschäftigt, kann die hier gezeigten Beispiele leicht auf Festival übertragen. Dies gilt in gleicher Weise auch für jede andere Text-to-Speech-Engine.

Information zu SSML findet man beim W3C unter http://www.w3.org/TR/speech-synthesis/.

Kapitel 9. Asterisk und VoiceXML

VoiceXML ist eine auf XML basierende Auszeichnungssprache für die Implementierung interaktiver Sprachdialoge.

Tipp

VoiceXML ist ein W3C Standard. Die aktuelle Version lautet 2.0 und ist unter folgender Addresse nachzulesen: *http://www.w3.org/TR/voicexml20/*. Auf Ken Rehor's Webseite *World of VoiceXML* (*http://www.kenrehor.com/voicexml/*) sind die wichtigsten Links zum Thema VoiceXML zusammengefasst.

Die zugrunde liegende Hardware soll dabei für den Anwendungsprogrammierer verborgen bleiben (Plattformunabhängigkeit), so dass er sich voll und ganz auf das Design der eigentlichen Sprachapplikation konzentrieren kann. Um sich eine Vorstellung über die Funktionsweise von VoiceXML zu machen, ist es hilfreich einen Vergleich zu anderen Auszeichnungssprachen, wie z.B. HTML, zu ziehen:

VoiceXML-Dokumente liegen genau wie HTML-Dokumente auf einem Webserver und werden über das HTTP Protokoll heruntergeladen. Das Herunterladen erfolgt wie bei HTML durch einen Browser, einen sogenannten *Voicebrowser*. Dieser Voicebrowser ist aber im Gegensatz zu den hinlänglich bekannten Webbrowsern *nicht* irgendeine Software die auf einem Gerät des Endanwenders läuft. Vielmehr handelt es sich um einen sehr leistungsfähigen Computer, der einerseits am Telefonnetz, andererseits am Internet angeschlossen ist. Der Voicebrowser ist sozusagen das Verbindungsglied zwischen der Telefonwelt und der IP-Welt.

Wie ein konventioneller Webbrowser muss auch der Voicebrowser von einem Endanwender bedient werden. Während man aber einen Webbrowser mit den Augen betrachtet und mit Maus und Tastatur steuert, ruft man einen Voicebrowser über ein Telefon an, lauscht seinen Ansagen und steuert ihn durch Sprache bzw. das Drücken der Nummerntasten auf dem Telefon. Der Funktionsumfang eines Voicebrowsers umfaßt also im Wesentlichen:

- Entgegennehmen von Anrufen
- Herunterladen von VoiceXML-Dokumenten.
- Interpretation der VoiceXML Dokumente
- Sprachsynthese (TTS - Text-to-Speech)
- Automatische Spracherkennung (ASR - Automatic speech recognition)
- Erkennung von DTMF Eingaben (Dual Tone Multi Frequency)
- Navigation zu weiteren VoiceXML Dokumenten
- Aufnehmen von Nachrichten des Anrufers
- Weitervermittlung des Anrufs

Bei näherer Betrachtung der Liste fällt auf, dass es sich hierbei um Funktionen handelt, für die der Asterisk eigentlich prädestiniert wäre.

Warnung

Zur Zeit der Drucklegung dieses Buches gibt es leider noch kein fertiges VoiceXML Add-On für Asterisk. Wer sich bis dahin schon einmal mit dem Thema VoiceXML auseinandersetzen will, der sei an dieser Stelle auf das Hastenix-AGI-Skript von *http://hastenix.hawhaw.de/* verwiesen.

1. Asterisk und Hastenix

Hastenix steht für *Hawhaw Adapter for aSTErisk aNd voIceXml*. Das *HAWHAW* Toolkit hat eigentlich nichts mit Asterisk zu tun, sondern beschäftigt sich mit der Erstellung von Webapplikationen für mobile Endgeräte. Neben

anderen Auszeichnungssprachen wie HTML und WML, unterstützt HAWHAW auch die Ausgabe von VoiceXML. Die eigentliche Intention des Hastenix Skripts ist es, einen Asterisk dahin gehend zu erweitern, dass er als Voicebrowser für HAWHAW-Anwendungen eingesetzt werden kann.

Das Skript unterstützt nur einen kleinen Teil des gesamten VoiceXML Sprachumfangs. Um einen ersten Einblick in die Thematik zu bekommen ist es aber dennoch hilfreich. Alle Beispiele der folgenden Kapitel wurden mit dem Skript erfolgreich getestet und können als Basis für eigene VoiceXML Anwendungen auf dem Asterisk dienen. Ein Anleitung zur Installation des Skripts und zur Einbindung in den Asterisk Dialplan findet sich auf der Webseite *http://hastenix.hawhaw.de/*.

Systemvoraussetzungen für den Einsatz des Hastenix Skripts sind:

- Asterisk 1.2 oder höher
- Perl 5.8 mit Thread Support
- Eine beliebiges TTS-System, z.B. Festival, Cepstral, MBROLA, usw.

Um das Hastenix Skript auf Ihrem Asterisk zu installieren gehen Sie wie folgt vor:

1. Laden Sie das Perlskript hastenix.pl von *http://www.hawhaw.de/download/hastenix.pl* mit **wget http://www.hawhaw.de/download/hastenix.pl** herunter und speichern Sie es in Ihrem AGI Verzeichnis (`/var/lib/asterisk/agi-bin/`) ab.

```
big-island:~# cd /var/lib/asterisk/agi-bin/
big-island:/var/lib/asterisk/agi-bin# wget http://www.hawhaw.de/download/hastenix.pl
--12:07:40--  http://www.hawhaw.de/download/hastenix.pl
           => `hastenix.pl'
Resolving www.hawhaw.de... 82.165.68.110
Connecting to www.hawhaw.de|82.165.68.110|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23,378 (23K) [text/plain]

100%[====================================>] 23,378        --.--K/s

12:07:40 (4.55 MB/s) - `hastenix.pl' saved [23378/23378]

big-island:/var/lib/asterisk/agi-bin#
```

2. Setzen Sie mit **chmod 755 hastenix.pl** die Dateirechte um das Perlskript ausführbar zu machen:

```
big-island:/var/lib/asterisk/agi-bin# chmod 755 hastenix.pl
big-island:/var/lib/asterisk/agi-bin#
```

3. Passen Sie den Konfigurationsteil am Anfang des Skripts Ihren Systemvoraussetzungen an. Setzen Sie die `$DIALOUT` variable auf *`Busy`*, um die Dialout-Funktionalität zu deaktivieren.

4. Binden Sie das Skript wie folgt in Ihren Dialplan ein:

```
;
; Hastenix Beispiel zur Sprachausgabe mit TTS
;
exten => 4291,1,Answer
exten => 4291,2,AGI(hastenix.pl|http://hastenix.hawhaw.de/aaw/sprachausgabe.vxml)
exten => 4291,3,Hangup
```

5. Starten Sie Ihren Asterisk im Konsolenmodus mit einem hohen Verbose-Level (mindestens 5):

   asterisk -vvvvvvc

Warnung

Wenn Sie sich mit der -r Option auf einen bereits laufenden Asteriskprozess reconnecten, werden aufgrund eines Asterisk Bugs evtl. Fehlermeldungen im Konsolenfenster nicht angezeigt. Sie können aber mit **set verbose 5** auch hier den Verbose-Level hochstellen.

6. Wählen Sie die 4291, beobachten Sie die Ausgaben im Konsolenfenster und lauschen Sie am Telefonhörer.

7. Installieren Sie evtl. fehlende Perlmodule nach, falls eine entsprechende Fehlermeldung im Konsolenfenster erscheint.

8. Fertig! Wenn alles klappt hören Sie einen Begrüßungstext.

2. Sprachausgabe

Beginnen wir mit einem möglichst einfachen VoiceXML Dokument:

Beispiel 9.1. sprachausgabe.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
<form>
<block><prompt>Willkommen bei der Apfelmus GmbH<break time="1000ms"/></prompt></block>
<block><prompt>Hier spricht der Asterisk.</prompt></block>
</form>
</vxml>
```

Die erste Zeile kommt jedem bekannt vor, der sich schon einmal mit XML beschäftigt hat. Sie hat noch nichts mit VoiceXML zu tun, sondern besagt lediglich, dass es sich um ein UTF-8 codiertes XML-Dokument handelt.

Der zweite Zeile mit dem XML Root-Element kommt dagegen schon zum Thema: Sie sagt dem Voicebrowser, dass es sich um ein VoiceXML Dokument der Version 2.0 handelt. Man sieht in diesem Element häufig auch diverse Namespace-Attribute. Erforderlich sind diese jedoch nicht und in den hier aufgezeigten Beispielen wollen wir der Einfachheit halber darauf verzichten.

Das <form> Element wird den Einen oder Anderen an dieser Stelle vielleicht verwundern. Mit einem Formular, wie man es von HTML kennt, hat es nämlich nicht viel gemeinsam. Wie man im Beispiel sieht, erfolgt hier lediglich eine Ausgabe von Text, ohne dass irgendwelche Daten aufgesammelt oder abgeschickt würden. Wenn man die Analogie zu HTML sucht, entspricht das <form> Element von VoiceXML eigentlich mehr einem HTML <body> Element.

Tipp

Dem interessierten Leser sei an dieser Stelle gesagt, dass es alternativ zu `<form>` in VoiceXML auch noch ein `<menu>` Element gibt. Es handelt sich hierbei um eine Spezialform des `<form>` Elements, die uns an dieser Stelle nicht weiter interessieren soll.

Die folgenden Element sind mehr oder weniger selbsterklärend: Innerhalb eines <form> Elements können mehrere <block> Elemente enthalten sein. In <block> Elementen können <prompt> Elemente stehen, die eine Sprachausgabe über das Text-to-Speech System des Voicebrowser bewirken. Ein Voicebrowser, der von einem Webserver das obige Dokument empfangen hat, sagt zuerst "Willkommen bei der Apfelmus GmbH", wartet dann eine Sekunde und sagt dann "Hier spricht der Asterisk".

Text-to-Speech Systeme für den Asterisk sind mittlerweile in recht guter Qualität verfügbar. Als die am weitesten verbreiteten Produkte sind hier Festival, MBROLA und Cepstral zu nennen (siehe Abschnitt 3, „Text2Speech (TTS)“).

3. Abspielen von Sounddateien

Wem die Computerstimme eines Text-to-Speech Systems zu synthetisch klingt, der kann unter Umständen auf das Abspielen vorab aufgenommener Sounddateien ausweichen. Es hängt natürlich stark von der jeweiligen Anwendung ab, ob sich die Sprachausgaben aus einer überschaubaren Anzahl von Sounddateien realisieren lassen oder ob der Einsatz von TTS unvermeidlich ist. Wenn man sich für die Variante mit den Sounddateien entscheidet, bietet VoiceXML folgendes Sprachkonstrukt an:

Beispiel 9.2. sounddatei.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
<form>
<block><prompt><audio src="demo-thanks.gsm">Vielen Dank!</audio></prompt></block>
</form>
</vxml>
```

Im Beispiel wird eine wohlbekannte gsm-Datei aus der Asterisk Demo abgespielt. Man beachte, dass sich sowohl die VoiceXML-Datei als auch die Sounddatei auf einem entfernten Webserver und nicht im lokalen Filesystem des Asterisk befinden. Das <audio> Element innerhalb des <prompt> Elements weist den Voicebrowser an, die Datei `demo-thanks.gsm` zu laden und abzuspielen. Falls aus technischen Problemen ein Abspielen der angegebenen Sounddatei nicht möglich sein sollte, wird der im <audio> Element vorhandene Text per TTS ausgegeben. Solange jedoch alles klappt, ist dieser Text ohne Bedeutung.

Das Format der Sounddatei spielt aus VoiceXML-Sicht keine Rolle. Der zur Anwendung kommende Voicebrowser muss das Format allerdings unterstützen. Für eine Asterisk-basierte VoiceXML Lösung bietet sich daher das gsm-Format an. Im einfachsten Fall wird man mit dem **record** Kommando einen Text aufnehmen und die erzeugte Datei im Anschluß auf den Webserver hochladen. Wer jedoch auf ein professionelles Auftreten Wert legt, sollte seine Aufnahmen in guter Qualität über ein Tonstudio anfertigen lassen.

Mit dem Hastenix Skript können Sie dieses Beispiel anhören, wenn Sie Ihren Dialplan wie folgt erweitern und danach die 4292 wählen:

```
;
; Hastenix Beispiel zum Abspielen von Sounddateien
;
exten => 4292,1,Answer
exten => 4292,2,AGI(hastenix.pl|http://hastenix.hawhaw.de/aaw/sounddatei.vxml)
exten => 4292,3,Hangup
```

4. DTMF Eingabe

Ein wesentlicher Bestandteil von VoiceXML ist das Thema Spracherkennung und die Definition von Grammatiken. Eine Sprachanwendung wird natürlich erst dann interaktiv, wenn der Anrufer durch eigene Eingaben in den Ablauf eingreifen kann. Mit den bisher behandelten Funktionen ließe sich ja gerade einmal eine Ansagemaschine realisieren.

Da der Asterisk mit Bordmitteln jedoch keine Spracherkennung unterstützt , wollen wir auf dieses Thema nicht näher eingehen, sondern uns dem kleinen Bruder der Spracherkennung zuwenden: Der DTMF-Erkennung.

Ein DTMF-Signal wird erzeugt, wenn der Anrufer während eines Sprachdialogs eine Taste seines Telefons drückt. In den vorherigen Kapiteln zum Thema Dialplan haben wir gesehen, dass der Asterisk hervorragend mit DTMF-Eingaben während einer Verbindung umgehen kann. Schreiben wir ein rudimentäres Sprachportal der Apfelmus GmbH einmal in VoiceXML-Schreibweise:

Beispiel 9.3. dtmf.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
<form>
<block><prompt>Willkommen im Sprachportal der Apfelmus GmbH
<break time="300ms"/></prompt></block>
<block><prompt>Für eine Beschreibung unserer neuen Produkte drücken Sie bitte die eins.
  <break time="300ms"/></prompt></block>
<block><prompt>Für allgemeine Informationen über unsere Firma drücken Sie bitte die zwei.
  <break time="300ms"/></prompt></block>
<field><prompt timeout="10s"/><noinput><exit/></noinput></field>
</form>
<link next="products.vxml" dtmf="1"></link>
<link next="info.vvxml" dtmf="2"></link>
</vxml>
```

Die ersten Zeilen dieses Dokuments verstehen wir bereits. Nach den drei Ansageblöcken kommt aber jetzt ein <field> Element. Dieses Element signalisiert dem Voicebrowser, dass er jetzt auf eine Eingabe des Benutzers warten soll. Bei dem <noinput> Element handelt es sich um ein sogenanntes Event, das der Voicebrowser in bestimmten Situationen sendet. In unserem Fall sendet er es dann, wenn 10 Sekunden lang keine Eingabe erfolgte. Das <exit> Element überträgt die Kontrolle wieder an den Interpreter, was hier nichts anderes heisst, als dass die Verbindung beendet wird.

Nach dem bisher gesagten würde der Voicebrowser anhand dieses Dokuments 3 Ansagen spielen, 10 Sekunden warten und dann die Verbindung beenden. Wenn da nicht noch die <link> Elemente am Ende des Dokuments wären! Hier wird nämlich dem Voicebrowser mitgeteilt was er tun soll, wenn er im Verlauf der Verbindung das Drücken der Taste 1 bzw. 2 erkennt: Er soll in diesem Fall das aktuelle VoiceXML Dokument verlassen und ein neues VoiceXML Dokument laden und ausführen.

Erwähnenswert ist noch die Position der <link> Elemente im Dokument. Je nach dem wo diese Elemente positioniert sind, entscheidet der Voicebrowser ob das Erkennung eines DTMF-Tones berücksichtigt werden soll oder nicht. In unserem Beispiel stehen die Links im Kontext des gesamten Dokuments. Somit muss der Voicebrowser während der gesamten Verbindung ein Ohr darauf haben, ob der Anrufer eine der Tasten 1 bzw. 2 gedrückt hat.

Mit dem Hastenix Skript können Sie dieses Beispiel interaktiv testen, wenn Sie Ihren Dialplan erneut erweitern und danach die 4293 wählen:

```
;
; Hastenix Beispiel zur DTMF-Eingabe
;
exten => 4293,1,Answer
exten => 4293,2,AGI(hastenix.pl|http://hastenix.hawhaw.de/aaw/dtmf.vxml)
exten => 4293,3,Hangup
```

5. Sprachaufnahme

Im Laufe eines Sprachdialogs möchte man manchmal dem Anrufer die Gelegenheit geben eine Nachricht zu hinterlassen. Während die Voicemailfunktion des Asterisk aufgenommene Nachricht im lokalen Filesystem speichert, hat man mit VoiceXML die Möglichkeit die aufgenommene Spracheingabe auf den Webserver hochzuladen und dort zu verarbeiten.

Erstellen wir also eine Beispielanwendung, bei welcher der Anrufer aufgefordert wird eine Nachricht zu hinterlassen und spielen wir ihm danach seine Worte noch einmal vor.

5.1. Upload von Sprachaufnahmen zum Webserver

Die Durchführung der Aufnahme und das anschließende Hochladen auf einen entfernten Webserver kann mit folgendem Dokument realisiert werden:

Beispiel 9.4. aufnahme-1.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
<form>
<record name="aufnahme" beep="true">
<prompt>Bitte hinterlassen Sie Ihre Nachricht nach dem Piepton.</prompt>
<filled>
<submit next="aufnahme-2-vxml.php" enctype="multipart/form-data" method="post" namelist="aufnahme"/>
</filled>
<catch event="connection.disconnect.hangup">
<submit next="aufnahme-2-vxml.php" enctype="multipart/form-data" method="post" namelist="aufnahme"/>
</catch>
</record>
</form>
</vxml>
```

Das <record> Element ist wie das <field> Element ein Eingabeelement innerhalb eines <form> Elements. Die Angabe eines Namens ist hier notwendig, um die Aufnahme später zum Webserver abschicken zu können und um die empfangenen Daten dort weiter verarbeiten zu können. Darüber hinaus verdienen die folgenden optionalen Attribute Erwähnung:

- beep (true/false): Wenn dieses Attribut auf "true" gesetzt wird, erzeugt der Voicebrowser einen kurzen Piepton bevor die Aufnahme beginnt.

- maxtime: Die Aufnahme endet wenn der im maxtime-Attribut angegebene Zeitwert überschritten wurde. Zeitwerte müssen in VoiceXML prinzipiell immer mit der richtigen Maßeinheit angegeben werden, z.B. "60s" oder "600ms".

- finalsilence: Hier kann man angeben nach wieviel Sekunden Stille die Aufnahme als beendet betrachtet werden soll. Hier gilt es einen vernünftigen Mittelwert zu finden. Zu kleine Werte beenden die Aufnahme vielleicht vorzeitig, weil der Anrufer nur mal Luft geholt oder kurz nachgedacht hat. Zu lange Werte spannen die Geduld des Anrufers unnötig auf die Probe, da er nach dem Ende seiner Ansprache keinen Fortschritt wahrnimmt. Die Voicebrowser haben normalerweise einen guten Mittelwert als Defaultwert fest voreingestellt, so dass man dieses Attribut nicht unbedingt setzen muss.[1]

Für das Absenden der Aufnahmedaten werden im obigen Dokument zwei Kriterien definiert:

1. Der Anrufer sagt nichts mehr oder drückt eine Taste seines Telefons. In diesem Fall kommt das <filled> Element zum Zug und sendet die aufgenommenen Sprachdaten zum im <submit> Element angegebenen PHP-Skript.

2. Der Anrufer legt nach seiner Ansprache auf. Auch in diesem Fall soll die Aufnahme nicht verloren gehen, sondern per <submit> zum Webserver übertragen werden. Man erreicht dies, indem man in einem <catch> Element das Event "connection.disconnect.hangup" fängt, welches die VoiceXML-Plattform aussendet, sobald sie das Auflegen des Anrufers erkannt hat.

Man sieht, dass die <submit> Anweisung für beide Fälle völlig identisch ist. Im next-Attribut wird das empfangende Skript angegeben. Die Sendemethode sollte bei Aufnahmen immer "post", der encoding type immer "multipart/form-data" lauten. Im namelist-Atttribut ist genau der Name einzutragen, der innerhalb des <record> Element im name-Attribut angegeben wurde.

5.2. Verarbeitung von Sprachaufnahmen auf dem Webserver

Wir haben im letzten Kapitel gesehen, dass die Sprachaufnahme mit der HTTP-Post Methode zu einem Skript namens `aufnahme-2-vxml.php` gesendet wird. Dieses Vorgehen ist vergleichbar mit dem Hochladen einer Datei vom Webbrowser aus. In beiden Fällen muß der Webserver Aktionen einleiten, um die empfangenen Daten zu verarbeiten. Dies kann auf unterschiedlichste Arten geschehen, je nach dem welche serverseitige Technologie zum Einsatz kommt. In unserem Beispiel verwenden wir ein kleines PHP-Skript:

[1] Der Anrufer kann die Aufnahme auch durch das Senden eines DTMF-Tones beenden, nur sollte man ihm dies vorher mitteilen.

Beispiel 9.5. aufnahme-2-vxml.php

```
<?php
  echo '<?xml version="1.0" encoding="UTF-8"?>';

  $filename = $_FILES['aufnahme']['name'];
  move_uploaded_file($_FILES['aufnahme']['tmp_name'],
                     "./aufnahmen/" . $filename);
?>

<vxml version="2.0">
<form>
<block><prompt>Ihre Nachricht war:<break time="300ms"/></prompt></block>
<block><prompt>
<audio src="./aufnahmen/<?php echo $filename; ?>"></audio>
<break time="300ms"/>
</prompt></block>
</form>
</vxml>
```

Das Beispielscript speichert die empfangene Sounddatei im Pfad `aufnahmen` ab. Alternativ könnte man die Aufnahme natürlich auch in einer Datenbank abspeichern oder als Email versenden. Bei der obigen Variante sollte man allerdings darauf achten, dass der Webserver die erforderlichen Zugriffsrechte hat, um in das `aufnahmen`-Verzeichnis schreiben zu können.

Damit der Anrufer seine aufgenommenen Worte hören kann, sendet der Webserver ein passendes VoiceXML Dokument zurück. Der Aufbau dieses Dokuments mit dem <audio> Element wurde schon im Kapitel "Abspielen von Sounddateien" behandelt.

Mit dem Hastenix Skript können Sie dieses Beispiel interaktiv testen, indem Sie Ihren Dialplan einmal mehr erweitern und die 4294 wählen:

```
;
; Hastenix Beispiel zur Sprachaufnahme
;
exten => 4294,1,Answer
exten => 4294,2,AGI(hastenix.pl|http://hastenix.hawhaw.de/aaw/aufnahme-1.vxml)
exten => 4294,3,Hangup
```

Anmerkung

Bevor Sie während des Tests eventuell unüberlegte Äußerungen machen: Seien Sie sich bewußt, dass Ihre Worte auf den HAWHAW-Server übertragen und dort wie oben beschrieben abgespeichert werden!

6. Anrufweiterleitung

Erinnern wir uns an das Kapitel zum Voicemailsystem des Asterisk und an die Einrichtung der verschiedenen Mailboxen für die Familie Meier. Nehmen wir einmal an der Asterisk der Familie Meier ist nur über eine Nummer von außen erreichbar, und die Mitglieder der Familie Meier wollen ein interaktives Anrufverteilsystem einsetzen. Das folgende Beispiel zeigt, wie so etwas mit VoiceXML realisiert werden könnte:

Beispiel 9.6. weiterleitung.vxml

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0">
<form>
<block><prompt>Hier ist der Telefonanschluss der Familie Meier.<break time="300ms"/></prompt></block>
<block><prompt>Druecken Sie die 1, um mit Horst Meier verbunden zu werden.<break time="300ms"/></prompt><
<block><prompt>Druecken Sie die 2, um mit Eva Meier verbunden zu werden.<break time="300ms"/></prompt></b
<block><prompt>Druecken Sie die 3, um mit Peter Meier verbunden zu werden.<break time="300ms"/></prompt><
<block><prompt>Oder druecken Sie die 4, um mit Lisa Meier verbunden zu werden.<break time="300ms"/></prom
<field><prompt timeout="10s"/><noinput><exit/></noinput></field>
</form>
<link next="#transfer1" dtmf="1"></link>
<form id="transfer1">Man
<transfer dest="tel:200" bridge="false" />
</form>
<link next="#transfer2" dtmf="2"></link>
<form id="transfer2">
<transfer dest="tel:201" bridge="false" />
</form>
<link next="#transfer3" dtmf="3"></link>
<form id="transfer3">
<transfer dest="tel:202" bridge="false" />
</form>
<link next="#transfer4" dtmf="4"></link>
<form id="transfer4">
<transfer dest="tel:203" bridge="false" />
</form>
</vxml>
```

Am Anfang des Dokumentes gibt der Voicebrowser Instruktionen an den Anrufer, was dieser zu tun hat, um seinen gewünschten Gesprächspartner zu erreichen. Das <field> Element, welches nach 10 Sekunden Untätigkeit die Verbindung beendet, kennen wir bereits aus dem Kapitel zur DTMF-Eingabe.

Auch das <link> Element haben wir dort kennengelernt. Allerdings ist hier die Syntax nun ein klein wenig anders. Der Unterschied liegt im #-Zeichen. Ein next-Attribut ohne Lattenzaun bewirkt das Laden eines neuen VoiceXML Dokuments, während bei einem next-Attribut mit Lattenzaun ein anderer Dialogabschnitt innerhalb des selben Dokuments angesprungen wird. Kommt Ihnen das irgendwie bekannt vor? Bei HTML-Dokumenten gibt es beim <a> Element eine ähnliche Logik!

Wie man in diesem Beispiel sieht, können durchaus mehrere <form> Elemente in einem VoiceXML Dokument existieren. Sie werden durch eine eindeutige ID unterschieden und über die DTMF-getriggerten <link> Elemente angesprungen. Die genaue Position der <link> Elemente im Dokument ist nur insofern von Bedeutung, als dass sie im Kontext des <vxml> Elements stehen und damit permanent aktiv sind. Man hätte die <link> Elemente aber auch zu einem Linkblock zusammenfassen können, ohne etwas an der Funktionalität zu ändern.

Wenn nun durch das Erkennen eines DTMF-Tons eines der unteren <form> Element erreicht wurde, leitet der Voicebrowser den Anruf an die angegebene Destination weiter. Das dest-Attribut muß gemäß RFC 2806 (URLs for Telephone Calls) http://www.ietf.org/rfc/rfc2806.txt angegeben werden.

Das Attribut *`bridge="false"`* bedeutet, dass sich der Voicebrowser nicht weiter um diese Verbindung kümmert, sondern mit der Weiterleitung des Anrufs an die angegebene Nummer seine Schuldigkeit getan hat.

Mit dem Hastenix Skript können Sie auch dieses Beispiel interaktiv testen. Sie werden aber nur dann erfolgreich weitervermittelt werden, wenn Sie auf Ihrem Asterisk entsprechende interne Teilnehmer mit den Rufnummern 200 bis 203 eingerichtet haben und die `$DIALOUT` Konfigurationsvariable so gesetzt haben, dass die Ausführung des **Dial** Kommandos erlaubt ist.

```
;
; Hastenix Beispiel zur Weitervermittlung
;
exten => 4295,1,Answer
exten => 4295,2,AGI(hastenix.pl|http://hastenix.hawhaw.de/aaw/weiterleitung.vxml)
exten => 4295,3,Hangup
```

Warnung

Beachten Sie, dass es sich bei einem Anruf-Transfer um ein sehr mächtiges Feature handelt, das aber bei falscher Anwendung nicht ungefährlich ist! Machen Sie sich bewußt, dass mit dem <transfer> Element ein entfernter Webserver die Kontrolle darüber hat, welche abgehenden Verbindungen ein Voicebrowser aufbaut. Im Hastenix-Skript können Sie jegliche Nutzung des <transfer> Elements dadurch unterbinden, dass Sie die `$DIALOUT` Variable im Konfigurationsteil mit dem Befehl **Busy** vorbelegen.

7. Fazit

VoiceXML hat das Potential die Programmierung von Sprachdialogen auf dem Asterisk einmal wesentlich zu vereinfachen. An die Stelle komplizierter und asterisk-proprietärer Dialplansequenzen in der `/etc/asterisk/extensions.conf`, treten standardisierte XML-Dokumente, die auf beliebigen Webservern verteilt werden können.

Komplexe Sprachanwendungen, die sich über viele VoiceXML Dokumente erstrecken, können leicht von einer Plattform auf eine andere portiert werden und müssen nicht dafür umprogrammiert werden.

Das Thema VoiceXML ist recht komplex und erfordert eine intensive Einarbeitung in die zugrunde liegenden Standards. Wer sich unabhängig von einer Asterisk Implementierung einmal näher mit dem Thema beschäftigen will, der sei auf die umfangreichen Entwickler-Programme der diversen VoiceXML Provider verwiesen. Mehrere Firmen bieten kostenlose Developer Accounts zum Testen eigener VoiceXML-Anwendungen an. Besonders genannt seien hier die Firmen *Voxeo* und *Tellme*.

Kapitel 10. Warteschleifen für Call-Center

Warteschleifen sind der Horror vieler Kunden. Trotzdem sind sie in kaum einer Firma mit einem telefonischen Kunden-Service mehr wegzudenken, daher soll hier beschrieben werden, wie man eine Warteschleife anlegt und welche Einstellungen möglich sind.[1]

Hier sind für uns insbesondere 4 Dateien wichtig:

`queues.conf`	Definiert Warteschlangen.
`agents.conf`	Definiert „Agenten“, also die Mitarbeiter, die die Anrufe entgegennehmen.
`musiconhold.conf`	Definiert die Wartemusik.
`extensions.conf`	Im Dialplan werden Anrufe mit der `Queue()`in eine Schlange eingereiht und Agenten mit `AgentLogin()` oder `AgentCallbackLogin()` am System registriert.

Eine häufige Ursache für Missverständnisse ist die Verwechslung von „Queue Members“ mit Anrufern. Mit den Mitgliedern einer Warteschlange sind immer nur die Agenten, *nicht* die Anrufer gemeint.[2]

Agenten können mehreren Schlangen angehören, und wir wollen, dass sich jeder Agent von einer beliebigen Extension aus einloggen kann („Hot-desking“). Das grundsätzliche Vorgehen zur Definition von Agenten und Warteschlangen sieht vereinfacht so aus:

`agents.conf`

```
[agents]
;    AgentenNr,Passwort,Name
agent => 1001,1234,Mark Spencer
agent => 1002,1234,Will Meadows
```

`queues.conf`

```
[support-schlange]
member => Agent/1001   ; Agent 1001 zur support-schlange hinzufügen
member => Agent/1002   ;   ... 1002 ...
```

`extensions.conf`

```
exten => 20,1,Queue(support-schlange)   ; => Warteschlange
exten => 25,1,AgentLogin()              ; bei Anruf Login
```

Und so funktioniert's: Die Agenten loggen sich über Extension `25` ein, hören fröhliche Musik und warten auf Anrufer. Anrufe auf Extension `20` werden in die Schlange aufgenommen und der Reihe nach (FIFO) abgearbeitet: Der Agent hört einen Piepton, und der erste Anrufer wird verbunden.

1. Extension zum Musiktest

Wir legen uns in der `extensions.conf` folgende Extension an, mit der wir gleich die ordnungsgemäße Funktion der Wartemusik überprüfen können:

```
exten => 222,1,Answer()                  ; abheben
exten => 222,n,Set(MUSICCLASS()=default) ; Musikklasse "default" setzen
```

[1] Ein Beispiel-Setup in AEL von Digium wird ab Asterisk 1.4 in `doc/queues-with-callback-members.txt` beschrieben.
[2] Hier wird der Einfachheit halber der Begriff „Agenten“ verwendet, Markus Bönke macht aber folgende Anmerkung:

> Sie übersetzen Queue Member mit "Agenten", wenn man aus der CallCenter-Welt kommt, ist das ein bisschen verwirrend, da QueueMember durchaus normale SIP-Extensions sein können.

```
exten => 222,n,WaitMusicOnHold(20)       ; 20 Sekunden Musik
exten => 222,n,Hangup()                  ; auflegen
```

2. `musiconhold.conf`

Beginnen wir mit dem Einfachsten, der Festlegung der Wartemusik (Music on Hold).

> Bitte beachten Sie, dass für das Abspielen von Musik ggf. Gebühren an die *GEMA* zu entrichten sind. Freie klassiche Werke finden Sie u.A. auf *http://www.classiccat.net/* zum Download. Oder Sie verwenden die 3 mitgelieferten Stücke.

Die Einstellmöglichkeiten werden in der Datei `musiconhold.conf` beschrieben, ein paar Infos zur MP3-Unterstützung durch das Programm **mpg123** stehen in `doc/README.mp3` (1.2) / `doc/mp3.txt` (1.4). Die optimale Abtastrate für die MP3s ist 8000 Hz, mono.

Die einfachste (Default-)Konfiguration mit nur einer Musik-Klasse (`default`):

```
[default]
mode=quietmp3
directory=/var/lib/asterisk/mohmp3
```

Jetzt stoppen und starten wir Asterisk, z.B. mit **asterisk -rx "stop now"** und **asterisk** um die neue Extension und die Streams zu aktivieren (ein Reload reicht nicht aus). Wenn Sie jetzt die Extension `222` (wie oben definiert) anrufen, sollten Sie Wartemusik hören (im o.g. Verzeichnis werden bereits 3 Dateien mitgeliefert).

Besser ist aber folgende Konfiguration, für die wir ein Verzeichnis `moh-native` in `/var/lib/asterisk/` anlegen:

```
[default]
mode=files   ; Dateien direkt in einem Asterisk-kompatiblen Format lesen
directory=/var/lib/asterisk/moh-native
random=yes   ; Dateien in zufälliger Reihenfolge spielen
```

Es ist dazu notwendig, die MP3-Dateien in ein Asterisk-kompatibles Format zu konvertieren, was z.B. mit den Programmen **sox**[3] und **lame**[4] so geht:

```
$  lame --decode musik.mp3 musik.wav
$  sox -V musik.wav -r 8000 -c 1 -w musik.raw
$  sox -V musik.wav -r 8000 -c 1 -w musik.gsm
```

Damit haben wir eine fiktive Datei `musik.mp3` ins WAV- und dann in die Formate RAW und GSM konvertiert (mit einer Abtastrate von 8 kHz). Wir können in unserem Verzeichnis auch Formate wie aLaw und uLaw (Dateiendungen für **sox**: `.al` und `.ul`) ablegen - Asterisk wird automatisch das am wenigsten CPU-intensive wählen. Wichtig ist, dass die Abtastrate immer 8000 Hz beträgt.

---

[3] **sox** (http://sox.sourceforge.net/) gibt es als Debian-Paket, also:

```
$  apt-get install sox
```

Für MacOS X gibt es - mit installierem Darwinports (http://darwinports.opendarwin.org/ oder http://darwinports.com/, siehe auch http://apfelwiki.de/wiki/Main/MacPorts) - auch ein Paket:

```
$  port install sox
```

[3] Auf anderen Plattformen müssen Sie die Sourcen runterladen und kompilieren, sofern Sie keine Binaries finden.

[4] **lame** (http://lame.sourceforge.net/) gibt es nicht als Debian-Paket, wir müssen als die Sourcen runterladen und mit den bekannten Befehlen **./configure**, **make** und **make install** kompilieren und installieren.

[4] Für MacOS X gibt es ein Paket über Darwinports (siehe Beschreibung **sox**):

```
$  port install lame
```

oder über Fink (http://fink.sourceforge.net/):

```
$  fink install lame
```

Natürlich geht das auch über den FinkCommander mit grafischem Interface.

Wichtig

Wenn Sie in der `modules.conf` nicht `autoload` verwenden, müssen Sie sicherstellen, dass die Module für die Formate, die Sie hier verwenden wollen, *vor* `res_musiconhold` geladen werden.

In unserem Test haben wir die im Verzeichnis `moh/` mitgelieferten WAV-Dateien in unser Verzeichnis `moh-native/` kopiert, mit **sox** nach RAW und GSM umkodiert und dann die WAV-Dateien, die Probleme bereiteten, aus dem Verzeichnis gelöscht:

```
$  cd /var/lib/asterisk/moh-native/
$  for i in *.wav; do \
sox $i -r 8000 -c 1 $(basename $i .wav).raw; \
sox $i -r 8000 -c 1 $(basename $i .wav).gsm; \
done
$  rm *.wav
```

Für CPU-kritische Anwendungen (große Installationen) sollten Sie einen Blick auf die Beschreibung in `contrib/utils/README.rawplayer` im ursprünglichen Asterisk-Verzeichnis werfen.

Weitere Musikklassen lassen sich definieren, indem man außer dem `[default]`-Abschnitt noch weitere Abschnitte, z.B. `[rock]` anlegt, dort ein anderes Verzeichnis, z.B. ein Unterverzeichnis, angibt und die Musikdateien in dieses Verzeichnis legt. Reload nicht vergessen!

3. `queues.conf`

In der `queues.conf` werden Warteschlangen definiert und entsprechende Einstellungen vorgenommen. Die Datei ist - wie wir das schon kennen - in Abschnitte unterteilt.

Unter `[general]` stellen wir nur `persistentmembers = yes` ein, damit beim Neustart von Asterisk die Agenten automatisch wieder in ihre Schlangen aufgenommen werden.

Jede Warteschleife stellt einen eigenen Abschnitt dar. Wir legen bespielsweise eine Schlange im Abschnitt `[support]` an. Folgende Parameter stehen zur Verfügung:

3.1. `musiconhold`

Legt die in `musiconhold.conf` definierte Musik-Klasse der Warteschleife fest. Wir wählen die oben angelegte Klasse `default`:

```
musiconhold = default
```

3.2. `announce`

Legt eine Audio-Datei fest, die einem Agenten vor dem Beantworten eines Anrufs vorgespielt wird. Dies dient dazu, dass ein Agent, der auf mehreren Schlangen Anrufe entgegennimmt, weiß, um welche Schange es sich handelt und z.B. den Anrufer entsprechend begrüßen kann.

```
;announce = schlange-support
```

Wenn Sie diese Einstellung verwenden, denken Sie daran, auch eine entsprechende Audio-Datei im `sounds/`-Verzeichnis anzulegen, also z.B. `schlange-support.gsm`.

3.3. `strategy`

Legt fest, in welcher Weise und Reihenfolge die Agenten bei einem Anruf angeklingelt werden sollen:

`ringall`	Alle anklingeln bis einer antwortet. (Default)
`roundrobin`	Der Reihe nach alle anklingeln bis einer antwortet.
`leastrecent`	Das Interface anklingeln, bei dem der letzte Anruf am längsten zurückliegt.

`fewestcalls` Den anklingeln, der am wenigsten Anrufe in dieser Schlange abgearbeitet hat.

`random` Ein zufällig gewähltes Interface anklingeln.

`rrmemory` Round-Robin mit Gedächtnis. Beginnt die Reihe bei dem, der nach dem letzten Anruf an der Reihe ist.

Es ist zu beachten, dass Agenten, für die ein niedrigerer Malus (penalty) festgelegt wurde, immer Agenten mit höherem Malus vorgezogen werden.

```
strategy = ringall
```

Die Einstellung ist abhängig von Ihren Gegebenheiten: `ringall` ist vielleicht lästig, die anderen Werte verursachen zusätzliche Wartezeit für den Anrufer, falls Agenten nicht ans Telefon gehen.

3.4. `servicelevel`

Legt die Zeit (in Sekunden) fest, in der Anrufe beantwortet sein sollen. Nur für statistische Auswertungen interessant („Wieviele Anrufe wurden innerhalb der Service-Zeit von x Sekunden beantwortet?“).

```
servicelevel = 60
```

3.5. `context`

Hier kann ein Kontext angegeben werden, in dem, wenn der Anrufer eine Extension mit nur *einer* Ziffer drückt, aus der Schlange rausgenommen und zu dieser Extension in diesem Kontext weitergeleitet wird.

```
context = supportschlange-kontext  ; so könnten wir es einstellen

; aber wir lassen das erst mal weg, also auskommentiert:
;context = supportschlange-kontext
```

3.6. `timeout`

Legt fest, wie lange (in Sekunden) ein Telefon klingeln soll, bis wir es als nicht besetzt (also Timeout) betrachten.

```
timeout = 15
```

3.7. `retry`

Bestimmt, wie lange (in Sekunden) gewartet werden soll, bevor erneut alle Agenten angeklingelt werden.

```
retry = 5
```

3.8. `weight`

Das Gewicht (Wichtigkeit) der Schlange, relativ zu anderen Schlangen. Wenn ein Agent auf mehreren Schlangen ist, werden ihm zuerst Anrufe von Schlangen mit höherem Gewicht durchgestellt. So könnte z.B. einer Notfallschlange ein höheres Gewicht gegeben werden.

```
weight = 0
```

3.9. `wrapuptime`

Die Wartezeit (in Sekunden), bevor ein Agent, der gerade einen Anruf beendet hat, erneut einen Anruf erhält (Default: 0).

```
; wir lassen unseren Agenten eine kleine Verschnaufpause, um
; einen Schluck Wasser zu trinken:
wrapuptime = 10
```

3.10. `maxlen`

Maximale Anzahl der in der Schlange wartenden Anrufer (Default: 0 für unbegrenzt).

```
maxlen = 0
```

3.11. `announce-frequency`

Bestimmt, in welchen Abständen (in Sekunden) den Anrufern ihre Position in der Schlange und/oder die geschätzte Wartezeit angesagt werden soll (0 für aus).

```
announce-frequency = 90
```

3.12. `announce-holdtime`

Ob die geschätzte Wartezeit in den Ankündigungen nach der Position genannt werden soll. Mögliche Werte sind `yes` (ja), `no` (nein) oder `once` (nur einmal).

```
announce-holdtime = yes
```

3.13. `announce-round-seconds`

Bei welchem Wert (1 - 59) Sekunden auf ganze Minuten auf- oder abgerundet werden sollen (offenbar werden die Sekunden aber trotzdem genannt). Mit der Einstellung 0 schalten wir die Ansage der Sekunden aus.

```
announce-round-seconds = 0
```

3.14. Sprachbausteine

Die folgenden Parameter stellen ein, welche Sprachbausteine für die Ansagen der Position und Wartezeit verwendet werden. I.d.R sind hier keine Veränderungen notwendig:

```
queue-youarenext = queue-youarenext     ; "You are now first in line."
queue-thereare = queue-thereare         ; "There are" ...
queue-callswaiting = queue-callswaiting ; ... "calls waiting."
queue-holdtime = queue-holdtime         ; "The current est. holdtime is" ...
queue-minutes = queue-minutes           ; ... "minutes"
queue-seconds = queue-seconds           ; ... "seconds"
queue-thankyou = queue-thankyou         ; "Thank you for your patience."
queue-lessthan = queue-less-than        ; ... "less than" ...
queue-reporthold = queue-reporthold     ; "Hold time" ...
periodic-announce = queue-periodic-announce  ; "All reps busy, wait for next"
```

Wenn diese Parameter nicht angegeben werden (also auskommentiert sind), gelten diese Default-Werte.

3.15. `periodic-announce-frequency`

Bestimmt, in welchen Abständen (in Sekunden) den Anrufern die periodische Ankündigung (`periodic-announce`, s.o., „Zur Zeit sind alle Mitarbeiter im Gespräch, bitte warten Sie.“) angesagt werden soll.

```
periodic-announce-frequency = 60
```

3.16. `monitor-format`

Die Angabe dieses Parameters schaltet die Aufzeichnung von Gesprächen ein (wie mit der `Monitor()`-Applikation) und bestimmt das Aufzeichnungsformat. (Bei auskommentiertem `monitor-format` werden keine Mitschnitte gemacht.) Geben Sie hier also `gsm`, `wav` (große Dateien!) oder `wav49` an.

```
; wenn Sie Aufzeichnungen wollen, diese Zeile einkommentieren:
;monitor-format = gsm
```

Per Default werden die Dateien (eine `-in` und eine `-out`) nach `${UNIQUEID}` benannt. Sie können das bei Bedarf verändern, indem Sie im Dialplan vor dem `Queue()`-Aufruf `Set(MONITOR_FILENAME=`*`dateiname`*`)` angeben. Wir nehmen aber im Beispiel keine Veränderung vor.

Weitere Informationen über Mitschnitte siehe Abschnitt 6.54, „`Monitor()`“

3.17. `monitor-join`

Kombiniert die beiden durch einen Gesprächs-Mitschnitt erzeugten Dateien `...-in` und `...-out` zu einer Datei. Werte: `yes` oder `no`.

```
monitor-join = yes
```

Weitere Informationen über Mitschnitte siehe Abschnitt 6.54, „`Monitor()`“

3.18. `joinempty`

Legt fest, ob Anrufer in eine Warteschleife ohne Agenten eingereiht werden.

`yes` Anrufer können in eine Schleife ohne Agenten oder nur nicht-verfügbaren Agenten eingereiht werden.

`no` Anrufer können nicht in eine Schleife ohne Agenten eingereiht werden.

`strict` Anrufer können nicht in eine Schleife ohne Agenten eingereiht werden (aber in eine Schlange mit nur nicht-verfügbaren Agenten).

Nicht-verfügbar ist nicht zu verwechseln mit im Gespräch (busy). Ein Agent ist nicht verfügbar, wenn er zwar der Schlange zugewiesen, aber tatsächlich nicht am System registriert ist (siehe auch Abschnitt 3.25, „`member`“). Achtung: Statisch definierte Mitglieder (siehe Abschnitt 3.25, „`member`“) gelten immer als verfügbar!

Sollte ein Anrufer nicht in eine Schlange aufgenommen werden, ist die `Queue()`-Applikation beendet, und der Dialplan wird fortgesetzt.

```
joinempty = no  ; wir wollen unsere Anrufer nicht unnötig warten lassen
```

3.19. `leavewhenempty`

Legt fest, ob Anrufer aus einer Schlange entfernt werden, sobald sich alle Agenten ausgeloggt haben. Mögliche Werte wie bei `joinempty`. Nach dem Verlassen der Schlange wir der Dialplan fortgesetzt.

```
leavewhenempty = strict  ; Anrufer nicht unnötig warten lassen
```

3.20. `eventwhencalled`

Stellt ein (`yes`|`no`), ob u.a. die folgenden Events für das Manager-Interface erzeugt werden: AgentCalled, AgentDump, AgentConnect, AgentComplete.

```
eventwhencalled = yes
```

3.21. `eventmemberstatus`

Stellt ein (`yes`|`no`), ob QueueMemberStatus-Events im Manager-Interface erzeugt werden (das können viele sein!).

```
eventmemberstatus = no
```

3.22. `reportholdtime`

Legt fest (`yes`|`no`), ob dem Agenten die Wartezeit des Anrufers angesagt werden soll, bevor dieser zu ihm durchgestellt wird. (Geschmacksfrage)

```
reportholdtime = no
```

3.23. `memberdelay`

Stellt ein, wie lange (in Sekunden) der Anrufer Stille hört, bevor er mit einem Agenten verbunden wird.

```
memberdelay = 1
```

3.24. `timeoutrestart`

Bestimmt, ob der Antwort-Timeout eines Agenten bei einem Besetzt- oder Leitung-überlastet-Signal zurückgesetzt wird. Kann nützlich für Agenten sein, die die Berechtigung haben, einen Anruf abzuweisen.

```
timeoutrestart = yes
```

3.25. `member`

Es ist möglich direkt in der `queues.conf` Agenten **statisch** in der Form

```
member => Technologie/Resource[,Malus]
```

- also z.B. `member => Zap/2` - anzulegen (darf mehrmals verwendet werden, siehe `queues.conf`). Das kann jedoch zu Problemen mit `joinempty` und `leavewhenempty` führen, da diese Agenten *immer* als verfügbar angesehen werden, auch wenn sie in Wirklichkeit nicht an ihrem Apparat sind. Zudem hat es den Nachteil, dass ein Agent immer fest einem Apparat zugeordnet ist und sich nicht von einem anderen Apparat aus anmelden kann.

Wir verwenden daher lieber die **dynamische** Form und ordnen unserer Schlange `support` in der Form

```
member => Agent/AgentenNr
```

zwei Agenten `1001` und `1002` zu:

```
member => Agent/1001
member => Agent/1002
```

Diese Agenten müssen wir noch in der `agents.conf` definieren (s.u.)[5]. Die `AgentenNr` ist eine frei wählbare Nummer, die jeden Agenten eindeutig identifiziert. Sie hat nichts mit den Extensions zu tun, die die Agenten evtl. verwenden.

4. `agents.conf`

In der `agents.conf` werden die Agenten, also Call-Center-Mitarbeiter, die auch „Members“ genannt werden, definiert und entsprechende Einstellungen vorgenommen. Auch diese Datei ist wieder in Abschnitte unterteilt.

Unter `[general]` stellen wir `persistentagents = yes` ein, damit die Logins der Agenten persistent (dauerhaft) sind und in der Asterisk-Datenbank gespeichert werden. Die Agenten werden dadurch beim Neustart von Asterisk automatisch wieder in ihre Schlangen aufgenommen.

Im Abschnitt `[agents]` nehmen wir weitere Einstellungen vor und definieren dann die Agenten. Folgende Parameter stehen zur Verfügung:

4.1. `autologoff`

Hiermit können wir einstellen, wie lange (in Sekunden) ein Telefon klingeln muss, bis der Agent automatisch ausgeloggt wird.

Beachten Sie, dass Agenten nicht ausgeloggt werden, wenn diese Dauer über dem timeout (Abschnitt 3.6, „`timeout`“) der Schlange liegt! Da haben wir 15 Sekunden eingestellt, also wählen wir hier 14.

```
autologoff = 14
```

[5] sonst wären es wirklich Geheimagenten ;-)

4.2. `ackcall`

Stellt ein (yes|no), ob Agenten, die mit `AgentCallbackLogin()` eingeloggt wurden, die Raute (#) drücken müssen, um ein Gespräch anzunehmen. Es ist zu beachten, dass in bisherigen Asterisk-Versionen der Voiceprompt dem Agenten nicht mitgeteilt hat, dass er # drücken muss.

```
ackcall = no
```

4.3. `endcall`

Stellte ein (yes|no), ob Agenten Anrufe durch Drücken der Stern-Taste (*) beenden können.

```
endcall = yes
```

4.4. `wrapuptime`

Legt (wie `wrapuptime` in der queues.conf, die doppelte Angabe ist nicht klar) die Wartezeit fest, bis ein Agent, der gerade einen Anruf beendet hat, erneut einen Anruf erhält- hier allerdings in Millisekunden. Default: 5000.

```
wrapuptime = 5000   ; 5 Sekunden Pause
```

4.5. `musiconhold`

Legt die Music-on-Hold-Klasse für die Agenten fest.

```
musiconhold = default
```

4.6. `updatecdr`

Den CDR-Record anpassen (yes|no), so dass die Kanalangabe *`Agent/AgentenNr`* enthält, damit wir wissen, welcher Agent das Gespräch geführt hat.

```
updatecdr = yes
```

4.7. `recordagentcalls`

Gespräche mit Agenten aufzeichnen (yes|no). Default: no.

```
recordagentcalls = no
```

4.8. `recordformat`

Aufzeichnungsformat für Mitschnitte. `gsm`, `wav` (große Dateien!) oder `wav49`. Default: `wav`.

```
recordformat = gsm
```

4.9. `createlink`

Legt fest (yes|no), ob im User-field (dem Freitextfeld) des CDR der Dateiname der Aufzeichnung gespeichert werden soll, damit man sie später zuordnen kann.

```
createlink = yes
```

4.10. `urlprefix`

Wenn Sie `createlink` verwenden, kann mit `urlprefix` direkt eine URL vor den Dateinamen gesetzt werden, unter der sie die Aufzeichnungen mit einem Web-Server im Intranet verfügbar gemacht haben.

```
; so könnte ein Beispiel aussehen:
;urlprefix = http://astbox/anrufe/
```

4.11. `savecallsin`

Hier können Sie angeben, in welchem Verzeichnis die Mitschnitte gespeichert werden sollen. Default: `/var/spool/asterisk/monitor`

```
; wenn wir z.B. die Mitschnitte in /var/calls/ speichern wollen:
;savecallsin = /var/calls
```

4.12. `agent`

`agent` wird zur Definition von Agenten verwendet (kann mehrfach vorkommen).

```
; Format: agent => AgentenNr,Passwort,Name
agent => 1001,0000,Mark Spencer
agent => 1002,0000,Will Meadows
```

5. `extensions.conf`

Im Dialplan (Kapitel 4, *Programmieren im Dialplan*) können sich zum einen die Agenten über eine bestimmte Extension in die Warteschleife einloggen, zum anderen werden Anrufe mit der Applikation `Queue()` (Abschnitt 6.69, „`Queue()`“) in eine Schlange eingereiht.

Eine minimale Extension, die die Anrufer in die Warteschleife einreiht, sähe so aus:

```
exten => 20,1,Answer()                    ; abheben
exten => 20,n,Set(MUSICCLASS()=default)   ; Musikklasse "default" setzen
exten => 20,n,Queue(support,t)            ; in Schlange "support" einreihen
exten => 20,n,Hangup()                    ; hinterher auflegen
```

Wir lieben es etwas vollständiger und verwenden diese Extension:

```
exten => 20,n,Wait(2)
exten => 20,n,Answer()                    ; abheben
exten => 20,n,Set(MUSICCLASS()=default)   ; Musikklasse "default" setzen
exten => 20,n,Queue(support,t)            ; in Schlange "support" einreihen
; zur Erinnerung: die Option t erlaubt es dem Agenten, den Anruf
; an eine andere Extension weiterzuleiten
; setzt QUEUESTATUS =
; TIMEOUT | FULL | JOINEMPTY | JOINUNAVAIL | LEAVEEMPTY | LEAVEUNAVAIL
exten => 20,n,Goto(q-${QUEUESTATUS},1)  ; je nach QUEUESTATUS springen

; keine Agenten auf der Schlange:
; (hier könnte man alternativ auf VoiceMail() leiten)
exten => q-JOINEMPTY,1,Wait(1)
exten => q-JOINEMPTY,n,Playback(vm-nobodyavail,noanswer)
exten => q-JOINEMPTY,n,Playback(vm-goodbye,noanswer)
exten => q-JOINEMPTY,n,Hangup()

; keine Agenten (oder nur nicht-verfügbare) auf der Schlange:
exten => q-JOINUNAVAIL,1,Goto(q-JOINEMPTY,1)   ; wie JOINEMPTY behandeln

; alle Agenten haben Schlange verlassen:
exten => q-LEAVEEMPTY,1,Goto(q-JOINEMPTY,1)    ; wie JOINEMPTY behandeln

; alle Agenten (auch nicht-verfügbare) haben Schlange verlassen:
exten => q-LEAVEUNAVAIL,1,Goto(q-JOINEMPTY,1)  ; wie JOINEMPTY behandeln

; kein Agent antwortet:
exten => q-TIMEOUT,1,Goto(q-JOINEMPTY,1)       ; wie JOINEMPTY behandeln

; max. Anruferzahl für Schlange erreicht:
; (hier könnte man alternativ auf VoiceMail() leiten)
exten => q-FULL,1,Busy(5)
exten => q-FULL,n,Hangup()
```

Wenn wir diese Extension anrufen, werden wir, da keine Agenten angemeldet sind, die Ansage hören, dass z.Zt. niemand unseren Anruf entgegennehmen kann. (Wenn Sie das verschleiern wollen, setzen Sie in der `queues.conf` die Parameter `joinempty=yes` und `leaveempty=no`.)

Es fehlt also noch eine Extension unter der sich die Agenten einloggen, d.h. anwesend melden:

```
exten => 25,1,Answer()       ; abheben
exten => 25,n,AgentLogin()   ; Agent einloggen
exten => 25,n,Hangup()       ; hinterher auflegen
```

Mit `AgentLogin()` bleibt die Verbindung zum Agenten dauerhaft bestehen, und die eingehenden Anrufe werden auf diesen bereits aufgebauten Sprachkanal aufgeschaltet. Wenn der Agent den Hörer auflegt, wird er abgemeldet. Das bedeutet auch, dass ein Agent nicht gleichzeitig eingeloggt sein kann und ausgehende Gespräche machen kann. `AgentLogin()` ist daher in der Praxis wirklich nur dann geeignet, wenn es sich um ein reines Inbound-Call-Center handelt.

`AgentCallbackLogin()` hingegen hat den Vorteil, dass sich Agenten einmal anmelden können und bei eingehenden Gesprächen zurückgerufen werden. Sie bleiben also auch beim Auflegen angemeldet und können ausgehende Gespräche führen. Diese Applikation ist aber seit Version 1.4 „deprecated“, wird also zukünftig nicht mehr vorhanden sein.

Digium verwendet im Beispiel `doc/queues-with-callback-members.txt` (in AEL, lässt sich aber leicht in einen „herkömmlichen“ Dialplan umwandeln)

```
Read(agentennr,agent-user);
VMAuthenticate(${agentennr},s);
```

zum Authentifizieren der Agenten, `AddQueueMember()` zum dynamischen Hinzufügen zur Schlange und `Dial()` zum Anrufen der Agenten.

Sollte AgentCallbackLogin() tatsächlich ersatzlos wegfallen, geht damit tatsächlich Funktionalität und das leichte Setup verloren. (An dieser Stelle nochmal vielen Dank an Markus Bönke!)

Aufgabe. Es bleibt dem Leser als Aufgabe überlassen, vor die Einreihung in eine Warteschlange noch ein Menü zu schalten, z.B. „Um eine Bestellung aufzugeben, drücken Sie die 1. Für Reklamationen drücken Sie die 2.“, und dann 2 Warteschlangen einzurichten, denen die Agenten der entsprechenden Abteilung angehören.

6. Log-Datei

Ereignisse, die auf den Warteschlangen auftreten, werden im Queue-Log, normalerweise `/var/log/asterisk/queue_log` detailliert festgehalten (natürlich werden auch Einträge im CDR-Log gemacht). Im `queue_log` steht ein Eintrag pro Zeile im Format:

```
Timestamp|Anruf-ID|Schlange|Kanal|Event|Param1[|Param2[|Param3]]
```

`Timestamp` Ist die Unix-Zeit, zu der der Event aufgetreten ist.

`Anruf-ID` Ist die einmalige ID des Anrufs (alphanumerisch).

`Schlange` Ist der Name der Warteschlange, z.B. `support`. Kann auch `NULL` sein.

`Kanal` Ist der Name des überbrückten Kanals, z.B. `Agent/1001`. Kann auch `NULL` sein.

`Event` Ist der Name des aufgetretenen Ereignisses (s.u.). Abhängig vom Event sind `Param1`, `Param2` und `Param3` angegeben.

Die möglichen Ereignisse sind u.a. (siehe auch `doc/queuelog.txt`):

`ABANDON` Anrufer hat seine Position in der Schlange durch Auflegen aufgegeben. Parameter: Position, Einstiegspoition, Wartezeit.

`AGENTDUMP` Agent hat den Anrufer während der Ankündigung der Schlange abgewiesen.

`AGENTLOGIN`	Agent hat sich eingeloggt. Parameter: Kanal (z.B. `SIP/127.0.0.1-0181ac00`).
`AGENTCALLBACKLOGIN`	Callback-Agent hat sich eingeloggt. Parameter: Login-Extension[@Kontext].
`AGENTLOGOFF`	Agent hat sich ausgeloggt. Parameter: Kanal, Login-Dauer.
`AGENTCALLBACKLOGOFF`	Agent hat sich (/wurde) ausgeloggt. Parameter: Login-Extension[@Kontext], Login-Dauer, Grund (z.B. Autologoff).
`COMPLETEAGENT`	Gespräch zwischen Anrufer und Agent wurde vom Agenten beendet. Parameter: Wartezeit, Anruf-Dauer, Einstiegsposition.
`COMPLETECALLER`	Gespräch zwischen Anrufer und Agent wurde vom Anrufer beendet. Parameter: Wartezeit, Anruf-Dauer, Einstiegsposition.
`CONFIGRELOAD`	Konfiguration wurde neu eingelesen (z.B. durch **asterisk -rx "reload"**).
`CONNECT`	Anrufer wurde zu einem Agenten durchgestellt. Parameter: Wartezeit.
`ENTERQUEUE`	Anrufer wurde in die Schlange aufgenommen. Parameter URL (?), Caller-ID.
`EXITWITHKEY`	Anrufer hat die Schlange durch Drücken einer Taste verlassen. Parameter: Taste, Position.
`EXITWITHTIMEOUT`	Anrufer war zu lange in der Schlange, und der Timeout ist abgelaufen. Parameter: Position.
`QUEUESTART`	Das Warteschlangensystem wurde gestartet. Dabei haben die Felder `Anruf-ID`, `Schlange` und `Kanal` den Wert `NULL`.
`RINGNOANSWER`	Ein verfügbarer Agent wurde angeklingelt, hat aber nicht abgenommen (Timeout). Parameter: Klingeldauer (in Millisekunden).
`SYSCOMPAT`	Agent hat Anruf angenommen, aber die Kanäle waren nicht kompatibel, und der Anruf wurde beendet.
`TRANSFER`	Anrufer wurde auf eine andere Extension umgeleitet. Parameter: Extension, Kontext.

Kommerzielle Log-Analyse- und Echtzeitüberwachungssysteme sind QueueMetrics[6] oder Easy PABX[7].

Asterisk kann derzeit das Queue-Log noch nicht direkt in eine SQL-Tabelle schreiben. Wie man das trotzdem erreichen kann, wird auf http://www.voip-info.org/wiki/view/Asterisk+queue_log+on+MySQL beschrieben. Eine Möglichkeit ist, die Datei `queue_log` durch eine Named-Pipe zu ersetzen und dann z.B. mit einem Perl-Skript die Einträge in eine Datenbank zu schreiben. Hier ein Perl-Skript von William Lloyd[8]:

```
#!/usr/bin/perl -w
#
# wlloyd at slap.net

# The asterisk version indpendant way to get queue stats into Mysql,
Postgres
# or whatever is supported by Perl DBI

# It's all about named pipes

# to setup this software
# stop asterisk
# rm /var/log/asterisk/queue_log
```

[6] http://queuemetrics.loway.it/
[7] http://easypabx.com/
[8] wlloyd at slap.net, veröffentlicht auf der Digium-Mailingliste `asterisk-users`, siehe: http://lists.digium.com/pipermail/asterisk-users/2005-July/109892.html

```
# mkfifo /var/log/asterisk/queue_log

# make sure permissions are setup
# chmod 777 /var/log/asterisk/queue_log

# run this program as root or under another user as you see fit.
# should start BEFORE asterisk.  Add to /etc/rc.d/rc.local or whatever

# restart asterisk

# requires a DB table like the following..
# CREATE TABLE csr_queue (
#  qname varchar(30) default NULL,
#  agent varchar(30) default NULL,
#  action text,
#  info1 text,
#  info2 text,
#  info3 text,
#  timestamp int(11) NOT NULL default '0',
#  id tinytext NOT NULL
#) TYPE=MyISAM;

use DBI;
use IO::File;

my $opt_debug = 0;

# if you want postgres change this to "Pg"
my $db_type = "mysql";
my $db_host = "127.0.0.1";
my $db_user_name = 'username';
my $db_password = 'password';
my $db_database = 'asteriskstat';

my $dbh = DBI->connect("DBI:$db_type:dbname=$db_database;host=
$db_host;", $db_user_name, $db_password);

open(FIFO, "< /var/log/asterisk/queue_log")        or die "Can't open
queue_log : $!\n";

while (1) {

     $message = <FIFO>;
     next unless defined $message;   # interrupted or nothing logged
     chomp $message;

     # remove chars that will cause DB problems
     $message =~ s/\"\'//g;

     @data = split(/\|/,$message);

     # these messages are almost useless for my purposes
     next if ($data[4] eq "QUEUESTART" );
     next if ($data[4] eq "CONFIGRELOAD" );

     if (!defined($data[5])) {
       $data[5] = '';
     }
     if (!defined($data[6])) {
       $data[6] = '';
     }
     if (!defined($data[7])) {
       $data[7] = '';
     }

     my $sql = "INSERT INTO csr_queue (timestamp, id, qname, agent,
action, info1, info2, info3) VALUES ('$data[0]', '$data[1]', '$data
[2]', '$data[3]', '$data[4]', '$data[5]', '$data[6]', '$data[7]')";

     print "$sql \n\n" if ($opt_debug);
```

```
    $dbh->do($sql);

# if you want an actual logfile you might want to uncomment this
#        if ( open(LOG, ">> /var/log/asterisk/queue_log_real") ) {
#            print LOG "$message\n";
#            close(LOG);
#        } else {
#            warn "Couldn't log to /var/log/asterisk_queue_log: $!\n";
#        }
#
}

$dbh->disconnect();

exit 0;
```

Bei QueueMetrics (kostenlose Demo-Version) wird ebenfalls ein Perl-Skript `queueLoader.pl` mitgeliefert.

Kapitel 11. Asterisk als Faxserver

Es gibt eine Reihe von Ansätzen und Projekten, mit Asterisk einen Faxserver zu implementieren. Lange Zeit war app_rxfax von *http://www.soft-switch.org/* die von vielen angepriesene Lösung. Allerdings gilt diese Variante als fehleranfällig. Faxe werden regelmäßig unvollständig übertragen.

Deshalb beschreiben wir hier das Projekt iaxmodem (siehe *http://iaxmodem.sourceforge.net/*). Das iaxmodem ist ein virtuelles Modem, das von einer beliebigen Faxsoftware angesteuert werden kann. Für die Faxserversoftware nehmen wir das populäre *Hylafax*.

1. Installation iaxmodem

IAXmodem ist eine Software, die ein Modem simuliert, und dieses Asterisk mit dem IAX2-Protokoll zur Verfügung stellt. Alle Schritte in diesem Kapitel werden als User root ausgeführt.

Um IAXmodem installieren zu können, benötigen wir noch einige Debian-Packete die mit **apt-get -y install g++ libtiff-tools libtiff4 libtiff4-dev** installiert werden.

```
debian:~# apt-get -y install g++ libtiff-tools libtiff4 libtiff4-dev
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
  g++ libjpeg62 libjpeg62-dev libtiffxx0 zlib1g-dev

[...]

Richte zlib1g-dev ein (1.2.2-4.sarge.2) ...
Richte libtiff4-dev ein (3.7.2-7) ...

debian:~#
```

Die Sourcen zu IAXmodem können auf der Webseite *http://iaxmodem.sourceforge.net* runtergeladen werden (in diesem Beispiel ist es die Version 0.1.16). Nach dem Download muss der TarBall mit **tar -xzf iaxmodem-0.1.16.tar.gz** enpackt werden. Das Kompilieren geschieht danach mit mit den Befehlen **./configure** und **make**. Das dabei entstehende Binary iaxmodem kopieren wir zum Schluss mit **cp iaxmodem /usr/bin** nach `/usr/bin`

```
debian:~# tar xfz iaxmodem-0.1.16.tar.gz
debian:~# cd iaxmodem-0.1.16
debian:~# ./configure
debian:~# make
debian:~# cp iaxmodem /usr/bin
```

Jetzt kommen wir zur Konfiguration des Modems. IAXmodem sucht seine Konfiguration im Verzeichnis `/etc/iaxmodem`. Dieses müssen wir jetzt anlegen und in ihm eine Datei erzeugen, in der die Konfiguration steht. In dieser Datei müssen folgende Parameter angegeben werden:

`device` Das ist das Device, das im `/dev` Verzeichnis angelegt wird. Über dieses Device kann später HylaFax auf das Modem zugreifen. Der Name des Devices ist frei wählbar, wir halten uns aber an die allgemeinen Konventionen und nennen es äquivalent zum Device für die serielle Schnittstelle ttyIAX0.

`owner` Das ist der Eigentümer des Devices in der Form *`user`*`:`*`group`*. Es sollte der selbe User und die selbe Gruppe sein, unter der HylaFax laufen soll.

`port` Der Port, auf dem das iaxmodem lauscht. Da Asterisk auf Port 4569 auf IAX2-Verbindungen hört, sollte man hier einen anderen Port verwenden, z.B. 4570.

`refresh` Das Intervall, nach dem sich das Modem erneut bei Asterisk registriert. Wenn dieses auf 0 steht, registriert sich das Modem nicht bei Asterisk.

`server` Der Server, auf dem der Asterisk läuft. Wenn der Server der selbe ist, auf dem auch das iaxmodem läuft, steht hier die lokale Adresse 127.0.0.1

`peername` Der Name, unter dem sich das IAXmodem bei Asterisk registriert.

`secret` Das Passwort zur Registrierung am Asterisk.

`codec` Der Codec, der vom iaxmodem verwendet wird. Erlaubt sind hier `alaw`, `ulaw` und `slinear`. Andere machen hier auch wenig Sinn, da diese komprimieren und die Übertragung dadurch nicht verlustfrei ist. Bei reiner Sprachübertragung wirkt sich das für einen Menschen nicht negativ aus, bei Faxübertragungen aber hätte es störende Fehler zur Folge.

Die Konfigurationsdatei für ttyIAX0 heißt `/etc/iaxmodem/ttyIAX0` und hat folgenden Inhalt:

```
device          /dev/ttyIAX0
owner           uucp:uucp
mode            660
port            4570
refresh         300
server          127.0.0.1
peername        iaxmodem
secret          password
codec           alaw
```

Nachdem das IAXmodem nun fertig konfiguriert ist, muss es gestartet werden. Der Start geschieht am besten über den init-Prozess. Also fügen wir folgenden Eintrag der Datei `/etc/inittab` hinzu:

```
IA00:23:respawn:/usr/bin/iaxmodem ttyIAX0
```

Wobei *`ttyIAX0`* der Name der Konfigurationsdatei unter `/etc/iaxmodem` ist. Mit dem Kommando **init q** aktivieren wir das neue Device.

2. Installation HylaFax

Bevor die Faxserversoftware gebaut werden kann, müssen mit **apt-get -y install libtiff-tools libtiff4 libtiff4-dev gs** noch ein paar Debian-Pakete installiert werden:

```
debian:~# apt-get -y install libtiff-tools libtiff4 libtiff4-dev gs
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig

[...]

Updating category gsfontderivative..
Updating category truetype..
Updating category cid..
Updating category cmap..
Updating category psprint..

debian:~#
```

Über den auf der Hylafax Homepage *http://www.hylafax.org* vorhandenen Download-Link die aktuellen Sourcen runterladen. Natürlich geht dies auch mit dem Command-Line-Befehl **wget ftp://ftp.hylafax.org/source/hylafax-4.3.1.tar.gz**

```
debian:~# cd /usr/src/
debian:/usr/src# wget ftp://ftp.hylafax.org/source/hylafax-4.3.1.tar.gz
--12:35:25--  ftp://ftp.hylafax.org/source/hylafax-4.3.1.tar.gz
           => `hylafax-4.3.1.tar.gz'
Auflösen des Hostnamen »ftp.hylafax.org«.... 66.179.245.142
Verbindungsaufbau zu ftp.hylafax.org[66.179.245.142]:21... verbunden.
Anmelden als anonymous ... Angemeldet!
==> SYST ... fertig.    ==> PWD ... fertig.
==> TYPE I ... fertig.  ==> CWD /source ... fertig.
==> PASV ... fertig.    ==> RETR hylafax-4.3.1.tar.gz ... fertig.
Länge: 1,498,444 (unmaßgeblich)

100%[====================================>] 1,498,444    229.94K/s    ETA 00:00
```

```
12:35:33 (217.14 KB/s) - »hylafax-4.3.1.tar.gz« gespeichert [1498444]

debian:/usr/src#
```

Danach mit **tar -xzf hylafax-4.3.1.tar.gz** entpacken und **./configure** aufrufen.

```
debian:/usr/src# tar xfz hylafax-4.3.1.tar.gz
debian:/usr/src# cd hylafax-4.3.1
debian:/usr/src/hylafax-4.3.1# ./configure

Configuring HylaFAX (tm) (aka FlexFAX) 4.3.1.

If configure does the wrong thing, check the file config.log for
information that may help you understand what went wrong.

Reading site-wide parameters from ./config.site.
Gosh, aren't you lucky to have a i686-pc-linux-gnu system!
Using /usr/bin/gcc for a C compiler (set CC to override).
Looks like /usr/bin/gcc supports the -g option.
Using " -g" for C compiler options.

[...]
```

Mit **./configure** werden die Sourcen für das Übersetzen vorbereitet, das heißt es wird überprüft, ob alle Voraussetzungen erfüllt sind. Wenn dies der Fall ist, fragt das Programm diverse Parameter an.

```
HylaFAX configuration parameters (part 1 of 2) are:

[ 1] Directory for applications:        /usr/local/bin
[ 2] Directory for lib data files:      /usr/local/lib/fax
[ 3] Directory for lib executables:     /usr/local/sbin
[ 4] Directory for system apps:         /usr/local/sbin
[ 5] Directory for manual pages:        /usr/local/man
[ 6] Directory for HTML documentation:  /var/httpd/htdocs/hylafax
[ 7] Directory for spooling:            /var/spool/hylafax
[ 8] Directory for uucp lock files:     /var/lock
[ 9] Uucp lock file scheme:             ascii
[10] PostScript imager package:         gs
[11] PostScript imager program:         /usr/bin/gs
[12] Manual page installation scheme:   bsd-source-cat
[13] Default page size:                 North American Letter
[14] Default vertical res (lpi):        98

Are these ok [yes]?
```

Hier müssen für den Betrieb in Deutschland die Werte für das Papierformat auf *A4* gesetzt werden:

```
Are these ok [yes]?13
Default page size [North American Letter]? A4

HylaFAX configuration parameters (part 1 of 2) are:

[ 1] Directory for applications:        /usr/local/bin
[ 2] Directory for lib data files:      /usr/local/lib/fax
[ 3] Directory for lib executables:     /usr/local/sbin
[ 4] Directory for system apps:         /usr/local/sbin
[ 5] Directory for manual pages:        /usr/local/man
[ 6] Directory for HTML documentation:  /var/httpd/htdocs/hylafax
[ 7] Directory for spooling:            /var/spool/hylafax
[ 8] Directory for uucp lock files:     /var/lock
[ 9] Uucp lock file scheme:             ascii
[10] PostScript imager package:         gs
[11] PostScript imager program:         /usr/bin/gs
[12] Manual page installation scheme:   bsd-source-cat
[13] Default page size:                 ISO A4
[14] Default vertical res (lpi):        98

Are these ok [yes]?
```

Tipp

Wer will kann hier auch die Auflösung ändern. Da im Geschäftsumfeld über einen solchen solchen Faxserver wahrscheinlich auch Briefe und Verträge verschickt werden, können Sie die Auflösung bei entsprechendem Bedarf auf 196 lpi einstellen (feine Auflösung).[1]

```
Are these ok [yes]? 14
Default vertical res (lpi) [98]? 196

HylaFAX configuration parameters (part 1 of 2) are:

[ 1] Directory for applications:        /usr/local/bin
[ 2] Directory for lib data files:      /usr/local/lib/fax
[ 3] Directory for lib executables:     /usr/local/sbin
[ 4] Directory for system apps:         /usr/local/sbin
[ 5] Directory for manual pages:        /usr/local/man
[ 6] Directory for HTML documentation:  /var/httpd/htdocs/hylafax
[ 7] Directory for spooling:            /var/spool/hylafax
[ 8] Directory for uucp lock files:     /var/lock
[ 9] Uucp lock file scheme:             ascii
[10] PostScript imager package:         gs
[11] PostScript imager program:         /usr/bin/gs
[12] Manual page installation scheme:   bsd-source-cat
[13] Default page size:                 ISO A4
[14] Default vertical res (lpi):        196

Are these ok [yes]?
```

Alle anderen Werte auf dieser Seite können so bleiben, wir bestätigen also mit **yes**.

```
Are these ok [yes]? yes

HylaFAX configuration parameters (part 2 of 2) are:

[15] Location of getty program:         /sbin/agetty
[16] Location of voice getty program:   /bin/vgetty
[17] Location of sendmail program:      /usr/sbin/sendmail
[18] Location of TIFF tools:            /usr/bin
[19] Location of SysV init scripts:     /etc/init.d
[20] Location of SysV start scripts:    ../rc2.d ../rc3.d ../rc4.d ../rc5.d
[21] Location of SysV stop scripts:     ../rc0.d ../rc1.d ../rc6.d
[22] Name of SysV start script:         S97hylafax
[23] Name of SysV stop script:          K05hylafax
[24] Init script starts faxq:           yes
[25] Init script starts hfaxd           yes
[26] Start old protocol:                no
[27] Start paging protocol:             no
Are these ok [yes]?
```

Auch hier muss nichts verändern werden (also einfach mit **yes** übernehmen). Danach werden automatisch alle zum Kompilieren benötigten Makefiles angelegt. Das Kompilieren starten wir mit **make** und danach können wir mit einem **make install** die Software installieren.

```
debian:/usr/src/hylafax-4.3.1# make
make[1]: Entering directory `/usr/src/hylafax-4.3.1'
= regex
make[2]: Entering directory `/usr/src/hylafax-4.3.1/regex'
make[3]: Entering directory `/usr/src/hylafax-4.3.1/regex'
make[4]: Entering directory `/usr/src/hylafax-4.3.1/regex'
/bin/bash ../port/mkdepend   -e 's@ /usr/include/@ /@' -c /usr/bin/gcc -D__ANSI_CPP__ -I. -I.. -I.././reg
make[4]: Leaving directory `/usr/src/hylafax-4.3.1/regex'

[...]

for i in README depend make.links preremove proto.local ; do    \
```

[1] Damit ist sichergestellt, das auch das Kleingedruckte noch gut zu lesen ist. ;-)

```
        [ -f $i ]  ||  cp .././pkg/$i . ;                       \
done
make[2]: Leaving directory `/usr/src/hylafax-4.3.1/pkg'
make[1]: Leaving directory `/usr/src/hylafax-4.3.1'
debian:/usr/src/hylafax-4.3.1# make install

[...]

if test -d /etc/config; then \
    /bin/bash ../port/install.sh  -idb hylafax.sw.server -F /etc/config -m 644 -src config.fax -O fax; \
fi
make[1]: Leaving directory `/usr/src/hylafax-4.3.1/etc'
debian:/usr/src/hylafax-4.3.1#
```

Der nächste Schritt ist das Setup des Faxservers. Hierzu rufen wir das Programm **faxsetup** auf:

```
debian:/usr/src/hylafax-4.3.1# faxsetup

[...]

There does not appear to be an entry for the FaxMaster either in
the YP/NIS database or in the /etc/aliases file.  The
FaxMaster is the primary point of contact for HylaFAX problems.
The HylaFAX client-server protocol server identifies this alias as
the place to register complaints and HylaFAX directs automatic mail
messages to this user when problems are identified on a server
machine or when the routine server maintainence scripts are run
(e.g. faxcron).

Should an entry be added for the FaxMaster to /etc/aliases [yes]?
```

Bei den folgenden 2-3 Fragen einfach immer Enter drücken.

```
 HylaFAX configuration parameters are:

        [1] Init script starts faxq:            yes
        [2] Init script starts hfaxd            yes
        [3] Start old protocol:                 no
        [4] Start paging protocol:              no
Are these ok [yes]?
```

Bei den folgenden Einstellungen müssen Sie die für Ihre Installation geeigneten Werte eingeben:

```
No scheduler config file exists, creating one from scratch.
Country code [1]? 49
Area code []? 69
Long distance dialing prefix [1]? 0
International dialing prefix [011]? 00
Dial string rules file (relative to /var/spool/hylafax) ["etc/dialrules"]?
Tracing during normal server operation [1]?
Default tracing during send and receive sessions [0xffffffff]?
Continuation cover page (relative to /var/spool/hylafax) []?
Timeout when converting PostScript documents (secs) [180]?
Maximum number of concurrent jobs to a destination [1]?
Define a group of modems []?
Time of day restrictions for outbound jobs ["Any"]?
Pathname of destination controls file (relative to /var/spool/hylafax) []?
Timeout before purging a stale UUCP lock file (secs) [30]?
Max number of pages to permit in an outbound job [0xffffffff]?
Syslog facility name for ServerTracing messages [daemon]?

The non-default scheduler parameters are:

CountryCode:            49
AreaCode:               69
LongDistancePrefix:     0
InternationalPrefix:    00

Are these ok [yes]?
```

Dabei sind an sich nur die ersten vier Fragen wichtig. Das erste ist die Landesvorwahl als zweistellige Zahl, für Deutschland also die `49`. Dann folgt die Ortsnetznummer ohne führende Null, in diesem Beispiel die `69` für Frankfurt am Main. Dann eine Null für Ferngespräche und zwei Nullen für internationale Gespräche. Dann mit **yes** bestätigen.

```
Restarting HylaFAX server processes.
Should I restart the HylaFAX server processes [yes]?

/etc/init.d/hylafax start
HylaFAX: faxq hfaxd (without old protocol & without SNPP support).

You do not appear to have any modems configured for use.  Modems are
configured for use with HylaFAX with the faxaddmodem(8C) command.
Do you want to run faxaddmodem to configure a modem [yes]?
```

Jetzt bestätigen wir den Restart des Serverprozesses mit **yes** und werden gefragt, ob wir in Modem installieren wollen. Da unser iaxmodem bereits eingerichtet ist, können wir hier direkt weitermachen und bestätigen mit **yes**.

```
Serial port that modem is connected to []? ttyIAX0

Ok, time to setup a configuration file for the modem.  The manual
page config(5F) may be useful during this process.  Also be aware
that at any time you can safely interrupt this procedure.

Reading scheduler config file /var/spool/hylafax/etc/config.

No existing configuration, let's do this from scratch.

Country code [1]? 49
Area code [415]? 69
Phone number of fax modem [+1.999.555.1212]? +49 69 12345678
Local identification string (for TSI/CIG) ["NothingSetup"]? Apfelmus GmbH
Long distance dialing prefix [1]? 0
International dialing prefix [011]? 00
Dial string rules file (relative to /var/spool/hylafax) [etc/dialrules]?
Tracing during normal server operation [1]?
Tracing during send and receive sessions [11]?
Protection mode for received facsimile [0600]?
Protection mode for session logs [0600]?
Protection mode for ttyIAX0 [0600]?
Rings to wait before answering [1]?
Modem speaker volume [off]?
Command line arguments to getty program ["-h %l dx_%s"]?
Pathname of TSI access control list file (relative to /var/spool/hylafax) [""]?
Pathname of Caller-ID access control list file (relative to /var/spool/hylafax) [""]?
Tag line font file (relative to /var/spool/hylafax) [etc/lutRS18.pcf]?
Tag line format string ["From %%l|%c|Page %%P of %%T"]?
Time before purging a stale UUCP lock file (secs) [30]?
Hold UUCP lockfile during inbound data calls [Yes]?
Hold UUCP lockfile during inbound voice calls [Yes]?
Percent good lines to accept during copy quality checking [95]?
Max consecutive bad lines to accept during copy quality checking [5]?
Max number of pages to accept in a received facsimile [25]?
Syslog facility name for ServerTracing messages [daemon]?
Set UID to 0 to manipulate CLOCAL [""]?
Use available priority job scheduling mechanism [""]?

The non-default server configuration parameters are:

CountryCode:            49
AreaCode:               69
FAXNumber:              +49 69 12345678
LongDistancePrefix:     0
InternationalPrefix:    00
DialStringRules:        etc/dialrules
SessionTracing:         11
RingsBeforeAnswer:      1
SpeakerVolume:          off
GettyArgs:              "-h %l dx_%s"
```

```
LocalIdentifier:        Apfelmus GmbH
TagLineFont:            etc/lutRS18.pcf
TagLineFormat:          "From %%l|%c|Page %%P of %%T"
MaxRecvPages:           25

Are these ok [yes]?
```

Fragen über Fragen, aber nur die wenigsten sind wirklich wichtig.[2] Wichtig sind natürlich die Faxnummer und der sogenannte LocalIdentifier. Das ist der Text, der auf jedem gesendeten Fax in der obersten Zeile steht.

Yes bringt uns weiter zur Modemerkennung.

```
Now we are going to probe the tty port to figure out the type
of modem that is attached.  This takes a few seconds, so be patient.
Note that if you do not have the modem cabled to the port, or the
modem is turned off, this may hang (just go and cable up the modem
or turn it on, or whatever).

Probing for best speed to talk to modem: 38400 OK.

About fax classes:

The difference between fax classes has to do with how HylaFAX interacts
with the modem and the fax protocol features that are used when sending
or receiving faxes.  One class isn't inherently better than another;
however, one probably will suit a user's needs better than others.

Class 1 relies on HylaFAX to perform the bulk of the fax protocol.
Class 2 relies on the modem to perform the bulk of the fax protocol.
Class 2.0 is similar to Class 2 but may include more features.
Class 1.0 is similar to Class 1 but may add V.34-fax capability.
Class 2.1 is similar to Class 2.0 but adds V.34-fax capability.

HylaFAX generally will have more features when using Class 1/1.0 than
when using most modems' Class 2 or Class 2.0 implementations.  Generally
any problems encountered in Class 1/1.0 can be resolved by modifications
to HylaFAX, but usually any problems encountered in Class 2/2.0/2.1 will
require the modem manufacturer to resolve it.

If you're unsure and your modem supports it, use Class 1.

This modem looks to have support for Class 1 and 1.0.
How should it be configured [1]?1
Hmm, this looks like a Class 1 modem.
Product code (ATI0) is "spandsp".
Other information (ATI3) is "www.soft-switch.org".
DTE-DCE flow control scheme [default]?
Modem manufacturer is "spandsp".
Modem model is "IAXmodem".

Using prototype configuration file iaxmodem...

The modem configuration parameters are:

ModemResetCmds:         "ATH1\nAT+VCID=1"

Are these ok [yes]?

Creating new configuration file /var/spool/hylafax/etc/config.ttyIAX0...
Creating fifo /var/spool/hylafax/FIFO.ttyIAX0 for faxgetty... done.
Done setting up the modem configuration.

Checking /var/spool/hylafax/etc/config for consistency...
...some parameters are different.

The non-default scheduler parameters are:
```

[2] Mögen die Leute verzeihen, die schon einmal ein HylaFax getunt haben, aber für den Anfang reichen die defaults.

```
CountryCode:            49
AreaCode:               69
LongDistancePrefix:     0
InternationalPrefix:    00
DialStringRules:        etc/dialrules

Are these ok [yes]?
```

Das Modem wurde erkannt und wir werden gefragt, ob es ein Class 1 Modem ist. Da das genau das ist, was wir wollen, bestätigen wir. Auch das Reset-Kommando für das Modem können wir so übernehmen. Wenn alles in Ordnung ist, bestätigen wir mit **yes**.

```
Creating new configuration file /var/spool/hylafax/etc/config...
...saving current file as /var/spool/hylafax/etc/config.sav.

Don't forget to run faxmodem(8C) (if you have a send-only environment)
or configure init to run faxgetty on ttyIAX0.
Do you want to run faxaddmodem to configure another modem [yes]? no

You do not appear to be using faxgetty to notify the HylaFAX scheduler
about new modems and/or their status.  This means that you must use the
faxmodem program to inform the new faxq process about the modems you
want to have scheduled by HylaFAX.  Beware that if you have modems that
require non-default capabilities specified to faxmodem then you should
read faxmodem(8C) manual page and do this work yourself (since this
script is not intelligent enough to automatically figure out the modem
capabilities and supply the appropriate arguments).

Should I run faxmodem for each configured modem [yes]?
/usr/local/sbin/faxmodem ttyIAX0

Done verifying system setup.
```

Ein zweites Modem haben wir nicht, also beantworten wir die Frage mit **no**. Das Ausführen von faxmodem macht durchaus Sinn, denn dadurch sind die Modems an Hylafax angebunden.

Jetzt ist das Hylafax zum Senden von Faxen eingerichtet. Aber vielleicht möchten Sie ja auch welche empfangen. Dazu brauchen wir einen getty, der am iaxmodem auf Verbindungen lauscht. Das erreicht man ganz einfach durch einen weiteren Eintrag in der Datei `/etc/inittab`.

```
mo00:23:respawn:/usr/local/sbin/faxgetty ttyIAX0
```

2.1. Faxe empfangen

Jetzt muss diese Faxlösung noch in Asterisk integriert werden. Dazu müssen wir Asterisk das IAXmodem bekannt machen. Dies erreichen wir, indem wir es als IAX2-Peer definieren. Die dazu erforderliche Datei heißt `/etc/asterisk/iax.conf` (siehe auch Abschnitt 4, „IAX“):

```
[general]
bindport = 4569
bindaddr = 0.0.0.0
disallow=all
allow=ulaw
allow=alaw

[iaxmodem]
type=friend
secret=password
port=4570
host=dynamic
disallow=all
context=fax-out
allow=alaw
```

Im Abschnitt `general` sind die globalen IAX2-Daten abgelegt. In diesem Beispiel wird der Bindport auf den Standart für IAX2 4569 gesetzt. Die Bindadresse gibt das Interface an, auf dem IAX2 lauscht, in diesem Falle auf allen Interfaces.

In der Konfiguration für das IAXmodem wird der *`type`* auf `friend` gesetzt, d.h. es sind eingehende und ausgehende Verbindungen erlaubt. *`secret`* und *`port`* entsprechen der Konfiguration des IAXmodems, der *`context`* ist der, der bei einer ausgehenden Verbindung angesprochen wird.

Mit dem Befehl **iax2 show peers** können wir jetzt in der Asterisk-Console (CLI) unser IAXmodem sehen:

```
*CLI> iax2 show peers
Name/Username    Host                 Mask             Port          Status
iaxmodem         127.0.0.1       (D)  255.255.255.255  4570          Unmonitored
1 iax2 peers [0 online, 0 offline, 1 unmonitored]
*CLI>
```

Damit Asterisk weiß, was es mit einem ankommenden Fax anstellen soll, müssen wir eine entsprechende Extension schreiben. Das Ziel soll sein, daß ein ankommendes Fax direkt an das Hylafax weitergeleitet wird. In diesem Beispiel gehen wir davon aus, das alle Faxe über einen SIP-Provider-Anschluss reinkommen. Eine entsprechende Konfiguration kann wie folgt aussehen:

```
[...]

[123456]
type=friend
insecure=very;
nat=yes
username=123456
fromuser=12345
fromdomain=mein-voip-provider.de
secret=secret
host=mein-voip-provider.de
qualify=yes
context=fax-in

[...]
```

Der entsprechende Context in der `extensions.conf` sieht dann so aus:

```
[fax-in]
exten => _.,1,Answer()
exten => _.,2,Dial(IAX2/iaxmodem)
exten => h,1,Hangup()
```

Jetzt wird das Fax vom Asterisk an Hylafax über IAXmodem übergeben und per Mail an den User gesendet, auf den das Mailalias *`Faxmaster`* zeigt.

2.2. Faxe versenden

Der nächste Schritt ist Fax senden. Auch hierfür brauchen wir einen Context in der `extension.conf`. Diesen haben wir bereits weiter oben bei der Konfiguration des IAXmodems für Asterisk mit *`fax-out`* angegeben:

```
[fax-out]
exten => _X.,1,Answer()
exten => _X.,2,Dial(SIP/5556300/${EXTEN})
exten => h,1,Hangup()
```

Testen können wir das Senden mit **sendfax -n -d <Faxnummer> <datei.txt>**

```
debian:~# sendfax -n -d 06912345678 /etc/issue.net
```

Im CLI sollten wir jetzt folgendes sehen:

```
 -- Accepting AUTHENTICATED call from 127.0.0.1:
       > requested format = alaw,
       > requested prefs = (),
```

```
      > actual format = alaw,
      > host prefs = (alaw),
      > priority = mine
   -- Executing Answer("IAX2/iaxmodem-3", "") in new stack
   -- Executing Dial("IAX2/iaxmodem-3", "SIP/123456/06912345678") in new stack
   -- Called 123456/06912345678
   -- SIP/123456-0818f630 is making progress passing it to IAX2/iaxmodem-3
   -- SIP/123456-0818f630 answered IAX2/iaxmodem-3
   -- parse_srv: SRV mapped to host mein-voip-provider.de, port 5060
  == Spawn extension (fax-out, 06912345678, 2) exited non-zero on 'IAX2/iaxmodem-3'
   -- Executing Hangup("IAX2/iaxmodem-3", "") in new stack
  == Spawn extension (fax-out, h, 1) exited non-zero on 'IAX2/iaxmodem-3'
   -- Hungup 'IAX2/iaxmodem-3'
```

Das Kommando **faxstat -s** zeigt während des Sendens folgendes:

```
debian:~# faxstat -s
HylaFAX scheduler on w077.example.com: Running
Modem ttyIAX0 (123456): Sending job 7

JID  Pri S  Owner Number       Pages Dials     TTS Status
7    127 R   root 06912345678  0:1   0:12
debian:~#
```

Fertig! Jetzt können Sie über Asterisk mit Hylafax Faxe versenden und empfangen.

Auf der Hylafax-Webseite *http://www.hylafax.org* finden Sie viele Hinweise und Howtos, wie Sie Ihren neuen Faxserver möglichst einfach in Ihre bestehende Büroinfrastruktur einbinden können.

2.3. Fax FAQ

2.3.1.1. Kann man anstatt Hylafax auch andere Faxserversoftware einsetzen?

Ja, das ist kein Problem. Allerdings hat sich Hylafax sehr bewährt.

2.3.1.2. Gibt es zu Hylafax ein eigenes FAQ?

Ja, dieses kann unter *http://www.hylafax.org/content/FAQ* eingesehen werden.

Kapitel 12. Asterisk Gateway Interface (AGI)

Das Asterisk Gateway Interface (AGI) ist vergleichbar mit CGI auf Webservern. Es sind externe Programme, die innerhalb des Dialplans (der `extensions.conf`) aufgerufen werden. AGI-Skripte können mit Asterisk interagieren und Befehle ausführen. AGI-Skripte können prinzipiell in allen Programmier- und Skriptsprachen (z.B. Unix-Shell Script) geschrieben werden, die mit den Standardsockets unter Unix umgehen können. Die folgenden Beschreibungen gelten also nur als Beispiele für eine Implementierung.

1. Datenströme

Für AGI-Skripte ist ein Verständnis des Unix-Datenstrom-Models unabdingbar. Im Anhang dieses Buches finden Sie Informationen zu diesem Prinzip.

1.1. STDIN

Das AGI-Skript bekommt über den „standard input“ (STDIN) einige Informationen von Asterisk. Das Format sieht folgendermaßen aus:

```
agi_request: programmname.php
agi_channel: Zap/1-2
agi_language: de
agi_callerid: Hans Beispiel
agi_context: extern
agi_extension: 1234
agi_priority: 2
```

Nach dem Übergeben dieser Grundinformationen sendet Asterisk noch eine Leerzeile, um dem AGI-Skript mitzuteilen, dass die Übertragung beendet ist.

1.1.1. AGI Headers

Die folgende Aufstellung listet alle von Asterisk weitergegebenen Header auf:

- `agi_request`

  Der Dateiname des aufgerufenen AGI-Programmes.

- `agi_channel`

  Der Channel.

- `agi_language`

  Die eingestellte Sprache (z.B. en für Englisch oder de für Deutsch).

- `agi_type`

  Die Channel-Art (z.B. `sip` für SIP oder `zap` für ISDN).

- `agi_uniqueid`

  Eine im System unique (einmalige) ID für das Gespräch.

- `agi_callerid`

  Die Caller-ID (z.B. `Hans Meier <2000>`)

- `agi_context`

  Den Ursprungs-Context.

- `agi_extension`

  Die Ursprungs-Extension.

- `agi_priority`

  Die Priorität in der Extension beim Aufruf des AGI-Skriptes.

- `agi_accountcode`

  Abrechnungscode.

- `agi_calleridname`

  Name aus der Caller-ID (z.B. `Hans Meier`)

- `agi_callingpres`

  Die Caller-ID im ZAP-Channel.

1.2. STDOUT

Nachdem das AGI-Skript die Grundinformationen von Asterisk erhalten hat, beginnt es mit seiner Arbeit und gibt Befehle über „standard out“ (STDOUT) zurück an Asterisk. Um sich diese Kommunikation anzuschauen, kann man im CLI mit **set verbose 5** den Verbosity-Level entsprechend hochsetzen.

1.3. STDERR

„Standard error“ (STDERR) ist der formale Weg, um Fehler- oder Debugmeldungen vom AGI-Skript auf die Console (das CLI) von Asterisk zu bringen.

2. Verzeichnisse und Rechte

AGI Skripte werden von Asterisk standardmässig im Verzeichnis `/var/lib/asterisk/agi-bin/` gesucht.

Achten Sie darauf, das das entsprechende Skript von Asterisk auch ausgeführt werden darf. Dies kann durch den Aufruf des Befehles **chmod 755 *`skriptname`*** gewährleistet werden.

3. Aufruf eines AGI im Dialplan

Ein AGI-Skript wird im Dialplan über die Applikation `agi` aufgerufen. Die kann wie folgt aussehen:

```
exten => 1234,1,Answer()
exten => 1234,2,AGI(mein-agi-programm.php)
exten => 1234,3,Hangup()
```

4. Perl

In der Standard-Asterisk-Installation ist ein Test AGI-Skript mit dem Namen `agi-test.agi` [1]im Verzeichnis `/var/lib/asterisk/agi-bin/` abgespeichert. Anhand dieses Perl--Programms werden wir die grundsätzliche Arbeit mit AGI-Skripten beschreiben.

Das Skript wird dabei in der `extensions.conf` wie folgt aufgerufen:

[1]Die Dateiendung `agi` ist dabei nicht zwingend notwendig. Man könnte die Datei auch `agi-test.pl` nennen.

```
exten => 1234,1,Answer()
exten => 1234,2,AGI(agi-test.agi)
exten => 1234,3,Hangup()
```

4.1. Schritt für Schritt Analyse des agi-test.agi Skriptes

Wir besprechen das Beispiel-Skript Zeilen- oder Abschnittsweise.

```
#!/usr/bin/perl
use strict;
```

Die ersten zwei Zeilen sagen dem ausführendem Betriebsystem, dass es sich um ein Perl-Programm handelt, das mit dem Interpreter `/usr/bin/perl` ausgeführt werden soll. `use strict` bewirkt eine konsequentere Behandlung von Fehlern innerhalb des Perl-Programms.

```
$|=1;
```

Diese kleine Zeile bringt Perl dazu die Ausgabe von Text nicht zu puffern. So können wir sicher sein, dass alle Ausgaben auch unmittelbar an Asterisk übergeben und nicht erst in einem Buffer zwischengespeichert werden.

```
# Setup some variables
my %AGI; my $tests = 0; my $fail = 0; my $pass = 0;
```

Hier werden verschiedene Variablen definiert. Das Hash `%AGI` nimmt die initialen Eingaben von Asterisk auf. Die restlichen Variablen zählen die Gesamtanzahl der Tests, die Anzahl der fehlgeschlagenen Tests und die Anzahl der funktionierenden Tests.

```
while(<STDIN>) {
  chomp;
  last unless length($_);
  if (/^agi_(\w+)\:\s+(.*)$/) {
    $AGI{$1} = $2;
  }
}
```

Die eben eingelesenen Werte werden zum Debuggen auf STDERR, also im Endeffekt auf dem CLI ausgegeben.

```
sub checkresult {
  my ($res) = @_;
  my $retval;
  $tests++;
  chomp $res;
  if ($res =~ /^200/) {
    $res =~ /result=(-?\d+)/;
    if (!length($1)) {
      print STDERR "FAIL ($res)\n";
      $fail++;
    } else {
      print STDERR "PASS ($1)\n";
      $pass++;
    }
  } else {
    print STDERR "FAIL (unexpected result '$res')\n";
    $fail++;
  }
}
```

Die Subroutine `checkresult` liest das Ergebnis eines Befehls an Asterisk aus und bestimmt, ob der Test erfolgreich war oder nicht. Entsprechend werden die Variablen `$fail` und `$pass` hochgezählt. Nachdem die Grundlagen gelegt sind, können die einzelnen Tests beginnen: Die Datei `beep.gsm` wird abgespielt.

```
print STDERR "1.  Testing 'sendfile'...";
print "STREAM FILE beep \"\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Der Text "hello world" wird an den Anrufer geschickt. Das funktioniert natürlich nur, wenn das Protokoll und das Endgerät diese Funktion unterstützen.

```
print STDERR "2.  Testing 'sendtext'...";
print "SEND TEXT \"hello world\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Das Bild "asterisk-image" wird an den Anrufer geschickt. Auch diese Funktion ist vom Protokoll und dem Endgerät abhängig.

```
print STDERR "3.  Testing 'sendimage'...";
print "SEND IMAGE asterisk-image\n";
my $result = <STDIN>;
&checkresult($result);
```

Die Zahl 192837465 wird dem Anrufer vorgelesen.

```
print STDERR "4.  Testing 'saynumber'...";
print "SAY NUMBER 192837465 \"\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Dieser Befehl wartet 1000 Millisekunden auf die Eingabe von DTMF-Tönen durch den Anrufer.

```
print STDERR "5.  Testing 'waitdtmf'...";
print "WAIT FOR DIGIT 1000\n";
my $result = <STDIN>;
&checkresult($result);
```

Ein 3000-Millisekunden-langes GSM-Soundfile mit dem Namen `testagi.gsm` wird aufgenommen. Die Aufnahme kann durch die Eingabe der Zahlen 1, 2, 3 oder 4 unterbrochen werden.

```
print STDERR "6.  Testing 'record'...";
print "RECORD FILE testagi gsm 1234 3000\n";
my $result = <STDIN>;
&checkresult($result);
```

Das soeben aufgenommene Soundfile wird abgespielt.

```
print STDERR "6a.  Testing 'record' playback...";
print "STREAM FILE testagi \"\"\n";
my $result = <STDIN>;
&checkresult($result);
```

Nun erfolgt die Ausgabe auf dem CLI, wie viele Tests funktioniert oder nicht funktioniert haben.

```
print STDERR "================== Complete ====================\n";
print STDERR "$tests tests completed, $pass passed, $fail failed\n";
print STDERR "================================================\n";
```

Warnung

Bei vielen AGI-Befehlen sehen Sie den folgenden Aufbau:

```
fwrite(STDOUT,"BEFEHL $value \"\"\n");
#                             ^^^^^^^
```

Der in dieser Zeile unterschlängelte Teil (zwischen `$value` und `);`) ist zwingend erforderlich, damit der Befehl korrekt ausgeführt wird. Es handelt sich hierbei um ein Argument ohne Inhalt, das durch zwei gequotete Anführungszeichen dargestellt wird. Abgeschlossen wird der gesamte Befehl durch ein "line feed" (ein Zeilenendzeichen). In diesem Fall ein `\n`.

5. PHP

PHP ist zu einer der populärsten Programmiersprachen für Webapplikationen geworden.[2] Da man aber aktuelle PHP Versionen auch für den Aufruf von Programmen auf der Kommandozeile benutzen kann, ist sie eine für AGI-Skripte gut geeignete Sprache. Als Beispiel benutzen wir ein kleines PHP-Programm (`lotto.php`), das 6 zufällige

[2] Böse Zungen sprechen auch vom BASIC des 21. Jahrhunderts. ;-)

Zahlen von 1 bis 49 auswählt und dem Anrufer vorspricht. Die Beschreibung der einzelnen Schritte erfolgt im Quellcode.

```
#!/usr/bin/php -q
<?php

# Sicherheitseinstellung. Das Skript laeuft nicht
# laenger als 8 Sekunden.
##################################################
set_time_limit(8);


# Output Buffer wird deaktiviert
# Alternativ könnten wir nach jeder Ausgabe
# fflush(STDOUT); aufrufen.
##################################################
ob_implicit_flush();


# PHP Error Reporting wird deaktiviert
##################################################
error_reporting(0);


# Fuer die Komunikation mit Asterisk benoetigen
# wir STDIN und STDOUT Filehandles
##################################################
if (!defined('STDIN'))
  define('STDIN' , fopen('php://stdin' , 'r'));
if (!defined('STDOUT'))
  define('STDOUT', fopen('php://stdout', 'w'));
if (!defined('STDERR'))
  define('STDERR', fopen('php://stderr', 'w'));


# Die von Asterisk uebergebenen Variablen und
# Werte werden ausgelesen und im Array $agi
# gespeichert.
##################################################

$agi = array();

while (!feof(STDIN))
{
  $tmp = trim(fgets(STDIN,4096));
  if (($tmp == '') || ($tmp == "\n"))
    break;
  $var1 = split(':',$tmp);
  $name = str_replace('agi_','',$var1[0]);
  $agi[$name] = trim($var1[1]);
}


# Ein Array mit 6 zufaelligen und nicht
# doppelten Zahlen von 1 bis 49 wird generiert.
##################################################

$Lottozahlen = array();
do {
  $Zahl = rand(1,49);
  if (array_search($Zahl, $Lottozahlen) == FALSE) {
    $Lottozahlen[] = $Zahl;
  }
} while (count($Lottozahlen) < 6);


# Vor der ersten Ansage wird eine Sekunde
# gewartet.
##################################################
```

```
fwrite(STDOUT,"EXEC Wait 1 \"\"\n");
fflush(STDOUT);


# Die Zahlen werden nacheinander vorgelesen.
# Zwischen den einzelnen Zahlen gibt es immer
# eine Pause von einer Sekunde.
################################################
foreach ($Lottozahlen as $value) {
  fwrite(STDOUT,"SAY NUMBER $value \"\"\n");
  fflush(STDOUT);
  fwrite(STDOUT,"EXEC Wait 1 \"\"\n");
  fflush(STDOUT);
}

?>
```

Das Programm `lotto.php` muss im Verzeichnis `/var/lib/asterisk/agi-bin/` abgespeichert werden und wird in der `extensions.conf` wie folgt aufgerufen:

```
exten => 1234,1,Answer()
exten => 1234,2,AGI(lotto.php)
exten => 1234,3,Hangup()
```

5.1. phpAGI

Wer PHP und AGI benutzen, aber nicht das Rad neu erfinden will, kann die fertige PHP-Klasse phpAGI benutzen. Informationen und die dazugehörigen Dateien finden Sie auf der Homepage des Projektes *http://phpagi.sourceforge.net/*.

Das folgende bei phpAGI mitgelieferte Programm gibt schnell einen Überblick über die Funktionsaufrufe[3]

```
<?php
/**
* @package phpAGI_examples
* @version 2.0
*/

function my_ip(&$agi, $peer)
   {
    $ip = 'unknown';
    $asm = $agi->new_AsteriskManager();
    if($asm->connect())
    {
      $peer = $asm->command("sip show peer $peer");
      $asm->disconnect();

      if(!strpos($peer['data'], ':'))
        echo $peer['data'];
      else
      {
        $data = array();
        foreach(explode("\n", $peer['data']) as $line)
        {
          $a = strpos('z'.$line, ':') - 1;
          if($a >= 0) $data[trim(substr($line, 0, $a))] = trim(substr($line, $a + 1));
        }
      }

      if(isset($data['Addr->IP']))
      {
        $ip = explode(' ', trim($data['Addr->IP']));
        $ip = $ip[0];
      }
    }
    $agi->text2wav("Your IP address is $ip");
```

[3] Für die Benutzung von `text2wav` muss ein Text-to-Speech System (z.B. Festival) installiert und konfiguriert sein.

```
  }
?>
```

6. Andere Programmiersprachen

Wie am Anfang dieses Kapitels schon beschrieben: Man kann AGI-Programme in jeder beliebigen Programmiersprache schreiben. Fertige Bibliotheken finden sich z.B. für

- Java
- Perl
- PHP
- Python
- Ruby
- C
- C#

Am einfachsten ist eine Suche in der Suchmaschine der Wahl nach den Schlüsselwörtern „AGI“ und dem Namen der Programmiersprache. Eine weiterer Startpunkt ist die Webseite http://www.voip-info.org/wiki-Asterisk+AGI.

7. AGI-Befehle

Die folgenden Befehle ermöglichen es einem AGI-Skript, mit Asterisk zu interagieren. Die meisten dieser Befehle sind von Dialplan-Applikationen (siehe Abschnitt 6, „Applikationen im Dialplan“) mit ähnlichen Namen abgeleitet.

> **Wichtig**
>
> Denken Sie daran, falls Sie Parameter, die nicht am Ende stehen, auslassen wollen, stattdessen "" anzugeben!

7.1. `ANSWER`

```
ANSWER
```

Funktioniert wie `Answer()` in der `extensions.conf`. Der Anruf wird entgegengenommen.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich

7.2. `CHANNEL STATUS`

```
CHANNEL STATUS [Kanalname]
```

Abfrage des Status eines Kanals. Wenn der *`Kanalname`* nicht angegeben ist, wird der aktuelle Kanal benutzt.

Ergebniswerte nach Ausführung:

0	Channel ist nicht aktiv und verfügbar.
1	Channel ist nicht aktiv, aber reserviert.
2	Channel ist off-hook.
3	DTMF-Eingaben wurden getätigt.
4	Der Channel wird gerade angerufen (es klingelt).
5	Die Leitung ist aktiv.
6	Die Leitung ist besetzt.

7.3. DATABASE DEL

```
DATABASE DEL Familie Schlüssel
```

Löscht den durch *Familie* und *Schlüssel* definierten Eintrag in der Asterisk-Datenbank.

Ergebniswerte nach Ausführung:

0	Fehler
1	Erfolgreich

7.4. DATABASE DELTREE

```
DATABASE DELTREE Familie
```

Löscht eine gesamte *Familie* aus der Asterisk-Datenbank.

Ergebniswerte nach Ausführung:

0	Fehler
1	Erfolgreich

7.5. DATABASE GET

```
DATABASE GET Familie Schlüssel
```

Gibt den entsprechenden Eintrag für *Familie*/*Schlüssel* aus der Asterisk-Datenbank aus.

Ergebniswerte nach Ausführung:

0	Fehler
1 (*Wert*)	Erfolgreich mit dem gefundenen Wert

7.6. DATABASE PUT

```
DATABASE PUT Familie Schlüssel Wert
```

Speichert den Wert *Wert* in der Asterisk-Datenbank unter *Familie*/*Schlüssel* ab. Das kann ein Hinzufügen oder ein Update sein.

Ergebniswerte nach Ausführung:

0	Fehler
1	Erfolgreich

7.7. EXEC

```
EXEC Applikation Argumente
```

Führt eine Dialplan-Applikation aus (siehe Abschnitt 6, „Applikationen im Dialplan“). Die *Argumente* werden dabei der Applikation übergegeben. Mehrere Argumente werden durch ein Pipe-Zeichen (|) getrennt.

Ergebniswerte nach Ausführung:

-2	Fehler. Konnte die Applikation nicht finden / aufrufen.
Wert	Erfolgreich. Ergebniswert der Applikation.

Beispiel:

```
EXEC Dial Zap/g1/123456
```

7.8. GET DATA

```
GET DATA Dateiname [Timeout] [maxZiffern]
```

Vergleichbar mit der Background()-Applikation in der `extensions.conf`. Spielt die Audiodatei mit dem Namen *`Dateiname`* ab und wartet auf DTMF-Eingaben. Diese werden durch *`maxZiffern`* limitiert. Die maximale Zeit wird durch *`Timeout`* begrenzt.

Ergebniswerte nach Ausführung:

`Wert`	DTMF-Eingaben

7.9. GET FULL VARIABLE

```
GET FULL VARIABLE Variablenname [Kanalname]
```

Gibt den Wert der Variable *`Variablenname`* aus dem Kanal *`Kanalname`* (oder dem aktuellen) aus.

Ergebniswerte nach Ausführung:

0	Fehler. Variable existiert nicht.
1 (*`Wert`*)	Erfolgreich. Wert der Variablen.

7.10. GET OPTION

```
GET OPTION Dateiname Escape_Ziffern [Timeout]
```

Verhält sich wie `STREAM FILE`. Hat aber zusätzlich die Möglichkeit, einen *`Timeout`* in Sekunden anzugeben.

Ergebniswerte nach Ausführung:

`Wert`	DTMF-Eingaben

7.11. GET VARIABLE

```
GET VARIABLE Variablenname
```

Ähnlich wie `GET FULL VARIABLE`. Allerdings versteht `GET VARIABLE` keine komplexen oder systemeigenen Variablen.

Warnung

Im Zweifelsfall sollte man `GET FULL VARIABLE` benutzen.

Ergebniswerte nach Ausführung:

0	Fehler. Kein Channel oder die Variable existiert nicht.
1 (*`Wert`*)	Erfolgreich. Wert der Variablen (in Klammern)

7.12. HANGUP

```
HANGUP [Kanalname]
```

Legt den virtuellen Hörer auf dem angegebenen Kanal auf. Wurde kein *`Kanalname`* angegeben, wird der aktuelle Kanal aufgelegt.

Ergebniswerte nach Ausführung:

-1	Fehler. Der angegebene Kanal existiert nicht.
1	Erfolgreich. Hangup wurde ausgeführt.

7.13. NOOP

```
NOOP [Text]
```

Es wird nichts gemacht (NOOP steht für No-Operation). *Text* wird auf dem CLI ausgegeben.

Ergebniswerte nach Ausführung:

0	Gibt immer 0 zurück.

7.14. RECEIVE CHAR

```
RECEIVE CHAR Timeout
```

Wenn ein Channel die Übermittlung von Text unterstützt, dann kann mit RECEIVE CHAR ein ein eizelnes Zeichen empfangen werden. *Timeout* gibt die Wartezeit für den Empfang in Millisekunden an (0 für unendlich).

Ergebniswerte nach Ausführung:

-1	Fehler oder Hangup
Wert (timeout)	Das empfangene Zeichen bei Timeout
Wert	Das empfangene Zeichen

7.15. RECEIVE TEXT

```
RECEIVE TEXT Timeout
```

Wenn ein Channel die Übermittlung von Text unterstützt, dann kann mit RECEIVE CHAR ein solcher Text empfangen werden. *Timeout* gibt die Wartezeit für den Empfang in Millisekunden an (0 für unendlich).

Ergebniswerte nach Ausführung:

-1	Fehler oder Hangup
Wert (timeout)	Der empfangene Text bei Timeout
Wert	Der empfangene Text

7.16. RECORD FILE

```
RECORD FILE Dateiname Format Escape_Ziffern Timeout [Offset] [BEEP] [s=Stille]
```

Funktioniert wie ein Rekorder, der alle Audiosignale in der Datei *Dateiname* im angegebenen *Format* aufzeichnet. Beendet werden kann dies durch einen definierten DTMF-Ton *Escape_Ziffern* oder durch einen *Timeout* in Millisekunden (dabei steht -1 für keinen Timeout).

BEEP spielt vor der Aufnahme einen Signalton. *Stille* definiert die Anzahl der Stille-Sekunden bis die Applikation auch ohne Eingabe von DTMF-Tönen einen Timeout ausgibt.

Mit *Offset* wird vor Beginn der Aufnahme soviele Sekunde wie angegeben gewartet.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich

7.17. SAY ALPHA

```
SAY ALPHA String [Escape_Ziffern]
```

Buchstabiert den übergebenen *String*. Die Ausgabe kann durch Eingabe einer Ziffer, die in *Escape_Ziffern* bestimmt wird, abgebrochen werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

7.18. SAY DATE

```
SAY DATE Timestamp [Escape_Ziffern]
```

Sagt das übergebene Datum an. *Timestamp* ist dabei die Anzahl der Sekunden seit 00:00:00 Uhr am 1. Januar 1970. Die Ausgabe kann durch Eingabe einer Ziffer, die in Escape_Ziffern bestimmt wird, abgebrochen werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

7.19. SAY DATETIME

```
SAY DATETIME Timestamp [Escape_Ziffern] [Format] [Zeitzone]
```

Sagt das übergebene Datum. *Timestamp* ist dabei die Anzahl der Sekunden seit 00:00:00 Uhr am 1. Januar 1970. Das *Format* bestimmt die Art der Ausgabe. Folgende Werte sind dabei möglich:

'*dateiname*'	Eine Sounddatei
${*VAR*}	Eine Variable
A oder a	Tag der Woche (z.B. Montag oder Dienstag)
B, b oder h	Name des Monats (z.B. Januar)
d oder e	Ordnungseinheit des Tages im Monat (z.B. erster, zweiter, dritter)
Y	Jahr
I oder i	Stunde (im 12-Std.-System)
H	Stunde (im 24-Std.-System). Dabei wir 07 „Null - Sieben“ ausgesprochen!
k	Stunde (im 24-Std.-System). Dabei wir 07 nur „Sieben“ ausgesprochen.
M	Minuten
P oder p	AM oder PM
Q	„Heute“, „Gestern“ oder der Inhalt der Werte ABdY
q	„ “ (für heute), „Gestern“, der Wochentag oder ABdY
R	Zeit inkl. Minuten im 24-Std.-System

Die *Zeitzone* wird wie in der Konfigurationsdatei /etc/asterisk/voicemail.conf angegeben.

Die Ausgabe kann durch Eingabe einer Ziffer, die in *Escape_Ziffern* bestimmt wird, abgebrochen werden. Wenn *Escape_Ziffern* oder *Format* ausgelassen werden soll, muss stattdessen "" angegeben werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

7.20. SAY DIGITS

```
SAY DIGITS Zahl [Escape_Ziffern]
```

Sagt die angegebene *Zahl* Ziffer für Ziffer an (z.B. 123 wird „eins, zwei, drei“ gesprochen). Die Ausgabe kann durch Eingabe einer Ziffer, die in Escape_Ziffern bestimmt wird, abgebrochen werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat.

7.21. SAY NUMBER

```
SAY NUMBER Zahl [Escape_Ziffern]
```

Sagt die angegebene *Zahl* an (z.B. 123 wird „ein-hundert-drei-und-zwanzig“ gesprochen). Die Ausgabe kann durch Eingabe einer Ziffer, die in Escape_Ziffern bestimmt wird, abgebrochen werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

7.22. SAY PHONETIC

```
SAY PHONETIC String [Escape_Ziffern]
```

Sagt den angegebenen *String* im phonetischen NATO-Alphabet an (z.B. a wird „alpha“ gesprochen). Die Ausgabe kann durch Eingabe einer Ziffer, die in Escape_Ziffern bestimmt wird, abgebrochen werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

7.23. SAY TIME

```
SAY TIME Timestamp [Escape_Ziffern]
```

Sagt die übergebene Zeit. *Timestamp* ist dabei die Anzahl der Sekunden seit 00:00:00 Uhr am 1. Januar 1970. Die Ausgabe kann durch Eingabe einer Ziffer, die in Escape_Ziffern bestimmt wird, abgebrochen werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich

Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

7.24. SEND IMAGE

```
SEND IMAGE Dateiname
```

Sendet das in *Dateiname* angegebene Bild auf den aktuellen Kanal. Der Dateiname darf keine Endung (z.B. `.gif` oder `.jpg`) enthalten. Diese Funktion wird von den meisten Channels nicht unterstützt.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich

7.25. SEND TEXT

```
SEND TEXT "zu sendender Text"
```

Sendet den angegebenen Text auf den aktuellen Kanal. Diese Funktion wird von den meisten Channels nicht unterstützt.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich

7.26. SET AUTOHANGUP

```
SET AUTOHANGUP Timeout
```

Stellt ein automatisches Auflegen nach der angegebenen Anzahl von Sekunden ein. Mit dem Wert `0` kann diese Funktion wieder deaktiviert werden.

Ergebniswerte nach Ausführung:

0	Erfolgreich (Wert wurde gesetzt)

7.27. SET CALLERID

```
SET CALLERID Nummer
```

Setzt die Caller-ID des aktuellen Kanals.

Ergebniswerte nach Ausführung:

1	Erfolgreich (Caller-ID wurde gesetzt)

7.28. SET CONTEXT

```
SET CONTEXT Kontextname
```

Setzt den Context, der nach dem Ausführen der AGI-Applikation angesprungen wird.

Ergebniswerte nach Ausführung:

0	Gibt immer 0 zurück.

Wichtig

Es wird nicht überprüft, ob der Kontext existiert. Wenn Sie einen ungültigen Kontext angeben, wird der Anruf nach Beendigung des Skriptes aufgelegt.

7.29. SET EXTENSION

```
SET EXTENSION Extension
```

Setzt die Extension, die nach dem Ausführen der AGI-Applikation angesprungen wird.

Ergebniswerte nach Ausführung:

0 Gibt immer 0 zurück.

Wichtig

Es wird nicht überprüft, ob die Extension existiert. Wenn Sie eine ungültige Extension angeben, wird Asterisk nach Beendigung des Skriptes bei der `i`-Extension weitermachen, falls vorhanden. Ansonsten wird der Anruf aufgelegt.

7.30. SET MUSIC

```
SET MUSIC [on [Klasse]|off]
```

Aktiviert (`on`) oder deaktiviert (`off`) die Warteschleifenmusik. Wenn *`Klasse`* nicht angegeben wurde, wird die Musikklasse `default` benutzt (siehe Konfigurationsdatei `/etc/asterisk/musiconhold.conf`).

Ergebniswerte nach Ausführung:

0 Gibt immer 0 zurück.

7.31. SET PRIORITY

```
SET PRIORITY Priorität
```

Setzt die Priorität, die nach dem Ausführen der AGI-Applikation angesprungen wird (als Priorität kann auch ein Label angegeben werden).

Ergebniswerte nach Ausführung:

0 Gibt immer 0 zurück.

Wichtig

Es wird nicht überprüft, ob die Priorität existiert. Wenn Sie eine ungültige Priorität angeben, wird der Anruf nach Beendigung des Skriptes aufgelegt.

7.32. SET VARIABLE

```
SET VARIABLE Variablenname Wert
```

Setzt die Kanal-Variable *`Variablenname`* auf den Wert *`Wert`*.

Ergebniswerte nach Ausführung:

1 Erfolgreich (Variable wurde gesetzt)

7.33. STREAM FILE

```
STREAM FILE Dateiname Escape_Ziffern [Offset]
```

Spielt die angegebene Datei *Dateiname* ab (dabei darf keine Dateiendung wie `.wav` oder `.gsm` angegeben werden). Das Abspielen kann dabei durch die Eingabe einer Ziffer, die in `Escape_Ziffern` definiert wurde, abgebrochen werden. Ist dies nicht erwünscht (es soll also kein Abbruch möglich sein), so muss an dieser Stelle `""` angegeben werden. Mit *Offset* kann ein Offset zum Start der Datei angegeben werden.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Erfolgreich (die Datei wurde komplett abgespielt)
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

Wichtig

`STREAM FILE` funktioniert manchmal nicht korrekt mit anderen Sprachen als Englisch. Stattdessen kann man `EXEC Playback` *Dateiname* verwenden.

7.34. TDD MODE

```
TDD MODE [on|off]
```

Aktiviert (`on`) oder deaktiviert (`off`) Übertragung und Empfang des TDD-Protokolls (Baudot) für TDD-Geräte (Telecommunications Device für the Deaf, Telekommunikationsgerät für Taubstumme, auch Schreibtelefon genannt) auf dem aktiven Kanal. Bisher wird TDD nur von Zap-Kanälen unterstützt.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Channel nicht TDD-fähig
1	Erfolgreich

7.35. VERBOSE

```
VERBOSE Meldung Level
```

Schreibt die *Meldung* auf das Command Line Interface (CLI). *Level* gibt dabei den Mindestwert des Verbosity-Levels an, damit die Ausgabe angezeigt wird.

Ergebniswerte nach Ausführung:

0	Gibt immer 0 zurück.

7.36. WAIT FOR DIGIT

```
WAIT FOR DIGIT Timeout
```

Wartet auf die Eingabe eines DTMF-Tones. *Timeout* gibt den Timeout der Applikation in Millisekunden an.

Ergebniswerte nach Ausführung:

-1	Fehler
0	Timeout
Wert	ASCII-Wert der Eingabe, die zum Abbruch geführt hat

Kapitel 13. Telefone

1. Einleitung

In der idealen Welt müsste man in diesem Kapitel alle zum Druckbeginn des Buches verfügbaren VoIP-Telefone beschreiben und am besten noch bewerten. Dies ist allerdings nicht möglich. Das liegt zum Teil auch daran, das nicht alle Hersteller Testgeräte zur Verfügung stellen. Ich hoffe, das im Laufe der Jahre auf der Webseite zum Buch und in zukünftigen Versionen des Buches mehr und mehr Telefone mit Beispielkonfigurationen aufgelistet werden können. Dazu benötige ich Ihr Feedback und Ihre Hilfe.

2. snom VoIP Telefone

VoIP Telefone der Firma *snom* sind in Deutschland sehr beliebt und werden deshalb an dieser Stelle im Detail beschrieben. Es gibt eine Reihe von verschiedenen snom Telefonen. Das snom 360 sticht allerdings mit einigen Features für den Einsatz bei grossen Firmen hervor. Alle snom Telefone bieten durch die umfangreiche SIP-Unterstützung in Kombination mit Asterisk Eigenschaften, die man sonst nur von herstellerspezifischen Anlagen mit Systemtelefonen kennt.

2.1. Konfiguration von Asterisk

Standard Snom-Telefone können mit dem SIP-Protokoll an eine Asterisk-Anlage angeschlossen werden. Um die Komformerkmale zu beschreiben, nehmen wir als Standardkonfiguration die aus Kapitel 2, *Installation und "Hello World"* und erweitern sie wie folgt:

Der Eintrag in der `sip.conf` Datei von Asterisk:

```
[general]
port = 5060
bindaddr = 0.0.0.0
context = sonstige

[2000]
type=friend
context=meine-telefone
secret=1234
host=dynamic
mailbox=2000

[2001]
type=friend
context=meine-telefone
secret=1234
host=dynamic
mailbox=2001
```

Der Eintrag `mailbox=...` sorgt dafür, dass die "Message"-Anzeige des snom Endgerätes angesprochen wird. Sobald eine Nachricht auf der entsprechenden Mailbox des Teilnehmers eingegangen ist, wird dies standardmäßig durch ein blinkendes "Message"-Lämpchen angezeigt. Dieser Mechanismus wird MWI genannt, "Message Waiting Indicator". Je nach Einstellung des Telefons kann ein Nachrichteneingang zusätzlich auch akustisch signalisiert werde. Es ist sicherzustellen, dass eine passende Mailbox innerhalb der `voicemail.conf` Datei angelegt wurde:

```
[general]
format = wav

[default]
2000 => 4711,Hans Mustermann,hansi@company.de
2001 => 0815,Ute Beispiel,ute.beispiel@company.de
```

Jedes snom Telefon verfügt über eine "Retrieve"-Taste, um neue Sprachnachrichten im Voicemailsystem abzurufen. Wir diese Taste gedrückt, wählt das snom Telefon in der Werkseinstellung die Extension *asterisk* an. Der Dialplan muss entsprechend um eine solche Extension erweitert werden:

```
[sonstige]

[meine-telefone]
exten => _2XXX,1,Dial(SIP/${EXTEN},20,j)
exten => _2XXX,2,VoiceMail(u${EXTEN})
exten => _2XXX,102,VoiceMail(b${EXTEN})

exten => 2999,1,VoiceMailMain(s${CALLERID(num)})

exten => asterisk,1,VoicemailMain(s${CALLERID(num)})
```

Um die für die Voicemail anzuwählende Extension zu ändern, muss in der Datei `sip.conf` der Parameter `vmexten=...` hinzugefügt werden.

```
[2001]
type=friend
context=meine-telefone
secret=1234
host=dynamic
mailbox=2001
vmexten=obelisk
```

Snom Telefone zeigen eingegangene Nachrichten auch im Display im Klartext an, daher sollten für diesen Fall durchaus "sprechende" Namen genutzt werden.

Warnung

Die "Retrieve"-Taste funktioniert nur, wenn die "Message"-Anzeige blinkt, d.h. nur nach dem Eingang einer neuen Nachricht. Um alte Nachrichten abzuhören oder die Mailbox zu konfigurieren, muss im Wählplan eine eigene Extension zum Abhören der Mailbox eingerichtet werden (im Beispiel ist dies die *2999*). Diese kann man dann entweder manuell anwählen, eine der Funktionstasten des Telefons belegen oder bei den snom 360 Modellen über die XML-Steuerung einen Menüpunkt zur Verfügung stellen.

Nun sind alle grundlegenden Schritte für die Integration der snom Telefone in Asterisk durchgeführt.

2.2. Konfiguration der Telefone

Konfiguriert man ein einzelnes Gerät, so kann dies durchaus manuell erfolgen. In diesem Fall sollte man dem Gerät dennoch per DHCP eine IP-Adresse zuweisen lassen und den Rest der Konfiguration dann bequem über einen Web-Browser durchführen, denn die snom Telefone verfügen alle über einen eingebauten WEB-Server, so dass alle Settings komfortabel von außen editiert werden können, ohne sich über die zehn Ziffern- und Menütasten zu quälen.

Wichtig

Achten Sie darauf, dass sich das Telefon im Admin-Modus befindet, nur so können Sie wirklich alle Einstellungen des Telefons verändern. Normalerweise befindet sich das Telefon in diesem Modus, falls nicht, kann dieser über die "Settings"-Taste aktiviert werden. Das Default-Admin-Passwort ist "0000".

2.2.1. Manuelle Konfiguration

Alle Details zur Einstellung am Gerät selbst finden sich auf der snom Homepage http://www.snom.com. Dort sind für alle Endgeräte Anleitungen im PDF-Format frei erhältlich. Die Konfiguration zur Anbindung eines Telefons an eine Anlage mit obiger Beispiel-Konfiguration ist sehr einfach:

Abbildung 13.1. snom Konfiguration

Bereits nach diesen Einstellungen sollten sich das Telefon erfolgreich mit der Anlage verbinden können.

Für ein Massen-Rollout ist ein solches Verfahren natürlich denkbar ungeeignet. Daher sind nachfolgend die Schritte beschrieben, die ein Rollout auch in großen Stückzahlen ermöglichen.

2.2.2. Automatische Konfiguration

Zunächst sollten Sie Ihren DHCP-Server mit ein paar zusätzlichen Einstellungen konfigurieren. Es wird davon ausgegangen, dass der Telefonserver und die Telefone in einem eigens dafür eingerichteten Netzwerk agieren. Sollte eine Integration in ein bestehendes Netzwerk durchgeführt werden, ist im Einzelfall zu überprüfen, ob sich die notwendigen Einstellungen mit bereits im Netz befindlichen Geräten vertragen.

Damit die Telefone in der Lage sind, sich selbständig zu konfigurieren, muss ein spezieller DHCP-Parameter beim DHCP-Request mit übergeben werden. Dies kann bei den meisten Linux-Systemen über die Datei `/etc/dhcp.conf` festgelegt werden:

```
option tftp-server-name "http://pbx.apfelmus-gmbh.de/snom/settings/snom.php?mac={mac}";
```

Nachdem die Geräte Ihre IP-Adresse erhalten haben, setzen Sie automatisch eine Abfrage an die oben angegebene URL ab, um Ihre Konfiguration zu erhalten. Dabei wird der Parameter `{mac}` automatisch mit der MAC-Adresse des Telefons belegt. Man sollte sich nicht verwirren lassen, dass ein HTTP- statt eines TFTP-Servers über diesen Parameter konfiguriert wird. Die Telefone erkennen automatisch, dass Sie eine HTTP-Request durchführen müssen. Es muss kein zusätzlicher TFTP-Server konfiguriert werden. Nachfolgend ein Beispiel für ein PHP-Skript, welches als Rückgabe die Settings für ein bestimmtes Telefon liefert:

```
<html>
<pre>

<?php
 // Die MAC-Adresse aus der URL (GET) lesen und in $mac abspeichern.
 $mac = $_GET['mac'];

 // Allgemeine Konfiguration
 readfile("snom-base.htm");

 // Spezielle Konfiguration fuer diese MAC-Adresse
 readfile("snom-".$mac.".htm");
?>

</pre>
</html>
```

Über das Skript wird der Inhalt mehrerer Dateien übermittelt. Neben dem Header, dem Footer und den Einstellungen, die bei allen Endgeräten konstant sein sollen, wird des weiteren eine Datei speziell für das Telefon mit der übergebenen MAC-Nummer übermittelt.

In der Datei `snom-base.htm` befinden sich die Settings, die für alle Telefone identisch sein sollen. In der Regel kann man hier Sprach- und Datumseinstellungen, feste Tastaturbelegungen, etc. konfigurieren.

```
# After each setting (before the colon) you can set a flag, which means respectively:
# ! means writeable by the user, but will not overwrite existing
# $ means writeable by the user, but will overwrite existing (available since version 4.2)
# & (or no flag) means read only, but will overwrite existing

# more settings can be found at the settings (dump) page of the phone's build in webinterface

# Language and Time settings

language$: Deutsch
web_language$: Deutsch
```

```
timezone$: GER+1
date_us_format&: off
time_24_format&: on
tone_scheme&: GER

# define the firmware update policy here
# valid values are <auto_update>, <ask_for_update>, <never_update_firm>,
# <never_update_boot>, <settings_only>
update_policy: auto_update

#define the firmware update interval here, amount in minutes, default is 1440 = 1 day
firmware_interval: 2880

setting_server!: http://pbx.apfelmus-gmbh.de/snom/settings/snom.php?mac={mac}
subscribe_config!: off
update_server!: http://pbx.apfelmus-gmbh.de/snom/settings/snom.php?mac={mac}
contrast!: 14

dtmf_speaker_phone!: on
web_language!: Deutsch
dkey_snom&: url http://pbx.apfelmus-gmbh.de/snom/webapps/mainmenu.xml

admin_mode: off
admin_mode_password$:4321
admin_mode_password_confirm$:4321

alert_internal_ring_text: alert-internal
alert_external_ring_text: alert-external
alert_group_ring_text: alert-group
```

Alle oben aufgeführten Einstellungen sind im Grunde Standard-Einstellungen. Besonderes Augenmerk gilt den Parametern `dkey_snom`, `alert_internal_ring_text`, `alert_external_ring_text` und `alert_group_ring_text`. Deren Bedeutung wird später noch genauer erläutert.

Passend zur allgemeinen Konfiguration bekommt jedes Telefon auch eine spezifische Konfiguration. Jeder Teilnehmer sollte in der Lage sein, bei einem Bürowechsel oder einem Reset des Telefons genau die Konfiguration an seinem Telefon wiederzufinden, die ihm vom Administrator zugewiesen wurde. Telefonspezifische Einstellungen werden daher in einer Datei `snom-{mac}.htm` abgespeichert, also bei einem Telefon mit MAC-Adresse *`00:04:13:23:8B:60`* in der Datei `snom-000413238B60.htm`. Der Aufbau des Dateinamens ist beliebig, wichtig ist dabei die Einhaltung der Kette zwischen den Einstellungen innerhalb der DHCP-Konfiguration, der dazugehörigen Skriptdatei und letztendlich der Settings-Datei des Telefons. Anbei die spezielle Konfiguration eines solchen Telefons:

```
# After each setting (before the colon) you can set a flag, which means respectively:
# ! means writeable by the user, but will not overwrite existing
# $ means writeable by the user, but will overwrite existing (available since version 4.2)
# & (or no flag) means read only, but will overwrite existing

# First account
user_active1!: on
user_realname1$: Hans Mustermann <2000>
user_name1$: 2000
user_host1&: pbx.apfelmus-gmbh.de
user_pass1$: 1234

# You may add up to 12 accounts

# set second account to active outgoing identity
active_line$: 1

# in order to perform automated updates, define the firmware setting file URL
# where you specify the final firmware image URL
firmware_status: http://pbx.apfelmus-gmbh.de/snom/firmware360.htm
```

Man erkennt in der Konfiguration exakt die Settings wieder, die auch oben in der manuellen Einstellung konfiguriert wurden. Wichtig sind die Flags, die man pro Parameter angeben kann:

!	Der Parameter kann vom Anwender über das Telefon geändert werden. Bestehende Einstellungen werden beim Laden des Settings-Files nicht geändert.
$	Der Parameter kann vom Anwender über das Telefon geändert werden. Bestehende Einstellungen werden beim Laden des Settings-Files überschrieben.
& oder kein Flag	Der Parameter kann vom Anwender nicht über das Telefon geändert werden. Bestehende Einstellungen werden beim Laden des Settings-Files überschrieben.

Man sollte sehr genau abwägen, welche Parameter man für die Anwender freigibt und welche man festzurrt. Allein die Reduktion auf den User-Mode statt den Admin-Mode des Telefons erleichtert Administratoren später den Alltag.

Tipp

Es ist sehr leicht möglich ein Template für die snom Telefone zu erstellen, ohne zunächst mühsam die komplette Dokumentation zu lesen. Am einfachsten konfiguriert man zunächst ein Telefon manuell über einen WEB-Browser. In der Navigationsleiste des WEB-Interface des Telefons befindet sich ganz unten ein Punkt "Einstellungen" (der Admin-Modus muss aktiv sein!). Dort werden dann im Browser alle aktuellen Einstellungen mit den passenden Parametern ausgegeben. Man braucht also nur noch die gewünschten Parameter in eine entsprechende Datei zu kopieren. Bitte übernehmen Sie auf gar keinen Fall alle Parameter aus der Liste sondern ausschließlich die, die für Ihre spezielle Konfiguration wichtig sind. Danach entscheiden Sie über die die oben angegebenen Flags, welcher der Parameter vom Anwender geändert werden dürfen und welche nicht.

Wichtig

Sollte die automatische Konfiguration aus irgendeinem Grund nicht funktionieren, so lohnt sich ein Blick in die Datei `/var/log/messages` um herauszufinden, welche DHCP-Anfragen gestellt wurden sowie in das Access-Log des Web-Servers, ob überhaupt die richtigen Dateien angefordert wurden. Bei einem Apache 2.0 auf einem Debian Linux wäre dies z.B. die Datei `/var/log/apache2/access_log`. Mittels dieser beiden Logs lassen sich die Fehler bei der Konfiguration in der Regel sehr schnell finden und beheben. Zur Fehlersuche kann man zunächst auch mit statischen Konfigurationsdateien statt mit einem PHP-Skript arbeiten. So wird die Komplexität verringern und es können Fehler in der Skipt-Datei ausgeschlossen werden.

Warnung

Die snom Telefone brauchen zum Laden und Aktivieren der Settings einige Sekunden. Auch wenn das Telefon sich zunächst in den Standard-Konfigurations-Modus versetzt, warten Sie erst einige Sekunden, bevor sie einen vermeindlich fehlerhaften Versuch abbrechen.

2.3. Unterschiedliche Klingeltöne

Nachfolgend ist ein kleines Beispiel aufgeführt, wie man eine unterschiedliche Signalisierung für interne, externe und Gruppenanrufe realisieren kann. Dabei spielen die Parameter `alert_internal_ring_text`, `alert_external_ring_text` und `alert_group_ring_text` eine Rolle. Über den SIP-Header ist es möglich, eine Signalisierungsinformation mit zu übergeben, auch Alert-Info genannt. Die Telefone erkennen dann über diese Info, ob sie ggfs. eine andere Ruf-Signalisierung nutzen sollen, soweit dies vom Anwender konfiguriert wurde. Dazu muss vor der eigentlichen Anwahl mittels des **Dial()**-Kommandos der **SIPAddHeader()**-Befehl abgesetzt werden. Anbei ein entsprechender Auszug aus der `extensions.conf` Datei.

```
exten => _2XXX,1,SIPAddHeader("Alert-Info: <http://pbx.apfelmus-gmbh.de>\;info=alert-internal\;x-line-id=
exten => _2XXX,2,Dial(SIP/${EXTEN})
```

Das Format des SIP-Headers ist vorgegeben, wichtig ist in diesem Fall eigentlich nur der zweite Parameter `info=alert-internal` . Hier muss einer der drei konfigurierten Texte stehen, an denen das Telefon die Art des Rufes erkennen soll. Welcher Text dort steht, ist eigentlich egal, aber er muss jeweils mit den Werten übereinstimmen, die bei den Telefon-Parametern `alert_internal_ring_text`, `alert_external_ring_text` oder `alert_group_ring_text` übergeben wurden. Es empfiehlt sich die jeweiligen Default-Werte zu übernehmen.

Wichtig

Bitte beachten Sie, dass die Semikolons im Alert-Info mittels "\"-Zeichen gequotet werden müssen. Ansonsten interpretiert Asterisk alles ab dem ersten Semikolon als Kommentar und der Befehl wird nur unvollständig oder gar nicht ausgeführt.

In der oberen Variante wird davon ausgegangen, dass das Telefon anhand des Info-Parameter selbst entscheidet, welchen Rufton es auswählen muss. Man kann diesen Befehl aber auch dazu nutzen, einen Rufton vorzugeben, indem man eine entsprechende URL angibt, die auf eine Sound-Datei verweist.

```
exten => _2XXX,1,SIPAddHeader(Alert-Info: <http://pbx.apfelmus-gmbh.de/snom/sounds/snom_trumpet.wav>)
exten => _2XXX,n,Dial(SIP/${EXTEN})
```

Es bietet sich an, einfach mit den verschiedenen Möglichkeiten zu experimentieren. Eine Unterscheidung zwischen internen, externen und Gruppenanrufen macht in den meisten Fällen durchaus Sinn.

2.4. Benutzerdefiniere Menüführung (nur snom 360)

Die Geräte der 360er Baureihe verfügen über von außen programmierbare Menüs. Die zugrundeliegenden XML-Strukturen sind unter http://www.snom.info/wiki/index.php/Xmlobjects ausführlich erläutert. In der oben angegebenen automatischen Konfiguration wurde bereits auf die Zeile `dkey_snom&: url http://pbx.apfelmus-gmbh.de/snom/webapps/mainmenu.xml` hingewiesen. Hier wurde die auf dem Telefon befindliche snom-Taste umbelegt, so dass der interne XML-Browser die dort verlinkte XML-Datei aufruft und interpretiert. Wie schon bei der automatischen Konfiguration der Telefone ist die Installation eines WEB-Servers auf dem Telefonanlagenrechner zu empfehlen, zumal die meisten Linux-Distributionen diese Option bereits von Hause aus anbieten.

Die Datei `mainmenu.xml` hat folgenden Aufbau:

```
<?xml version="1.0" encoding="UTF-8"?>
<SnomIPPhoneMenu>
<Title>Menu</Title>
<MenuItem>
<Name>Globales Adressbuch</Name>
<URL>http://pbx.apfelmus-gmbh.de/snom/webapps/phonebook/phonebook.php</URL>
</MenuItem>
<MenuItem>
<Name>Sondernummern</Name>
<URL>http://pbx.apfelmus-gmbh.de/snom/webapps/specialnums.xml</URL>
</MenuItem>
</SnomIPPhoneMenu>
```

Ein Druck auf die snom-Taste führt zu einem Menü mit den beiden Einträgen "Globales Adressbuch" und "Sondernummern". Wie von der Benutzerführung gewohnt, kann man innerhalb dieser Menüs mit den Telefon-Cursor-Tasten navigieren und Einträge mit der Häckchentaste auswählen. Eine Ebene zurück gelangt man mittes der X-Taste.

Tipp

Den "xml version" Header kann man auch weglassen, der eingebaute Browser ist in der Lage, das XML-Menü auch ohne diese Header korrekt zu interpretieren.

Diese Menüform führt noch keine Wahlvorgänge durch, sondern dient nur der Anzeige von Menüeinträgen, mit denen man wiederum neue URLs aufrufen kann.

Weiter oben wurde erläutert, dass man mittels der "Retrieve"-Taste ausschließlich dann die Voicemail anrufen kann, wenn neue Nachrichten eingegangen sind. Es wurde daher eine Extension "2999" definiert, welche die Abfrage der Mailbox zu jeder Zeit ermöglicht. Diese kann man durch ein spezielles XML-Menü anwählen lassen. Das Beispiel `specialnums.xml` zeigt genau dies:

```
<?xml version="1.0" encoding="UTF-8"?>
<SnomIPPhoneDirectory>
<Title>Spezialnummern</Title>
<Prompt>Prompt</Prompt>
<DirectoryEntry>
<Name>Mailbox</Name>
<Telephone>2999</Telephone>
</DirectoryEntry>
</SnomIPPhoneDirectory>
```

Der Anwender bekommt nach Auswahl den Menüs "Sondernummern" nun einen Eintrag "Mailbox" angezeigt, den er wie gehabt auswählen kann. Durch das XML-Objekt `SnomIPPhoneDirectory` weiß das Telefon, dass nachfolgend Rufnummern gelistet sind, die es nach der Auswahl durch den Anwender anrufen soll. Nachdem der Anwender also den Eintrag "Mailbox" gewählt hat, ruft das Telefon die *2999* an. Nach dem oben angegebenen Wählplan also die Abfrage der eigenen Voicemail.

Warnung

snom Telefone, die mit der Firmware 6.5.2 oder früher ausgestattet sind, zeigen kurz vor der Anwahl einen xml response error im Display an. Dies ist ein Fehler der Firmware und hat nichts mit den XML-Konfigurations-Dateien zu tun. Einfluss auf die Funktionalität hat dieser Fehler nicht.

2.5. Ansteuerung der Leitungs-Tasten und -LEDs

snom Telefone bieten die Möglichkeit, die Leitungstasten nicht nur mit speziellen URLs oder Kurzwahlen zu belegen, man kann diese auch zur Überwachung und Übernahme (PickUp) anderer Leitungen nutzen, was z.B. bei Gruppenanrufen eine sehr nützliche Angelegenheit ist. Momentan setzt dies die Junghanns bristuffed Version von Asterisk voraus (siehe ???)!

Zunächst muss in der Datei `sip.conf` die `[general]`-Sektion erweitert werden:

```
[general]
allowsubscribe=yes              ; Enable support for subscriptions.
notifyringing=yes               ; Notify subscriptions on RINGING state (default: no)
notifyhold=yes                  ; Notify subscriptions on HOLD state (default: no)
useclientcode=yes
```

Des Weiteren muss in der Sektion des Telefons, welches die überwachende Rolle übernimmt, eine sogenannte "Subscription" eingetragen werden. Damit meldet das Telefon bei Asterisk an, dass es über Status-Änderungen anderer Telefone informiert werden will. Der Kontext, für den sich das Telefon anmeldet, muss identisch sein mit dem Kontext, in dem die sogenannten "Hints" der Telefone gesetzt werden, die überwacht werden sollen. In der Regel ist dieser identisch mit dem Kontext, in dem sich die eigentlichen Telefone bewegen.

```
[2000]
type=friend
context=meine-telefone
subscribecontext=meine-telefone
secret=1234
host=dynamic
mailbox=2000
```

Für die zu überwachenden Telefone muss innerhalb der `sip.conf` kein Eintrag hinzugefügt werden, dies geschieht ausschließlich in der `extentions.conf`. Im Wählplan müssen die oben bereits erwähnten "Hints" gesetzt werden. Über diese Hints wird Asterisk quasi erlaubt, Informationen über den Zustand der dort definierten Extensions weiterzugeben, nämlich genau an die oben konfigurierten Subscriber. Es ist wichtig, dass die Hints genau in dem Kontext gesetzt werden, in dem sich das überwachende Telefon über die Zeile `subscribecontext=...` angemeldet hat, ansonsten werden die Status-Informationen nicht weitergeleitet. Dies ist durchaus Absicht, damit man nicht

per se von allen Telefonen alle weiteren überwachen, sondern durch die Wahl des Kontextes eine Auswahl treffen kann.

```
[sonstige]

[meine-telefone]
exten => 2001,hint,SIP/2001

exten => _2XXX,1,Dial(SIP/${EXTEN},20,j)
exten => _2XXX,2,VoiceMail(u${EXTEN})
exten => _2XXX,102,VoiceMail(b${EXTEN})

exten => 2999,1,VoiceMailMain(s${CALLERID(num)})

exten => asterisk,1,VoicemailMain(s${CALLERID(num)})
```

Wichtig

Die hint-Priorität war in der Vergangenheit immer für einige Tücken und Probleme bekannt. In einigen älteren Asterisk-Versionen ist die Priorität case-sensitive, muss also auf alle Fälle klein geschrieben werden. Ebenso funktioniert auch nicht in allen Asterisk-Versionen die Benutzung von Variablen für die zu überwachenden Extensions. Der sicherste Weg ist daher für jede zu überwachende Extension einen Eintrag exakt in der obigen Form anzulegen. Der Übersichtlichkeit halber kann man alle gesammelten Hints auch in einen eigenen Kontext legen und diesen überall dort mittels `include =>...` einbinden, wo er benötigt wird.

Nun muss noch das überwachende Telefon konfiguriert werden. Dazu melden Sie sich wie gewohnt mit einem WEB-Browser am Telefon an und gehen in das Untermenü "Funktionstasten". Dort belegen Sie die Funktionstaste Ihrer Wahl (im Beispiel die P6) mit dem Typ "Ziel" und tragen die zu überwachende Extension ein, hier also die 2001.

Nachdem die Einstellungen gespeichert wurden, ändert das Telefon den Eintrag automatisch in eine passende URI der Form <sip:2001@xxx.xxx.xxx.xxx;user=phone> (xxx.xxx.xxx.xxx ist die IP-Adresse der Telefonanlage) ab:

Nun sind alle Einstellungen getätigt. Zunächst muss Asterisk neu gestartet werden und anschließend die Telefone. Der Telefon-Neustart nach dem Asterisk-Neustart ist für die saubere Anmeldung der Subscription des Telefons notwendig. Zunächst sollte überprüft werden, ob alles richtig konfiguriert wurde. Dazu geben sie im Asterisk-CLI die Zeile **show hints** (**core show hints** in Version 1.4) ein:

```
*CLI> show hints

    -= Registered Asterisk Dial Plan Hints =-
   2001               : SIP/2001             State:Unavailable     Watchers  0
----------------
- 1 hints registered
*CLI>
```

In obigem Beispiel wird deutlich, dass das zu überwachende Telefon noch nicht am Asterisk-Server angemeldet ist (*unavailable*). Ebenso ist auch noch kein überwachendes Telefon an den Status dieser Extension angeklinkt (Watchers-Wert ist 0). Letzteres wird auch durch die Auflistung der aktiven Subscriptions bestätigt, welche durch das Kommando **sip show subscriptions** initiiert wird.

```
*CLI> sip show subscriptions
Peer             User        Call ID      Extension        Last state     Type
0 active SIP subscriptions
*CLI>
```

Nun melden Sie zunächst das überwachende Telefon an und setzen den Befehl erneut ab:

```
*CLI> sip show subscriptions
Peer             User        Call ID      Extension        Last state     Type
192.168.0.2      2000        815d944554e  2001             Unavailable    dialog-info+xml
1 active SIP subscription
*CLI>
```

Hier wird deutlich, dass der User 2000 die Extension 2001 überwacht. Bereits jetzt leuchtet die Lampe am Telefon, die zur Überwachung konfiguriert wurde. Es ist eine Eigenschaft des bristuff-Patches, dass überwachte, aber noch nicht angemeldete Endgeräte, über eine dauerhaft leuchtende LED angezeigt werden.

Nun melden Sie das zu überwachende Telefon erneut an. Bereits mit der Anmeldung sollte folgende Zeile auf dem Asterisk-CLI erscheinen:

```
Extension Changed 2001 new state Idle for Notify User 2000
```

Die LED am überwachenden Telefon erlischt zeitgleich. Setzen Sie den Befehl **show hints** (**core show hints** in Version 1.4) erneut ab:

```
*CLI> show hints

    -= Registered Asterisk Dial Plan Hints =-
   2001               : SIP/2001             State:Idle           Watchers  1
----------------
- 1 hints registered
*CLI>
```

Im Gegensatz zu vorher sieht man, dass das Telefon einen definierten Zustand hat ("Idle") und zudem ein weiteres Telefon den Zustand überwacht (Watchers-Wert ist 1). Die Konfiguration ist nun komplett, Asterisk meldet Statuswechsel der überwachten Extension sofort an das überwachende Telefon. Befindet sich das überwachte Telefon im Gespräch, leuchtet die LED dauerhaft. Wird das überwachte Telefon angerufen, blinkt die LED. Ohne Aktivität ist auch die LED gelöscht. Die unterschiedlichen Stati sind ebenfalls auf der Konsole sichtbar:

```
Extension Changed 2001 new state InUse for Notify User 2000
Extension Changed 2001 new state Ringing for Notify User 2000
Extension Changed 2001 new state Idle for Notify User 2000
```

In der bisherigen Konfiguration werden Gespräche des überwachten Teilnehmers nur angezeigt. Gerade bei Gruppenanrufen ist es nützlich, dass ein anderer Teilnehmer diese Anrufe auch übernehmen kann. Um dies zu ermöglichen, muss noch ein weiterer Eintrag in der extensions.conf hinzugefügt werden.

```
[meine-telefone]
exten => 2001,hint,SIP/2001

; Gespraechsuebernahme (PickUp) fuer bristuff
exten => _*8.,1,PickUpChan(SIP/${EXTEN:2})
```

Jetzt können am überwachten Telefon eingehende Anrufe einfach am überwachenden Telefon angenommen werden, indem die zur blinkenden LED passende Taste gedrückt wird. Das Gespräch wird dann sofort übernommen, die Nummer des Anrufers im Display angezeigt.

Tipp

Um bereits zur Anrufzeit und nicht erst nach der Übernahme die Nummer des Anrufers im überwachenden Telefon angezeigt zu bekommen, müssen in den "Erweiterten Einstellungen" des snom Telefons zwei weitere Einstellungen geändert werden. Das Setting "Dialog-Info Call Pickup" muss aktiviert sein, "Pakete vom Registrar filtern" muss deaktiviert werden. Die erweiterten Einstellungen sind nur im Admin-Modus des Telefons verfügbar.

Sollen die in diesem Kapitel erläuterten Änderungen in Form einer automatischen Konfiguration ins Telefon geladen werden, müssen folgende Einstellungen Teil der speziellen Konfigurarion des Telefons werden:

```
fkey5: dest <sip:2001@pbx.apfelmus-gmbh.de;user=phone>
filter_registrar: off
callpickup_dialoginfo: on
```

Wichtig

Die Funktionstasten sind 0-indiziert. Um wie im Beispiel die Funktionstaste 6 zu belegen, muss der Parameter fkey5 gesetzt werden. Die Funktionstaste 1 wird mit fkey0 angesprochen.

2.6. Weitere verfügbare Applikationen

Es gibt bereits einige frei verfügbare Applikationen, vor allem im Bereich der Telefonbücher. Zwei empfehlenswerte Implementationen finden sich hier:

- Phonebook for snom phones: Ein Telefonbuch-Browser mit Suchfunkton und LDAP-, Datei- oder SQL-Anbindung für die Telefondaten (http://www.bevuta.com/phoneapps/)
- IPPhone XML library for PHP: Sehr empfehlenswert, eigentlich eine Bibliothek zur Entwicklung von PHP-basierten Telefonbüchern auch für andere Telefontypen wie Cisco, aber mit einem Beispiel für einen T-9 geführten Telefonbuch-Browser mit Datei- oder SQL-Anbindung für die Telefondaten (http://swt.uni-stuttgart.de/~langausd/asterisk/).

Vor allem letzteres Beispiel ist eine sehr gute Grundlage für zentrale Telefonbücher. Die T-9-Führung passt ideal auf das snom Telefon, welches über keine echte alphanummerische Tastatur verfügt, wie viele Systemtelefone. Wer seine Benutzerdaten bereits in einem zentralen LDAP-Server verwaltet, sollte sich dagegen die erste Lösung genauer ansehen.

Kapitel 14. Danksagungen

1. Das Kernteam

Eigentlich ist es bei einem technischen Buch völlig unmöglich allen Personen, Projekten, Webseiten oder sonstigen Institutionen zu danken die einem bei der Realisierung geholfen haben. So habe ich oft bei Ungereimtheiten[1] im IRC Channel `#asterisk` Hilfe aus der ganzen Welt bekommen! Oft habe ich auch in den Kommentaren auf *http://www.voip-info.org* Möglichkeiten gefunden bestimmte Probleme noch besser zu lösen. Diesen ganzen anonymen Helfern möchte ich hiermit meinen Dank aussprechen.

Natürlich gibt es auch Personen die ganz besonders geholfen haben. Eigentlich ist es ungerecht, das als Autor nur eine Person genannt wird. Denn die folgenden Personen haben wertvolle Hilfe und Unterstützung zu diesem Buchprojekt geliefert. Ohne sie wäre dieses Buch nie möglich gewesen!

- PhilippKempgen<pkempgen@uni-koblenz.de>KoblenzGermany

  Philipp hat große Teile des Buches aus der ersten LaTeX Version in die DocBook Version konvertiert. Dabei hat er viele Fehler im Text korrigiert. Nachdem er sich dann in die Materie eingearbeitet hatte, hat er noch sehr viel am Buch gearbeitet. Ein herzliches Danke an dieser Stelle!

- MilisavRadmanic<radmanic@suse.de>NürnbergGermany

  Ohne meinen langjährigen Freund Milisav wäre das ganze Buch nie zustande gekommen oder es wäre in einer unlesbaren Form gewesen. Er hat jeden Absatz korrigiert und sprachlich feingeschliffen. Zusätzlich hat er technische Ungereimtheiten aufgedeckt und dabei meistens direkt korrigiert.

- FlorenceMauriceDr.<florence_maurice@web.de>MünchenGermany

  http://www.maurice-web.de

  Dr. Florence Maurice hat das gesamte Buch auf Rechtschreibfehler und sprachliche Fehler durchsucht und diese korrigiert. Dabei mußte sie sich am Anfang mit LaTeX und später mit DocBook XML rumschlagen. Danke! :-)

- BorisKarnikowski<bkarnikowski@pearson.de>MünchenGermany
  <affiliation></affiliation>
  Pearson Education Deutschland GmbH
  Markt+Technik/Addison Wesley

  http://www.pearson.de

  Boris Karnikowski ist der geistige Vater dieses Buches. Als Lektor von Pearson hat er mich (Stefan Wintermeyer) gefragt, ob ich Lust habe ein Buch über Asterisk zu schreiben. In diesem Prozess habe ich es ein ganzes Jahr lang geschaft nicht eine Deadline einzuhalten und mich am Ende gewundert, das er überhaupt noch mit mir spricht. ;-)

  Boris hat geholfen, das dieses Buch unter der GNU Free Documentation Licence veröffentlicht werden konnte. Dafür bin ich ihm und dem Verlag besonders dankbar!

- Kevin P.Fleming<kpfleming@digium.com>USA
  <affiliation></affiliation>
  Digium

  Kevin hat freundlicherweise die current Links auf dem FTP- und HTTP-Server eingerichtet. Ohne diese Links wäre eine brauchbare Installationsanleitung in ??? schier unmöglich gewesen.

[1] Und davon gibt es bei Asterisk einige! ;-)

2. Spezialisten

In dieser Rubrik möchte ich Personen danken, die ganz ganze Abschnitte oder Kapitel entworfen oder mitentworfen haben. Vielen Dank für die Hilfe!

- Mathias Paezolt <mathias@paezolt.de>

  Mathias hat das Kapitel über die Faxserverlösung mit Hylafax geschrieben. Siehe Kapitel 11, *Asterisk als Faxserver*

- Norbert Huffschmid <norbert.huffschmid@hawhaw.de>

  Norbert hat das Kapitel über VoiceXML geschrieben. Siehe Kapitel 9, *Asterisk und VoiceXML*

- Norbert A. Richartz <richartz@networkers.de>

  Norbert Richartz von der Firma *Networkers AG* hat die Beschreibung zu den snom Telefonen geschrieben. Siehe Abschnitt 2, „snom VoIP Telefone“

3. Helfer beim Beta-Test

Die folgenden Personen haben beim offiziellen Beta-Test des Buches Fehler gefunden und gemeldet. Die Liste ist chronologisch geordnet. An dieser Stelle noch einmal ein herzliches DANKE an alle!

- Fabian Müller <fabian_mueller@open-tk.de>
- Ulrich Papendick <u.papendick@lambdalogic.de>
- Florian Esser <flo@gmxpro.net>
- Stefan Strauss <seneka.core@googlemail.com>
- Johannes Hubertz <johannes@hubertz.de>
- Detlef Ralf Pauly <dettodet@gmail.com>
- Matthias Krauß <mogi@mogi.de>
- Joern Allmers <web@allmers.de>
- Philippe Devaux <ph.devaux@bluewin.ch>
- Ronald Wimmer <ronald.wimmer@ronzo.at>
- Bernd Langehegermann <b.langehegermann@bls-integration.de>
- Joern Allmers <web@allmers.de>
- Marcel Freese <marcel@freese.name>
- Christian Bräunlich <c.braeunlich@negele.net>
- Tobias Kowatsch <tobias.kowatsch@googlemail.com>
- Christian Krick <mail@christian-krick.name>
- Sascha John Hesse
- Richard Karsch <r.karsch@web.de>
- Andreas Schulze <asterisk-buch@andreasschulze.de>

- Andreas Rehm <mail@andreasrehm.de>
- Dirk Paulsen <big.albatros@gmx.de>
- Bernhard M. Wiedemann <bernhard@lsmod.de>
- Marcus Franke <marcus.franke@gmx.net>
- Stephan Goeldi <stephan@goeldi.org>
- Beata Wroblewska
- Markus Zielonka <zielonka.markus@googlemail.com>
- Markus Bönke <markus.boenke@gmx.de>
- Gernot Stöckl <gernot.stoeckl@snet.at>
- Tobias Siebenlist <mail@tobias-siebenlist.de>
- Thomas Netousek <thomas@netousek.com>

Glossar

Hier werden die im Zusammenhang mit Asterisk verwendeten Begriffe und Abkürzungen kurz erklärt und gegeneinander abgegrenzt.

ADSI
: Analog Display Services Interface. Ein von einigen analogen Telefonen mit Text-Display unterstütztes Verfahren, das z.B. die Anrufer-Nummer oder die Anzahl der eingegangenen Nachrichten überträgt und ein entsprechendes Blinklicht aktiviert. Wird per FSK (FSK) übertragen.

AGI
: Asterisk Gateway Interface. Ein Interface, über das Asterisk mit externen Skripten kommunizieren kann (ähnlich CGI bei Web-Servern). Die weiteren Formen EAGI, FastAGI und DeadAGI werden in ??? erklärt.

AMA
: Automatic Message Accounting. Spezifiziert Mechanismen zur Erzeugung und Übermittlung von CDRs (CDR).

Applikation
: Applikation im Dial plan (Dial Plan). ....

  Nicht zu verwechseln mit einer Funktion (Funktion).

AstDB
: Die in Asterisk integrierte Datenbank, aufbauend auf der Berkeley DB (BDB, Berkeley DB)

BDB
: Siehe Berkeley DB.

Befehl
: In Bezug auf den Dial-plan siehe Applikation

Berkeley DB
: Eine alte und in der UNIX-Welt (und Derivaten) weit verbreitete Datenbank, die von vielen Anwendungen genutzt wird und meist nicht extra installiert werden muss.

Caller ID
: Auch kurz CID. Anrufer-Identifikation/-Kennung. Die Telefonnummer und ggf. auch der Name eines Anrufers. Bekannt z.B. bei ISDN oder Mobilfunk-gesprächen von der Rufnummernanzeige im Display.

CDR
: Call Detail Record oder Call Data Record, im Dt. auch Kommunikationsdaten-satz (KDS). Solche Anrufprotokolle können z.B. zur Abrechnung von Gesprächen verwendet werden. Asterisk verwendet das folgende Format:

  ```
  acct code, src, dst, dst context, cid, chan, dst channel, last app, last app
  ```

Channel
: So werden in Asterisk Verbindungen oder Verbindungsdefinitionen zu Teil-nehmern (users) oder anderen Telefonanlagen (peers) bezeichnet. Es gibt mehrere mögliche Übertragungsprotokolle, u.a. IAX (IAX), SIP (SIP), H.323, CAPI, ... die in ??? vorgestellt und beschrieben werden.

Codec
: So bezeichnet man Verfahren zur Kodierung und Dekodierung von Datenströ-men, hier Audio-Daten. Das sind z.B. GSM, WAV oder das bekannte MP3. Meist ist auch immer ein entsprechendes Dateiformat gemeint.

Context
: So wird in Asterisk ein Bereich oder Abschnitt in der `extensions.conf` oder `voicemail.conf` genannt, der z.B. Telefonnummern einer Firma zu einer Abteilung gruppiert. Ein besonderer Kontext ist der `[default]` Kontext, der immer existiert und oft auch ausreichend ist. Siehe "Kapitel Voicemailsystem -> Freie Kontexte" (linken!!!).

Dial Plan
: Die in der Datei `extensions.conf` festgelegten Aubläufe, die aufgerufen werden, sobald eine Extension (Extension) gewählt wird.

DNS — Domain Name System/Server/Service. Ein Name-Server löst Domain-Namen (z.B. example.com) zu IP-Adressen auf.

DoS — Denial of Service. Bezeichnung für einen Angriff auf einen Rechner, bei dem dieser mit Anfragen überschwemmt wird, wodurch er nicht mehr in der Lage ist zu reagieren.

Durchwahl — Siehe Extension.

DSCP — Differentiated Service CodePoint. Eine Standardisierung von ToS-Flags (ToS) in RFC 2474. Verwendet die ersten 6 Bits im ToS-Byte von IPv4- bzw. das Class Field in IPv6-Paketen zur Priorisierung von Übertragungen.

Da so eine Priorisierung natürlich missbraucht werden kann[2], werden diese Flags von Internet-Providern in aller Regel überschrieben (Trust boundary). Sie können aber sicherstellen, dass Ihre internen Switches diese Flags erlauben (also keine Trust boundaries bilden).

Die sinnvollste Einstellung des DSCP für VoIP-Anwendungen ist EF (Expedited Forwarding, also Express-Übertragung), was geringe Latenz (Latenz), geringe Verlustrate, und wenig Jitter (Jitter) bedeutet. Alternativ sind verschiedene Klassen von AF (Assured Forwarding) vorgesehen, die sich jedoch für Telefonie weniger eignen. Die Standard-Einstellung ist BE (Best Effort), die normale (niedrigste) Priorität.

DTMF — Dual-Tone Multi-Frequency. Wähltöne beim Mehrfrequenz-Wählverfahren (MFV, auch Tonwahlverfahren oder Touch Tone), wird von den meisten analogen Telefonen benutzt. Im Gegensatz zum älteren Impuls-Wählverfahren (IWV), Pulse dialing. Die meisten Telefone lassen sich bei abgenommenem Hörer vorübergehend mit der Tastenfolge `Set * Set` zwischen den beiden Verfahren umstellen (`Set` ist die Taste mit einem Pfeil nach rechts mit Karo). Mit DTMF sind außer den Ziffern `0-9` die Symbole `*` und `#` sowie die Buchstaben `A-D` (kaum verwendet) möglich.

DUNDi — Distributed Universal Number Discovery. Ein von Mark Spencer (Digium) entwickeltes Peer-to-Peer-System um Gateways zu Telefondiensten zu finden. Im Gegensatz zu z.B. ENUM (ENUM) arbeitet DUNDi dezentral und kann beispielsweise in einer Firma oder auch im ganzen Internet eingesetzt werden. Weitere Informationen: http://www.dundi.info/, http://www.dundi.com/, http://de.wikipedia.org/wiki/DUNDi, `doc/README.dundi` (1.2) / `doc/dundi.txt` (1.4)

E.164 — Die Richtlinie E.164 The international public telecommunication numbering plan der ITU-T regelt international die Adressierung der Telefonnetze. In E.164 ist festgelegt, aus welchen Bestandteilen eine Telefonnummer besteht und wie viele Stellen sie enthalten darf. Weiterhin sind die internationalen Vorwahlen (wie 43 für Österreich, 49 für Deutschland und 41 für die Schweiz) festgelegt.

—http://de.wikipedia.org/wiki/E.164

ENUM — E.164 number and DNS. Ein Verfahren um DNS (DNS) zur Verbreitung von E.164-Telefonnummern (E.164) einzusetzen. Eine neuere Methode ist DUNDi (DUNDi). Weitere Informationen: `doc/README.enum` (1.2) / `doc/enum.txt` (1.4), RFC 3761

[2] Microsoft Windows 2000 z.B. setzt Gerüchten zufolge immer eine ungerechtfertigt hohe Priorität

Extension

Die deutsche Übersetzung für *Extension* ist *Durchwahl-Nummer*, und das ist auch die grundlegende Bedeutung. In dieser Verwendung besteht eine Extension nur aus Ziffern (obwohl theoretisch auch die Symbole `*` und `#` und sogar Buchstaben möglich wären, was sich aber auf einem normalen Telefon nicht wählen lässt).

In Asterisk gibt es allerdings zusätzlich noch spezielle - i.d.R. einbuchstabige - Extensions, die nicht direkt am Telefon eingetippt werden, sondern zu denen man unter bestimmten Umständen oder durch Befehle im Dialplan (Dial Plan) verbunden wird.

Die besonderen Extensions sind:

`a` (Abort) Wird aufgerufen, wenn der Anrufer während der Voicemail (Voicemail)-Begrüßung `*` drückt.

`h` (Hangup) Gespräch wird aufgelegt. Hier kann z.B. noch eine Nachricht („Auf Wiederhören“) gespielt werden.

`i` (Invalid) Ungültige, nicht vergebene Nummer gewählt oder ungültige Eingabe in einem IVR (IVR)-Menü.

`o` (Operator). In einem IVR (IVR)-Menü `0` (Null) gedrückt um Verbindung mit einem Operator herzustellen.

`s` (Start) Start der Befehlsabarbeitung für eingehende Anrufe.

`t` (Timeout) Zeitberschreitung (User antwortet nicht auf Prompt).

`T` (AbsoluteTimeout) Durch die Dialplan-Applikation `AbsoluteTimeout()`.

`failed` Automatischer Anruf nach „draußen“ (auto dial-out; mit definiertem Kontext, Priorität und Extension) fehlgeschlagen.

`fax` Fax-Erkennung auf Zap-Kanälen.

`talk` Von der Dialplan-Applikation `BackgroundDetect()` ausgelöst.

Mit einer Extension sind oft auch Extension-Muster (Regular Expression) gemeint, die eine Gruppe von Extensions beschreiben.

Aus der Sicht des Administrators sind Extensions eine Reihe von in der `extensions.conf` festgelegten Kommandos (siehe Applikation), die abgearbeitet werden. Eine Extension ist also erstmal nur ein Einstiegspunkt im Dialplan; zu einer vollwertigen Nebenstellennumer eines Endgerätes wird die Extension erst durch das Verbinden mit einem entsprechenden Kanal durch `Dial()`. Im Dialplan können auch Extensions wie z.B. das Muster `_0.` für ausgehende Anrufe verwendet werden.

FSK

Frequency Shift Keying, dt.: Frequenzumtastung, ist digitale Frequenzmodulation. Man kennt das z.B. vom Piepsen eines Faxgerätes, das man vom Telefon aus anruft.

Funktion

Funktion im Dial plan (Dial Plan). ....

Nicht zu verwechseln mit einer Applikation (Applikation).

Hook

In Bezug auf Telefone die „Gabel“. On-hook (auf der Gabel) heißt also aufgelegt, Off-hook abgenommen.

IAX	InterAsterisk eXchange Protokoll (Hier ist meist IAX2, also Version 2 gemeint.), der Asterisk-eigene Channel (Channel).
IP	Internet Protocol. Ein grundlegendes Übertragungsprotokoll für IP-Pakete, auf dem die üblichen Protokolle TCP und UDP basieren, auf die wiederum Protokolle wie HTTP oder IAX aufgebaut sind. Die Übertragung von IP-Paketen ist nicht garantiert. IP-Pakete werden z.B. über eine Ethernet-Verbindung übertragen. IP wird oft auch nicht ganz korrekt kurz für IP-Adresse verwendet.
IVR	Interactive Voice Response system. Mit IVR kann man komplexe Menüstrukturen bauen, in denen der Anrufer mit seinem Ziffernblock am Telefon oder durch gesprochene Eingaben navigieren kann.
Jitter	Bezeichnet die unterschiedliche Übertragungsdauer (Latenz) von IP-Paketen, die bei VoIP (VoIP) in Abhängigkeit vom eingesetzten Codec (Codec) zu Sprach-Verzerrungen oder kurzen Aussetzern führen kann. Wird in Asterisk (wie auch in anderen Telefonanlagen oder Endgeräten) durch Jitter Buffer ausgeglichen.
Kanal	Siehe Channel.
Kommando	In Bezug auf den Dial-plan siehe Applikation
Kontext	Siehe Context.
Latenz	(engl. latency) Die auch Delay genannte Verzögerung oder Übertragungsdauer von IP-Paketen (IP). Gerade für Echtzeitkommunikation (Telefongespräche) ist eine niedrige Latenz wünschenswert.
MAC-Adresse	Media Access Control Adresse. Die feste, weltweit einmalige Hardware-Adresse (auch Ethernet-ID genannt) eines Netzwerk-Interfaces (Netzwerkkarte). (Kann nicht physikalisch, aber u.U. durch Software geändert werden.) Nicht zu verwechseln mit der IP-Adresse[3].
MOH	Music on Hold. Warteschleifenmusik. Bitte beachten Sie, dass Sie entgegen der verbreiteten Meinung keine „kommerziellen“ CDs oder MP3s abspielen dürfen; hierfür fallen ggf. GEMA-Gebühren an (http://www.gema.de/). Wenn Sie selber ein Volkslied singen, sind Sie auf der sicheren Seite :-)
MySQL	Die MySQL-Datenbank (http://www.mysql.org/)
NTP	Network Time Protocol. Ein standardiertes Protokoll (und entsprechende Algorithmen) mit dem sich die Uhr eines Rechners auf Millisekunden genau automatisch mit einem Zeit-Server synchronisieren läßt.
Pattern	Siehe Regular Expression.
Peer	In Asterisk die Bezeichnung für eine andere Telefonnanlage (PBX), zu der eine Verbindung über einen Channel (Channel) hergestellt wird. Also z.B. ein Asterisk-System an einem anderen Standort.
PBX	Private Branch eXchange. Eine Telefonanlage. Verbindet Endgeräte (i.d.R. Telefone) untereinander und mit dem öffentlichen Telefonnetz. Im Dt. auch TVA (Telefonvermittlungsanlage) oder (allgemeiner) TK-Anlage (Telekommunikationsanlage).

[3] oder den bekannten Computern :-)

Pound	Pound sign = Raute (#), Pound key = Rautetaste. Auf diesen Begriff werden Sie öfter in amerikanischen Beschreibungen stoßen. Der britische Ausdruck ist hash. Das #-Zeichen ist auch als Number sign oder Sharp bekannt.
Prompt	Siehe Voice Prompt.
PSTN	Public Switched Telephone Network. Das öffentliche Telefonnetz, insbesondere das Festnetz (landline).
QoS	Quality of Service (dt.: Dienstgüte). Bezeichnet die Gesamtheit der Qualität eines Übertragungsweges, die von Latenz (Latenz), Jitter (Jitter), Durchsatz, Fehlerrate und Verlustrate bestimmt wird. Kann in IP-Paketen durch die ToS-Flags (ToS) beeinflusst werden.
RegExp	auch RegEx oder RE. Siehe Regular Expression
Regulärer Ausdruck	Siehe Regular Expression.
Regular Expression	Ein „regulärer Ausdruck“. Im Falle des Dial-plans (Dial Plan) ein Suchmuster, das auf mehrere Extensions (Extension) zutrifft und aus Gründen der Übersichtlichkeit und leichteren Wartbarkeit verwendet wird. Pattern werden immer durch _ (Unterstrich) eingeleitet. Möglich im Pattern sind freie Platzhalter wie `[0-9]`, `[3-6]`, `[56]` etc. oder die vordefinierten Kurzformen `X`, `Z`, `N` und `.` (Punkt). Beispiel: `_2[3-6]X` RegExe sind in Asterisk nicht die aus Perl bekannten PCREs. Eine ausführliche Erklärung finden Sie im Abschnitt "Der Wählplan -> Platzhalter (in Kap. 2)" (linken!!!).
SIP	Session Initiation Protocol (eigentlich eine Sammlung von Protokollen), in Asterisk ein Channel (Channel).
TCP	Transmission Control Protocol. Eine Protokollschicht über IP (IP), die sicherstellt, dass bei der Übertragung keine IP-Pakete verlorengehen, was z.B. für HTTP wichtig ist, damit eine vollständige Übertragung gewährleistet ist.
ToS	Type of Service. Eine Form von Traffic shaping. Manche Router beachten solche Flags in IP-Paketen (IP), die dazu dienen, bestimmten Diensten (z.B. Telefongesprächen) Priorität gegenüber anderen (z.B. Dateiübertragungen) zu geben und damit die QoS (QoS) zu verbessern. Kann in Asterisk bei einigen Channels (Channel) eingestellt werden. Die ToS-Flags waren früher in RFC 791 spezifiziert und wurden in RFC 2474 von der IETF als DSCP (DSCP) rückwärtskompatibel überarbeitet. Die alten Flags waren low-delay, throughput und reliability. Die neuen finden Sie unter DSCP.
UDP	User Datagram Protocol. Wie TCP eine Protokollschicht über IP (IP), bei der allerdings die vollständige Übertragung und richtige Reihenfolge aller Pakete nicht garantiert ist. Eignet sich aber durch den geringeren Overhead besonders gut für Echtzeitanwendungen wie die Übertragung von Ton- oder Videodaten, da z.B. bei einem Video-Stream ein Bild sowieso nicht einfach 2 Sekunden später eingefügt werden könnte.
Voicemail	Ein Ausdruck für Sprach-Nachrichten. Also Nachrichten auf einem (hier meist virtuellen) Anrufbeantworter (Voice Mailbox).

Voice Mailbox	Moderner Ausdruck für einen (meist virtuellen) Anrufbeantworter. In Asterisk werden Voice-Mailboxen in der Datei `voicemail.conf` definiert.
Voice Message	Siehe Voicemail.
Voice Prompt	Ein Sprachbaustein - z.B. eine Eingabeaufforderung - der in einem IVR-Menü (IVR) gespielt wird.
VoIP	Voice over IP. Bezeichnet die Echtzeitübertragung von Gesprächen zerlegt in IP-Pakete (IP) anstatt über analoge Leitungen. Oft eine kostengünstige Möglichkeit zu telefonieren. Die meisten Anbieter von Breitbandanschlüssen für Privathaushalte meinen damit ihren eigenen kostenpflichtigen Dienst oder die Möglichkeit, an einer mitgelieferten DSL-Modem-Router-Einheit direkt analoge Telefone anschließen zu können. Bekannte von Hausanschlüssen unabhängige Provider sind Skype oder Sipgate, die auch ein Gateway zum und vom öffentlichen Telefonnetz anbieten.
Wählplan	Siehe Dial Plan.

Appendix A. GNU Free Documentation License

Version 1.2, November 2002

Free Software Foundation, Inc.
51 Franklin St, Fifth Floor,
Boston,
MA
02110-1301
USA

Version 1.2, November 2002

1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

GNU FDL Modification Conditions

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See *http://www.gnu.org/copyleft/*.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Sample Invariant Sections list

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any

> later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> ### Sample Invariant Sections list
>
> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Anhang B. Spezielle SIP-Provider

1. Einleitung

Die meisten SIP-Provider haben auf Ihren Webseiten eine gute Beschreibung, wie man Asterisk anschliesst. Diese kann man oft einfach per cut-and-paste übernehmen. Für alle anderen Fälle gibt es diesen Anhang.

2. SIP-Provider T-Online

Der Internet-Provider T-Online bietet ebenfalls seit einiger Zeit VoIP-Dienste an. Auch wenn der Provider seine eigene VoIP-Software propagiert ist es durchaus möglich, auch Asterisk an den T-Online SIP-Dienst anzukoppeln. Nachfolgend sind die Schritte beschrieben, die eine Anbindung von Asterisk an T-Online als VoIP-Provider ermöglicht.

Warnung

T-Online hat in der Vergangenheit öfters die serverseitige SIP-Software gewechselt, wodurch es mehrfach zu Problemen und notwendigen Anpassungen an der Client-Konfiguration kam. Die nachfolgend beschriebene Konfiguration war zum Zeitpunkt der Drucklegung funktionsfähig. Bei Problemen konsultieren Sie bitte die online Version unter *http://www.das-asterisk-buch.de*

2.1. Asterisk bei T-Online registrieren

Es wird davon ausgegangen, dass Sie Kunde bei T-Online sind, die VoIP-Dienstleistung in Ihrem Account haben freischalten lassen und von T-Online bereits eine VoIP-Telefonnummer bekommen haben. Ihre aktuellen Einstellungen können Sie über das T-Online-Kundencenter konfigurieren (http://www.t-online.de/kundencenter).

Zunächst wird in der `[general]`-Sektion der Datei `sip.conf` der folgenden Registrierungs-Eintrag eingefügt:

```
[general]
;
; Register to SIP Provider T-Online
; register => user[:secret[:authuser]]@host[:port][/extension]
;
maxexpirey=240                  ; Wichtig fuer T-Online
defaultexpirey=240              ; Wichtig fuer T-Online, mindestens 240
register => 03222xxxxxxx:secret:hans.mustermann@tel.t-online.de/siptonline
```

Die beiden Parameter `maxexpirey` und `defaultexpirey` müssen für das T-Online-SIP-Gateway wie oben gesetzt werden, ansonsten nimmt T-Online die Registrierung nicht an. Werden diese Einstellungen nicht oder falsch gesetzt, meldet T-Online bei der Registrierung verschiedenste Fehlermeldungen, je nach aktuell eingesetzter Server-Software. Die Parameter des `register`-Befehls haben folgende Bedeutung:

03222xxxxxxx	Die VoIP-Telefonnumer, die Ihnen von T-Online zugewiesen wurde
secret	Ihr T-Online-WEB-Passwort
hans.mustermann	Ihr T-Online-Benutzername (ohne @t-online.de)
siptonline	Extension, die im Wählplan für eingehende Anrufe angesprungen werden soll

Wird die `sip.conf` in Asterisk neu einladen und dann den Befehl **sip show registry aufrufen**, sollten Sie folgende Zeile sehen:

```
*CLI> sip show registry
Host                            Username       Refresh State                Reg.Time
tel.t-online.de:5060            03222xxxxxxx       225 Registered           Wed, 27 Dec 2006 19:01:07
```

Werden auf der Konsole keine Registrierungsfehler ausgegeben, hat bereits alles geklappt und das Asterisk-System ist bei T-Online erfolgreich registriert. Nun muss in der Datei `extensions.conf` der Wählplan entsprechend angepasst werden.

Wird der Registrierungs-Befehl wie oben angegeben, landen eingehende Anrufe im [default]-Kontext des Wählplans. Alles andere ist wie immer, man erstellt eine entsprechende Extension und wählt darüber ein passendes Endgerät an.

```
[default]
exten => siptonline,1,Dial(SIP/2000)
exten => siptonline,2,VoiceMail(u2000)
exten => siptonline,102,VoiceMail(s2000)
exten => siptonline,103,Hangup()
```

Nun sollte ein Anruf an Ihrer T-Online-VoIP-Nummer bereits ohne Probleme durchgestellt werden können.

2.2. Über T-Online telefonieren

Um über den Provider auch raustelefonieren zu können, muss zunächst ein weiterer Eintrag in der Datei `sip.conf` erfolgen. Dort werden die Account-Informationen des Providers eingetragen, die ähnlich sind, wie oben bereits beschrieben

```
[tonlineout]
type=peer
username=hans.mustermann@t-online.de
secret=secret
host=tel.t-online.de
fromdomain=tel.t-online.de
insecure=very
```

Wichtig

Wird der Eintrag `insecure=very` nicht gesetzt, würden eingehende(!) Anrufe durch diesen Eintrag abgewiesen. Zudem muss der Username diesmal zusammen mit der T-Online Domäne gesetzt werden, da ansonsten die unten aufgeführte Rufnummernunterdrückung nicht funktioniert (Danke an Simon Baatz (http://simon.baatz.info/) für diesen entscheidenden Tipp).

In dem Kontext der abgehenden Verbindung muss nun in der Datei `extensions.conf` noch ein entsprechender Eintrag getätigt werden, in diesem Fall werden z.B. alle mit 0 beginnenden Rufnummern über das SIP-Gateway geleitet:

```
exten => _0.,1,Set(CALLERID(num)=03222xxxxxxx)
exten => _0.,n,Set(CALLERID(name)=03222xxxxxxx)
exten => _0.,n,Dial(SIP/${EXTEN:1}@tonlineout)
```

Wichtig

Werden nicht sowohl die CallerID-Nummer wie auch der CallerID-Name auf die vom Provider zugewiesene Rufnummer gesetzt, so wird der Anruf vom T-Online SIP-Gateway abgewiesen.

Möchte man eine Rufnummernunterdrückung erzwingen, so müssen CallerID-Nummer und CallerID-Name beide auf den Wert `anonymous` gesetzt werden:

```
exten => _9.,1,Set(CALLERID(num)=anonymous)
exten => _9.,n,Set(CALLERID(name)=anonymous)
exten => _9.,n,Dial(SIP/${EXTEN:1}@tonlineout)
```

Anhang C. IAX vs. SIP

1. Original E-Mail von Mark Spencer

```
Date: Mon, 5 Jul 2004 18:59:52 -0500 (CDT)
From: Mark Spencer <markster@digium.com>

Let me summarize some differences between SIP and IAX, and it might help you
make a decision about what is best for you.

1) IAX is more efficient on the wire than RTP for any number of calls,
any codec. The benefit is anywhere from 2.4k for a single call to
approximately tripling the number of calls per megabit for G.729 when
measured to the MAC level when running trunk mode.

2) IAX is information-element encoded rather than ASCII encoded. This
makes implementations substantially simpler and more robust to buffer
overrun attacks since absolutely no text parsing or interpretation is
required. The IAXy runs its entire IP stack, IAX stack, TDM interface,
echo canceler, and callerid generation in 4k of heap and stack and 64k of
flash. Clearly this demonstrates the implementation efficiency of its
design. The size of IAX signaling packets is phenomenally smaller than
those of SIP, but that is generally not a concern except with large
numbers of clients frequently registering. Generally speaking, IAX2 is
more efficient in its encoding, decoding and verifying information, and it
would be extremely difficult for an author of an IAX implementation to
somehow be incompatible with another implementation since so little is
left to interpretation.

3) IAX has a very clear layer2 and layer3 separation, meaning that both
signaling and audio have defined states, are robustly transmitted in a
consistent fashion, and that when one end of the call abruptly disappears,
the call WILL terminate in a timely fashion, even if no more signaling
and/or audio is received. SIP does not have such a mechanism, and its
reliability from a signaling perspective is obviously very poor and
clumsy requiring additional standards beyond the core RF3261.

4) IAX's unified signaling and audio paths permit it to transparently
navigate NAT's and provide a firewall administrator only a *single* port to
have to open to permit its use. It requires an IAX client to know
absolutely nothing about the network that it is on to operate. More
clearly stated, there is *never* a situation that can be created with a
firewall in which IAX can complete a call and not be able to pass audio
(except of course if there was insufficient bandwidth).

5) IAX's authenticated transfer system allows you to transfer audio and
call control off a server-in-the-middle in a robust fashion such that if
the two endpoints cannot see one another for any reason, the call
continues through the central server.

6) IAX clearly separates Caller*ID from the authentication mechanism of
the user. SIP does not have a clear method to do this unless
Remote-Party-ID is used.

7) SIP is an IETF standard. While there is some fledgling documentation
courtesy Frank Miller, IAX is not a published standard at this time.

8) IAX allows an endpoint to check the validity of a phone number to know
whether the number is complete, may be complete, or is complete but could
be longer. There is no way to completely support this in SIP.

9) IAX always sends DTMF out of band so there is never any confusion about
what method is used.

10) IAX support transmission of language and context, which are useful in
an Asterisk environment. That's pretty much all that comes to mind at the
```

```
moment.


Mark


RS:
I Guess there must be some advantages to SIP (or we should call the writers of
it stupid).

So here a few questions to elaborate how IAX handles:

1) Bandwidth indications

2) New codecs

3) extensibility

4) Call Hold and other complex scenarios

5) Video telephone

I have got the impression this has all been better aranged in SIP
```

Anhang D. Standard-Datenströme

Anmerkung

Dieser Artikel basiert auf dem Artikel Stdout aus der freien Enzyklopädie Wikipedia und steht unter der GNU-Lizenz für freie Dokumentation. In der Wikipedia ist eine Liste der Autoren verfügbar.

Die Standard-Datenströme (englisch: standard streams) sind drei Datenströme für die Ein- und Ausgabe im Betriebssystem Unix oder in verwandten Betriebssystemen. Sie werden auch von der Standard C Library unterstützt. Viele Programme verwenden automatisch die Standardein- bzw. Standardausgabe, wenn auf der Kommandozeile keine Dateien für die Ein- oder Ausgabe angegeben werden.

1. Standardeingabe (stdin)

Über die Standardeingabe können Daten in ein Programm eingelesen werden. Normalerweise ist sie mit der Tastatur verbunden, d.h. Programme empfangen die Benutzereingaben über die Standardeingabe. Unter Unix entspricht der Standardeingabe die Gerätedatei `/dev/stdin`, der Datei-Deskriptor hat die Nummer 0.

1.1. Beispiel

Hier wird die Standardeingabe für das Programm less aus der Datei eingaben.txt gelesen:

```
$ less < eingaben.txt
```

2. Standardausgabe (stdout)

Über die Standardausgabe kann ein Programm Daten ausgeben. Normalerweise ist sie mit dem Monitor verbunden, d.h. Programme senden Ausgabetexte über die Standardausgabe an den Benutzer. Unter Unix entspricht der Standardausgabe die Gerätedatei `/dev/stdout`, der Datei-Deskriptor hat die Nummer 1.

2.1. Beispiel

Hier wird die Standardausgabe des Programms find in die Datei `ausgaben.txt` umgeleitet:

```
$ find . -name '*.html' > ausgaben.txt
```

3. Standardfehlerausgabe (stderr)

Die Standardfehlerausgabe ist ein zweiter Ausgabedatenstrom, der dazu gedacht ist, Fehler- und Statusmeldungen auszugeben. Normalerweise ist er ebenfalls mit dem Monitor verbunden; allerdings kann er getrennt von der Standardeingabe umgeleitet werden, so dass Fehlermeldungen nicht mit den ausgegebenen Nutzdaten vermischt werden. Unter Unix entspricht der Standardfehlerausgabe die Gerätedatei `/dev/stderr`, der Datei-Deskriptor hat die Nummer 2.

3.1. Beispiel

Hier werden die Fehlermeldungen des Programms find in die Datei `fehlermeldungen.txt` umgeleitet, während die Standardausgabe an das Programm less weitergeleitet wird:

```
$ find . -name '*.html' 2> fehlermeldungen.txt | less
```

Stichwortverzeichnis

N

O

P

Q

R

S

T

U

V

W

Z