

CapiSuite 0.4.5

Gernot Hillier

`gernot@hillier.de`

CapiSuite 0.4.5
von Gernot Hillier

2. Users Guide.....	27
2.1. Einführung in Python.....	27
2.1.1. Python-Grundlagen.....	27
2.1.2. Blöcke, Funktionen und Exceptions.....	29
2.1.3. Mit Modulen arbeiten.....	30
2.2. Ein erster Blick auf das Incoming- und Idle-Skript.....	31
2.2.1. Das Incoming-Skript.....	31
2.2.2. Das Idle-Skript.....	32
2.3. Verwendete Dateiformate.....	33
2.3.1. Format der Sprach-Dateien (invertiert A-Law, 8kHz, mono).....	34
2.3.1.1. Erzeugen von A-Law-Dateien.....	34
2.3.1.2. Abspielen von A-Law-Dateien.....	35
2.3.2. Format von Fax-Dateien (Structured Fax Files).....	35
2.3.2.1. Erzeugen eines SFF.....	36
2.3.2.2. Ansehen / konvertieren von SFF.....	36
2.3.2.3. Farbfaxe - das CFF-Format.....	36
2.4. Tutorial: Ein Incoming-Skript schreiben.....	37
2.4.1. Grundlagen und ein wirklich dummer Anrufbeantworter.....	37
2.4.2. Verbesserungen in eine brauchbaren (?) Zustand.....	40
2.4.3. Verwendung vernünftiger Dateinamen.....	41
2.4.4. Automatische Fax-Erkennung und -Empfang.....	42
2.5. Beispiel eines Idle-Skripts.....	45
2.6. Struktureller überblick über die Standard-Skripte.....	49
2.6.1. incoming.py.....	49
2.6.1.1. Funktion <code>callIncoming</code>	50
2.6.1.2. Funktion <code>faxIncoming</code>	50
2.6.1.3. Funktion <code>voiceIncoming</code>	50
2.6.1.4. Funktion <code>remoteInquiry</code>	51
2.6.1.5. Funktion <code>newAnnouncement</code>	52
2.6.2. idle.py.....	52
2.6.2.1. Funktion <code>idle</code>	52
2.6.2.2. Funktion <code>sendfax</code>	53
2.6.2.3. Funktion <code>movejob</code>	53
2.6.3. capisuitefax.....	53
2.6.4. cs_helpers.py.....	54
2.7. CapiSuite-Befehlsreferenz.....	55
A. Danksagungen.....	57
B. CAPI 2.0 Fehler-Codes.....	59
B.1. CAPI-Fehler, die Verbindungsprobleme beschreiben.....	59
B.1.1. Protokoll-Fehler.....	59
B.1.2. ISDN-Fehler-Codes.....	60
B.2. Interne CAPI-Fehler.....	62

B.2.1. Informative Werte (kein Fehler).....	62
B.2.2. Fehler bezüglich CAPI_REGISTER.....	62
B.2.3. Nachrichtenaustausch-Fehler	63
B.2.4. Resource/Coding-Fehler.....	63
B.2.5. Fehler bezüglich angeforderter Services	64

Beispiele

2-1. example.py	37
2-2. example.py, verbessert	40
2-3. Verwendung eindeutiger Dateinamen	41
2-4. Hinzufügen der Fax-Funktionen	42
2-5. idle_example.py	45
2-6. idle_example.py, Version für CapiSuite	47

Einleitung

1. Willkommen bei CapiSuite

Willkommen bei CapiSuite, einer durch Python-Skripte erweiterbaren ISDN-Telekommunikations-Suite. Sie verwendet die neue CAPI-Schnittstelle für den Zugriff auf Ihre ISDN-Hardware - Sie brauchen also eine Karte, für die ein CAPI-kompatibler Treiber verfügbar ist. Momentan sind dies alle Karten, die von AVM hergestellt werden, und einige Eicon-Karten.

Dieses Handbuch soll Ihnen helfen, CapiSuite so schnell wie möglich zu benutzen. Weil ich das Lesen von Dokumentationen genau so hasse wie Sie, lassen Sie uns am besten gleich anfangen.

2. Was zum Teufel ist "CapiSuite"?!

CapiSuite versucht, es dem Anwender zu ermöglichen, seine eigenen ISDN-Applikationen zu programmieren, ohne sich mit all den blöden Programmierdetails wie Callback-Funktionen, Datenpuffer, Protokolleinstellungen usw. herumschlagen zu müssen.

Ich nahm eine Skript-Sprache, die (meiner Meinung nach) sehr einfach zu verstehen, zu benutzen und zu lernen ist - besonders für Anfänger: Python. Ich erweiterte sie um einige Funktionen, die die ISDN-Basisfunktionalität für Benutzeranwendungen zur Verfügung stellen. Hinter diesen Funktionen implementiert CapiSuite all die Details, die den Anwender gar nicht interessieren. Mein Ziel war es, die Skript-Programmierung so einfach wie möglich zu machen und gleichzeitig Ihnen die Flexibilität zu geben, realisieren zu können, was Sie wollen.

Um Ihnen einen kleinen Eindruck zu geben, wie einfach die Programmierung eines Anrufbeantworters ist:

```
def callIncoming (call, service, call_from, call_to):
    connect_voice (call, 10)           # Ruf nach 10 Sek annehmen
    audio_send (call, "announcemnt.la") # Ansage abspielen
    audio_send (call, "beep.la")       # Piepton abspielen
    audio_receive (call, "call.la", 10) # Anruf aufzeichnen
```

Einleitung

Natürlich fehlen hier einige Details wie z.B. das Erzeugen eines eindeutigen Dateinamens oder das Speichern von zusätzlichen Informationen (angerufene und anrufende Teilnehmernummer, Uhrzeit, ...) - aber ich nehme an, Sie verstehen das Prinzip.

Und - keine Angst - wenn Sie nur einen normalen Anrufbeantworter haben möchten oder ein paar Faxe verschicken oder empfangen wollen, können Sie einfach die Standard-Skripte verwenden, die mit CapiSuite ausgeliefert werden. Diese haben schon einige nette Funktionen - z.B. ist der Anrufbeantworter multiuserfähig, unterstützt automatische Faxerkennung und Fernabfragefunktionen. Sie müssen CapiSuite nur noch einige Details mitteilen wie Ihre eigene Nummer, eine eigene Ansage aufnehmen und das war's.

CapiSuite ist also schon für Ihre täglichen Telekommunikationsbedürfnisse ausgestattet - aber wenn Sie die Dinge nicht so mögen wie ich - ändern Sie sie oder implementieren Sie sie komplett selbst. Und wenn Sie schöne Skripte schreiben oder Änderungen an meinen Standard-Skripten vornehmen, wäre es nett, wenn ich sie bekommen und allen Anwendern zur Verfügung stellen könnte.

3. Aufbau des Handbuchs

Dieses Handbuch ist aufgeteilt in drei große Teile.

Der erste Teil (Kapitel 1) erklärt, wie CapiSuite installiert wird, was Sie nach der Installation mit den Standard-Skripten machen können und wie Sie sie konfigurieren können. Hier wird Ihnen keine Zeile Code präsentiert. Wenn Sie nur die Standard-Skripte nutzen wollen, sollte dies alles sein, was Sie lesen müssen.

Der zweite Teil (Kapitel 2) erklärt Ihnen, wie Sie Ihre eigenen Skripte schreiben können. Er gibt Ihnen eine sehr, sehr kurze Einführung in Python und eine komplette Referenz der Befehle, die CapiSuite hinzufügt. Und schließlich enthält er einen Überblick über die Standard-Skripte, wo deren Funktionsweise beschrieben ist, sodass Sie diese einfach als Ausgangspunkt für Ihre eigenen Applikationen verwenden können.

Der letzte Teil wendet sich an Programmierer, die bei der Entwicklung des CapiSuite-Kerns helfen möchten. Er gibt einen Überblick über das System und eine detaillierte Beschreibung für jede einzelne Klasse, jede Methode und jedes Attribut. Da er automatisch aus den Sourcen von CapiSuite erstellt wird, ist er nicht Teil dieses Dokuments. Sie finden ihn entweder lokal unter `../reference/index.html` oder online unter <http://www.capisuite.de/reference/index.html>.

Es gibt dort auch noch einige zusätzliche Teile, in denen "was ich sonst noch sagen wollte" enthalten ist:

Da CapiSuite als Diplomarbeit begonnen wurde, möchte ich in Anhang A allen danken, die mir bisher geholfen haben.

Wenn Sie Ihre eigenen Skripte schreiben wollen oder wenn die bei der Entwicklung des CapiSuite-Kerns mithelfen möchten, werden Sie bald über einige spezielle ISDN- und CAPI-Fehler-Codes stolpern, die in Anhang B erklärt sind.

Wenn Sie weitere Informationen oder Support benötigen, schauen Sie bitte auf die CapiSuite-Homepage unter <http://www.capisuite.de>. Sie finden dort Links to einem Bug-Tracking-System, aktueller Dokumentation, Downloads und anderen Ressourcen. Wenn Sie Fragen haben, Support benötigen oder uns Ihre Ideen oder Meinung bezüglich CapiSuite mitteilen möchten, sind Sie auf den CapiSuite-Mailinglisten willkommen. Bitte schreiben Sie mir keine persönlichen Mail mit solchen Fragen, da dies anderen Benutzern nicht hilft und ich kann nicht die gleiche Frage zehnmal am Tag beantworten, sorry. Informationen, wie Sie sich auf den Listen anmelden können und wo das Archiv zu finden ist, befinden sich ebenfalls auf der Homepage.

Ich hoffe, ich konnte Ihren Appetit wecken - lassen Sie uns nun also wirklich loslegen, damit Sie bereit werden, CapiSuite zu benutzen.

Einleitung

Kapitel 1. Getting Started

1.1. Voraussetzungen und Installation von CapiSuite

1.1.1. Voraussetzungen

1.1.1.1. Hardware und Treiber

Da CapiSuite das CAPI (Common ISDN Application Programming Interface) für den Zugriff auf Ihre ISDN-Hardware benutzt, brauchen Sie eine Karte, für die ein CAPI-kompatibler Treiber verfügbar ist.

Momentan sind dies alle Karten, die von AVM hergestellt werden, und einige Eicon-Karten. Wenn Sie eine der passiven Karten von AVM nutzen, müssen Sie sich deren CAPI-Treiber herunterladen und installieren.

Es gibt auch einige Distributionen (z.B. aktuelle Versionen von SUSE), bei denen die Capi4Linux-Treiber von AVM bereits dabei sind - Sie müssen sie dann nur noch aktivieren (unter SUSE mit YaST2). Wenn Sie eine aktive Karte von AVM besitzen (z.B. die B1, C2 oder C4), dann haben Sie bereits alles benötigte installiert.

Nein, es gibt keine Möglichkeit, CapiSuite mit der alten ISDN4Linux-Schnittstelle zum Laufen zu bringen. Vermutlich wird es die auch nie geben, weil das ISDN4Linux-Projekt eine CAPI-kompatible Schnittstelle mittlerweile in dem neuen mISDN zur Verfügung stellt. Selbstverständlich funktioniert CapiSuite auch damit.

CapiSuite wurde hauptsächlich mit ISDN-Karten von AVM getestet, insbesondere mit der Fritz!PCI, der Fritz!USB und der B1 auf der i386-Plattform, aber es sollte mit anderen CAPI-kompatiblen Treibern für andere Karten oder auf anderen Plattformen keine Probleme geben. Trotzdem müssen nicht von allen CAPI-kompatiblen Karten alle Features unterstützt werden, sodass Sie evtl. nicht mit allen Karten faxen oder vom Sprach- in den Fax-Modus umschalten können.

1.1.1.2. Software

CapiSuite benötigt einige Pakete, die installiert sein müssen, bevor CapiSuite benutzt werden kann.

Ich liste sie hier zusammen mit einer kurzen Information, warum das Paket benötigt wird und wo Sie weitere Informationen zur Installation finden können, auf. Es ist

Kapitel 1. Getting Started

immer eine gute Idee, zuerst einmal mit dem Installationstool Ihrer Lieblingsdistribution zu prüfen, ob sie bereits dabei sind, bevor Sie alle aus dem Internet herunterladen und installieren. Keine Angst, weil es so viele sind - die meisten sind bei fast jeder Distribution dabei und vermutlich bereits auf Ihrem System installiert.

Python >= 2.2

CapiSuite verwendet einen eingebauten Python-Interpreter, um die Skripte zu interpretieren - deshalb brauchen Sie eine installierte und lauffähige Version von Python. Diese sollte in nahezu jeder aktuellen Linux-Distribution enthalten sein. Weitere Infos zu Python, ein schönes Tutorial und vieles mehr finden Sie unter <http://www.python.org>

sox >= 12.17.3

Dies ist das Schweizer Offiziersmesser für die Konvertierung von Audio-Formaten. Es wird nicht vom CapiSuite-Kern benötigt, ist aber sehr nützlich, wenn Sie Sprachdateien für Anrufe auf Ihrer Maschine anhören oder aufnehmen möchten. Es wird auch benötigt, wenn Sie die Standard-Skripte von CapiSuite benutzen möchten. Ich möchte wetten, dass es in Ihrer Distribution dabei ist und sehr wahrscheinlich ist es auf Ihrem System bereits installiert. Um sicher zu gehen, probieren Sie einfach mal aus, **sox** zu starten. Wie mir Helmut Gruber mitteilte, benötigen Sie mindestens Version 12.17.3, da erst diese die hier genutzten Inversed-A-Law-Dateien unterstützt. Sie finden weitere Details unter <http://sox.sourceforge.net>

sfftobmp

CapiSuite speichert Fax-Dateien im CAPI-spezifischen Format Structured Fax File (SFF). sfftobmp ist ein kleiner, aber sehr nützlicher Konverter, um diese Dateien in gängigere Formate wie JPEG, TIFF oder BMP zu konvertieren. Sie bekommen es unter <http://sfftools.sourceforge.net/sfftobmp.html>. Es wird ebenfalls nicht vom CapiSuite-Kern benötigt, aber von den Standard-Skripten.

sffview

Dieses Tool ist ein einfacher, aber nützlicher SFF-Betrachter. Es wird von keiner CapiSuite-Komponente benötigt, ist aber sehr nützlich, wenn Sie mal eben ein Fax ansehen möchten, ohne es vorher erst konvertieren zu müssen. Sie finden es unter <http://sfftools.sourceforge.net/sffview.html>.

tiff2ps

Ein kleines Utility, um TIFF-Dateien ins Postscript-Format zu konvertieren. Es wird von den Standard-Skripten benötigt, um Faxe in PDF-Dateien

umzuwandeln (SFF->TIFF->PS->PDF :-}). Es ist oft in einem Paket namens `tiff` oder `tiff-tools` enthalten. Details unter <http://www.libtiff.org>

ps2pdf

Noch ein kleines Utility für die Kette SFF->PDF - diesmal für die Konvertierung von Adobe PostScript in Adobe PDF. Es ist bei Ghostscript dabei, sodass Sie es höchst wahrscheinlich bereits haben.
(<http://www.gnu.org/software/ghostscript/ghostscript.html>)

aktueller Ghostscript mit cfax-Patch

Aktuelle Ghostscript-Versionen enthalten ein Device, um die oben erwähnten SFF-Dateien zu erstellen. Wenn Sie eine ältere Version haben, brauchen Sie den Patch von <http://sfftools.sourceforge.net/ghostscript.html>. Um zu prüfen, ob Ihre GhostScript-Version diesen Patch bereits hat, rufen Sie bitte **gs --help** auf und sehen nach, ob Sie das Device `cfax` in der langen Liste der unterstützten Devices finden können.

jpeg2ps

Das **jpeg2ps**-Kommando wird zur Konvertierung von Farbfaxen in das PostScript-Format für die Mailzustellung benötigt. Sie brauchen es nicht, wenn Sie keine Farbfaxe empfangen wollen. Unglücklicherweise gibt es wegen eines Treiberfehlers momentan keine Möglichkeit, den Empfang von Faxdokumenten bei AVM-ISDN-Karten komplett zu deaktivieren. Wenn Ihnen also jemand ein Farbfax schicken sollte (was nach meiner Erfahrung nur sehr selten vorkommt), dann bekommen Sie eine Fehlermeldung per Mail, wenn Sie dieses Programm nicht installiert haben. Wenn Ihre Linux-Distribution dieses Paket nicht enthält, können Sie es von <http://www.pdflib.com/jpeg2ps/> herunterladen.

Da das Farbfaxprotokoll für die Übertragung mehrseitiger Dokumente aneinandergehängte JPEG-Dateien benutzt, sollten Sie außerdem meinen multiJPEG-Patch von <http://www.hillier.de/linux/jpeg2ps-multi.php3> herunterladen und anwenden.

1.1.2. Installation

Zuerst sollten Sie prüfen, ob Ihr CAPI-Treiber korrekt eingerichtet wurde. Rufen Sie einfach **capiinfo** als root in einer Shell auf.

Wenn Sie viele Zeilen erhalten, funktioniert Ihr CAPI-Treiber. Wenn Sie nur eine Fehlermeldung erhalten, müssen Sie einen CAPI-kompatiblen Treiber installieren.

Bitte schauen Sie in der Dokumentation Ihres ISDN-Kartenherstellers, Ihrer Linux-Distribution und/oder auf einer ISDN-Mailingliste deswegen nach. Wenn Sie wirklich niemanden finden können, der Ihnen dabei hilft, können Sie *als letzten Ausweg* mich auf der CapiSuite-Mailingliste um Rat fragen.

Der Rest der Installation hängt davon ab, ob Sie Binär- oder Source-Pakete verwenden, um CapiSuite zu installieren. Wenn die die CapiSuite-Sourceen nicht ändern möchten, empfehle ich, die Binärpakete zu verwenden, falls diese für Ihre Distribution und Plattform verfügbar sind.

Sie können sowohl die Binär- als auch die Source-Pakete im Download-Bereich unter <http://www.capisuite.de> herunterladen. Wenn Sie selbst Pakete für andere Distributionen erstellt haben, schicken Sie mir diese bitte und ich werde sie dorthin kopieren...

1.1.2.1. Installation der Binär-Pakete

Wenn Sie Binär-Pakete für Ihre Distribution und Plattform bekommen können, empfehle ich, diese zu verwenden. Diverse Pakete für diverse Distributionen sind bereits verfügbar, die von verschiedenen Personen gepflegt werden. CapiSuite ist mittlerweile auch Bestandteil von aktuellen Versionen von SUSE, Debian und Gentoo.

Wenn es Ihnen gelungen ist, CapiSuite auf einem nicht erwähnten System zu installieren, lassen Sie es mich bitte wissen und ich werde die Beschreibung hier aufnehmen. Wenn Sie Binär-Pakete für andere Distributionen erstellt haben, wäre ich ebenfalls erfreut, wenn ich auf Ihren Download-Bereich verweisen könnte oder Sie auf meiner Seite anbieten könnte.

Jetzt sollte alles soweit sein, dass wir loslegen können. Lesen Sie also weiter im Abschnitt 1.2.

1.1.2.1.1. Installation der RPM-Pakete (SUSE, Mandrake, Fedora & Co.)

Um die CapiSuite-RPM-Pakete zu installieren, können Sie entweder Ihr Lieblings-Setup-Tool verwenden - entweder von Ihrem Distributor oder aus der Community - oder Sie können es manuell (als root) machen:

```
rpm -Uvh capisuite-version.rpm
```

1.1.2.1.2. Installation anderer Pakete

Informationen über die Installation anderer Paketformate sollten Sie in der Dokumentation Ihrer Distribution, in den Paketen selbst oder auf den Homepages der jeweiligen Maintainer finden. Wenn Sie eine Installationsbeschreibung für

bestimmte Pakete für diese Anleitung verfassen wollen, wenden Sie sich bitte an mich.

1.1.2.2. Installation der Source-Pakete

Wenn Sie keine Binär-Pakete verwenden können oder Sie möchten alles selbst machen, können Sie sich die Sourcen aus dem Download-Bereich herunterladen.

Laden Sie das neueste Tar-Archiv (capisuite-X.Y.tar.gz) von der CapiSuite-Homepage herunter und kopieren Sie es irgendwo hin. Wechseln Sie dorthin und geben Sie die folgenden Befehle ein:

```
./configure
make
su # jetzt als root
make install
```

Dies installiert CapiSuite komplett im `/usr/local`-Baum. Wenn Sie in ein anderes Verzeichnis installieren möchten, schauen Sie sich bitte die Kommandozeilen-Hilfe an, die von

```
./configure --help
```

ausgegeben wird. Hier finden Sie Optionen zum ändern des Installationsverzeichnisses.

1.1.2.3. Installation aus dem Subversion-Repository

Wenn Sie ganz innovativ sein und immer die neuesten Features testen wollen, können Sie sich die aktuellen Sourcen von CapiSuite aus dem Repository holen.

Dies ist nicht empfohlen, wenn Sie nicht die neuesten Features testen oder bei der Entwicklung von CapiSuite helfen wollen! Die Entwicklungsversionen können OK sein, können aber auch nicht funktionieren oder noch nicht mal kompilieren. Es ist Ihr eigenes Risiko!

Sie müssen die üblichen Entwicklungs-Tools installiert und lauffähig haben, wie z.B. GNU make, gcc/g++ und alle Komponenten, die oben beschrieben wurden (insbesondere die Entwickler-Pakete von Python).

Wenn Sie die Dokumentation aus den Sourcen erstellen wollen, brauchen Sie zusätzlich Doxygen und funktionsfähige Docbook/XML-Tools.

Eine Anleitung, wo Sie das Repository finden und wie Sie die Sourcen auschecken können, finden Sie im Download-Bereich auf der CapiSuite-Homepage unter <http://www.capisuite.de>.

Wenn Sie die Sourcen in ein Verzeichnis ausgecheckt haben, machen Sie bitte ein

```
make -f Makefile.cvs
```

Jetzt können Sie ganz normal mit der Installation weiter machen wie im Abschnitt 1.1.2.2 beschrieben.

1.1.3. Update von früheren Versionen

Dieser Abschnitt enthält einen Überblick über das Vorgehen, wenn Sie bereits eine frühere Version von CapiSuite installiert haben.

Generell sollte der übliche Updatemechanismus, passend zu Ihrer Installationsart verwendet werden - zum Beispiel also der Update-Modus Ihres Paketmanagementtools wenn Sie ein Binärpaket installiert haben. Sollten Sie CapiSuite von den Sourcen gebaut haben, sollten Sie ihre alte Konfiguration vor dem Aufruf von **make install** sichern. Im Prinzip entspricht das Vorgehen hier dem bei jedem anderen Software-Paket, das Sie benutzen - daher möchten wir hier nicht weiter darauf eingehen.

Was hier hauptsächlich behandelt werden soll, sind die Änderungen der Konfigurationsdateien und der Anforderungen an andere installierte Tools zwischen verschiedenen CapiSuite-Versionen, so dass Sie Ihre Installation schnell auf den aktuellen Stand bringen können. Eine umfassendere Liste der neuen Features und wichtigen Änderungen finden Sie in der Datei `NEWS`, die in den CapiSuite-Paketen enthalten ist. Zusätzlich enthält das `ChangeLog` alle einzelnen Änderungen in den Source-Dateien im Detail, was aber wohl nur für Entwickler interessant sein dürfte.

1.1.3.1. 0.4.4 auf 0.4.5

Die *Standardskripte* verwenden nun eine SMTP-Verbindung zu localhost, anstatt das **sendmail**-Kommando manuell aufzurufen, wie es frühere Versionen taten. Dies wurde auf Grund von diversen Stabilitätsproblemen mit dem alten Mechanismus geändert. Das bedeutet, dass Sie nun einen laufenden SMTP-Dämon benötigen, der Verbindungen auf localhost annimmt. Da dies in der Standardeinrichtung der meisten Distributionen der Fall ist, sollte das kein Problem sein.

In `answering_machine.conf` und `fax.conf` stehen zwei neue Optionen zur Verfügung: `fax_email_from` und `voice_email_from` erlauben die Festlegung der von CapiSuite beim Senden von Mails an die User verwendeten Absenderadresse. Diese Einträge sind allerdings optional - sind sie nicht gesetzt, wird wie bisher der Username als Absender angegeben.

1.2. Wie funktioniert CapiSuite, wie wird es konfiguriert und gestartet

Lassen Sie uns zuerst anfangen mit einer kurzen Einführung, was CapiSuite wirklich ist und wie es funktioniert. Danach wird die Konfiguration und das Starten von CapiSuite kurz erklärt.

1.2.1. Wie funktioniert CapiSuite?

CapiSuite ist ein Daemon (Programm, das im Hintergrund läuft), dessen Hauptaufgabe es ist, darauf zu warten, dass ein Anruf ankommt. Wenn das passiert, startet er ein spezielles Python-Skript - das *Incoming-Skript* - und macht, was dieses Skript ihm sagt, z.B. einen Sprachanruf aufzeichnen, um einen Anrufbeantworter zu implementieren.

Für ausgehende Rufe gibt es ein weiteres Skript, das regelmäßig aufgerufen wird - das *Idle-Skript*. Es kann alle Ressourcen prüfen, um Anweisungen zum Durchführen eines Anrufs zu erhalten - es ist z.B. vorstellbar, einen speziellen Mail-Account zu prüfen oder spezielle Verzeichnisse zu beobachten, in die der Anwender bestimmte Aufträge ablegen kann.

Es werden also alle für den Benutzer sichtbaren Aktionen und das Verhalten von CapiSuite über diese beiden Skripte definiert.

Sie müssen nun zwei Dinge tun:

- die Skripte zur Verfügung stellen, indem Sie entweder
 - die Standard-Skripte, die mit CapiSuite geliefert werden, benutzen und anpassen oder
 - Ihre eigenen Skripte schreiben (vielleicht, indem Sie die Standard-Skripte als Vorlage verwenden)
- CapiSuite selbst konfigurieren und ihm sagen, wo die beiden Skripte zu finden sind

Diese Seite konzentriert sich auf die allgemeine Konfiguration von CapiSuite - diese besteht hauptsächlich aus Optionen, die festlegen, welche Skripte verwendet werden und wo und wie die Aktivitäten protokolliert werden. Danach werden einige Details zum Starten von CapiSuite beschrieben.

Die nächste Seite gibt dann eine Einführung zu den Standard-Skripten, die Sie bereits zusammen mit CapiSuite installiert haben und zeigt Ihnen, wie der Anrufbeantworter und die Fax-Funktionen genutzt werden können.

Die Details, wie Sie eigene Skripte schreiben können, werden in einem anderen Teil der Dokumentation (Kapitel 2) behandelt.

1.2.2. Konfiguration von CapiSuite

CapiSuite verwendet eine allgemeine Konfigurationsdatei für die Kernfunktionen. Diese Datei sollte unter `/etc/capisuite/capisuite.conf` oder `/usr/local/etc/capisuite/capisuite.conf` liegen (hängt davon ab, wie Sie CapiSuite installiert haben).

Die meisten Optionen sind für die Verwendung der Standard-Skripte bereits mit brauchbaren Voreinstellungen belegt - wenn Sie wollen, können Sie diesen Abschnitt also überspringen und im Abschnitt 1.2.3 weiterlesen.

Die Optionen werden hier kurz vorgestellt - für weitere Details schauen Sie sich bitte die Kommentare in der Konfigurationsdatei selbst an.

Optionen in `capisuite.conf`

```
incoming_script="/path/to/incoming.py"
```

Diese Option sagt CapiSuite, welches Skript bei eingehenden Anrufen ausgeführt werden soll. ändern Sie dies nur, wenn Sie Ihr eigenes Skript verwenden möchten.

```
idle_script="/path/to/idle.py"
```

Diese Option gibt den Pfad und den Namen des Idle-Skripts wider. Dieses Skript wird in regelmäßigen Intervallen aufgerufen, um zu prüfen, ob ein ausgehender Ruf getätigt werden soll. Wie oben sollten auch hier die Voreinstellungen OK sein, wenn Sie nicht Ihr eigenes Skript verwenden wollen.

```
idle_script_interval="30"
```

Hier können Sie festlegen, wie oft das Idle-Skript ausgeführt werden soll. Die angegebene Zahl ist das Intervall zwischen zwei Aufrufen in Sekunden. Kleinere Zahlen resultieren in einer schnelleren Reaktion auf abgesetzte Jobs, aber auch in einer höheren Systemlast. Die Voreinstellung sollte in den meisten Fällen OK sein.

```
log_file="/path/to/capisuite.log"
```

Diese Datei wird verwendet für alle "normalen" Meldungen, die von CapiSuite ausgegeben werden und die Ihnen mitteilen, was CapiSuite gerade macht. Fehlermeldungen werden in ein spezielles Log geschrieben (siehe unten).

```
log_level="1"
```

Sie können festlegen, wie detailliert die Protokollierung von CapiSuite sein soll. Die Voreinstellung gibt ein paar informative Meldungen für jeden eingehenden und ausgehenden Ruf und sollte im Normalfall ausreichen. Ich empfehle, den Werte nur zu erhöhen, wenn Probleme auftreten. Logs mit höherem Level sind hauptsächlich für Entwickler gedacht. Sie sollten Sie also nur verwenden, wenn Sie ein Problem melden wollen oder sich ein bisschen mit der CAPI-Schnittstelle und den Internas von CapiSuite auskennen.

```
log_error="/path/to/capisuite.error"
```

Alle Fehler, die CapiSuite intern und in Ihren Skripten feststellt, landen hier. Sie werden in eine extra Datei geschrieben, sodass sie nicht im normalen Log untergehen. Bitte schauen Sie sich dieses Log regelmäßig an - insbesondere, wenn Probleme auftreten. Bitte schicken Sie alle Meldungen, die Sie nicht verstehen und die nicht durch Ihre eigenen Änderungen an den Skripten verursacht wurden, an das CapiSuite-Team.

```
DDI_length="0"
```

Wenn Ihre ISDN-Karten mit einem Anlagenanschluss verbunden ist, benutzen Sie normalerweise auch DDI. DDI heißt, Sie haben nur eine Hauptrufnummer und können Ihre Durchwahlen nach Belieben selbst festlegen. In diesem Fall müssen Sie die Länge Ihrer Durchwahlen hier festlegen. Wenn Sie zum Beispiel 1234-000 bis 1234-999 benutzen, dann ist Ihre DDI_length 3. Wenn hier 0 gesetzt ist, dann ist DDI deaktiviert.

Sollten Sie jetzt nur Bahnhof verstanden haben, dann haben Sie vermutlich kein DDI und können die Einstellung so belassen, wie sie ist.

```
DDI_base_length="0"
```

Diese Option wird nur benutzt, wenn DDI_length nicht 0 ist. Hier wird die Länge der Basisrufnummer festgelegt - im obigen Beispiel wäre das 4.

```
DDI_stop_numbers=""
```

Wenn Sie üblicherweise Durchwahlen einer bestimmten Länge verwenden, aber auch einige kürzere Ausnahmen haben (beispielsweise die "-0" für die Zentrale), dann können Sie diese Ausnahmen hier durch Kommata getrennt auflisten.

1.2.3. Start von CapiSuite

Da CapiSuite ein Daemon ist, wird es normalerweise beim Systemstart aktiviert. Sie müssen lediglich den Aufruf von

```
/path/to/capisuite -d
```

in Ihre Start-Skripte einfügen. In LSB-konformen Linux-Distributionen finden Sie die Start-Skripte unter `/etc/init.d`. Für eine detaillierte Beschreibung, wie dort ein Service hinzu gefügt wird, schauen Sie bitte in der Dokumentation Ihrer Distribution nach. Ein Beispiel eines Start-Skripts für SUSE-Linux ist in der Source-Distribution enthalten (see `rc.capisuite`). Dieses sollte (hoffentlich) auch mit anderen LSB-konformen Distributionen funktionieren. Wenn Sie es anpassen müssen, würde ich mich über Ihr Feedback freuen und würde hier gerne Anleitungen für andere Distributionen aufnehmen.

Wenn Sie die richtigen RPM-Pakete von CapiSuite verwenden, sollten die benötigten Skripte bereits enthalten sein. Bitte verwenden Sie das Konfigurations-Tool Ihres Distributors, um sie zu aktivieren. Wenn Sie das RPM verwenden, das mit SUSE-Linux geliefert wird, und Sie bei den Standard-Skripten bleiben wollen, sollte alles "out of the box" funktionieren. Sobald Sie die Standard-Skripte konfiguriert haben, rufen Sie einfach **rccapisuite restart** auf.

Zu Debug-Zwecken können Sie CapiSuite auch jederzeit manuell starten, indem Sie

```
/path/to/capisuite
```

aufrufen

Es gibt auch noch ein paar weitere Kommandozeilen-Optionen:

Kommandozeilen-Optionen von CapiSuite

```
--help, -h
```

Zeigt eine kurze Zusammenfassung der Kommandozeilen-Optionen an

```
--config=file, -c file
```

Verwendet eine benutzerdefinierte Konfigurationsdatei anstelle von `/etc/capisuite/capisuite.conf` oder `/usr/local/etc/capisuite/capisuite.conf`.

```
--daemon, -d
```

Läuft als Daemon (verwendet im Start-Skript, siehe oben)

CapiSuite kann theoretisch von jedem Benutzer aufgerufen werden. Es braucht lediglich read/write-Zugriffsrechte auf `/dev/capi20`. Wenn Sie jedoch die Standard-Skripte verwenden, *muss* CapiSuite als `root` ausgeführt werden.

1.3. Features und Konfiguration der Standard-Skripte

Wie bereits oben erwähnt, werden mit CapiSuite Standard-Skripte geliefert, die Ihnen die am weitesten verbreiteten Kommunikationsfunktionen (Anrufbeantworter und Fax) zur Verfügung stellen.

Dieser Abschnitt soll Ihnen helfen, diese für Ihre täglichen Bedürfnisse zu nutzen.

1.3.1. Skript-Features

Die mit CapiSuite gelieferten Skripte stellen folgende Hauptfunktionen zur Verfügung:

- Multi-User-Anrufbeantworter
 - verschiedene Benutzer können verschiedene Nummern haben und unterschiedliche Ansagetexte
 - eingehende Anrufe werden gespeichert und per E-Mail an den Benutzer verschickt
 - die Verzögerung, bis ein Anruf angenommen wird, und die maximale Aufnahmelänge ist konfigurierbar
 - Stille wird erkannt und der Anruf nach einer einstellbaren Zeit beendet
 - eingehende Fax-Anrufe werden automatisch erkannt und empfangen
 - komfortable, menügesteuerte Fernabfragefunktionen, die Ihnen Datum/Uhrzeit des Anrufs sowie die angerufene und die anrufende Nummer mitteilen, stehen zu Verfügung
 - über das Fernabfragemenü können Sie Ihren eigenen Ansagetext aufnehmen
 - fast alle Einstellungen sind global konfigurierbar, können aber für jeden Benutzer überschrieben werden
- Fax-Gerät
 - verschiedene Benutzer können verschiedene Nummern haben

- eingehende Faxe werden gespeichert und per E-Mail an den Benutzer verschickt
- Kommandozeilen-Tools zum faxen von PostScript-Dokumenten sind vorhanden
- Anzahl von Versuchen und die Verzögerung beim Senden von Faxen ist konfigurierbar
- momentan wird nur ein ISDN-Controller für ausgehende Faxe unterstützt

Da meine Muttersprache deutsch ist, sind alle mitgelieferten WAV-Dateien nur in deutsch verfügbar. Wenn jemand englische WAV-Dateien (oder in irgendeiner anderen Sprache) zur Verfügung stellen möchte, nehmen Sie mit mir Kontakt auf. Danke!

1.3.2. Wie die Skripte funktionieren

Es folgt nun ein grober Überblick, wie die Skripte funktionieren. Ich werde hier nur das Verhalten beschreiben, das für den Benutzer interessant ist. Wenn Sie die Interna verstehen möchten, schauen Sie sich bitte den Abschnitt 2.6 an.

Wenn ein eingehender Anruf empfangen wird, wird die anrufende Nummer in verschiedenen Listen für die verschiedenen Benutzer gesucht. Jeder Benutzer kann in der Konfiguration eigene Nummern definieren (siehe unten). Die Skripte entscheiden also anhand der angerufenen Nummer, an welchen Benutzer der Anruf gerichtet ist. Wenn Sie die Nummer in der Sprach- oder Fax-Nummernliste eines Benutzers finden, beantworten Sie den Anruf mit diesem Service und geben dem Anrufer die Möglichkeit, seine Nachricht zu hinterlassen oder sein Fax zu senden.

Das empfangene Dokument wird dann in einem lokalen Verzeichnis in einem eigenen Format und auch konvertiert in ein gängiges Format gespeichert und zusammen mit einigen Details des Anrufs an den Benutzer gemailt. Sprachanrufe werden als WAV-Anhang geschickt, während Faxe als PDF-Dokumente an die Mail angehängt werden.

Sie bekommen also normalerweise Ihre eingehenden Anrufe als Mail an die angegebene Adresse geschickt - sie werden zur Sicherheit aber auch noch im lokalen Dateisystem gespeichert. Es ist Ihre Aufgabe, alte Dateien, die nicht mehr benötigt werden, zu löschen. Weitere Informationen finden Sie im Abschnitt 1.3.4.

Für den Anrufbeantworter besteht die Möglichkeit zur Fernabfrage. Der Anrufer bekommt ein Menü, wo er wählen kann, ob er seine Ansage aufnehmen oder gespeicherte Anrufe abhören möchte. Er bekommt gesagt, wieviele Anrufe vorhanden sind, von wem und wann sie empfangen wurden usw. Darüber hinaus kann er aufgenommene Anrufe, die er nicht mehr benötigt, löschen.

Ein anderes Skript prüft regelmäßig ein spezielles Queue-Verzeichnis nach ausgehenden Fax-Aufträgen. Mit Hilfe des Kommandozeilen-Tools **capisuitefax** werden Aufträge in diesem Verzeichnis abgelegt. Weitere Details dazu finden sich in Abschnitt 1.4.

1.3.3. Skript-Konfiguration

Es gibt einige wichtige Optionen, die die Skripte wissen müssen, bevor Sie sie benutzen können - Dinge, wie die Nummern des Benutzers und einige Details, wie Anrufe zu behandeln sind.

Diese Optionen werden aus zwei Konfigurationsdateien gelesen. Jede Konfigurationsdatei ist in einen oder mehrere Abschnitte unterteilt. Ein Abschnitt beginnt mit dem Abschnittsnamen in eckigen Klammern, z.B. [Abschnitt], während die Optionen Schlüssel="wert" Zeilen sind.

Jede Datei muss einen speziellen Abschnitt namens [GLOBAL] haben und einen Abschnitt für jeden angerufenen Benutzer namens [<username>] (wo <username> ein gültiger System-Benutzer ist).

Der Abschnitt [GLOBAL] definiert einige globale Einstellungen wie Pfadnamen und Standard-Einstellungen für Optionen, die in Benutzer-Sektionen überschrieben werden können. Die Benutzer-Abschnitte enthalten alle Optionen, die zu einem bestimmten Benutzer gehören.

Alle Optionen für die beiden Dateien werden weiter unten kurz beschrieben. Alle Details hierzu finden Sie in den Kommentaren in den Beispiel-Konfigurationsdateien, die mit CapiSuite installiert werden.

1.3.3.1. Konfiguration des Fax-Dienstes

Diese Datei enthält alle verfügbaren Optionen für die Fax-Dienste (Fax-Empfang und -Versand).

Sie liegt unter `/etc/capisuite/fax.conf` oder `/usr/local/etc/capisuite/fax.conf` (hängt von der Installation ab).

1.3.3.1.1. Der Abschnitt [GLOBAL]

```
spool_dir="/path/to/spooldir/"
```

Dieses Verzeichnis wird verwendet für die Archivierung von gesendeten (oder fehlgeschlagenen) Aufträgen. Es muss existieren und der Benutzer, unter dem CapiSuite läuft, muss Schreibrechte für seine Unterverzeichnisse haben. Es gibt zwei Unterverzeichnisse:

Kapitel 1. Getting Started

`spooldir/done/`

Erfolgreich erledigte Aufträge werden in dieses Verzeichnis verschoben.

`spooldir/failed/`

Fehlgeschlagene Aufträge landen schließlich hier.

Diese Option ist zwingend erforderlich.

`fax_user_dir="/path/to/userdir/"`

In diesem Verzeichnis werden Fax-Aufträge und empfangene Dokumente gespeichert. Es muss existieren und der Benutzer, unter dem CapiSuite läuft, muss dafür Schreibrechte besitzen. Es enthält für jeden konfigurierten Benutzer ein Unterverzeichnis (benannt nach seiner User-ID). Die folgenden Unterverzeichnisse befinden sich unterhalb des benutzerspezifischen Verzeichnisses:

`user_dir/username/received/`

Hier werden empfangene Faxe gespeichert.

`user_dir/username/sendq/`

Hier werden Fax-Dateien von **capsuitedfax** abgelegt, die versendet werden sollen.

Diese Option ist zwingend erforderlich.

`send_tries="10"`

Wenn ein Fax aus irgendeinem Grund nicht an den Empfänger verschickt werden kann, wird es mehrmals versucht. Diese Einstellung begrenzt die Anzahl der Versuche. Wenn alle Versuche fehlgeschlagen sind, wird der Auftrag im Verzeichnis für fehlgeschlagene Aufträge abgelegt (siehe `fax_spool_dir`) und der Benutzer bekommt eine Mail.

Diese Option ist optional. Wenn nichts angegeben wird, werden standardmäßig 10 Versuche unternommen.

`send_delays="60,60,60,300,300,3600,3600,18000,36000"`

Wenn ein Fax aus irgendeinem Grund nicht an den Empfänger verschickt werden kann, wird es nochmals versucht. Diese Einstellung gibt die Verzögerung in Sekunden zwischen zwei Versuchen an. Die verschiedenen

Werte werden durch Kommas getrennt und es dürfen *keine Leerzeichen* enthalten sein. Die Liste sollte `send_tries-1` (siehe `fax_send_tries`) Werte haben - wenn nicht, werden überzählige Einträge ignoriert und fehlende Einträge mit dem letzten Wert aufgefüllt. Die Voreinstellung sollte mit 10 größer werdenden Verzögerungen für bis zu 10 Versuchen OK sein.

Diese Option ist optional. Wenn nichts angegeben wird, wird die oben gezeigte Liste genommen.

```
send_controller="1"
```

Wenn Sie mehr als einen ISDN-Controller installiert haben (einige aktive Karten für mehr als einen Basisanschluß wie die AVM C2 oder C4 werden für CAPI-Anwendungen wie CapiSuite auch als mehrere Controller abgebildet), können Sie entscheiden, welcher Controller (und damit, welcher Basisanschluß) für das Versenden von Faxen genutzt werden soll. Alle Controller sind nummeriert, beginnend mit 1. Wenn Sie sich nicht sicher sind, welcher Controller welche Nummer hat, erhöhen Sie den Log-Level in CapiSuite auf mindestens 2 (siehe Abschnitt 1.2.2), starten es erneut und schauen sich die Log-Datei an, in der dann alle Controller aufgelistet werden. Leider kann CapiSuite im Moment nicht mehrere Controller für das Versenden von Faxen nutzen, sodass hier keine Liste erlaubt ist. Wenn Sie nur einen Controller haben, lassen Sie den Wert einfach auf 1 stehen.

Diese Option ist optional. Wenn nichts angegeben wird, wird standardmäßig der Controller 1 verwendet.

```
outgoing_MSN="<Ihre MSN>"
```

Diese Nummer wird als Ihre eigene Nummer für ausgehende Rufe verwendet. Wenn Sie nicht angegeben wird, wird die erste Nummer von `fax_numbers` verwendet (siehe `fax_numbers`). Ist diese ebenfalls nicht gesetzt, so kann der Benutzer kein Fax verschicken. Bitte ersetzen Sie den Eintrag mit einer gültigen MSN Ihres ISDN-Anschlusses oder lassen ihn leer. Dieser Wert kann in den Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist optional. Wenn nichts angegeben wird, wird standardmäßig die erste MSN von `fax_numbers` verwendet.

```
outgoing_timeout="60"
```

Standard-Einstellung, die angibt, wieviele Sekunden nach dem Wählen der Nummer auf eine Verbindung gewartet wird. Dieser Wert kann in den

Kapitel 1. Getting Started

Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist optional. Wenn nichts angegeben wird, wird standardmäßig 60 Sekunden gewartet.

```
dial_prefix=" "
```

Wenn hier etwas angegeben ist, wird es als Präfix vor jede Nummer eingefügt, die an **capisuitefax** übergeben wird. Dies ist z.B. sehr nützlich, wenn Ihr ISDN-Adapter an einer Telefonanlage hängt, die eine "0" für externe Anrufe benötigt. Es kann aber auch später für ein einzelnes Fax-Dokument deaktiviert werden, sodass diese Einstellung nicht interne Anrufe ohne Präfix verhindert.

Diese Option ist optional. Wenn nichts angegeben wird, wird standardmäßig kein Präfix verwendet.

```
fax_stationID="<Ihre faxID> "
```

Absenderkennung, die beim Versenden eines Fax-Dokuments verwendet wird. Die Absenderkennung ist normalerweise Ihre Fax-Nummer im internationalen Format, z.B. "+49 89 123456" für eine Nummer in München in Deutschland. Absenderkennungen dürfen nur aus dem "+"-Zeichen, Leerzeichen und den Ziffern 0-9 bestehen. Die maximale Länge ist 20. Dieser Wert kann in den Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist zwingend erforderlich.

```
fax_headline="<Ihre Fax-Kopfzeile> "
```

Fax-Kopfzeile, die beim Versenden eines Fax-Dokuments verwendet wird. Wo und ob diese Kopfzeile angezeigt wird, hängt von der Implementierung Ihres CAPI-Treibers ab. Die Kopfzeile sollte eine vernünftige Länge haben, damit sie auf den oberen Rand einer Seite passt, aber es gibt keine bestimmte Begrenzung.

Diese Option ist optional. Wenn nichts angegeben wird, wird standardmäßig keine Kopfzeile verwendet.

1.3.3.1.2. Die Benutzer-Abschnitte

outgoing_MSN

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

outgoing_timeout

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

fax_stationID

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

fax_headline

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

fax_numbers=" <Nummer1> , <Nummer2> , . . . "

Eine Liste mit Nummern, auf denen dieser Benutzer eingehende Fax-Anrufe empfangen möchte. Diese Nummern werden benutzt, um zwischen Benutzern zu unterscheiden - die selbe Nummer darf also nicht in mehr als einem Benutzerabschnitt erscheinen! Die Nummern sind durch Kommas getrennt und es sind *keine Leerzeichen* erlaubt. Die erste Nummer der Liste dient auch als Ihre eigene Nummer für ausgehende Faxe, wenn outgoing_MSN nicht angegeben wurde (siehe outgoing_MSN)

Wenn Sie die selbe Nummer zum Empfangen von Fax- und Sprach-Anrufen nutzen möchten, geben Sie sie hier bitte *nicht* an. Verwenden Sie stattdessen die voice_numbers Option (siehe voice_numbers) - der Anrufbeantworter hat eine eingebaute Fax-Erkennung und kann auch Faxe empfangen.

Wenn die Liste nur den * enthält, werden *alle* eingehenden Anrufe für diesen Benutzer angenommen (bitte mit Vorsicht benutzen!). Dies ist nur sinnvoll bei einem Setup mit nur einem Benutzer, der alle Anrufe als Fax empfangen möchte.

Wenn aus irgendeinem Grund für spezielle MSNs *keine Zielnummer* übertragen wird (die österreichische Telekom macht das für die Haupt-MSN, die wird als "Global Call" bezeichnet), können Sie das besondere Zeichen - benutzen. Dies bedeutet "no destination number available".

Diese Option ist optional. Wenn nichts angegeben wird, kann der Benutzer keine Fax-Dokumente empfangen.

```
fax_email=" "
```

Wenn hier etwas angegeben wird, werden die Strings als E-Mail-Adressen interpretiert, an die empfangene Faxe geschickt werden. Mehrere Adressen werden durch Kommas getrennt. Wenn der Eintrag leer ist, werden sie an den System-Account auf dem System, auf dem CapiSuite läuft, geschickt. Die Adresse wird auch für Status-Reports für versendete Fax-Aufträge benutzt. Wenn Sie überhaupt nicht möchten, dass E-Mails verschickt werden, verwenden Sie die action-Option (siehe `fax_action`).

Diese Option ist optional. Wenn nichts angegeben wird, wird die Mail an den System-Account geschickt.

```
fax_action="MailAndSave"
```

Hier können Sie festlegen, welche Aktionen ausgeführt werden, wenn ein Anruf empfangen wurde. Momentan werden zwei mögliche Aktionen unterstützt:

`MailAndSave`

Der empfangene Anruf wird an die angegebene Adresse (siehe `fax_email`) geschickt und im `fax_user_dir` (siehe Abschnitt 1.3.3.1.1) gespeichert.

`SaveOnly`

Der empfangene Anruf wird nur im `fax_user_dir` (siehe Abschnitt 1.3.3.1.1) gespeichert.

Diese Option ist zwingend erforderlich.

1.3.3.2. Konfiguration des Anrufbeantworters

Diese Datei enthält alle verfügbaren Einstellungen für den Anrufbeantworter.

Sie liegt unter `/etc/capisuite/answering_machine.conf` oder `/usr/local/etc/capisuite/answering_machine.conf` (hängt von der Installation ab).

1.3.3.2.1. Der Abschnitt [GLOBAL]

```
audio_dir="/path/to/audiodir/"
```

Das Anrufbeantworter-Skript nutzt verschiedene WAV-Dateien, z.B. eine globale Ansage, wenn der Benutzer keine eigene hat, sowie einige gesprochene Worte für die Fernabfrage und das dabei verwendete Menü. Diese Audio-Dateien werden in diesem Verzeichnis gesucht. Wenn `user_audio_files` aktiviert ist (siehe `user_audio_files`), kann jeder Benutzer seine eigenen Audio-Schnipsel in seinem `user_dir` (siehe `voice_user_dir`) verwenden.

Diese Option ist zwingend erforderlich.

```
voice_user_dir="/path/to/userdir/"
```

In diesem Verzeichnis werden benutzerspezifische Daten gespeichert. Es muss existieren und der Benutzer, unter dem CapiSuite läuft, muss dafür Schreibrechte besitzen. Es enthält für jeden konfigurierten Benutzer ein Unterverzeichnis (benannt nach seiner User-ID). Die folgenden Unterverzeichnisse befinden sich unterhalb des benutzerspezifischen Verzeichnisses:

```
user_dir/username/
```

Hier kann der Benutzer seine eigenen Audio-Dateien (siehe `user_audio_files`) ablegen. Die benutzerdefinierte Ansage wird ebenfalls hier gespeichert.

```
user_dir/username/received/
```

Hier werden empfangene Sprach-Anrufe gespeichert.

Diese Option ist zwingend erforderlich.

```
user_audio_files="0"
```

Wenn dies auf 1 gesetzt wird, kann jeder Benutzer seine eigenen Audio-Dateien in seinem Benutzerverzeichnis verwenden (siehe `voice_user_dir`). Wenn dies auf 0 gesetzt wird, wird nur das `audio_dir` (siehe `voice_audio_dir`) durchsucht.

Diese Option ist optional. Wenn nichts angegeben wird, werden keine benutzerspezifischen Audio-Dateien verwendet (0).

Kapitel 1. Getting Started

```
voice_delay="15"
```

Setzt den Standard-Wert für die Verzögerung, mit der eingehende Anrufe angenommen werden (in Sekunden). Ein Wert von 10 bedeutet, dass der Anrufbeantworter einen eingehenden Anruf 10 Sekunden nach der eingehenden Verbindungsanforderung annimmt. Dieser Wert kann in den Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist zwingend erforderlich.

```
announcement="announcement.la"
```

Setzt den Standard-Namen für die Benutzer-Ansage. Die Ansagen werden dann in `user_dir/username/announcement` gesucht. Wenn sie dort, nicht gefunden wird, wird eine globale Ansage, die die angerufene MSN enthält, abgespielt. Dieser Wert kann in den Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist optional. Wenn nichts angegeben wird, wird "announcement.la" angenommen.

```
record_length="60"
```

Einstellung für die maximale Aufnahmelänge in Sekunden. Dieser Wert kann in den Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist optional. Wenn nichts angegeben wird, beträgt die maximale Aufnahmelänge 60 Sekunden.

```
record_silence_timeout="5"
```

Einstellung für den Stille-Timeout während der Aufnahme in Sekunden. Wenn dieser Wert größer als 0 ist, wird die Aufnahme beendet, wenn für die angegebene Zeit Stille erkannt wird. Um den Timeout zu deaktivieren, setzen Sie ihn auf 0. Dieser Wert kann in den Benutzer-Abschnitten individuell überschrieben werden.

Diese Option ist optional. Wenn nichts angegeben wird, beträgt der Timeout 5 Sekunden.

1.3.3.2.2. Verfügbare Optionen für die Benutzer-Abschnitte in der Anrufbeantworter-Konfiguration

voice_delay

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

announcement

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

record_length

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

record_silence_timeout

Benutzerspezifischer Wert für die globale Option im Abschnitt [GLOBAL]
oben

voice_numbers="<Nummer1>,<Nummer2>,..."

Eine Liste mit Nummern, auf denen dieser Benutzer eingehende Sprach-Anrufe empfangen möchte. Diese Nummern werden benutzt, um zwischen Benutzern zu unterscheiden - die selbe Nummer darf also nicht in mehr als einem Benutzerabschnitt erscheinen! Die Nummern sind durch Kommas getrennt und es sind *keine Leerzeichen* erlaubt. Das Anrufbeantworter-Skript macht auch eine automatische Fax-Erkennung, sodass ein Fax an diese Nummer geschickt werden kann. Wenn die Liste nur den * enthält, werden *alle* eingehenden Anrufe für diesen Benutzer angenommen (bitte mit Vorsicht benutzen!). Dies ist nur sinnvoll bei einem Setup mit nur einem Benutzer, der alle Anrufe empfangen möchte.

Wenn aus irgendeinem Grund für spezielle MSNs *keine Zielnummer* übertragen wird (die österreichische Telekom macht das für die Haupt-MSN, die wird als "Global Call" bezeichnet), können Sie das besondere Zeichen - benutzen. Dies bedeutet "no destination number available".

Diese Option ist optional. Wenn nichts angegeben wird, kann der Benutzer keine Sprach-Anrufe empfangen.

voice_email=""

Wenn hier etwas angegeben wird, werden die Strings als E-Mail-Adressen interpretiert, an die empfangene Faxe und Sprach-Anrufe geschickt werden.

Kapitel 1. Getting Started

Mehrere Adressen werden durch Kommas getrennt. Wenn der Eintrag leer ist, werden sie an den System-Account auf dem System, auf dem CapiSuite läuft, geschickt. Wenn überhaupt nicht möchten, dass E-Mails verschickt werden, verwenden die action Option (siehe `voice_action`).

Diese Option ist optional. Wenn nichts angegeben wird, wird die Mail an den System-Account geschickt.

```
pin="<Ihre PIN>"
```

Der Anrufbeantworter hat auch eine Fernabfrage-Funktion. Diese Funktion kann benutzt werden, wenn, während die Ansage abgespielt wird, eine PIN (Personal Identification Number) eingegeben wird. Diese PIN kann hier definiert werden. Wenn die Fernabfrage-Funktion nicht nutzen möchten, geben Sie hier einfach nichts an. Die PIN hat keine maximale Länge - aber Sie sollten vielleicht keine 200 Ziffern verwenden, da Sie sie sich ansonsten nicht so gut merken können (zumindest könnte ich das nicht). ;-)

Diese Option ist optional. Wenn nichts angegeben wird, wird die Fernabfrage-Funktion deaktiviert.

```
voice_action="MailAndSave"
```

Hier können Sie festlegen, welche Aktionen ausgeführt werden, wenn ein Anruf empfangen wurde. Momentan werden drei mögliche Aktionen unterstützt:

`MailAndSave`

Der empfangene Anruf wird an die angegebene Adresse (siehe `voice_email`) geschickt und im `voice_user_dir` (siehe `voice_user_dir`) gespeichert.

`SaveOnly`

Der empfangene Anruf wird nur im `voice_user_dir` (siehe `voice_user_dir`) gespeichert.

`None`

Es wird nur die Ansage abgespielt - es wird nichts aufgenommen.

Diese Option ist zwingend erforderlich.

1.3.4. Alte Dateien löschen

Wie oben beschrieben, werden alle ein- und ausgehenden Anrufe im lokalen Dateisystem gespeichert, um sicher zu sein, dass nichts verloren geht. CapiSuite führt keine Aufräumarbeiten durch, sodass diese Dateien für immer auf Ihrem System verbleiben, wenn Sie nicht von Zeit zu Zeit aufräumen.

Da es nicht sehr bequem ist, dies von Hand zu machen, empfehle ich, diesen Prozess zu automatisieren. cron ist prädestiniert für solch eine Aufgabe. Auf den meisten modernen GNU/Linux-Distributionen können Sie einfach Skripte in `/etc/cron.daily` ablegen, die dann automatisch einmal am Tag ausgeführt werden.

Ein Beispiel für ein Bash-Skript, das Sie verwenden können, liegt dem CapiSuite-Paket bei. Kopieren Sie einfach `capisuite.cron` nach `/etc/cron.daily/capisuite` und stellen Sie sicher, dass es die richtigen Zugriffsrechte besitzt (Owner root, Executable-Bit gesetzt).

Editieren Sie nun die Datei `cronjob.conf` und kopieren Sie es in Ihr CapiSuite-Konfigurationsverzeichnis (normalerweise `/etc/capisuite` oder `/usr/local/etc/capisuite`). Es sagt dem Cron-Job, wie lange die Dateien in den verschiedenen Verzeichnisse gespeichert bleiben sollen.

Die folgenden Optionen sind verfügbar:

`MAX_DAYS_RCVD=" <Wert> "`

Dateien, die in den Empfangs-Verzeichnissen der Benutzer gespeichert sind und auf die in den letzten `<Wert>` Tagen nicht zugegriffen wurde, werden gelöscht. Setzen Sie diesen Wert auf 0, um das automatische Löschen zu deaktivieren.

`MAX_DAYS_DONE=" <Wert> "`

Dateien, die im globalen Erledigt-Verzeichnis gespeichert sind und auf die in den letzten `<Wert>` Tagen nicht zugegriffen wurde, werden gelöscht. Setzen Sie diesen Wert auf 0, um das automatische Löschen zu deaktivieren.

`MAX_DAYS_FAILED=" <Wert> "`

Dateien, die im globalen Fehlgeschlagen-Verzeichnis gespeichert sind und auf die in den letzten `<Wert>` Tagen nicht zugegriffen wurde, werden gelöscht. Setzen Sie diesen Wert auf 0, um das automatische Löschen zu deaktivieren.

1.4. CapiSuite mit den Standard-Skripten verwenden

1.4.1. Anrufe empfangen

Dies ist ein schön kurzer Abschnitt. Wenn Sie CapiSuite und die Skripte einmal konfiguriert haben CapiSuite erfolgreich gestartet haben, brauchen Sie nichts mehr zu tun. Sie bekommen Ihre Mails wie im Abschnitt 1.3.2 beschrieben und das war's. Sie müssen nur Ihr Mail-Programm so einrichten, dass Sie lokale Mails empfangen können. Viel Spaß! :-)

1.4.2. Eine Fernabfrage machen

Um eine Fernabfrage zu machen, geben Sie bitte Ihre PIN (siehe Abschnitt 1.3.3.2.2) ein, während die Ansage des Anrufbeantworters abgespielt wird. Nach einigen Sekunden bekommen Sie ein "Sprach-Menü", das Ihnen sagt, wie Sie Ihre eigene Ansage für den Anrufbeantworter aufnehmen können oder wie Sie empfangene Anrufe abspielen können.

1.4.3. Faxe Versenden

Die Standard-Skripte für CapiSuite enthalten auch ein Kommandozeilen-Tool zum Versenden von Faxen namens **capisuitefax**.

capisuitefax wird mit einigen Parametern aufgerufen, um ihm zu sagen, welche Datei (momentan werden nur PostScript- und PDF-Dateien unterstützt) an welche Nummer verschickt werden soll. Es fügt dann den in das richtige Format konvertierten Job in die Send-Queue ein, von wo aus er von einem anderen CapiSuite-Skript genommen und an den Empfänger verschickt wird. Wenn das Versenden erfolgreich beendet wurde oder nach mehreren Versuchen endgültig fehlgeschlagen ist, erhält der/die entsprechende Benutzer(in) ein E-Mail, die ihm/ihr mitteilt, was passiert ist.

capisuitefax kennt die folgenden Optionen:

```
capisuitefax [-q] [-n] [-u user] [-A addr] [-S subj] -d number file1 [file2 ...]
```

```
capisuitefax [-q] -a id
```

```
capisuitefax -l
```

```
capisuitefax -h
```

-a id

Bricht den Auftrag mit der angegebenen ID ab. Um die ID eines Auftrags zu ermitteln, können Sie die Option -l verwenden.

-A adr

Der Adressat des Faxes. Diese Angabe dient (zur Zeit) nur zur leichteren Identifikation eines Jobs. Sie wird in der versandten Status-Mail später zitiert.

-d dialstring

Die Nummer, die angerufen werden soll (Empfänger des Faxes).

-h

Zeigt eine kurze Kommandozeilen-Hilfe an.

-l

Zeigt die Aufträge an, die sich momentan in der Send-Queue befinden.

-n

Nicht den konfigurierten Präfix für diesen Auftrag verwenden. Dies ist nützlich für interne Aufträge.

-q

Sei still, gib keine informativen Meldungen aus!

-S subj

Der Betreff des Faxes. Diese Angabe dient (zur Zeit) nur zur leichteren Identifikation eines Jobs. Sie wird in der versandten Status-Mail später zitiert.

-u user

Fax als anderer Benutzer versenden. Nur möglich, wenn **capisuitefax** als Benutzer `root` aufgerufen wird. Diese Option ist hauptsächlich für Erweiterungen, z. B. zum Faxen im Netzwerk, nützlich.

file1 [file2...]

Eine oder mehrere PostScript/PDF-Dateien, die an diesen Empfänger verschickt werden sollen. Mehrere Dateien erzeugen mehrere separate Fax-Aufträge).

Kapitel 2. Users Guide

Im vorigen Kapitel haben Sie gesehen, wie die mit CapiSuite gelieferten Standard-Skripte genutzt werden. Das Hauptziel bei der Entwicklung von CapiSuite war aber nicht, eine perfekte "Ready-to-Use"-Anwendung zur Verfügung zu stellen. Ich wollte ein Tool entwickeln, mit dem jeder auf einfache Weise seine *eigenen* Anwendungen schreiben kann. Ich werde Ihnen in den nächsten Abschnitten zeigen, wie das geht.

2.1. Einführung in Python

Als ich über die Skriptsprache nachdachte, die ich in CapiSuite integrieren wollte, war meine erste Idee, eine eigene, einfache zu entwickeln. Aber je tiefer ich in das Thema einstieg, desto mehr stellte sich heraus, dass eine allgemeine Sprache sehr viel nützlicher sein würde, als jedes benötigte Rad neu zu erfinden. Also suchte ich nach einer einfach zu integrierenden (und zu lernenden) Sprache. Die, die mir am besten gefiel, war Python - und sie hatte eine schöne Dokumentation über das Einbetten in andere Applikation. Deshalb habe ich mich für sie entschieden und ich habe es bis jetzt nicht bereut. :-)

Also, die erste Sache, die zu tun ist, ist Python zu lernen. Aber keine Angst - sie wurde als eine Anfängersprache entwickelt und Guido (Guido van Rossum, der Erfinder von Python) hat das meiner Meinung nach sehr gut gemacht.

In den nächsten Abschnitten werde ich Ihnen eine kurze Einführung in die Features von Python geben, die Sie am häufigsten für CapiSuite brauchen werden. Da dies weder ein Python-Handbuch noch ein Programmier-Tutorial sein soll, gehe ich davon aus, dass Sie bereits mit den Grundkonzepten heutiger verbreiteter prozeduraler und objektorientierter Sprachen vertraut sind.

Wenn nicht, empfehle ich Ihnen, mit Hilfe eines Buchs Python zu erlernen - es gibt viele davon in verschiedenen Sprachen. Auf der Python-Homepage unter <http://www.python.org> finden Sie ein reichhaltiges Angebot an kostenlosen Handbüchern und Tutorials.

2.1.1. Python-Grundlagen

Python unterstützt die meisten Features, die Sie von anderen Sprachen kennen. Hier ist die Syntax der Basisoperationen einer Python-Session. Eine Python-Session ist ein nettes Feature des Interpreters: sie wird einfach gestartet, indem Sie **python** in einer Shell eingeben und Sie erhalten eine Eingabeaufforderung:

```
gernot@linux:~> python
```

Kapitel 2. Users Guide

```
Python 2.2.1 (#1, Sep 10 2002, 17:49:17)
[GCC 3.2] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Wie Sie sehen, ist die Eingabeaufforderung von Python `>>>`. Wenn Sie mehrzeilige Befehle eingeben, zeigt Python eine zweite Eingabeaufforderung an: ...

```
>>> if (1==2):
...     print "Now THAT's interesting!"
... 
```

Ok, gehen wir weiter:

```
>>> # Kommentare beginnen mit # am Anfang der Zeile
>>> # jetzt die gewöhnlichen ersten Schritte
>>> print "hello world"
hello world
>>> # Variablen
>>> a=5 # keine separate Deklarationen erforderlich
>>> b=a*2
>>> print b
10
>>> b='hello'
>>> print b,'world'
hello world
>>> # Python ist sehr mächtig bzgl. Sequenzen
>>> a=(1,2,3) # definiert ein Tuple (nicht änderbar!)
>>> print a
(1, 2, 3)
>>> a[1]=2 # dies geht schief
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
>>> a=[1,2,3] # definiert eine Liste (änderbar)
>>> a[1]=7
>>> print a
[1, 7, 3]
>>> # Kontrollstrukturen
>>> if (b=='hello'):
...     print "b is hello"
... else:
...     print "???"
...
b is hello
>>> # das for-Statement kann über Sequenzen iterieren
>>> for i in a:
```

```

...     print i
...
1
7
3
>>> # ersetze Positionen 1 bis 3 (ohne 3) mit 0
>>> a[1:3]=[0]
>>> a
[1, 0]
>>> # a[-i] ist das i-te Element von hinten gezählt
>>> a[-1]=7; a[-2]=8
>>> a
[8, 7]

```

2.1.2. Blöcke, Funktionen und Exceptions

Blöcke werden nur durch Einrückung gebildet. Kein `begin`, `end`, Klammern (`{, }`) oder ähnliches ist nötig. Dies sieht auf den ersten Blick etwas unbequem aus, aber es ist wirklich nett - Sie müssen Ihren Code immer so strukturieren, wie er *gemeint* ist:

```

>>> for i in [1,2,3]:
...     print 2*i
...
2
4
6
>>> i=0
>>> while (i!=3):
...     print i
...     i+=1
...
0
1
2

```

Schauen wir uns nun an, wie Funktionen definiert werden und wie man mit Exceptions arbeitet:

```

>>> def double_it(a):
...     return (2*a)
...
>>> print double_it(9)
18
>>> print double_it("hello")
hellohello
>>>

```

```
>>> # wir lösen eine Exception aus
>>> a=1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ZeroDivisionError: integer division or modulo by zero
>>>
>>> # jetzt reagieren wir darauf
>>> try:
...     a=1/0
... except ZeroDivisionError,e:
...     print "You divided by zero, message was:",e
...
You divided by zero, message was: integer division or modulo by zero
```

2.1.3. Mit Modulen arbeiten

Module sind eine Möglichkeit, Funktionen zusammen zu packen. Sie müssen importiert werden, bevor Sie sie benutzen können und sie stellen Ihnen ein neues Objekt zur Verfügung, das alle Funktionen enthält. Lassen Sie uns mit einigen etwas herumspielen:

```
>>> import time
>>> # was ist in time?
>>> dir(time)
['__doc__', '__file__', '__name__', 'accept2dyear', ...]
>>> # Was machen all diese Funktionen? Python sagt es uns...
>>> print time.__doc__
This module provides various functions to manipulate time values.
```

```
[...]
```

Variables:

```
[...]
```

Functions:

```
time() -- return current time in seconds since the Epoch as a float
ctime() -- convert time in seconds to string
[...]
```

```
>>> # Kannst du mir ctime bitte etwas genauer erklären?
>>> print time.ctime.__doc__
ctime(seconds) -> string
```

Convert a time in seconds since the Epoch to a string in local time.

This is equivalent to `asctime(localtime(seconds))`. When the time tuple is not present, current time as returned by `localtime()` is used.

```
>>> time.time()
1044380131.186987
>>> time.ctime()
'Tue Feb  4 18:35:36 2003'
>>> import os
>>> os.getuid()
500
>>> import pwd
>>> pwd.getpwuid(500)
('hans', 'x', 500, 100, 'Hans Meier', '/home/gernot', '/bin/bash')
```

Ok, ich hoffe, Sie haben einen kleinen Eindruck von Python bekommen. Viel Spaß damit. Ich hatte ihn... :-)

Wenn Sie weitere Fragen haben, würde ich Ihnen *wirklich* empfehlen, mit einem guten Buch oder der Dokumentation auf <http://www.python.org> weiter zu machen. Bitte stellen Sie keine allgemeinen Python-Fragen auf den CapiSuite-Listen...

2.2. Ein erster Blick auf das Incoming- und Idle-Skript

Im Abschnitt 1.2.1 sagte ich schon, dass zwei Typen von Skripten in CapiSuite genutzt werden. Schauen wir sie uns jetzt mal genauer an.

2.2.1. Das Incoming-Skript

Immer, wenn ein Anruf von CapiSuite empfangen wird, wird das Incoming-Skript aufgerufen - um genau zu sein, es wird eine Funktion namens `callIncoming` in einem Python-Skript irgendwo auf Ihrer Festplatte aufgerufen. Dieses "irgendwo" wurde im Abschnitt 1.2.2 definiert, erinnern Sie sich?

Das Skript muss also immer eine Funktion mit folgender Signatur definieren:

```
def callIncoming(call, service, call_from, call_to):
    # Funktionsrumpf
    ...
```

Die von CapiSuite übergebenen Parameter sind:

call

Referenz auf den eingehenden Anruf. Diese wird später in allen CapiSuite-Funktionen, die Sie aufrufen, dazu genutzt, dem System mitzuteilen, welcher Anruf gemeint ist. Sie werden diesen Parameter nur an andere Funktionen weitergeben - das Skript kann nichts damit anfangen (er ist *unsichtbar*).

service (Integer)

Service des eingehenden Anrufs wie er vom ISDN signalisiert wird. Folgende Werte sind möglich:

- SERVICE_VOICE: Sprach-Anruf
- SERVICE_FAXG3: analoger Fax-Anruf
- SERVICE_OTHER: anderer Service, der oben nicht gelistet ist

call_from (String)

Die Nummer des Anrufers (Quelle des Anrufs) als Python-String

call_to (String)

Die angerufene Nummer (Ziel des Anrufs) als Python-String

Die erste Aufgabe der Funktion sollte es sein, zu entscheiden, ob sie den Anruf annehmen oder ablehnen möchte. Wenn sie ihn annimmt, wird sie normalerweise etwas damit machen (ein Fax empfangen, einen Sprach-Anruf aufnehmen, eine nette Ansage abspielen, ...) und die Verbindung dann beenden. Nachdem alle erforderliche Arbeit getan ist, sollte sie sich sofort beenden. In einem späteren Kapitel werde ich Ihnen ein paar Beispiele zeigen, um das etwas zu verdeutlichen.

Natürlich können Sie Ihre Anwendung in mehrere Funktionen und vielleicht mehrere Skripte aufteilen, die aufgerufen und/oder rekursiv importiert werden - aber der Startpunkt ist immer das *Incoming-Skript*, das `callIncoming` enthält. Wenn Python und CapiSuite richtig installiert sind, sollten Sie auch Python-Module importieren und benutzen können.

2.2.2. Das Idle-Skript

Während das Incoming-Skript nur gestartet wird, wenn ein Anruf herein kommt, brauchen wir einen anderen Mechanismus, um einen ausgehenden Anruf auszulösen. Da CapiSuite nicht wissen kann, wann Sie dies beabsichtigen, ruft es in regelmäßigen Abständen eine Funktion namens `idle` im sog. "Idle-Skript" auf. Wie die Abstände konfiguriert werden und wo sich das Skript befindet, finden Sie unter Abschnitt 1.2.2.

Die aufgerufene Funktion muss folgende Signatur haben:

```
def idle(capi):  
    # Funktionsrumpf  
    ...
```

Der einzige von CapiSuite übergebene Parameter ist:

`capi`

Dies ist eine Referenz auf eine interne Klasse von CapiSuite, die die Kommunikation mit der CAPI-Schnittstelle abwickelt. Sie müssen diesen Parameter an einige CapiSuite-Funktionen übergeben. Sie können darüber hinaus in Ihrem Skript nichts sinnvolles damit anfangen. Dieser Parameter ist nur zum internen Gebrauch und wird möglicherweise (hoffentlich) irgendwann entfernt werden. Bis dahin übergeben Sie ihn einfach, wo es erforderlich ist.

Jetzt können Sie in dieser Funktion machen, was Sie wollen. Wahrscheinlich werden Sie in einem E-Mail-Account nach einem Auftrag sehen, nach einer zu sendenden Datei in einem speziellen Verzeichnis oder ähnliches und einen Anruf tätigen, um den Auftrag an den richtigen Empfänger zu schicken.

Theoretisch könnten Sie auch jede andere periodische Aufgabe auf Ihrem System mit dem Idle-Skript erledigen - aber wir sollten solche allgemeinen Dinge lieber Anwendungen überlassen, die für solche Sachen entwickelt wurden wie z.B. cron. ;-)

Wie oben erwähnt, kann `idle` andere Funktionen oder Skripte aufrufen, wenn Sie wollen, und alle Python-Module können importiert werden.

2.3. Verwendete Dateiformate

Bevor wir mit dem Schreiben von Skripten weiter machen, lassen Sie mich Ihnen bitte einige Worte zu den Dateiformaten sagen, die der CapiSuite-Kern verwendet.

CapiSuite liest und schreibt Dateien immer im dem Format, das die CAPI-ISDN-Treiber erwarten bzw. übergeben. Deshalb muss nicht alles in oder aus anderen Formaten konvertiert werden, wodurch unnötiger Overhead vermieden wird.

Da diese Formate nicht so bekannt sind und Sie spezielle Tools für die Konvertierung oder zum Anschauen/Abspielen brauchen, gebe ich Ihnen einen kurzen Überblick, wie das geht.

Wahrscheinlich werden Ihre Skripte die speziellen ISDN-Dateiformate in bekannte Formate konvertieren, um sie z.B. per E-Mail an Sie zu verschicken. Trotzdem

empfehle ich Ihnen, die empfangenen und gesendeten Dateien irgendwo im CapiSuite-eigenen Format zu speichern. Dies schützt Sie vor Datenverlust, wenn die Konvertierung schief geht und hilft Ihnen beim debuggen Ihrer Skripte.

Alle Tools, die hier erwähnt werden, sind im Abschnitt 1.1.1.2 beschrieben. Dort finden Sie Informationen, wo Sie diese bekommen.

2.3.1. Format der Sprach-Dateien (invertiert A-Law, 8kHz, mono)

ISDN überträgt Sprach-Daten als Wave-Dateien mit einer Sample-Rate von 8kHz in Mono. Um Bandbreite zu sparen, wird eine Kompression namens A-Law verwendet (zumindest in Europa, andere Länder wie die USA verwenden u-Law, das A-Law sehr ähnlich ist). Aus irgendeinem Grund, den ich nicht verstehe, verwenden sie eine Form von A-Law mit umgedrehter Bitfolge namens "invertiert A-Law".

2.3.1.1. Erzeugen von A-Law-Dateien

Es gibt zwei Möglichkeiten, A-Law-Dateien zu erzeugen.

Die erste ist, Ihren Computer mit Ihrem Telefon anzurufen (verwenden Sie entweder das Standard-Anrufbeantworter-Skript und konfigurieren Sie es wie im Abschnitt 1.3.3.2 beschrieben oder schreiben Sie selbst ein einfaches Skript). Sie können dann alles aufnehmen, was Sie wollen. Danach können Sie die Datei nehmen (wenn Sie die Standard-Skripte verwenden, nehmen Sie bitte die Datei aus dem `user_dir`, nicht den Anhang der Mail, da dieser bereits konvertiert wurde) und verwenden.

Eventuell möchten Sie die aufgenommene Datei kürzen und unerwünschte Geräusche und Stille am Anfang und am Ende entfernen. Dies kann einfach mit **sox** und **play** erledigt werden (beide befinden sich im **sox**-Paket).

Mit **sox** kann man eine Datei konvertieren, während man sie mit **play** nur abspielen kann. Beide unterstützen die selben Effekte, einschließlich der Trim-Option. Beide erkennen auch, welchen Dateityp sie verwenden, indem Sie sich den Dateinamenerweiterung Ihrer Datei ansehen. Alle Ihre invertierten A-Law-Dateien sollten daher in der Form `something.la` benannt sein (`.la` ist die umgekehrte Form von `.al`, was für A-Law steht).

Lassen Sie uns also zuerst die optimalen Werte für den Trim-Effekt herausfinden, indem wir **play** aufrufen:

```
play myfile.la trim <Start-Offset> <Dauer>
```

Spielen Sie nun ein bisschen mit dem Start-Offset und der Dauer (beide angegeben in Sekunden), bis Sie die richtigen Werte haben. Wenn Sie sie gefunden haben, können Sie **sox** verwenden, um die benötigte Datei zu erzeugen:

```
sox myfile.la outfile.la trim <Start-Offset> <Dauer>
```

Sie erhalten dann eine Datei namens `outfile.la`, die jetzt enthalten sollte, was Sie wollen.

Die zweite Möglichkeit, eine invertierte A-Law-Datei zu erzeugen, ist, eine normale WAV-Datei mit Ihrem Lieblings-Sound-Tool aufzunehmen und sie mit **sox** in das Zielformat zu konvertieren. Sie erzielen die besten Ergebnisse, wenn Ihre WAV-Dateien bereits das Format 8kHz, Mono, 8 Bit haben. **sox** kann auch andere WAV-Dateien wenn nötig konvertieren, aber dies resultiert normalerweise in einer schlechteren Qualität. Empfehlenswert ist auch, die Aufnahme auf maximal 50% der maximalen Amplitude zu normalisieren.

Sie können WAV in invertiert A-Law konvertieren, indem Sie aufrufen (danke an Carsten Heesch für den Tipp):

```
sox myfile.wav -r 8000 -c 1 -b outfile.la resample -ql
```

2.3.1.2. Abspielen von A-Law-Dateien

Auch hier gibt es wieder zwei Möglichkeiten. Der **play**-Befehl von **sox** kann das invertierte A-Law-Format ohne Konvertierung abspielen. Rufen Sie einfach **play** mit dem Dateinamen als Parameter auf:

```
play myfile.la
```

Sie können aber auch **sox** verwenden, um die A-Law-Dateien in das gebräuchlichere WAV-Format zu konvertieren, indem die aufrufen:

```
sox myfile.la outfile.wav
```

Die erzeugte Datei `outfile.wav` kann mit fast jedem Audio-Player problemlos abgespielt werden.

2.3.2. Format von Fax-Dateien (Structured Fax Files)

CAPI-konforme Treiber erwarten und übergeben Fax-Dateien als sog. Structured Fax File (SFF). Da dies ein CAPI-spezifisches Format zu sein scheint, gibt es nicht sehr viele Tools für GNU/Linux, die es verarbeiten können. Ich habe schließlich einige kleine Tools gefunden, die Peter Schäfer geschrieben hat und die wir hier verwenden werden.

CapiSuite kann auch Farbfaxe empfangen, welche in einem speziellen Dateiformat abgelegt werden, das ich CFF genannt habe.

2.3.2.1. Erzeugen eines SFF

In aktuellen Ghostscript-Versionen gibt es einen Patch von Peter, um SF-Dateien zu erstellen. Um zu prüfen, ob Ihr Ghostscript das schon unterstützt, geben Sie **gs --help** ein und suchen nach dem sog. **cfax**-Device in der langen Device-Liste, die angezeigt wird. Wenn es nicht aufgelistet ist, müssen Sie einen neueren Ghostscript verwenden oder Ihn neu übersetzen, sorry. Ich kenne keine andere Möglichkeit, momentan SFF zu erzeugen.

Sie brauchen erstmal eine PostScript-Datei (wie sie von fast jedem Linux-Programm erstellt wird, wenn Sie "in Datei drucken" auswählen). Jetzt können Sie GhostScript aufrufen, um sie in ein SFF zu konvertieren:

```
gs -dNOPAUSE -d BATCH -sDEVICE=cfax -sOutputFile=outfile.sff file.ps
```

Wenn Sie sich nicht sicher sind, ob es funktioniert hat, können Sie **sffview** wie weiter unten beschrieben verwenden.

2.3.2.2. Ansehen / konvertieren von SFF

Um einfach ein empfangenes SFF anzusehen, können Sie das **sffview**-Programm verwenden. Dies ist ein einfaches, aber sehr nützliches Tool zum Ansehen von SF-Dateien, ohne sie konvertieren zu müssen. Starten Sie es einfach und Sie bekommen ein GUI, wo Sie die gewünschte Datei öffnen können.

Wenn Sie eine Fax-Datei in ein gebräuchlicheres Format konvertieren möchten, empfehle ich **sfftobmp**. Es unterstützt einige Ausgabe-Formate wie JPEG, TIFF, PBM oder BMP. Ich bevorzuge mehrseitige TIFF-Dateien, da dies das einzige Format ist, das in der Lage ist, mehrere Seiten in einer Datei zu speichern. Um SFF in ein mehrseitiges TIFF zu konvertieren, rufen Sie auf:

```
sfftobmp -tif myfile.sff outfile.tiff
```

Sie erhalten eine TIFF-Datei, die Sie mit den TIFF-Tools (z.B. **tiff2ps**) in nahezu jedes andere nützliche Format konvertieren können.

2.3.2.3. Farbfaxe - das CFF-Format

Es gibt eine Erweiterung zum Fax-Standard, die die Übertragung von farbigen Dokumenten erlaubt. Sie wird nur selten benutzt, aber da einige User sie trotzdem

benötigten, habe ich den Empfang solcher Dokumente mittlerweile in CapiSuite realisiert.

Das CFF-Format (ich weiß nicht sicher, ob das ein offizieller Name für das Format ist) scheint eine JPEG-Datei mit spezieller Kodierung zu sein. Die meisten Programme, die JPEG-Dateien unterstützen, sollten CFF-Dateien ebenfalls öffnen können. Eventuell müssen Sie die Datei von `.cff` in `.jpg` umbenennen, bevor Sie sie öffnen können.

Leider kenne ich momentan keinen Weg, dieses Format manuell zu erzeugen. Daher unterstützt CapiSuite bis jetzt nur den Empfang solcher Dokumente. Wenn Sie mehr darüber wissen sollten oder den JPEG-Standard gut kennen, dann kontaktieren Sie mich bitte!

2.4. Tutorial: Ein Incoming-Skript schreiben

In diesem Abschnitt werde ich Ihnen Schritt für Schritt zeigen, wie Sie Ihr eigenes Incoming-Skript schreiben können. Wir fangen an, indem wir einfach jeden eingehenden Anruf annehmen und einen Piep abspielen. Das letzte Beispiel ist ein sehr einfacher, aber nützlicher Anrufbeantworter mit Fax-Erkennung und -Empfang.

2.4.1. Grundlagen und ein wirklich dummer Anrufbeantworter

Lassen Sie uns mit einem sehr einfachen Fall anfangen: wir nehmen alle eingehenden Anrufe an, piepen und nehmen etwas auf, sodass wir eine Audio-Datei haben, mit der wir später ein bisschen rumspielen können. Legen Sie zuerst irgendwo ein neues Verzeichnis an, für das `root` Schreibrechte haben muss. Wir brauchen auch eine Test-Audio-Datei, um sie zu verschicken. Lassen Sie uns den Piep nehmen, der mit CapiSuite geliefert wird.

```
mkdir capisuite-examples
chmod 777 capisuite-examples # schreibbar für alle
cd capisuite-examples
cp /usr/local/share/capisuite/beep.la .
```

Vielleicht müssen Sie den Pfad in der letzten Zeile anpassen, damit er zu Ihrer Installation passt.

Kopieren Sie das hier gezeigte Beispiel jetzt in eine Datei namens `example.py` in diesem Verzeichnis. Vergessen Sie nicht, den Pfad `my_path` zu ändern.

Beispiel 2-1. example.py

```
import capisuite❶

my_path="/path/to/the/just/created/capisuite-examples/"❷

def callIncoming(call,service,call_from,call_to):❸
    capisuite.connect_voice(call,10)❹
    capisuite.audio_send(call,my_path+"beep.la")❺
    capisuite.audio_receive(call,my_path+"recorded.la",20,3)❻
    capisuite.disconnect(call)❼
```

Lassen Sie uns das Skript Zeile für Zeile durchgehen:

- ❶ Das capisuite-Modul, das alle CapiSuite-spezifischen Funktionen enthält, wird importiert. Alle CapiSuite-Objekte (Funktionen, Konstanten) in diesem Modul können nun über `capisuite.objectname` angesprochen werden. Sie können auch ein `"from capisuite import *"` machen, das alle Objekte in den aktuellen Namespace einfügt - aber dies wird nicht empfohlen, da sie mit anderen globalen Objekten kollidieren können.

Anmerkung: Das importierte Modul `capisuite` ist nicht als extra Modul verfügbar, sodass Sie dies nicht in einer interaktiven Python-Session tun können. Es wird in das CapiSuite-Binary eingebunden und ist nur in Skripten, die von CapiSuite interpretiert werden, verfügbar.

- ❷ ändern Sie dies bitte auf den echten Pfad, aus dem Sie diese Beispiele aufrufen.
- ❸ Definieren Sie die erforderliche Funktion wie im Abschnitt 2.2.1 beschrieben.
- ❹ Dies ist die erste CapiSuite-Funktion, die wir verwenden: Sie nimmt den wartenden Anruf an. Der erste Parameter teilt CapiSuite mit, welchen Anruf Sie meinen. Dieser Parameter ist für fast alle CapiSuite-Funktionen erforderlich. Ok, wir haben jetzt nur einen Anruf - aber stellen Sie sich ein Incoming-Skript vor, das zur selben Zeit auch noch einen ausgehenden Anruf tätigen will (zum Beispiel, um einen Anruf weiterzuleiten). In diesem Fall wüßte CapiSuite, welchen Anruf Sie meinen - deshalb müssen Sie die übergebene Referenz an alle Funktionen weiterreichen, die mit der Verbindung zu tun haben.

Sie können CapiSuite auch sagen, dass es eine bestimmte Zeit warten soll, bevor es einen Anruf annehmen soll - dafür wird der zweite Parameter benutzt. Diese Skript wartet also 10 Sekunden bevor die Verbindung zum Anrufer hergestellt wird. Glauben Sie nicht, dass dieser Parameter überflüssig ist und dass Sie stattdessen eine Python-Funktion (z.B. `time.sleep()`) aufrufen können, um zu warten. Dies wird für Verzögerungen länger als 4 (oder 8, hängt von Ihrem ISDN-Setup ab) Sekunden nicht funktionieren, da der Anruf abgewiesen wird,

wenn er nicht von einem ISDN-Device "vor-angenommen" wird, indem Ihrem Netz-Provider mitgeteilt wird, dass es klingelt. CapiSuite macht das, wenn nötig - nutzen Sie also bitte diesen Parameter.

- ⑤ Dieser Aufruf sollte ziemlich selbsterklärend sein. Er sendet die Audio-Datei, die unter `beep.la` gespeichert ist.
- ⑥ Nimmt eine Audio-Datei von maximal 20 Sekunden auf - es wird eher aufgehört, wenn mehr als 3 Sekunden Stille erkannt werden.
- ⑦ Last, but not least - wird die Verbindung beendet. Auflegen. Fertig. Es ist vorbei.

Die CapiSuite-Konfiguration muss geändert werden, um das gerade erstellte Skript nutzen zu können. Editieren Sie dazu Ihre `capisuite.conf` und ersetzen Sie den `incoming_script`-Wert durch den Pfad zur Datei, die Sie gerade erstellt haben (siehe Abschnitt 1.2.2) und starten Sie CapiSuite neu.

Probieren Sie's mal aus: Rufen Sie irgendeine Nummer Ihrer ISDN-Karte an - wenn sie am ISDN-Anschluss hängt, geht jede Nummer (MSN) - wenn sie an einer Telefonanlage hängt, müssen Sie eine Nummer wählen, die in der Anlage für die Karte konfiguriert ist.

Sie sollten einen Piep hören und dann können Sie etwas auf diesen primitiven Anrufbeantworter sprechen. Legen Sie bitte nicht auf, bevor das Skript dies macht, da dieser Fall noch nicht behandelt wird. Warten Sie einfach 3 Sekunden, nachdem Sie etwas gesagt haben - es sollte die Verbindung nach dieser Stille-Periode trennen.

Wenn das nicht funktioniert, haben Sie vielleicht einen Fehler beim Kopieren des Skripts gemacht. In diesem Fall können Sie sich das Log und Error-Log von CapiSuite anschauen. Sie finden dies unter `/var/log/capisuite` oder `/usr/local/var/log/capisuite`, wenn Sie den Pfad nicht geändert haben.

Ein guter Trick, um auf Syntaxfehler zu prüfen, ist, Ihre Skripte durch den normalen Python-Interpreter laufen zu lassen. Rufen Sie dazu **python** `/path/to/your/example.py` auf. Natürlich wird er sich über das `import capasuite` beschweren, da dies kein Standard-Python-Modul ist. Aber bevor er dies macht, prüft er die Syntax Ihres Skripts - wenn Sie also irgendeinen *anderen* Fehler bekommen, beheben Sie ihn und probieren Sie es nochmal. Wenn Sie nur

```
Traceback (most recent call last):
  File "../scripts/incoming.py", line 16, in ?
    import capasuite,cs_helpers
ImportError: No module named capasuite
```

bekommen, hat Ihr Skript die korrekte Syntax.

Ich hoffe, Sie haben Ihr Skript jetzt zum Laufen bekommen - wenn nicht, zögern Sie nicht, auf der CapiSuite-Mailingliste zu fragen, *wenn* Sie zuvor ein Python-Tutorial gelesen haben.

Im nächsten Abschnitt werden wir eine Ansage verwenden. Nehmen Sie also einige Worte mit diesem einfachen Skript auf und verschieben Sie die erzeugte Datei `recorded.la` nach `announce.la`.

2.4.2. Verbesserungen in eine brauchbaren (?) Zustand

Gut, es ist wirklich nicht so schön, dass der Anrufer nicht auflegen darf - und es ist schlecht, dass wir alle eingehenden Anrufe annehmen - dadurch nehmen wir vielleicht der Mutter ihre wichtigen Anrufe weg.

Lassen Sie uns das schnell verbessern.

Beispiel 2-2. `example.py`, verbessert

```
import capisuite

my_path="/path/to/the/just/created/capisuite-examples/"

def callIncoming(call,service,call_from,call_to):
    try:❶
        if (call_to=="123"):❷
            capisuite.connect_voice(call,10)
            capisuite.audio_send(call,my_path+"announce.la")❸
            capisuite.audio_send(call,my_path+"beep.la")
            capisuite.audio_receive(call,my_path+"recorded.la",20,3)
            capisuite.disconnect(call)
        else:
            capisuite.reject(call,1)❹
            capisuite.disconnect(call)❺
    except capisuite.CallGoneError:
        capisuite.disconnect(call)❻
```

- ❶ CapiSuite sagt dem Skript, dass die Gegenstelle aufgelegt hat, indem die Exception namens `CallGoneError` ausgelöst wird. Sie sollten deshalb Ihren Code immer in ein `try`-Statement einfassen und die ausgelöste Exception am Ende Ihres Skripts (oder früher, wenn nötig) behandeln. Diese Exception kann durch ein CapiSuite-Kommando ausgelöst werden.
- ❷ Wir schauen uns die angerufene Nummer an (ersetzen Sie bitte 123 durch die Nummer, die CapiSuite akzeptieren soll)...
- ❸ Die Ansage, die wir im letzten Abschnitt aufgenommen haben, wird abgespielt. Wenn Sie das nicht möchten, nehmen Sie einfach eine neue auf und verschieben Sie die Datei `recorded.la` wieder nach `announce.la`.

- ④ Der Anruf wird ignoriert. Der zweite Parameter teilt den genauen Grund für die Ablehnung mit - Sie können den Anruf ignorieren (jedes andere ISDN-Device oder Telefon klingelt für diese Nummer weiter), indem Sie 1 angeben, aktiv die Verbindung beenden, indem Sie 2 angeben, oder irgend einen Fehler-Code, der in der ISDN-Spezifikation enthalten ist (siehe Abschnitt B.1.2 für die verfügbaren Codes).
- ⑤ Sie müssen immer am Ende Ihres Skripts `disconnect` aufrufen, da dieses immer auf das Ende des Anrufs wartet, während `reject` nur die Ablehnung des Anrufs auslöst. Andernfalls erhalten Sie eine Warning im Error-Log.
- ⑥ Dies ist der Exception-Handler für `CallGoneError` - die Exception, die CapiSuite auslöst, wenn der Anruf vom anderen Gesprächspartner beendet wird. Sie sollten hier auch `disconnect` aufrufen, um zu warten bis der Anruf vollständig beendet wurde.

Speichern Sie dies wieder als `example.py` und probieren Sie es aus. Es ist nicht erforderlich, CapiSuite neu zu starten, da alle Skripte jedesmal, wenn sie ausgeführt werden, neu gelesen werden. Jetzt dürfen Sie auch auflegen ;-).

2.4.3. Verwendung vernünftiger Dateinamen

Wir haben immer den selben Namen verwendet, um die aufgenommene Nachricht zu speichern, was natürlich nicht sinnvoll ist. Wir sollten wirklich für jeden neuen Anruf einen neuen Namen wählen. Dies ist nicht so einfach, wie es sich anhört - sie müssen sicherstellen, dass der verwendete Algorithmus auch noch mit mehreren Anrufen, die zur selben Zeit ankommen, funktioniert. Glücklicherweise hatte der hilfsbereite Programmierer von CapiSuite das selbe Problem, sodass wir seinen (hmmm... meinen?) Code verwenden können.

Das Python-Modul `cs_helpers.py` enthält einige hilfreiche Funktionen, die von den Standard-Skripten, die mit CapiSuite geliefert werden, benötigt werden, die aber auch in Ihren eigenen Skripten ganz nützlich sein können. Es enthält die Funktion `uniqueName`, die genau das macht, was wir hier brauchen. Die Syntax lautet:

```
filename=cs_helpers.uniqueName(directory,prefix,suffix)
```

Die Funktion sucht einen neuen eindeutigen Dateinamen im angegebenen `directory`. Der erzeugte Dateiname ist "`prefix-xxx.suffix`", wobei `xxx` die nächste freie Nummer, angefangen bei 0, ist. Die nächste freie Nummer wird in der Datei `prefix-nextnr` vermerkt und der erzeugte Name wird zurück gegeben.

Wir können diesen Aufruf einfach in unser Skript einbauen:

Beispiel 2-3. Verwendung eindeutiger Dateinamen

```
import capisuite,cs_helpers

my_path="/path/to/the/just/created/capisuite-examples/"

def callIncoming(call,service,call_from,call_to):
    try:
        if (call_to=="123"):
            filename=cs_helpers.uniqueName(my_path,"voice","la")
            capisuite.connect_voice(call,10)
            capisuite.audio_send(call,my_path+"announce.la")
            capisuite.audio_send(call,my_path+"beep.la")
            capisuite.audio_receive(call,filename,20,3)
            capisuite.disconnect(call)
        else:
            capisuite.reject(call,1)
    except capisuite.CallGoneError:
        capisuite.disconnect(call)
```

Wenn es Sie interessiert, welche anderen Funktionen in `cs_helpers.py` definiert sind, schauen Sie sich die Referenz unter dem Abschnitt 2.6.4 an.

2.4.4. Automatische Fax-Erkennung und -Empfang

Als letzten Schritt möchte ich Ihnen zeigen, wie Fax-Erkennung und -Empfang funktionieren und wie man vom Sprach- in den Fax-Modus umschaltet.

Hier ist das letzte und komplizierteste Beispiel dieses Abschnitts. Es führt vier neue CapiSuite-Funktionen ein und zeigt, wie die Funktionalität in eine andere Funktion aufgeteilt werden kann, die von `callIncoming` genutzt wird. Es gibt viele Änderungen, die weiter unten beschrieben sind - aber die meisten davon sollten nahezu selbsterklärend sein. Ich glaube daher nicht, dass dieser letzte Schritt zu umfangreich ist. Und Sie wollen ja nicht noch 10 weitere Schritte lesen, stimmt's? ;-)

Beispiel 2-4. Hinzufügen der Fax-Funktionen

```
import capisuite,cs_helpers,os❶

my_path="/path/to/the/just/created/capisuite-examples/"

def callIncoming(call,service,call_from,call_to):
    try:
```

```

if (call_to=="123"):
    filename=cs_helpers.uniqueName(my_path,"voice","la")
    capisuite.connect_voice(call,10)
    capisuite.enable_DTMF(call)❷
    capisuite.audio_send(call,my_path+"announce.la",1)❸
    capisuite.audio_send(call,my_path+"beep.la",1)
    capisuite.audio_receive(call,filename,20,3,1)
    dtmf=capisuite.read_DTMF(call,0)❹
    if (dtmf=="X"):❺
        if (os.access(filename,os.R_OK)):❻
            os.unlink(filename)
            faxIncoming(call)❼
            capisuite.disconnect(call)
    else:
        capisuite.reject(call,1)
except capisuite.CallGoneError:
    capisuite.disconnect(call)

def faxIncoming(call):
    capisuite.switch_to_faxG3(call,"+49 123 45678","Test headline")❸
    filename=cs_helpers.uniqueName(my_path,"fax","sff")
    capisuite.fax_receive(call,filename)❹

```

- ❶ In diesem Beispiel brauchen wir zum ersten Mal ein normales Python-Modul. Das `os`-Modul enthält Funktionen für alle Arten von Betriebssystem-Diensten und wird hier benötigt, um eine Datei zu löschen.
- ❷ DTMF ist die Abkürzung für Dual Tone Multi Frequency. Dies sind die Töne, die generiert werden, wenn Sie die Ziffern auf Ihrem Telefon drücken, und werden normalerweise genutzt, um zu wählen. Sie werden auch von modernen Fax-Geräten gesendet, bevor die Übertragung beginnt. Deshalb können die selben Funktionen zur Erkennung von gedrückten Tasten und von Fax-Geräten verwendet werden.

Bevor irgendein DTMF von CapiSuite erkannt wird, muss die entsprechende Funktion mit `enable_DTMF` aktiviert werden.
- ❸ Alle Audio-Sende- und -Empfangsfunktionen unterstützen das Abbrechen, wenn ein DTMF-Ton erkannt wird. Dies wird aktiviert, indem eine "1" als letzter Parameter übergeben wird. Es verhindert auch, dass die Funktion startet, wenn ein DTMF-Zeichen *zuvor* erkannt, aber noch nicht vom Skript gelesen wurde.
- ❹ CapiSuite speichert alle empfangenen DTMF-Signale in einem Puffer, von wo sie jederzeit wieder gelesen werden können. Gelesen werden sie von `read_DTMF`, das auch den Puffer löscht. Sie gibt alle empfangenen Zeichen in

einem String zurück. Wenn der Anrufer "3","5","*" drückt, bekommen Sie "35*".

Die 0 weist CapiSuite an, nicht auf DTMF-Signale zu warten - wenn keine vorhanden sind, wird einfach ein leerer String zurück gegeben. Es ist auch möglich zu sagen, dass eine bestimmte Zeit gewartet werden soll oder bis eine bestimmte Anzahl von Signalen empfangen wurden.

Anmerkung: Bitte beachten Sie, dass es nicht erforderlich ist, nach jeder Audio-Sende- oder -Empfangsfunktion auf empfangene DTMF zu prüfen. Aktivieren Sie einfach den DTMF-Abbruch in allen Befehlen in einem Block und prüfen Sie auf empfangene Töne nach dem ganzen Block.

- ⑤ Fax-Geräte senden einen speziellen Ton, der vom CAPI als "x" repräsentiert wird. Wenn Sie also den String "X" empfangen, ruft ein Fax-Gerät an und wir sollten unsere Fax-Routinen starten.
- ⑥ Möglicherweise war die Ansage so kurz, dass die Aufnahme schon begonnen hat, bevor das Fax erkannt wurde. Wir wollen keine Datei speichern, die nur das Fax-Piepen enthält. Deshalb testen wir, ob sie erzeugt wurde (`os.access` prüft die Existenz einer Datei) und löschen sie wenn nötig, indem wir `os.unlink` aufrufen.
- ⑦ Fax-Behandlung wurde in einer separaten Funktion realisiert, die hier aufgerufen wird.
- ⑧ Bis jetzt ist diese Verbindung im Sprach-Modus (der von `connect_voice` gesetzt wurde). Wenn wir nun ein Fax empfangen wollen, müssen wir den Modus auf Fax umstellen. Dies wird mit `switch_to_faxG3` gemacht. Da das Fax-Protokoll einige zusätzliche Parameter benötigt, müssen sie hier angegeben werden. Der erste String ist die sog. *Fax-Geräteerkennung*, die an das anrufende Fax geschickt und die im Protokoll angezeigt wird, während der zweite eine *Fax-Titelzeile* ist. Diese Titelzeile wird hauptsächlich beim Versenden von Faxen verwendet. Um ehrlich zu sein, ich persönlich weiss nicht, ob es irgendeinen Sinn macht diese anzugeben, wenn Sie nur Faxe empfangen möchten. Aber es schadet sicher auch nicht ;-). Wenn das jemand sicher weiss, sage er's mir bitte.

Anmerkung: Wenn Sie eine eigene Nummer nur für Fax-Zwecke nutzen möchten, sollten Sie *nicht* `switch_to_faxG3` benutzen. Benutzen Sie stattdessen `connect_faxG3`.

- ⑨ Nachdem die Verbindung erfolgreich in den Fax-Modus umgeschaltet wurde, können wir schließlich das Fax-Dokument empfangen. Die benutzte Funktion

`fax_receive` bekommt einen neuen Namen, der wieder von `cs_helpers.uniqueName` erzeugt wird wie oben.

Glückwunsch. Sie haben mein kleines Tutorial beendet. Jetzt sind Sie dran - Sie können ein bisschen mit dem erstellten Skript herumspielen und versuchen, es noch weiter zu vervollständigen. Es gibt noch viel zu tun - empfangene Anrufe per E-Mail an den Benutzer schicken, Verbindungen protokollieren, ... Wenn Sie dieses Skript vervollständigen wollen, ist der Abschnitt 2.7 hilfreich. Sie können auch hier weiter lesen, um einen kurzen Blick auf das Idle-Skript zu werfen, gefolgt von einem schnellen Überblick über die Struktur der Standard-Skripte, die mit CapiSuite geliefert werden.

2.5. Beispiel eines Idle-Skripts

Nachdem wir gesehen haben, wie eingehende Anrufe behandelt werden, folgt nun eine sehr kurze Einführung, wie ausgehende Anrufe mit Hilfe des Idle-Skripts initiiert werden.

Wie bereits erwähnt, wird das Idle-Skript in regelmäßigen Abständen von CapiSuite aufgerufen, um irgendwo nach gespeicherten Aufträgen zu suchen und diese dann zu verschicken.

Das hier vorgestellte Beispiel sucht nach einer Datei `job-xxxx.sff` im Beispiel-Verzeichnis, das wir im letzten Abschnitt erzeugt haben. Diese Datei wird an das durch `xxxx` angegebene Ziel gefaxt. Wenn Sie kein gültiges Ziel haben, an das Sie zum Testen Faxe schicken können, warum nehmen Sie nicht einfach CapiSuite als Absender und Empfänger gleichzeitig? Ersetzen Sie in diesem Fall `xxxx` durch die Nummer, die Ihr Incoming-Skript behandelt. Dies funktioniert nicht, wenn Ihre ISDN-Karte nicht zwei Fax-übertragungen parallel durchführen kann (einige alte AVM-B1-Karten haben z.B. diese Beschränkung).

Wir brauchen jetzt für unsere Tests ein oder mehrere Fax-Dateien im SFF-Format. Erzeugen Sie bitte welche mit einem Namen wie der oben gezeigte. Wenn Sie nicht wissen, wie Sie dies tun sollen, schauen Sie bitte unter dem Abschnitt 2.3.2.1 nach.

Wenn ich ein CapiSuite-Skript entwickeln möchte, aber nicht sicher bin, wie es geht, fange ich oft mit einem normalen Skript an, das ich ohne CapiSuite testen kann. Lassen Sie uns also zuerst ein Skript erstellen, das die Dateien durchsucht und die Empfängernummern extrahiert. Wenn das funktioniert, können wir weiter machen und die CapiSuite-spezifischen Aufrufe einfügen.

Beispiel 2-5. `idle_example.py`

```
import os, re
```

```
my_path="/path/to/your/capisuite-examples/"

files=os.listdir(my_path)❷
files=filter (lambda s: re.match("job-.*\.sff",s),files)❸

for job in files:❹
    destination=job[4:-3]❺ # Hmm.. Ist das richtig?
    print "found",job,"to destination",destination
```

- ❶ Wir kennen das `os`-Modul bereits. `re` stellt Funktionen zum Suchen nach regulären Ausdrücken zur Verfügung. Wenn Sie nicht wissen, was reguläre Ausdrücke sind, lesen Sie bitte z.B. die Python-Dokumentation des `re`-Moduls oder eine andere Dokumentation dazu. Sie sind zu kompliziert, um sie hier zu erklären.
- ❷ `os.listdir` gibt die Dateien eines Verzeichnisses als Liste zurück.
- ❸ Diese Zeile ist ein bisschen trickreicher. Sie filtert alle Dateinamen aus der Liste heraus, die nicht zur Regel mit *"job-"* beginnen, dann eine beliebige Anzahl von Zeichen, mit *".sff"* aufhören passen. Dies wird mit der `filter`-Funktion gemacht. Die Funktion erwartet den Namen einer Funktion, die die Regel als ersten Parameter prüft und die zu filternde Liste der Dateien als zweiten.

Wir können jetzt eine neue Funktion definieren und ihren Namen hier verwenden, aber das `lambda`-Schlüsselwort erlaubt eine viel elegantere Lösung: Es definiert eine "namenlose Funktion" mit dem Parameter `s`. Der Funktionsrumpf folgt direkt danach und besteht aus einem Aufruf von `re.match`, der prüft, ob der übergebene String `s` auf den Ausdruck passt.
- ❹ Iterieren über alle gefundenen Dateinamen.
- ❺ Das Ziel wird aus dem gegebenen Dateinamen mit Hilfe von String-Indizes extrahiert.

Speichern Sie das Skript jetzt als `idle_example.py` in unserem Beispielverzeichnis und lassen Sie es laufen, indem Sie **python idle_example.py** aufrufen.

Wenn Sie SFF-Dateien mit den richtigen Namen angelegt haben, sollten Sie nun Zeile für Zeile angezeigt werden. Aber... Offensichtlich funktioniert irgendetwas hier nicht richtig. Das Ziel enthält den ".". Ich habe tatsächlich einen Fehler beim indizieren des Strings gemacht. Es sollte `destination=job[4:-4]` anstatt `[4:-3]` heißen. ändern wir das also und probieren es nochmal. Es sollte jetzt funktionieren. Das ist der Grund, warum ich solche Skripte lieber zuerst außerhalb von CapiSuite schreibe. Das Debuggen geht so viel schneller...

Da wir jetzt wissen, dass die grundlegenden Teile funktionieren, können wir die richtigen Kommunikationsfunktionen hinzufügen.

Speichern Sie dieses Beispiel bitte wieder als `idle_example.py` in Ihrem Beispielverzeichnis.

Beispiel 2-6. `idle_example.py`, Version für CapiSuite

```
import os,re,capisuite

my_path="/path/to/your/capisuite-examples/"
my_number="678"❶
my_stationID="+49 123 45678"
my_headline="example headline"

def idle(capi):❷
    files=os.listdir(my_path)
    files=filter(lambda s: re.match("job-.*\.sff",s),files)

    for job in files:
        destination=job[4:-4]
        capisuite.log("sending "+job+" to destination "+destination,1)❸
        try:
            (call,result)=capisuite.call_faxG3(capi,1,my_number,destination,
                60,my_stationID,my_headline)❹
            if (result!=0):❺
                capisuite.log("job "+job+" failed at call setup with reason "
                    +str(hex(result)),1)
                os.rename(my_path+job,my_path+"failed-"+job)❻
                return❼
            capisuite.fax_send(call,my_path+job)❸
            (result,resultB3)=capisuite.disconnect(call)❾
        except capisuite.CallGoneError:
            (result,resultB3)=capisuite.disconnect(call)

        if (result in (0,0x3400,0x3480,0x3490) and resultB3==0):(10)
            capisuite.log("job "+job+" was successful",1)
            os.rename(my_path+job,my_path+"done-"+job)
            return
        else:
            capisuite.log("job "+job+" failed during send with reasons "
                +str(hex(result))+","+str(hex(resultB3)),1)
            os.rename(my_path+job,my_path+"failed-"+job)
```

- ❶ Einige Parameter für das Senden des Faxes werden hier gesetzt. `my_number` ist Ihre eigene Nummer, die zum Senden des Faxes verwendet wird.
- `my_stationID` ist die Fax-Geräteerkennung, die an die Gegenstelle übermittelt

und auf der gesendeten Fax-Seite angezeigt wird. Es sind nur Ziffern und das "+" erlaubt. Sie können außerdem in `fax_headline` einen kurzen Text definieren, der in der Fax-Kopfzeile angezeigt wird.

- ② Wie im Abschnitt 2.2.2 erklärt, müssen Sie eine Funktion namens `idle` definieren, die dann in regelmäßigen Abständen von CapiSuite ausgeführt wird. Deshalb wurde aller Code in diese Funktion verschoben.
- ③ Wir können keine Meldungen nach `stdout` ausgeben, da das Skript im Kontext eines Daemons läuft. Deshalb stellt CapiSuite Funktionen bereit, um Einträge in den CapiSuite-Log-Dateien zu erzeugen. `log` erwartet mindestens zwei Parameter: die Meldung und einen Log-Level. Dieser Level entspricht der Log-Level-Einstellung in der globalen CapiSuite-Konfiguration (siehe Abschnitt 1.2.2). Wenn der Level der Meldung *kleiner oder gleich* verglichen mit dem Level in der Konfiguration ist, wird sie in die Logs geschrieben. Sie können also Meldungen zu Debug-Zwecken einfügen, die im Normalbetrieb nicht in die Logs geschrieben werden, indem Sie Level größer als 1 verwenden.
- ④ Diese Funktion initiiert einen ausgehenden Anruf, bei dem der Fax-Service verwendet wird. Die Parameter sind:
 - Referenz auf das von CapiSuite erhaltene CAPI-Objekt (Parameter für `idle`).
 - Die Nummer des Controllers, der für ausgehende Anrufe verwendet wird. Der erste Controller hat immer die Nummer "1".
 - Die eigene Nummer, die für ausgehende Anrufe verwendet wird
 - Die Empfängernummer des Anrufs
 - Maximale Zeit in Sekunden, die auf einen erfolgreichen Verbindungsaufbau gewartet werden soll
 - Die Fax-Geräteerkennung
 - Die Fax-Kopfzeile

Die Funktion gibt ein Tuple zurück, das eine Referenz auf den erzeugten Anruf und einen Fehler-Code enthält.

- ⑤ Dieser Block prüft, ob der Verbindungsaufbau erfolgreich war. Eine detaillierte Beschreibung der möglichen Fehler-Codes finden Sie im Abschnitt 2.7. 0 bedeutet "alles war OK, die Verbindung wurde aufgebaut".
- ⑥ Wenn der Anruf nicht erfolgreich war, wird die Fax-Datei umbenannt, um zu verhindern, dass die selbe Datei immer wieder verschickt wird.
- ⑦ Vergessen Sie nicht, die `idle`-Funktion zu verlassen, wenn die Verbindung nicht aufgebaut werden konnte!
- ⑧ Ein weiteres sehr einfaches CapiSuite-Kommando: verschickt die gegebene Datei als Fax-Dokument.

⑨ Wir haben bislang die Gründe, *warum* eine Verbindung beendet wurde, ignoriert. Jetzt müssen wir sie analysieren, weil wir wissen müssen, ob die Datei erfolgreich übertragen wurde. Deshalb gibt `disconnect` ein Tuple zurück, das den physikalischen und logischen Fehler-Code enthält. Jede ISDN-Verbindung enthält eine physikalische und (mindestens) eine logische Verbindung. Man kann sich die physikalische Verbindung als "den Draht" vorstellen, der uns mit dem Empfänger verbindet, während sich die logische Verbindung auf das Fax-Protokoll bezieht, das diesen "Draht" verwendet. Sie müssen sich beide Werte ansehen, um entscheiden zu können, ob alles OK war.

(10) Erlaubte Werte für den physikalischen Verbindungsabbau sind `0,0x3400,0x3480` und `0x3490`. Diese bedeuten alle "kein Fehler aufgetreten, Verbindung wurde normal beendet". Der logische Wert darf nur `0` sein, wenn alles OK ist. Weitere Informationen zu den Fehler-Codes finden Sie im Abschnitt 2.7.

Nachdem Sie die Datei gespeichert und die Standard-Werte an Ihre eigene Konfiguration angepasst haben, ändern Sie bitte den Wert von `idle_script` in der CapiSuite-Konfiguration, damit er auf dieses Skript verweist (wie im Abschnitt 1.2.2 beschrieben).

Starten Sie CapiSuite neu und beobachten Sie die Logs. Einige Minuten später sollten die Dateien `job-xxx.sff` verschickt worden und entweder in `done-job-xxx.sff` oder `failed-job-xxx.sff` umbenannt worden sein. Wenn der Auftrag fehlgeschlagen ist, schauen Sie bitte in das Error-Log und prüfen die Fehler-Codes, die im Abschnitt 2.7 und Anhang B beschrieben sind.

Hoffentlich hat Ihnen dieses Tutorial geholfen zu verstehen, wie Sie Ihre eigenen Skripte schreiben können. Machen Sie bitte weiter, indem Sie die Beispiele oder die Dateien, die mit CapiSuite geliefert werden, modifizieren (lesen Sie den Abschnitt 2.6 zuvor). Sie finden eine komplette Referenz der verfügbaren Kommandos im Abschnitt 2.7.

Wenn Sie Schwierigkeiten haben, Ihre eigenen Skripte zum Laufen zu bringen, nutzen Sie bitte die CapiSuite-Mailingliste. Und vergessen Sie nicht, Spaß dabei zu haben. ;-)

2.6. Struktureller Überblick über die Standard-Skripte

2.6.1. `incoming.py`

Das Incoming-Skript behandelt alle eingehenden Verbindungen. Es liest zwei

Konfigurationsdateien, die alle benötigten Daten enthalten, die detailliert im Abschnitt 1.3.3 beschrieben wurden. Die Gesamtstruktur wird hier beschrieben und gibt Ihnen einen Überblick, wie es implementiert ist.

Zuerst (nachdem einige benötigte Module importiert wurden) wird die benötigte Funktion `callIncoming` definiert, die bei Bedarf andere Funktionen aufruft.

2.6.1.1. Funktion `callIncoming`

Diese Funktion beginnt mit einem Aufruf von `cs_helpers.readConfig`, um die Konfiguration zu lesen. Sie iteriert dann über alle Abschnitte, die die konfigurierten Benutzer repräsentieren (außer `GLOBAL`), um zu prüfen, ob die angerufene Nummer zu irgendeinem Benutzer gehört. Wenn eine Übereinstimmung gefunden wurde, wird der Benutzer und der definierte Service unter `curr_user` und `curr_service` gespeichert.

Wenn keine Übereinstimmung gefunden wurde (`curr_user` ist leer), wird der Anruf abgewiesen und die Funktion verlassen. Andernfalls wird das Verzeichnis für eingehende Faxe oder Sprach-Daten ermittelt und erzeugt, falls es noch nicht existiert.

Zuletzt wird noch ein Log-Eintrag erzeugt, der Anruf mit dem richtigen Service (Fax oder Sprache) angenommen und entweder `faxIncoming` oder `voiceIncoming` aufgerufen. Die Funktion definiert außerdem einen Exception-Handler für `capisuite.CallGoneError`.

2.6.1.2. Funktion `faxIncoming`

`faxIncoming` ist ziemlich einfach: sie erzeugt einen eindeutigen Dateinamen, ruft `capisuite.fax_receive` auf, baut die Verbindung ab und protokolliert den Grund des Verbindungsabbaus. Danach prüft sie, ob wirklich ein Fax erfolgreich empfangen wurde (d.h. ob die Datei existiert). Wenn ja, erzeugt sie eine Beschreibungsdatei dafür, ändert den Eigentümer der Datei auf den richtigen Benutzer und verschickt die Datei als Mail.

2.6.1.3. Funktion `voiceIncoming`

`voiceIncoming` hat viel mehr Aufgaben zu erledigen, wie Fax-Erkennung und Wechsel in den Fax-Modus, die Fernabfrage starten usw.

Sie fängt mit dem Bestimmen des zu verwendenden Verzeichnisses und dem Erzeugen eines eindeutigen Dateinamens an. Außerdem wird die PIN für die Fernabfrage in einer privaten Variable gespeichert. Es gibt jetzt zwei Möglichkeiten: der Benutzer hat bereits eine eigene Ansage - in diesem Fall wird

sie jetzt abgespielt. Andernfalls wird eine vordefinierte Ansage, die einen allgemeinen Text und die angerufene Nummer enthält, abgespielt. Wenn das Aufnehmen einer Nachricht nicht deaktiviert wurde (indem `voice_action` auf `None` gesetzt wurde), start die Aufnahme jetzt nach dem Piep.

Alle bisher benutzten Aufrufe von `audio_send` und `audio_receive` hatten DTMF-Abbruch aktiviert, sodass das Skript alle Anrufe "überspringt", nachdem ein DTMF-Signal erkannt wurde. Danach wird `read_DTMF` verwendet, um zu prüfen, ob so ein Signal erkannt wurde. "x" steht für den Fax-Ton und schaltet auf das Fax-Protokoll um und ruft `faxIncoming` auf. Alle anderen empfangenen Signale werden als Teil der PIN für die Fernabfrage interpretiert. Deshalb wird eine Schleife durchlaufen, die 3 Sekunden nach jedem Ton auf den nächsten wartet. Wenn eine gültige PIN eingegeben wird, startet sie `remoteInquiry`. Nach drei falschen Versuchen bricht sie die Verbindung ab.

Nach dem Verbindungsabbau und dem Protokollieren wird ein eine Beschreibungsdatei geschrieben (wenn die aufgenommene Datei existiert), bei beiden Dateien wird der Eigentümer auf den richtigen Benutzer geändert und die aufgenommene Nachricht wird an ihn/sie gemailt.

2.6.1.4. Funktion `remoteInquiry`

`remoteInquiry` startet mit dem Erzeugen einer Lock-Datei und dem Anfordern einer exklusiven Sperre darauf, um zu verhindern, dass zwei Fernabfragen für den selben Benutzer parallel laufen. Wenn die Sperre nicht angefordert werden kann, wird eine Fehlermeldung abgespielt und die Funktion wird verlassen. Wenn das Sperren erfolgreich war, wird eine Liste der aufgenommenen Sprachanrufe erstellt, indem das Benutzerverzeichnis aufgelistet, gefiltert und sortiert wird. Jetzt wird eine Datei namens `last_inquiry` gelesen, wenn sie existiert. Sie enthält die Nummer der zuletzt angehörten Nachricht. Mit dieser Information können die alten Nachrichten in eine separate Liste heraus gefiltert werden, sodass sich der Anrufer zuerst die Nachrichten anhören kann, die er noch nicht kennt.

Die Anzahl neuer Nachrichten wird angesagt, gefolgt von einem kleinen Menü, in dem der Anrufer auswählen kann, ob er entweder eine Ansage aufnehmen oder eine aufgezeichnete Nachricht anhören möchte. Wenn er die Aufnahme einer Ansage auswählt, wird die Funktion `newAnnouncement` aufgerufen, andernfalls wird mit `remoteInquiry` weiter gemacht.

Jetzt läuft eine Schleife zuerst über alle neuen und dann über alle alten Nachrichten. Sie beginnt, indem sie dem Anrufer mitteilt, wieviele Nachrichten gefunden wurden. Dann werden alle Nachrichten abgespielt, indem die folgenden Schritte für jede wiederholt werden:

- Lesen der Beschreibungsdatei der aktuellen Nachricht

- Abspielen eines Informationsblocks, der die aktuelle Nummer der Nachricht, den Absender, den Empfänger sowie Datum und Uhrzeit des Anrufs enthält.
- Abspielen der Nachricht
- Anbieten eines Menüs, in dem der Anrufer zur nächsten oder vorigen Nachricht gehen kann, die aktuelle Nachricht wiederholen oder löschen kann.

Am Ende wird der Anrufer informiert, dass keine weiteren Nachrichten vorhanden sind und dass die Verbindung beendet wird. Danach wird die Lock-Datei wieder freigegeben und gelöscht.

2.6.1.5. Funktion `newAnnouncement`

`newAnnouncement` präsentiert dem Anrufer zuerst einige Instruktionen. Danach wird die neue Ansage in eine temporäre Datei aufgezeichnet. Um dem Benutzer die Möglichkeit zu geben, die Ansage zu überprüfen, wird sie nun noch einmal abgespielt, gefolgt von einem Menü, das es dem Benutzer erlaubt, die Ansage zu speichern oder die Aufnahme zu wiederholen. Wenn der Benutzer sie speichern will, wird sie von der temporären Datei nach `announcement.la` im Sprach-Verzeichnis des Benutzers verschoben und der Eigentümer auf ihn geändert. Der Anruf wird abgeschlossen mit einer Bestätigung an den Anrufer, dass die Ansage erfolgreich gespeichert wurde.

2.6.2. `idle.py`

Das Idle-Skript ist verantwortlich für das Einsammeln der Jobs aus den Send-Queues (wo sie von `capisuitifax` gespeichert werden) und deren Versendung an die angegebenen Empfänger. Es liest seine Konfiguration ebenfalls aus den Dateien, die im Abschnitt 1.3.3 beschrieben wurden.

2.6.2.1. Funktion `idle`

Nach dem Lesen der Konfiguration durch den Aufruf von `cs_helpers.readConfig` und dem Prüfen der Existenz der benötigten Archiv-Verzeichnisse wird die Benutzerliste aus der Liste der vorhandenen Abschnitte erstellt.

Für jeden Benutzer, der ein gültiges Fax-Setup besitzt (andernfalls wird dieser Benutzer übersprungen), wird die Send-Queue überprüft. Wenn die benötigten Queue-Verzeichnisse nicht existieren, werden sie erzeugt. Danach wird eine Liste namens `files` mit den Namen aller Dateien in der Send-Queue erstellt und gefiltert, damit sie nur Fax-Aufträge enthält.

Für jeden gefundenen Auftrag wird ein Sicherheits-Check gemacht, um zu prüfen, ob er vom richtigen Benutzer erzeugt wurde. Wenn dieser Check erfolgreich war, wird eine Lock-Datei erzeugt und eine Sperre darauf angefordert. Dies verhindert, dass der **capisuitefax**-Befehl einen Auftrag abbricht, während er übertragen wird. Danach wird die Existenz der Datei überprüft (vielleicht wurde der Auftrag abgebrochen, bevor wir eine Sperre bekommen konnten?).

Jetzt wird die Beschreibungsdatei für diesen Auftrag gelesen und die Startzeit wird überprüft. Wenn sich noch nicht erreicht ist, macht das Skript mit dem nächsten Auftrag weiter. Andernfalls werden einige Parameter aus der Konfiguration genommen und ein Log-Eintrag erzeugt. Die Datei wird durch den Aufruf von `sendfax` übertragen. Das Ergebnis wird gespeichert und protokolliert. Wenn der Auftrag erfolgreich war, wird er ins Done-Verzeichnis verschoben und eine Bestätigung an den Benutzer gemailt. Wenn er nicht erfolgreich war, wird die Verzögerungszeit aus der Konfiguration bestimmt und die neue Startzeit wird berechnet, indem die alte Startzeit um das Intervall erhöht wird. Ein Zähler für die Versuche wird erhöht und die Beschreibungsdatei wird mit den neuen Werten neu geschrieben. Wenn die Anzahl der Versuche das konfigurierte Maximum überschreitet, wird der Auftrag in das Failed-Verzeichnis verschoben und der Fehler wird dem Benutzer per Mail berichtet.

Zuletzt wird die Lock-Datei entsperrt und gelöscht.

2.6.2.2. Funktion `sendfax`

Diese Funktion realisiert den Sendeprozess. Nachdem die zu verwendende MSN entweder aus der `outgoing_msn`-Einstellung oder aus der `fax_numbers`-Liste bestimmt wurde, wird ein Anruf zum Ziel initiiert. Wenn dies schief geht, wird die Funktion verlassen; andernfalls wird die Datei gesendet und die Verbindung beendet.

2.6.2.3. Funktion `movejob`

Dies ist eine kleine Hilfsfunktion, die zum Verschieben eines Auftrags und der zugehörigen Beschreibungsdatei in ein anderes Verzeichnis verwendet wird.

2.6.3. `capisuitefax`

capisuitefax erlaubt das Einreihen von Fax-Aufträgen, das Auflisten der aktuellen Queue und das Abbrechen von Aufträgen. Es wird nicht direkt vom CapiSuite-System benutzt - es ist ein Frontend für das Send-Queue-Verzeichnis des

Benutzers. Es hat verschiedene Kommandozeilenoptionen - eine Beschreibung finden Sie im Abschnitt 1.4.3.

Es werden zunächst drei Hilfsfunktionen definiert. *usage* gibt einen kleinen Hilfetext aus, wenn "--help" oder "-h" als Parameter angegeben wurde oder wenn ein unbekannter Parameter übergeben wurde. *showlist* erstellt ein Listing des Send-Queue-Verzeichnisses des Benutzers und gibt es schön formatiert als Tabelle aus. *abortjob* entfernt einen Auftrag aus der Queue. Sie macht dies sicher, indem eine Lock-Datei verwendet wird, damit es keine Konflikte mit dem Sendeprozess gibt.

Der Haupt-Code dieses Skripts prüft zuerst die angegebenen Kommandozeilenoptionen. Es setzt verschiedene Variablen auf die übergebenen Werte. Nach einigen Gültigkeitprüfungen der Optionen, der Benutzerrechte für das Versenden von Faxen und der Existenz der erforderlichen Verzeichnisse, erfüllt es die gewünschte Aufgabe. Entweder wird *listqueue* aufgerufen, um ein Listing der aktiven Aufträge anzuzeigen, *abortjob*, um einen Job abzubrechen oder die angegebenen Dateien werden verarbeitet und in die Queue geschrieben.

Um einen Auftrag zu verarbeiten, wird seine Existenz und sein Format geprüft. Momentan ist nur PostScript erlaubt. Der CapiSuite-Kern selbst unterstützt nur das SFF-Format. Deshalb werden die Dateien mit **ghostscript** von PostScript nach SFF konvertiert. Zuletzt wird die Beschreibungsdatei für den Auftrag erstellt, die die angegebenen Parameter wie die Empfängernummer enthält.

2.6.4. cs_helpers.py

Das `cs_helpers.py`-Skript enthält viele kleine Hilfsfunktionen, die in anderen Skripten verwendet werden. Diese sind:

`readConfig`

Liest entweder die Konfigurationsdateien, die im Abschnitt 1.3.3 beschrieben sind oder eine beliebige Konfigurationsdatei wie die Beschreibungsdateien, die zu jeder empfangenen Datei oder zu jedem ausgehenden Auftrag gehören.

`getOption`

Liest eine Option aus dem angegebenen Benutzerabschnitt oder nimmt den globalen Abschnitt, wenn keiner gefunden wurde.

`getAudio`

Liest eine Audio-Datei aus dem Benutzerverzeichnis oder dem globalen CapiSuite-Verzeichnis.

`uniqueName`

Konstruiert einen neuen Dateinamen im angegebenen Verzeichnis aus einem gegebenen Präfix & Suffix und fügt einen Zähler hinzu. Siehe auch Abschnitt 2.4.3.

`sendMIMEMail`

Sendet eine E-Mail mit Attachment an den angegebenen Benutzer. Sie unterstützt auch automatische Format-Konvertierung SFF -> PDF und invertiert A-Law -> WAV.

`sendSimpleMail`

Sendet eine normale E-Mail ohne Attachment an den angegebenen Benutzer.

`writeDescription`

Erzeugt eine Beschreibungsdatei, die später von `readConfig` gelesen werden kann.

`sayNumber`

Unterstützt die Ansage einer Nummer, indem verschiedene Wave-Fragmente verwendet werden. Funktioniert momentan nur für deutsche Ausgabe.

Eine detaillierte Beschreibung aller Funktionen und ihrer Verwendung finden Sie in den Skripten selbst. Dort gibt es Kommentare, die jede Funktion im Detail beschreiben.

2.7. CapiSuite-Befehlsreferenz

CapiSuite stellt ein internes Python-Modul namens `capisuite` zur Verfügung, das ganz normal mit `import capisuite` importiert werden kann. Intern bedeutet, dass es in das CapiSuite-Binary einkompiliert ist und dass es nur gefunden wird, wenn CapiSuite das Skript interpretiert.

Eine komplette Referenz aller Funktionen dieses Moduls wird automatisch aus den CapiSuite-Sourcen generiert, sodass Sie es im Referenzhandbuch lokal unter `./reference/group__python.html` oder online unter http://www.capisuite.de/reference/group__python.html finden.

Da es keinen Sinn macht, die Informationen hier doppelt zu halten, schauen Sie sich bitte dort an.

Anmerkung: Diese Funktionen sind intern in C implementiert, sodass das Referenz-Dokument die C-Funktionsköpfe anstelle der Köpfe, wie sie in

Kapitel 2. Users Guide

Python definiert würden, zeigt. Ignorieren Sie daher die dort gezeigten Funktionsköpfe und schauen Sie sich nur die Beschreibungen und die Parameter unter *args* an. Wenn dies zu verwirrend ist, sagen Sie mir dies bitte. Vielleicht finde ich irgendwann eine bessere Lösung, um das Dokument automatisch zu generieren...

Anhang A. Danksagungen

CapiSuite wurde als Diplomarbeit im Wintersemester 2002/03 begonnen. Ich möchte den folgenden Leuten für ihre Hilfe danken:

- Karsten Keil von SUSE Linux AG (mein Betreuer der Arbeit) für seine unschätzbare Unterstützung und Geduld beim Beantworten meiner vielen ISDN-Fragen, beim Testen und für seine Vorschläge bzgl. der Architektur von CapiSuite
- Prof. Dr. Wolfgang Jürgensen vom UAS Landshut (mein Betreuer vom UAS) für seine Hilfe bei ISDN-Fragen während der Arbeit und dass er mir vorher in seiner Vorlesung die ISDN-Grundlagen beigebracht hat
- Prof. Dr. Peter Scholz vom UAS Landshut (zweiter Betreuer vom UAS) für seine Unterstützung und seine Vorschläge
- meiner Freundin Claudia und ihrer Schwester Bethina für das Korrekturlesen meiner Arbeit
- Peter Reinhart von SUSE für das Korrekturlesen meiner Arbeit
- vielen Kollegen bei SUSE für ihre Hilfe bei technischen Problemen, insbesondere Andreas Jaeger, Andreas Schwab, Thorsten Kukuk und Andi Kleen
- Achim Bohnet, weil er der erste aus der Community war, der versucht hat, die CVS-Version zu kompilieren und der einige Verbesserungsvorschläge gemacht hat

Anhang A. Danksagungen

Anhang B. CAPI 2.0 Fehler-Codes

Die hier verwendete CAPI-Schnittstelle hat ihre eigene Kodierung der Standard-ISDN-Fehler-Codes. Die meisten Fehler, die im Abschnitt B.2 beschrieben werden, sind nur für Entwickler des CapiSuite-Kerns interessant. Als Anwender brauchen Sie nur die Codes wie Abschnitt B.1, da sie in den Python-Funktionen von CapiSuite wie `capisuite.disconnect` verwendet werden.

Sie finden weiter unten eine Liste aller Codes und eine kurze Beschreibung. Eine detaillierte Beschreibung der CAPI-Codes finden Sie in der CAPI-Spezifikation, die Sie unter <http://www.capi.org> bekommen.

Alle Nummern sind *hexadecimal* angegeben!

B.1. CAPI-Fehler, die Verbindungsprobleme beschreiben

Alle hier beschriebenen Fehler deuten auf Verbindungsprobleme hin. Diese Fehler sind auch für Skript-Schreiber interessant, da sie von einigen CapiSuite-Python-Funktionen zurück gegeben werden. Siehe Abschnitt 2.7 für weitere Details.

B.1.1. Protokoll-Fehler

Protokoll-Fehler deuten auf Übertragungsfehler hin. Es werden hier nur Meldungen des transparenten (Sprache) und des Fax-Protokolls, die von CapiSuite gesprochen werden, angezeigt.

- 0 - Normal call clearing, no error
- 3301 - Protocol error layer 1 (broken line or B-channel removed by signalling protocol)
- 3302 - Protocol error layer 2
- 3303 - Protocol error layer 3
- 3304 - Another application got that call
- 3311 - T.30 (fax) error: Connection not successful (remote station is not a G3 fax device)
- 3312 - T.30 (fax) error: Connection not successful (training error)

Anhang B. CAPI 2.0 Fehler-Codes

- 3313 - T.30 (fax) error: Disconnect before transfer (remote station doesn't support transfer mode, e.g. wrong resolution)
- 3314 - T.30 (fax) error: Disconnect during transfer (remote abort)
- 3315 - T.30 (fax) error: Disconnect during transfer (remote procedure error)
- 3316 - T.30 (fax) error: Disconnect during transfer (local transmit data underflow)
- 3317 - T.30 (fax) error: Disconnect during transfer (local receive data overflow)
- 3318 - T.30 (fax) error: Disconnect during transfer (local abort)
- 3319 - T.30 (fax) error: Illegal parameter coding (e.g. defective SFF file)

B.1.2. ISDN-Fehler-Codes

Diese Codes sind ISDN-Fehler-Codes, die im Standard ETS 300 102-01 im Detail beschrieben sind. Er ist momentan für den privaten Gebrauch kostenlos unter <http://www.etsi.org> erhältlich. Details, wie die ISDN-Codes auf die CAPI-Nummern gemappt werden, finden Sie in der CAPI-Spezifikation, Parameter "Info".

- 3400 - Normal termination, no reason available
- 3480 - Normal termination
- 3481 - Unallocated (unassigned) number
- 3482 - No route to specified transit network
- 3483 - No route to destination
- 3486 - Channel unacceptable
- 3487 - Call awarded and being delivered in an established channel
- 3490 - Normal call clearing
- 3491 - User busy
- 3492 - No user responding
- 3493 - No answer from user (user alerted)
- 3495 - Call rejected
- 3496 - Number changed
- 349A - Non-selected user clearing
- 349B - Destination out of order
- 349C - Invalid number format

- 349D - Facility rejected
- 349E - Response to STATUS ENQUIRY
- 349F - Normal, unspecified
- 34A2 - No circuit / channel available
- 34A6 - Network out of order
- 34A9 - Temporary failure
- 34AA - Switching equipment congestion
- 34AB - Access information discarded
- 34AC - Requested circuit / channel not available
- 34AF - Resources unavailable, unspecified
- 34B1 - Quality of service unavailable
- 34B2 - Requested facility not subscribed
- 34B9 - Bearer capability not authorized
- 34BA - Bearer capability not presently available
- 34BF - Service or option not available, unspecified
- 34C1 - Bearer capability not implemented
- 34C2 - Channel type not implemented
- 34C5 - Requested facility not implemented
- 34C6 - Only restricted digital information bearer capability is available
- 34CF - Service or option not implemented, unspecified
- 34D1 - Invalid call reference value
- 34D2 - Identified channel does not exist
- 34D3 - A suspended call exists, but this call identity does not
- 34D4 - Call identity in use
- 34D5 - No call suspended
- 34D6 - Call having the requested call identity has been cleared
- 34D8 - Incompatible destination
- 34DB - Invalid transit network selection
- 34DF - Invalid message, unspecified
- 34E0 - Mandatory information element is missing
- 34E1 - Message type non-existent or not implemented

Anhang B. CAPI 2.0 Fehler-Codes

- 34E2 - Message not compatible with call state or message type non-existent or not implemented
- 34E3 - Information element non-existent or not implemented
- 34E4 - Invalid information element contents
- 34E5 - Message not compatible with call state
- 34E6 - Recovery on timer expiry
- 34EF - Protocol error, unspecified
- 34FF - Interworking, unspecified

B.2. Interne CAPI-Fehler

Diese Fehler sind hauptsächlich für Entwickler der CapiSuite-Kerns interessant. Wenn Sie ein Anwender sind, werden Sie sie normalerweise nicht brauchen.

B.2.1. Informative Werte (kein Fehler)

Diese Werte sind nur Warnings und können in ausführlichem CapiSuite-Log in Meldungen von der CAPI vorkommen.

- 0000 - No error, request accepted
- 0001 - NCPI not supported by current protocol, NCPI ignored
- 0002 - Flags not supported by current protocol, flags ignored
- 0003 - Alert already sent by another application

B.2.2. Fehler bezüglich CAPI_REGISTER

Diese Fehler können auftreten, wenn die Anwendung startet und deuten meistens auf Probleme mit Ihrer Treiber-Installation hin.

- 1001 - Too many applications.
- 1002 - Logical Block size too small; must be at least 128 bytes.
- 1003 - Buffer exceeds 64 kbytes.
- 1004 - Message buffer size too small, must be at least 1024 bytes.
- 1005 - Max. number of logical connections not supported.

- 1006 - reserved (unknown error).
- 1007 - The message could not be accepted because of an internal busy condition.
- 1008 - OS Resource error (out of memory?).
- 1009 - CAPI not installed.
- 100A - Controller does not support external equipment.
- 100B - Controller does only support external equipment.

B.2.3. Nachrichtenaustausch-Fehler

Diese Fehler sind wirklich intern: sie treten auf, wenn die Anwendung die CAPI falsch aufruft. Wenn sie auftreten, ist das meistens ein Bug, den sie den CapiSuite-Entwicklern melden sollten.

- 1101 - Illegal application number.
- 1102 - Illegal command or subcommand, or message length less than 12 octets.
- 1103 - The message could not be accepted because of a queue full condition.
- 1104 - Queue is empty.
- 1105 - Queue overflow: a message was lost!!
- 1106 - Unknown notification parameter.
- 1107 - The message could not be accepted because on an internal busy condition.
- 1108 - OS resource error (out of memory?).
- 1109 - CAPI not installed.
- 110A - Controller does not support external equipment.
- 110B - Controller does only support external equipment.

B.2.4. Resource/Coding-Fehler

Diese Fehler treten auf, wenn die Anwendung versucht, eine nicht verfügbare Resource zu benutzen. Dies sind meistens auch Bugs in der Anwendung. Melden Sie sie uns bitte.

- 2001 - Message not supported in current state
- 2002 - Illegal Controller / PLCI / NCCI
- 2003 - Out of PLCI

Anhang B. CAPI 2.0 Fehler-Codes

- 2004 - Out of NCCI
- 2005 - Out of LISTEN
- 2007 - Illegal message parameter coding

B.2.5. Fehler bezüglich angeforderter Services

Diese Fehler treten auf, wenn die Anwendung versucht, einen Service falsch anzufordern. Auch dies sind Bugs, die Sie uns melden sollten.

- 3001 - B1 protocol not supported
- 3002 - B2 protocol not supported
- 3003 - B3 protocol not supported
- 3004 - B1 protocol parameter not supported
- 3005 - B2 protocol parameter not supported
- 3006 - B3 protocol parameter not supported
- 3007 - B protocol combination not supported
- 3008 - NCPI not supported
- 3009 - CIP Value unknown
- 300A - Flags not supported (reserved bits)
- 300B - Facility not supported
- 300C - Data length not supported by current protocol
- 300D - Reset procedure not supported by current protocol