Cantina SIP Phone Provisioning System

Stefan Wintermeyer <stefan.wintermeyer@amooma.de>
Philipp Kempgen <philipp.kempgen@amooma.de>
Sascha Daniels <sascha.daniels@amooma.de>

Cantina SIP Phone Provisioning System

von Stefan Wintermeyer, Philipp Kempgen, Sascha Daniels und

Inhaltsverzeichnis

Kapitel 1. Installation

Nachfolgend wird beschrieben wie zuerst Ruby on Rails installiert wird und dann das eigentliche Provisionierungssystem.

1.1. Installation von Ruby on Rails 3.0 mit RVM auf Debian 5.0

Stefan Wintermeyer <stefan.wintermeyer@amooma.de>
Philipp Kempgen <philipp.kempgen@amooma.de>

Diese Beschreibung setzt ein frisch installiertes Debian GNU/Linux 5.0 („Lenny“) voraus. Ein ISO-Image für die Installation finden Sie auf `http://www.debian.org/releases/lenny/debian-installer/`. Ich empfehle das etwa 160 MByte große Netzinstallations-CD-Image. Eine Debian-GNU/Linux-Installationsanleitung findet sich unter `http://www.debian.org/releases/lenny/i386/`; ein allgemeines Anwenderhandbuch unter `http://debiananwenderhandbuch.de`.

1.1.1. Vorbereitungen

Wenn Sie auf dem Zielsystem Root-Rechte besitzen, dann können Sie mit folgendem Befehl sichergehen, dass alle notwendigen Programme für eine erfolgreiche Installation von RVM bereitstehen. Falls Sie keine Root-Rechte haben, müssen Sie entweder hoffen, dass Ihr Admin bereits alles so installiert hat oder ihm die entsprechende Zeile kurz mailen.

Als Erstes ein Update der Paketlisten:

```
debian:~# aptitude update
[...]
Hole:1 http://ftp.de.debian.org lenny/main Translation-de [1820kB]
[...]
1820kB wurden in 12s heruntergeladen (150kB/s)
Paketlisten werden gelesen... Fertig
```

Anmerkung

Natürlich können Sie optional auf dem System einen SSH-Server installieren, um dann auf dem System per SSH statt auf der Konsole zu arbeiten:

```
debian:~# aptitude install openssh-server
```

Und jetzt die Installation der zur Installation von RVM benötigten Pakete:

```
debian:~# aptitude -y install curl git-core patch file
[...]
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  ca-certificates{a} curl file git-core less{a} libcurl3{a}
  libcurl3-gnutls{a} libdigest-sha1-perl{a} liberror-perl{a}
  libexpat1{a} libidn11{a} libldap-2.4-2{a} libmagic1{a}
  libssh2-1{a} openssl{a} patch rsync{a}
0 Pakete aktualisiert, 17 zusätzlich installiert, 0 werden entfernt und 0 nicht
 aktualisiert.
Muss 6501kB an Archiven herunterladen. Nach dem Entpacken werden 16,1MB
 zusätzlich belegt sein.
[...]
Richte curl ein (7.18.2-8lenny4) ...
```

```
[...]
Richte git-core ein (1:1.5.6.5-3+lenny3.2) ...
[...]
```

Damit wir uns nicht später nochmal als `root` einloggen müssen, installieren wir nachfolgend auch direkt die Abhängigkeiten für und zum Bauen von Ruby:

```
debian:~# aptitude -y install build-essential bison  \
  openssl zlib1g-dev libssl-dev libreadline5-dev libxml2-dev \
  libreadline5-dev libxml2-dev
[...]
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  binutils{a} bison build-essential bzip2{a} cpp{a} cpp-4.3{a}
  dpkg-dev{a} g++{a} g++-4.3{a} gcc{a} gcc-4.3{a} libc6-dev{a}
  libgmp3c2{a} libgomp1{a} libmpfr1ldbl{a} libncurses5-dev{a}
  libreadline5-dev libssl-dev libstdc++6-4.3-dev{a}
  libtimedate-perl{a} libxml2{a} libxml2-dev linux-libc-dev{a}
  m4{a} make{a} sgml-base{a} xml-core{a} zlib1g-dev
0 Pakete aktualisiert, 28 zusätzlich installiert, 0 werden entfernt und 0 nicht
 aktualisiert.
Muss 25,5MB an Archiven herunterladen. Nach dem Entpacken werden 80,1MB
 zusätzlich belegt sein.
[...]
Richte g++ ein (4:4.3.2-2) ...
Richte build-essential ein (11.4) ...
[...]
```

Zum Schluss wird es aber noch mal etwas wilder. Wir brauchen für Ruby on Rails 3 eine neuere Version von SQLite (>= 3.6.x) als die, die auf Debian 5 („Lenny“) normalerweise verfügbar ist (3.5.9). Wir haben aber Glück, denn eine passende Version (3.6.21) ist in den „Lenny-Backports“ vorhanden. [1]Folgendermaßen wird Lenny-Backports als Paketquelle für APT hinzugefügt:

```
debian:~# echo \
  'deb http://ftp.de.debian.org/backports.org/ lenny-backports main' \
  > /etc/apt/sources.list.d/lenny-backports.list
```

Kontrolle des Datei-Inhalts:

```
debian:~# cat /etc/apt/sources.list.d/lenny-backports.list
deb http://ftp.de.debian.org/backports.org/ lenny-backports main
```

Paketlisten aktualisieren:

```
debian:~# aptitude update
[...]
Hole:2 http://ftp.de.debian.org lenny-backports Release [74,3kB]
[...]
Hole:3 http://ftp.de.debian.org lenny-backports/main Packages [477kB]
[...]
552kB wurden in 5s heruntergeladen (102kB/s)
Paketlisten werden gelesen... Fertig

Aktueller Status: 824 Neue [+824].
```

[1]siehe `http://packages.debian.org/lenny-backports/sqlite3`

Benötigte SQLite-Pakete aus den Lenny-Backports installieren:

```
debian:~# aptitude -y -t lenny-backports install \
  sqlite3 libsqlite3-dev
[...]
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  libsqlite3-0{a} libsqlite3-dev sqlite3
0 Pakete aktualisiert, 3 zusätzlich installiert, 0 werden entfernt und 27 nicht
 aktualisiert.
Muss 699kB an Archiven herunterladen. Nach dem Entpacken werden 1659kB zusätzlich
 belegt sein.
[...]
Richte libsqlite3-0 ein (3.6.21-2~bpo50+1) ...
Richte libsqlite3-dev ein (3.6.21-2~bpo50+1) ...
Richte sqlite3 ein (3.6.21-2~bpo50+1) ...
[...]
```

1.1.2. Ruby 1.9.2 mit RVM installieren

Loggen Sie sich mit Ihrem normalen Benutzer-Account ein (in unserem Fall ist das der User `xyz`).

RVM lässt sich auf verschiedene Wege installieren. Ich empfehle das folgende Monsterkommando (bitte exakt kopieren), mit dem das aktuellste RVM installiert wird:

```
xyz@debian:~$ bash < <( curl http://rvm.beginrescueend.com/releases/rvm-install-
head )

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   683  100   683    0     0    422      0  0:00:11  0:00:11 --:--:--  222k

Initialized empty Git repository in /home/xyz/.rvm/src/rvm/.git/
got 6d48919a9a6984a38a576e1ec19496d24b2a2696
walk 6d48919a9a6984a38a576e1ec19496d24b2a2696
Getting alternates list for http://github.com/wayneeseguin/rvm.git
Getting pack list for http://github.com/wayneeseguin/rvm.git
[...]

  RVM:  Shell scripts enabling management of multiple ruby environments.
  RTFM: http://rvm.beginrescueend.com/
  HELP: http://webchat.freenode.net/?channels=rvm (#rvm on irc.freenode.net)

Installing RVM to /home/xyz/.rvm/
    Correct permissions for base binaries in /home/xyz/.rvm/bin...
    Copying manpages into place.
cat: /etc/*-release: Datei oder Verzeichnis nicht gefunden


Notes for Linux (  )

# NOTE: MRI stands for Matz's Ruby Interpreter (1.8.X, 1.9.X), ree stands for
 Ruby Enterprise Edition and rbx stands for Rubinius.
# curl is required.
# git is required.
# patch is required (for ree, some ruby head's).
# If you wish to install rbx and/or any MRI head (eg. 1.9.2-head) then you must
 install and use rvm 1.8.7 first.
```

```
# If you wish to have the 'pretty colors' again, set 'export
 rvm_pretty_print_flag=1' in ~/.rvmrc.

dependencies:
# For RVM
  rvm: bash curl git-core

# For JRuby (if you wish to use it) you will need:
  jruby: aptitude install curl sun-java6-bin sun-java6-jre sun-java6-jdk

# For MRI & ree (if you wish to use it) you will need (depending on what you are
 installing):
  ruby: aptitude install build-essential bison openssl libreadline5 libreadline-
dev curl git-core zlib1g zlib1g-dev libssl-dev libsqlite3-0 libsqlite3-dev
 sqlite3 libxml2-dev
  ruby-head: git subversion autoconf

# For IronRuby (if you wish to use it) you will need:
  ironruby: aptitude install curl mono-2.0-devel


  You must now complete the install by loading RVM in new shells.

  1) Place the folowing line at the end of your shell's loading files
     (.bashrc or .bash_profile for bash and .zshrc for zsh),
     after all PATH/variable settings:

     [[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm"  # This
 loads RVM into a shell session.

     You only need to add this line the first time you install rvm.

  2) Ensure that there is no 'return' from inside the ~/.bashrc file,
     otherwise rvm may be prevented from working properly.


  This means that if you see something like:

    '[ -z "$PS1" ] && return'

  then you change this line to:

  if [[ -n "$PS1" ]] ; then

    # ... original content that was below the '&& return' line ...

  fi # <= be sure to close the if at the end of the .bashrc.

  # This is a good place to source rvm v v v
  [[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm"  # This
 loads RVM into a shell session.

EOF - This marks the end of the .bashrc file

     Be absolutely *sure* to REMOVE the '&& return'.
```

```
    If you wish to DRY up your config you can 'source ~/.bashrc' at the bottom
 of your .bash_profile.

    Placing all non-interactive (non login) items in the .bashrc,
    including the 'source' line above and any environment settings.

  3) CLOSE THIS SHELL and open a new one in order to use rvm.


  WARNING:  you have a 'return' statement in your ~/.bashrc
              This could cause some features of RVM to not work.


  This means that if you see something like:

    '[ -z "$PS1" ] && return'

  then you change this line to:

  if [[ -n "$PS1" ]] ; then

    # ... original content that was below the '&& return' line ...

  fi # <= be sure to close the if at the end of the .bashrc.

  # This is a good place to source rvm v v v
  [[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm"  # This
 loads RVM into a shell session.

EOF - This marks the end of the .bashrc file

  Even if you are using zsh you should still adjust the ~/.bashrc
  If you have any questions about this please visit
    #rvm on irc.freenode.net.


Installation of RVM to /home/xyz/.rvm/ is complete.


xyz,

Thank you very much for using RVM! I sincerely hope that RVM helps to
make your work both easier and more enjoyable.

If you have any questions, issues and/or ideas for improvement please
join#rvm on irc.freenode.net and let me know, note you must register
(http://bit.ly/5mGjlm) and identify (/msg nickserv <nick> <pass>) to
talk, this prevents spambots from ruining our day.

My irc nickname is 'wayneeseguin' and I hang out in #rvm typically

  ~09:00-17:00EDT and again from ~21:00EDT-~23:00EDT

If I do not respond right away, please hang around after asking your
question, I will respond as soon as I am back.  It is best to talk in
#rvm itself as then other users can help out should I be offline.
```

```
Be sure to get head often as rvm development happens fast,
you can do this by running 'rvm update --head' followed by 'rvm reload'
or opening a new shell

  w□□□t

    ~ Wayne

xyz@debian:~$
```

RVM ist jetzt fertig installiert, aber Sie müssen zum täglichen Gebrauch noch die beim Starten Ihrer Bash eingelesene `.bashrc` so verändern, dass `rvm` immer automatisch konfiguriert wird. Das ist etwas kniffelig, weil Sie dafür Elemente aus der vorhergehenden Ausgabe wiederverwenden müssen und diese bei jeder Installation angepasst werden.

Jetzt gibt es zwei Möglichkeiten: Entweder Sie haben eine ganz normale Debian-Installation und eine `.bashrc` Default-Datei oder eben nicht. Die folgenden Zeilen funktionieren nur bei den Default-Dateien. Sie sind also eine 80-%- oder 90-%-Lösung. Wenn Sie ein stark „optimiertes“ System haben, müssen Sie oben in der Ausgabe des RVM-Installers unter „`You must now complete the install by loading RVM in new shells`“ nachlesen, was genau zu machen ist. Alle mit Default kopieren Folgendes:

```
xyz@debian:~$ cp .bashrc .bashrc-ORIG
xyz@debian:~$ echo '[[ -s $HOME/.rvm/scripts/rvm ]] && source $HOME/.rvm/scripts/
rvm' >> .bashrc
xyz@debian:~$ perl -pi -e 's/\[ -z \"\$PS1\" \] \&\& return/if \[\[ -n \"\$PS1\"
 \]\] \; then/' .bashrc
xyz@debian:~$ echo 'fi' >> .bashrc
xyz@debian:~$
```

Wichtig

Jetzt müssen Sie das Terminal schließen und ein neues Terminal öffnen [2], denn ansonsten wirkt diese Veränderung noch nicht.

Um zu testen, ob `rvm` richtig installiert ist, rufen wir es einmal mit `-v` auf:

```
xyz@debian:~$ rvm -v

rvm 1.0.14 by Wayne E. Seguin (wayneeseguin@gmail.com) [http://
rvm.beginrescueend.com/]
```

Sollte hierbei als Antwort der Bash „`command not found`“ ausgegeben werden, müssen Sie die obigen Schritte noch einmal überprüfen. Wahrscheinlich haben Sie einen Schritt ausgelassen oder bei der Arbeit in der `.bashrc` einen Fehler gemacht. Bitte lesen Sie noch einmal die Ausgabe vom Skript `rvm-install-head` und führen Sie die entsprechenden Schritte durch.

Mit `rvm` müssen Sie zuerst das Paket `zlib` installieren:

```
xyz@debian:~$ rvm package install zlib

Fetching zlib-1.2.5.tar.gz to /home/xyz/.rvm/archives

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
```

```
100  531k  100  531k    0     0   120k      0  0:00:04  0:00:04 --:--:--  147k

Extracting zlib-1.2.5.tar.gz to /home/xyz/.rvm/src
Configuring zlib in /home/xyz/.rvm/src/zlib-1.2.5.
Compiling zlib in /home/xyz/.rvm/src/zlib-1.2.5.
Installing zlib to /home/xyz/.rvm/usr
xyz@debian:~$
```

Der folgende Befehl zeigt Ihnen die „Rubies“ (Ruby-Interpreter) an, die RVM kennt [3]:

```
xyz@debian:~$ rvm list known
# MRI Rubies
[ruby-]1.8.6[-p399]
[ruby-]1.8.6-head
[ruby-]1.8.7[-p302]
[ruby-]1.8.7-head
[ruby-]1.9.1-p243
[ruby-]1.9.1[-p376]
[ruby-]1.9.1-p429
[ruby-]1.9.1-head
[ruby-]1.9.2-preview1
[ruby-]1.9.2-preview3
[ruby-]1.9.2-rc1
[ruby-]1.9.2-rc2
[ruby-]1.9.2[-p0]
[ruby-]1.9.2-head
ruby-head
[...]
```

Anmerkung

Es gibt für Ruby verschiedene Interpreter. Relevant ist hier für uns der ganz normale **MRI** (*Matz's Ruby Interpreter*) [4], dessen Kern seit Ruby 1.9 die Ruby-VM namens **YARV** (*Yet Another Ruby VM*) [5] bildet.

Jetzt können Sie mit RVM Ruby 1.9.2 installieren:

```
xyz@debian:~$ rvm install ruby-1.9.2

/home/xyz/.rvm/rubies/ruby-1.9.2-p0, this may take a while depending on your
 cpu(s)...

ruby-1.9.2-p0 - #fetching
ruby-1.9.2-p0 - #downloading ruby-1.9.2-p0, this may take a while depending on
 your connection...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 8296k  100 8296k    0     0   112k      0  0:01:14  0:01:14 --:--:--  122k

ruby-1.9.2-p0 - #extracting ruby-1.9.2-p0 to /home/xyz/.rvm/src/ruby-1.9.2-p0
ruby-1.9.2-p0 - #extracted to /home/xyz/.rvm/src/ruby-1.9.2-p0
ruby-1.9.2-p0 - #configuring
```

[3] siehe `http://rvm.beginrescueend.com/rubies/list/`

```
ruby-1.9.2-p0 - #compiling
ruby-1.9.2-p0 - #installing
ruby-1.9.2-p0 - updating #rubygems for /home/xyz/.rvm/gems/ruby-1.9.2-p0@global
ruby-1.9.2-p0 - updating #rubygems for /home/xyz/.rvm/gems/ruby-1.9.2-p0
ruby-1.9.2-p0 - adjusting #shebangs for (gem).
ruby-1.9.2-p0 - #importing default gemsets (/home/xyz/.rvm/gemsets/)
Install of ruby-1.9.2-p0 - #complete
xyz@debian:~$
```

Noch haben Sie per Default kein **ruby** (oder ggf. nur das Ruby 1.8.x des Systems, falls Sie vorher bereits eines installiert haben sollten):

```
xyz@debian:~$ ruby -v
bash: ruby: command not found
xyz@debian:~$ which ruby
xyz@debian:~$
```

Mit dem Befehl **rvm ruby-1.9.2** oder kurz **rvm 1.9.2** können Sie aber auf ein Ruby 1.9.2 wechseln [6].

```
xyz@debian:~$ rvm 1.9.2
xyz@debian:~$ ruby -v
ruby 1.9.2p0 (2010-08-18 revision 29036) [i686-linux]
xyz@debian:~$ which ruby
/home/xyz/.rvm/rubies/ruby-1.9.2-p0/bin/ruby
xyz@debian:~$
```

Dummerweise müssten Sie jetzt in jeder neuen Shell **rvm 1.9.2** eingeben. Sie können für diesen User aber auch die Ruby-Version 1.9.2 als Default einstellen:

```
xyz@debian:~$ rvm --default 1.9.2
```

Anmerkung

Sollten Sie vorher bereits eine System-Ruby-Version 1.8.x installiert haben, so können Sie mit dem Befehl **rvm system** jederzeit wieder auf diese zurückwechseln.

Tipp

Geben Sie einfach mal den Befehl **rvm** ein, um eine entsprechende Hilfe-Seite mit weiteren Befehlen angezeigt zu bekommen.

1.1.3. Rails 3.0 installieren

Zuerst überprüfen wir, ob ein Ruby in der Version 1.9.2 aufgerufen wird. Sollte das nicht der Fall sein, lesen Sie sich bitte *Abschnitt 1.1.2, „Ruby 1.9.2 mit RVM installieren“* durch.

```
xyz@debian:~$ ruby -v
ruby 1.9.2p0 (2010-08-18 revision 29036) [i686-linux]
```

[6] Dieses wird dann in den sogenannten PATH (also den Such-Pfad für ausführbare Programme) des aktuellen Benutzers eingetragen. Weitere Informationen zum PATH: http://en.wikipedia.org/wiki/PATH_(variable) [http://en.wikipedia.org/wiki/PATH_%28variable%29]

Anschließend stellen wir sicher, dass das `gem`-Paketmanagement aktuell ist:

```
xyz@debian:~$ gem update --system
Updating RubyGems
Nothing to update

xyz@debian:~$ gem update
Updating installed gems
Nothing to update
```

Der Rest ist einfach:

```
xyz@debian:~$ gem install rails --version 3.0.0
Successfully installed activesupport-3.0.0
Successfully installed builder-2.1.2
Successfully installed i18n-0.4.1
[...]
Successfully installed thor-0.14.0
Successfully installed railties-3.0.0
Successfully installed rails-3.0.0
22 gems installed
Installing ri documentation for activesupport-3.0.0...
Installing ri documentation for builder-2.1.2...
Installing ri documentation for i18n-0.4.1...
[...]
Installing RDoc documentation for railties-3.0.0...
Installing RDoc documentation for rails-3.0.0...
xyz@debian:~$
```

Perfekt. Jetzt haben Sie Rails 3.0 installiert.

```
xyz@debian:~$ rails -v
Rails 3.0.0
```

1.2. Installation des Cantina SIP Phone Provisioning Systems

Philipp Kempgen
Voraussetzung: Installation von Ruby on Rails 3 auf Debian 5 (siehe *Abschnitt 1.1, „Installation von Ruby on Rails 3.0 mit RVM auf Debian 5.0“*).

Cantina ist verfügbar auf `https://github.com/amooma/Cantina` und kann von dort wie folgt per Git installiert werden:

```
debian:~# cd /usr/src
debian:/usr/src# git clone git@github.com:amooma/Cantina.git
Cloning into Cantina...
remote: Counting objects: 3688, done.
remote: Compressing objects: 100% (1505/1505), done.
remote: Total 3688 (delta 2481), reused 3249 (delta 2114)
Receiving objects: 100% (3688/3688), 2.07 MiB | 446 KiB/s, done.
Resolving deltas: 100% (2481/2481), done.
debian:/usr/src#
```

1.2.1. Anlegen der Datenbank

Initial muß die Datenbank des Provisioning-Servers angelegt werden. Wechseln Sie dazu in das Installations-Verzeichnis (abhängig von Ihrer Installation, beispielsweise `/home/cantina/`):

```
debian:/usr/src# cd /usr/src/Cantina/
```

Dann geben Sie den Befehl **rake db:setup** ein:

```
debian:/usr/src/Cantina# rake db:setup
(in /usr/src/Cantina)
[...]
```

Die Datenbank ist nun angelegt.

1.2.2. Provisioning-Server starten

Um den Provisioning-Server zu starten wechseln Sie in das Installations-Verzeichnis (abhängig von Ihrer Installation, beispielsweise `/home/cantina/`):

```
debian:~# cd /usr/src/Cantina/
```

Dann starten Sie den mitgelieferten Web-Server mit dem Befehl **rails server** (oder kurz: **rails s**):

```
debian:/usr/src/Cantina# rails server
=> Booting WEBrick
=> Rails 3.0.3 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2010-11-23 09:35:30] INFO  WEBrick 1.3.1
[2010-11-23 09:35:30] INFO  ruby 1.9.2 (2010-08-18) [i686-linux]
[2010-11-23 09:35:30] INFO  WEBrick::HTTPServer#start: pid=2910 port=3000
```

Der Web-Server läuft nun auf allen IP-Adressen (0.0.0.0) des Rechners auf Port 3000. Wie angezeigt läßt er sich mit **Ctrl**+**C** stoppen.

Wenn man an den Befehl **rails server** die Option **-h** oder **--help** anhängt bekommt man eine Hilfe angezeigt:

```
debian:/usr/src/Cantina# rails server -h
Usage: rails server [mongrel, thin, etc] [options]
  -p, --port=port         Runs Rails on the specified port.
                          Default: 3000
  -b, --binding=ip        Binds Rails to the specified ip.
                          Default: 0.0.0.0
  -c, --config=file       Use custom rackup configuration file
  -d, --daemon            Make server run as a Daemon.
  -u, --debugger          Enable ruby-debugging for the server.
  -e, --environment=name  Specifies the environment to run
                          this server under
                          (test/development/production).
                          Default: development
  -P, --pid=pid           Specifies the PID file.
                          Default: tmp/pids/server.pid
```

```
    -h, --help              Show this help message.
```

Will man den Server beispielsweise nur auf einer bestimmten IP-Adresse des Rechners betreiben so geht dies mit der Option **-b** oder **--binding**, also z.B.

```
debian:/usr/src/Cantina# rails server -b 127.0.0.1
=> Booting WEBrick
=> Rails 3.0.3 application starting in development on http://127.0.0.1:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[...]
```

1.2.3. Installation und Konfiguration des DHCP Servers

Um Telefone vollautomatisch konfigurieren zu können, benötigen Sie einen DHCP Server.

Über den DHCP Server erhalten die Telefone nicht nur eine IP, sondern auch die Information wo eine Konfiguration bereitgestellt wird. Im folgendn Beispiel gehen wir davon aus, dass noch kein DHCP Server im Netz Ihrer Telefone vorhanden ist. Folgende Eckdaten setzen wir für das Beispiel voraus:

1. Cantina und der DHCP Server laufen auf dem gleichen Server.
2. IP des Servers 192.168.1.254
3. Netzmaske 255.255.255.0

Als erstes muss ein DHCP Server installier werden. Wir gehen bei unserem Beispiel von der Basis eines Debian "Squeeze" Systems aus.

Zuerst aktualisieren wir die Paketdatenbank:

```
root@server:~# aptitude update
Hole:1 http://ftp.de.debian.org squeeze Release.gpg [835 B]
Hole:2 http://ftp.de.debian.org/debian/ squeeze/main Translation-de [1.510 kB]

Hole:3 http://www.debian-multimedia.org squeeze Release.gpg [198 B]

Ign http://www.debian-multimedia.org/ squeeze/main Translation-de

Ign http://www.debian-multimedia.org/ squeeze/main Translation-en

Ign http://www.debian-multimedia.org/ squeeze/non-free Translation-de

Ign http://www.debian-multimedia.org/ squeeze/non-free Translation-en

Hole:4 http://www.debian-multimedia.org squeeze Release [26,7 kB]

Hole:5 http://www.debian-multimedia.org squeeze/main i386 Packages/DiffIndex
 [2.023 B]
Hole:6 http://www.debian-multimedia.org squeeze/non-free i386 Packages/DiffIndex
 [2.023 B]
Hole:7 http://www.debian-multimedia.org squeeze/main i386 Packages [73,5 kB]

Hole:8 http://security.debian.org squeeze/updates Release.gpg [835 B]

Ign http://security.debian.org/ squeeze/updates/main Translation-de
```

```
Ign http://security.debian.org/ squeeze/updates/main Translation-en

Hole:9 http://security.debian.org squeeze/updates Release [44,6 kB]

Hole:10 http://security.debian.org squeeze/updates/main Sources [7.934 B]

Hole:11 http://security.debian.org squeeze/updates/main i386 Packages [47,6 kB]

Hole:12 http://www.debian-multimedia.org squeeze/non-free i386
 2011-01-06-1715.55.pdiff [2.376 B]
Hole:13 http://www.debian-multimedia.org squeeze/non-free i386
 2011-01-06-1715.55.pdiff [2.376 B]
Hole:14 http://www.debian-multimedia.org squeeze/non-free i386
 2011-01-06-1715.55.pdiff [2.376 B]
Ign http://ftp.de.debian.org/debian/ squeeze/main Translation-en

Hole:15 http://www.debian-multimedia.org squeeze/non-free i386
 2011-01-11-1621.09.pdiff [1.357 B]
Hole:16 http://www.debian-multimedia.org squeeze/non-free i386
 2011-01-11-1621.09.pdiff [1.357 B]
Hole:17 http://www.debian-multimedia.org squeeze/non-free i386
 2011-01-11-1621.09.pdiff [1.357 B]
Hole:18 http://ftp.de.debian.org squeeze Release [89,9 kB]
Hole:19 http://ftp.de.debian.org squeeze/main Sources/DiffIndex [2.038 B]
Hole:20 http://ftp.de.debian.org squeeze/main i386 Packages/DiffIndex [2.038 B]
Hole:21 http://ftp.de.debian.org squeeze/main Sources [3.724 kB]
Hole:22 http://ftp.de.debian.org squeeze/main i386 Packages [6.566 kB]
12,1 MB wurden in 3 s heruntergeladen (3.499 kB/s)
```

Als nächstes installieren wir den DHCP Server.

```
root@server:~# aptitude install isc-dhcp-server
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  isc-dhcp-server
Die folgenden Pakete werden aktualisiert:
  isc-dhcp-common
1 Pakete aktualisiert, 1 zusätzlich installiert, 0 werden entfernt und 140 nicht
 aktualisiert.
Muss 693 kB an Archiven herunterladen. Nach dem Entpacken werden 885 kB
 zusätzlich belegt sein.
Wollen Sie fortsetzen? [Y/n/?] y

Hole:1 http://ftp.de.debian.org/debian/ squeeze/main isc-dhcp-common i386 4.1.1-
P1-15 [315 kB]
Hole:2 http://ftp.de.debian.org/debian/ squeeze/main isc-dhcp-server i386 4.1.1-
P1-15 [377 kB]
693 kB wurden in 0 s heruntergeladen (1.698 kB/s)
Lese Changelogs... Fertig
Vorkonfiguration der Pakete ...
(Lese Datenbank ... 49000 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereiten zum Ersetzen von isc-dhcp-common 4.1.1-P1-12 (durch .../isc-dhcp-
common_4.1.1-P1-15_i386.deb) ...
Entpacke Ersatz für isc-dhcp-common ...
Wähle vormals abgewähltes Paket isc-dhcp-server.
Entpacke isc-dhcp-server (aus .../isc-dhcp-server_4.1.1-P1-15_i386.deb) ...
Verarbeite Trigger für man-db ...
Richte isc-dhcp-common ein (4.1.1-P1-15) ...
```

```
Richte isc-dhcp-server ein (4.1.1-P1-15) ...
Generating /etc/default/isc-dhcp-server...
Starting ISC DHCP server: dhcpdcheck syslog for diagnostics. ... failed!
 failed!
invoke-rc.d: initscript isc-dhcp-server, action "start" failed.

Aktueller Status: 140 Aktualisierungen [-1].
root@server:~#
```

Die Fehlermeldung ist in dem Fall in Ordnung, da der Server ohne Konfiguration nicht starten kann.

Zuerst müssen wir dem Server sagen, an welcher Netzwerk Schnittstelle er lauschen soll.

```
root@server:~# vi /etc/default/isc-dhcp-server

# Defaults for dhcp initscript
# sourced by /etc/init.d/dhcp
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES=""
```

Wir gehen davon aus, dass der Server nur eine Netzwerkkarte besitzt und damit der Name der Karte "etho" ist.

Wir ändern die Konfiguration wie folgt ab:

```
# Defaults for dhcp initscript
# sourced by /etc/init.d/dhcp
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```

Jetzte speichern wir die Konfiguration wie folgt ab:

"**ESC**"

:wq

Jetzt müssen wir noch eine Konfiguration für den DHCP Server anlegen.

```
root@server:~# mv /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf_DEBIAN
root@server:~# vi /etc/dhcp/dhcpd.conf
```

Folgende Minimalkonfiguration reicht aus um die Telefone mit den benötigten Informationen zu versorgen.

```
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

ddns-updates off;

ddns-update-style none;
log-facility local7;

subnet 192.168.1.0 netmask 255.255.255.0 {
              range 192.168.1.1 192.168.1.100;
               option tftp-server-name "http://192.168.1.254:3000/settings";
}
```

Die Werte für "domain-name" und "domain-name-server" müssen Ihren Gegebenheiten angepasst werden, haben aber für einen ersten Test keine Bedeutung. Weitere Informationen zu sinnvollen Parametern entnehmen Sie bitte der Dokumentation des DHCP Servers. Natürlich müssen Sie die Daten des Subnets auch an Ihre Gegebenheiten anpassen.

Wir können den Server jetzt starten.

```
root@start:~# /etc/init.d/isc-dhcp-server start
```

Jetzt können Sie Ihr erstes Telefon provisionieren.

1.2.4. Telefon provisionieren am Beispiel eines Snom 360/370

Wir empfehlen zum ersten Testen des Provisionings ein eigenes Testnetz anzulegen. Dann können Sie leichter das System ausprobieren und verstehen.

- Stellen Sie sicher, das auf dem benutzen Snom 360/370 eine aktuelle 8er Firmware installiert ist.
- Starten Sie den Rails-Server.
- Starten Sie den DHCP-Server.
- Stecken Sie das Telefon an das Netzwerk.

Kapitel 2. Daten-Modell

####### #######
##########

Wie bei jedem Ruby-on-Rails-Projekt befinden sich die Datenbank-„*Models*“ im Unterverzeichnis `app/models/`. In jeder Datei wird eine Klasse definiert, die eine entsprechende Tabelle in der Datenbank hat. Das komplette `ActiveRecord`-Schema findet sich in `db/schema.rb`. Auch in den jeweiligen *Models* ist jeweils oben in einem Kommentar die Tabellenstruktur beschrieben.

Ein Daten-Modell hat eine Reihe von Attributen (bzw. Feldern), Relationen zu anderen Daten-Modellen, sowie Methoden zur Validierung von übergebenen Daten für die Attribute.

In Cantina gibt es folgende Daten-Modelle:

Modell	**Dateiname**
`Codec`	`codec.rb`
`Manufacturer`	`manufacturer.rb`
`Oui`	`oui.rb`
`PhoneKeyFunctionDefinition`	`phone_key_function_definition.rb`
`PhoneKey`	`phone_key.rb`
`PhoneKeyToFunctionMapping`	`phone_key_to_function_mapping.rb`
`PhoneModelCodec`	`phone_model_codec.rb`
`PhoneModelKey`	`phone_model_key.rb`
`PhoneModelMacAddress`	`phone_model_mac_address.rb`
`PhoneModel`	`phone_model.rb`
`Phone`	`phone.rb`
`ProvisioningLogEntry`	`provisioning_log_entry.rb`
`RebootRequest`	`reboot_request.rb`
`SipAccountCodec`	`sip_account_codec.rb`
`SipAccount`	`sip_account.rb`

Die Dateinamen und Klassen folgen der Standard-Konvention von Ruby-on-Rails-Projekten, was die EInstiegshürde für andere Entwickler niedrig hält.

Eine grafische Übersicht zu den Datenstrukturen findet sich in der mitgelieferten Datei `ERD.pdf` in Form eines Entity-Relationship-Diagramms.

Im folgenden wird erklärt was die einzelnen Daten-Modelle enthalten. Wir gehen hier zwar zwecks leichterer Nachschlagbarkeit in alphabetischer Reihenfolge vor, jedoch sei schon mal vorab gesagt daß die zentralen Daten-Modelle `Phone` (die Telefone) und `SipAccount` (die SIP-Accounts) sind.

Dabei sind immer auch die entsprechenden direkten Relationen angegeben. Wir verwenden hierfür eine an `ActiveRecord`-Assoziationen angelehnte Schreibweise, allerdings gleich mit einem Verweis bzw. Link auf das jeweilige referenzierte Daten-Modell. Die verwendeten Assoziationen bedeuten: `has_many` = „hat viele“ (0, 1, oder mehrere), `belongs_to` = „gehört zu“ (genau eine).

2.1. Codec

Enthält die relevanten Audio-Codecs (*codecs*). Diese werden verwendet zur Zuordnung zu auf Telefon-Modellen unterstützten Codecs (*phone model codecs*) und den von den SIP-Accounts zu verwendenden Codecs (*sip account codecs*).

Beispiele für Codecs sind: „`gsm`“ (GSM-FR 06.10 [2]), „`alaw`“ (G.711 [3] A-law [4]), „`ulaw`“ (G.711 µ-law [5]), „`g722`“ (G.722 [6]) etc.

Verknüpfungen.

- `has_many` *Abschnitt 2.14, „SipAccountCodec“*
- `has_many` *Abschnitt 2.7, „PhoneModelCodec“*

2.2. Manufacturer

Enthält die Hersteller (*manufacturer*) der Telefon-Modelle (*phone models*).

Beispiele für Hersteller sind: „Snom Technology“, „Aastra DeTeWe“, „Tiptel“ etc.

Verknüpfungen.

- `has_many` *Abschnitt 2.10, „PhoneModel“*
- `has_many` *Abschnitt 2.3, „Oui“*

2.3. Oui

Enthält die IEEE-Ethernet-OUIs (*http://de.wikipedia.org/wiki/MAC-Adresse#Herstellerkennungen*) der Hersteller (*manufacturers*).

Beispiele für OUIs sind: „`000413`“, „`003042`“, „`00085d`“, „`001565`“ etc.

Verknüpfungen.

- `belongs_to` *Abschnitt 2.2, „Manufacturer“*

2.4. PhoneKeyFunctionDefinition

Enthält die verfügbaren Funktionen von Telefon-Tasten (*phone keys*).

Beispiele sind: „`BLF`“, „`Speed dial`“, „`Line`“, „`ActionURL`“ etc.

Verknüpfungen.

- `has_many` *Abschnitt 2.5, „PhoneKey“*
- `has_many` *Abschnitt 2.6, „PhoneKeyToFunctionMapping“*

2.5. PhoneKey

Enthält die zu den SIP-Accounts (*sip accounts*) gesetzten Softkeys. Ein `PhoneKey` ist jeweils einem SIP-Account zugeordnet sowie einer bestimmten Taste auf einem Telefon-Modell und bezieht sich auf eine `PhoneKeyFunctionDefinition`.

EIn Beispiel wäre sinngemäß: Taste 1 auf Snom 370 zu SIP-Account XY belegt als Zielwahl mit der Nummer 123.

Verknüpfungen.

[2] *http://de.wikipedia.org/wiki/Global_System_for_Mobile_Communications#Full_Rate_Codec_.28FR.29*
[3] *http://de.wikipedia.org/wiki/G.711*
[4] *http://de.wikipedia.org/wiki/A-law*
[5] *http://de.wikipedia.org/wiki/%CE%9C-law*
[6] *http://de.wikipedia.org/wiki/G.722*

- `belongs_to` *Abschnitt 2.15, „SipAccount“*
- `belongs_to` *Abschnitt 2.8, „PhoneModelKey“*
- `belongs_to` *Abschnitt 2.4, „PhoneKeyFunctionDefinition“*

2.6. PhoneKeyToFunctionMapping

Enthält eine Umsetzung der Telefon-Tasten auf Tasten-Funktionen.

Verknüpfungen.
- `belongs_to` *Abschnitt 2.8, „PhoneModelKey“*
- `belongs_to` *Abschnitt 2.4, „PhoneKeyFunctionDefinition“*

2.7. PhoneModelCodec

Enthält die auf den jeweiligen Telefon-Modellen (*phone models*) verfügbaren Codecs (*codecs*).

Verknüpfungen.
- `belongs_to` *Abschnitt 2.10, „PhoneModel“*
- `belongs_to` *Abschnitt 2.1, „Codec“*

2.8. PhoneModelKey

Enhält die auf dem jeweiligen Telefon-Model (*phone model*) verfügbaren Tasten.

Auf einem Snom 370 wären dies beispielsweise P 1 (`fkey` 0) bis P 12 (`fkey` 11).

Verknüpfungen.
- `belongs_to` *Abschnitt 2.10, „PhoneModel“*
- `has_many` *Abschnitt 2.6, „PhoneKeyToFunctionMapping“*
- `has_many` *Abschnitt 2.4, „PhoneKeyFunctionDefinition“*
- `has_many` *Abschnitt 2.5, „PhoneKey“*

2.9. PhoneModelMacAddress

Enthält bekannte MAC-Adressen zu den Telefon-Modellen.

Verknüpfungen.
- `belongs_to` *Abschnitt 2.10, „PhoneModel“*

2.10. PhoneModel

Enthält die verfügbaren Telefon-Modelle (*phone models*).

Beispiele für Telefon-Modelle sind: Snom 360, Snom 370, Aastra 55i, Aastra 57i, Tiptel IP 286, etc.

Verknüpfungen.
- `belongs_to` *Abschnitt 2.2, „Manufacturer“*
- `has_many` *Abschnitt 2.8, „PhoneModelKey“*
- `has_many` *Abschnitt 2.9, „PhoneModelMacAddress“*
- `has_many` *Abschnitt 2.11, „Phone“*

- `has_many` *Abschnitt 2.7, „PhoneModelCodec“*

2.11. Phone

Enthält die Telefone (*phones*).

Verknüpfungen.
- `belongs_to` *Abschnitt 2.10, „PhoneModel“*
- `has_many` *Abschnitt 2.5, „PhoneKey“*
- `has_many` *Abschnitt 2.15, „SipAccount“*
- `has_many` *Abschnitt 2.12, „ProvisioningLogEntry“*
- `has_many` *Abschnitt 2.13, „RebootRequest“*

2.12. ProvisioningLogEntry

Enthält Provisioning-Log-Einträge (*provisioning log entries*) zu den Telefonen (*phones*).

Verknüpfungen.
- `belongs_to` *Abschnitt 2.11, „Phone“*

2.13. RebootRequest

Enthält Neustart-Anfragen (*reboot requests*) zu den Telefonen (*phones*).

Verknüpfungen.
- `belongs_to` *Abschnitt 2.11, „Phone“*

2.14. SipAccountCodec

Enthält die den SIP-Accounts (*sip accounts*) zugeordneten Codecs (*codecs*).

Verknüpfungen.
- `belongs_to` *Abschnitt 2.15, „SipAccount“*
- `belongs_to` *Abschnitt 2.1, „Codec“*

2.15. SipAccount

Enthält die auf die Telefone (*phones*) zu provisionierenden SIP-Accounts. Zu einem SIP-Account gehören Softkeys (*phone keys*) und zu verwendende Codecs (*codecs*). Es ist jedoch zu beachten daß nicht alle Telefon-Modelle die Zuordnung von Codecs pro SIP-Account unterstützen, auf einigen Geräten können die Codecs lediglich generell für alle SIP-Accounts gesetzt werden.

Verknüpfungen.
- `belongs_to` *Abschnitt 2.11, „Phone“*
- `has_many` *Abschnitt 2.5, „PhoneKey“*
- `has_many` *Abschnitt 2.14, „SipAccountCodec“*

Kapitel 3. API

####### #######

Wie bei Ruby-on-Rails-Projekten üblich sind die einzelnen „Ressourcen“ (siehe *Kapitel 2, Daten-Modell*) über eine API zugänglich. Dabei gibt es meist mehrere Formate, üblicherweise HTML und XML. HTML – das Standard-Format – dient zur Anzeige im Browser, XML dient zur automatisierten Ansteuerung, beispielsweise aus Skripten heraus.

3.1. API im REST-Stil

Betrachten wir als Beispiel mal die Ressource „`phone_models`“ zum `PhoneModel` *model*, die die verfügbaren Telefon-Modelle bezeichnet (siehe *Abschnitt 2.10, „PhoneModel“*).

Voraussetzung: Der Web-Server ist gestartet, wie in *Abschnitt 1.2.2, „Provisioning-Server starten“* beschrieben.

```
debian:/usr/src/Cantina# rails server
=> Booting WEBrick
=> Rails 3.0.3 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2010-11-23 09:35:30] INFO  WEBrick 1.3.1
[2010-11-23 09:35:30] INFO  ruby 1.9.2 (2010-08-18) [i686-linux]
[2010-11-23 09:35:30] INFO  WEBrick::HTTPServer#start: pid=2910 port=3000
```

Auf dem Server „`http://{host}:3000`“ unter dem Pfad „`/`“ ist jetzt die Applikation erreichbar. Unter dem Pfad „`/phone_models`“ bekommen Sie im Browser eine Liste der Telefon-Modelle angezeigt. Woher weiß die Applikation daß HTML ausgegeben werden soll? HTML ist – etwas vereinfacht gesagt – das Standard-Format.

Anmerkung

Tatsächlich wird das zurückzugebende Format folgendermaßen bestimmt: Wenn der Client in der URL eine Dateiendung angibt (`.html` oder `.xml`), dann bekommt er das entsprechende Format.

Wenn der Client keine Dateiendung angibt, dann kommt „*Content Negotiation*“ (*http://tools.ietf.org/html/rfc2616#section-12.1*) zum Einsatz. Clients senden üblicherweise einen HTTP-Header namens `Accept` (*http://tools.ietf.org/html/rfc2616#section-14.1*), in dem sie dem Server die von ihnen verstandenen Formate mitteilen. Ein Web-Browser sendet beispielsweise:

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
```

Der Server erkennt daran daß der Client das HTML-Format bevorzugt. Würde der Client folgendes senden:

```
Accept: text/xml,application/xml
```

so weiß der Server daß der Client XML bevorzugt.

In der Ausgabe des Rails-Servers (bzw. im Log mit **tail -f log/*.log**) sieht man:

```
Started GET "/phone_models" [...]
  Processing by PhoneModelsController#index as HTML
[...]
```

Ebensogut kann man an den Pfad noch die Endung „`.html`“ anhängen: Unter dem Pfad „`/phone_models.html`“ bekommt man also die gleiche Ausgabe im Browser. Geloggt wird dabei:

```
Started GET "/phone_models.html" [...]
  Processing by PhoneModelsController#index as HTML
[...]
```

Wenn man nun die Liste im XML-Format haben will, so bekommt man diese mit der Endung „`.xml`“, also unter dem Pfad „`/phone_models.xml`“.

Anmerkung

Auch dies läßt sich im Web-Browser ausprobieren. Bitte beachten Sie jedoch daß jeder Browser solche „nackten“ XML-Daten anders darstellt. U.U. müssen Sie im Browser die Funktion „Seitenquelltext anzeigen“ o.ä. aufrufen.

Geloggt wird dabei:

```
Started GET "/phone_models.xml" [...]
  Processing by PhoneModelsController#index as XML
[...]
```

Das XML-Format sieht etwa wie folgt aus (hier etwas gekürzt):

```
<?xml version="1.0" encoding="UTF-8"?>
<phone-models type="array">

    <phone-model>
        <id type="integer">4</id>
        <name>Snom 360</name>
        <manufacturer-id type="integer">1</manufacturer-id>
        <number-of-keys type="integer">12</number-of-keys>
        <!-- ... -->
    </phone-model>

    <phone-model>
        <id type="integer">5</id>
        <name>Snom 370</name>
        <manufacturer-id type="integer">1</manufacturer-id>
        <number-of-keys type="integer">12</number-of-keys>
        <!-- ... -->
    </phone-model>

    <!-- ... -->

</phone-models>
```

Betrachten wir nun mal im Überblick welche URL-Pfade und welche HTTP-Methoden es zu der Ressource „`phone_models`“ in der REST-Architektur (siehe *http://de.wikipedia.org/wiki/Representational_State_Transfer*) gibt. Zuerst die für das HTML-Format. Die Endung „`.html`“ ist dabei optional und wird üblicherweise nicht mit angegeben.

URL-Pfad	HTTP-Methode	Controller-Aktion	Datenbank
/phone_models	GET	index	SELECT „
/phone_models/new	GET	new	- *
/phone_models	POST	create	INSERT „
/phone_models/1	GET	show	SELECT „ WHERE „
/phone_models/1/edit	GET	edit	SELECT „ WHERE „ *
/phone_models/1	PUT	update	UPDATE „ WHERE „
/phone_models/1	DELETE	destroy	DELETE „ WHERE „

Anmerkung

* Die URL-Pfade /phone_models/new und /phone_models/1/edit, die per GET aufgerufen werden, sind eine Besonderheit für das HTML-Interface im Browser. Unter diesen URLs wird ein Formular zur Eingabe dargestellt. Erst beim Absenden des Formulars wird dann die manipulierende Operation (Erzeugen / Verändern) auf der Datenbank vorgenommen.

Die Anhänge new und edit werden übrigens „Modifikatoren“ (*modifiers*) genannt.

Anmerkung

Die meisten Web-Browser unterstützen nur die HTTP-Methoden GET und POST, nicht jedoch PUT und DELETE. Ruby on Rails emuliert daher im Browser PUT und DELETE indem die HTTP-Methode POST genutzt wird, jedoch in einem nicht sichtbaren Parameter (_method) die eigentliche Methode übertragen wird. Dies geschieht automatisch und für den Anwender transparent.

Die gleichen URLs, jedoch mit der Endung „.xml“, gibt es – wie für das HTML-Format – auch für das XML-Format.

URL-Pfad	HTTP-Methode	Controller-Aktion	Datenbank
/phone_models.xml	GET	index	SELECT „
/phone_models.xml	POST	create	INSERT „
/phone_models/1.xml	GET	show	SELECT „ WHERE „
/phone_models/1.xml	PUT	update	UPDATE „ WHERE „
/phone_models/1.xml	DELETE	destroy	DELETE „ WHERE „

Anmerkung

Details zu den HTTP-Methoden GET, POST, PUT und DELETE findet man übrigens im RFC das HTTP/1.1 spezifiziert: *http://tools.ietf.org/html/rfc2616#section-9*

3.2. curl

Natürlich läßt sich die API mit den bekannten HTTP-Tools wie curl oder wget benutzen.

```
debian:~# curl http://localhost:3000/phone_models.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<phone-models type="array">
  <phone-model>
    <id type="integer">4</id>
    <name>Snom 360</name>
    <manufacturer-id type="integer">1</manufacturer-id>
    <number-of-keys type="integer">12</number-of-keys>
    <!-- ... -->
  </phone-model>
  <phone-model>
    <id type="integer">5</id>
    <name>Snom 370</name>
    <manufacturer-id type="integer">1</manufacturer-id>
    <number-of-keys type="integer">12</number-of-keys>
    <!-- ... -->
  </phone-model>
  <!-- ... -->
</phone-models>
```

Im folgenden Beispiel verwenden wir die Ressource `phones` statt `phone_models`.

Liste aller Telefone (`index`):

```
debian:~# curl -D - http://localhost:3000/phones.xml

HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
[...]

<?xml version="1.0" encoding="UTF-8"?>
<phones type="array">
  <phone>
    <created-at type="datetime">2011-01-10T14:42:41Z</created-at>
    <http-password nil="true"></http-password>
    <http-user nil="true"></http-user>
    <id type="integer">1</id>
    <ip-address nil="true"></ip-address>
    <last-ip-address nil="true"></last-ip-address>
    <mac-address>001565123456</mac-address>
    <phone-model-id type="integer">15</phone-model-id>
    <updated-at type="datetime">2011-01-10T14:42:41Z</updated-at>
  </phone>
  <phone>
    <created-at type="datetime">2011-01-10T14:42:41Z</created-at>
    <http-password nil="true"></http-password>
    <http-user nil="true"></http-user>
    <id type="integer">2</id>
    <ip-address nil="true"></ip-address>
    <last-ip-address nil="true"></last-ip-address>
    <mac-address>000413123456</mac-address>
    <phone-model-id type="integer">4</phone-model-id>
    <updated-at type="datetime">2011-01-10T14:42:41Z</updated-at>
  </phone>
  <!-- ... -->
</phones>
```

Geloggt wird dabei:

```
Started GET "/phones.xml" [...]
  Processing by PhonesController#index as XML
  Phone Load (3.4ms)  SELECT "phones".* FROM "phones"
Completed 200 OK [...]
```

Ein einzelnes Telefon abfragen (`show`):

```
debian:~# curl -D - http://localhost:3000/phones/1.xml

HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
[...]

<?xml version="1.0" encoding="UTF-8"?>
<phone>
  <created-at type="datetime">2011-01-10T14:42:41Z</created-at>
  <http-password nil="true"></http-password>
  <http-user nil="true"></http-user>
  <id type="integer">1</id>
  <ip-address nil="true"></ip-address>
  <last-ip-address nil="true"></last-ip-address>
  <mac-address>001565123456</mac-address>
  <phone-model-id type="integer">15</phone-model-id>
  <updated-at type="datetime">2011-01-10T14:42:41Z</updated-at>
</phone>
```

Geloggt wird dabei:

```
Started GET "/phones/1.xml" [...]
  Processing by PhonesController#show as XML
  Parameters: {"id"=>"1"}
  Phone Load (0.7ms)  SELECT "phones".* FROM "phones" WHERE ("phones"."id" = 1)
 LIMIT 1
Completed 200 OK [...]
```

Nun wollen wir ein neues Telefon anlegen (`create`). Nehmen wir mal ein Snom 360. Zum Anlegen des Telefons müssen wir die `phone-model-id` kennen. Diese bekommen wir wie weiter oben aus der XML-Liste der `phone_models`. Das Modell „Snom 360“ hat die `phone-model-id` 4.

Probieren wir das mal aus:

```
debian:~# curl -D - -X POST \
  -H 'Content-Type: application/xml' \
  --data-binary \
    '<?xml version="1.0" encoding="UTF-8"?>
    <phone>
      <mac-address>00:04:13:00:00:01</mac-address>
      <phone-model-id>4</phone-model-id>
    </phone>' \
  http://localhost:3000/phones.xml

HTTP/1.1 201 Created
Location: http://localhost:3000/phones/7
Content-Type: application/xml; charset=utf-8
[...]
```

```
<?xml version="1.0" encoding="UTF-8"?>
<phone>
  <created-at type="datetime">2011-01-10T15:16:50Z</created-at>
  <http-password nil="true"></http-password>
  <http-user nil="true"></http-user>
  <id type="integer">7</id>
  <ip-address nil="true"></ip-address>
  <last-ip-address nil="true"></last-ip-address>
  <mac-address>000413000001</mac-address>
  <phone-model-id type="integer">4</phone-model-id>
  <updated-at type="datetime">2011-01-10T15:16:50Z</updated-at>
</phone>
```

Das neue Telefon hat die ID 7 bekommen.

Geloggt wird dabei:

```
Started POST "/phones.xml" [...]
  Processing by PhonesController#create as XML
  Parameters: {"phone"=>{"mac_address"=>"00:04:13:00:00:01",
 "phone_model_id"=>"4"}}
  Phone Load (0.1ms)  SELECT "phones"."id" FROM "phones" WHERE
 ("phones"."mac_address" = '00:04:13:00:00:01') LIMIT 1
  PhoneModel Load (0.1ms)  SELECT "phone_models"."id" FROM "phone_models" WHERE
 ("phone_models"."id" = 4) LIMIT 1
  Oui Load (0.2ms)  SELECT "ouis".* FROM "ouis" WHERE ("ouis"."value" = '000413')
 LIMIT 1
  CACHE (0.0ms)  SELECT "ouis".* FROM "ouis" WHERE ("ouis"."value" = '000413')
 LIMIT 1
  Manufacturer Load (0.2ms)  SELECT "manufacturers".* FROM "manufacturers" WHERE
 ("manufacturers"."id" = 1) LIMIT 1
  PhoneModel Load (0.5ms)  SELECT "phone_models".* FROM "phone_models" WHERE
 ("phone_models"."id" = 4) LIMIT 1
  CACHE (0.0ms)  SELECT "manufacturers".* FROM "manufacturers" WHERE
 ("manufacturers"."id" = 1) LIMIT 1
  AREL (0.6ms)  INSERT INTO "phones" ("mac_address", "phone_model_id",
 "ip_address", "last_ip_address", "http_user", "http_password", "created_at",
 "updated_at") VALUES ('000413000001', 4, NULL, NULL, NULL, NULL, '2011-01-10 15
:16:50.085697', '2011-01-10 15:16:50.085697')
Completed 201 Created [...]
```

Das Telefon bearbeiten (`update`), z.B. die IP-Adresse verändern:

```
debian:~# curl -D - -X PUT \
  -H 'Content-Type: application/xml' \
  --data-binary \
    '<?xml version="1.0" encoding="UTF-8"?>
    <phone>
      <ip-address>10.0.0.55</ip-address>
    </phone>' \
  http://localhost:3000/phones/7.xml

HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
[...]
```

Geloggt wird dabei:

```
Started PUT "/phones/7.xml" [...]
  Processing by PhonesController#update as XML
  Parameters: {"phone"=>{"ip_address"=>"10.0.0.55"}, "id"=>"7"}
  Phone Load (0.7ms)  SELECT "phones".* FROM "phones" WHERE ("phones"."id" = 7)
 LIMIT 1
  Phone Load (0.2ms)  SELECT "phones"."id" FROM "phones" WHERE
 ("phones"."mac_address" = '000413000001') AND ("phones".id <> 7) LIMIT 1
  Phone Load (0.1ms)  SELECT "phones"."id" FROM "phones" WHERE
 ("phones"."ip_address" = '10.0.0.55') AND ("phones".id <> 7) LIMIT 1
  PhoneModel Load (0.7ms)  SELECT "phone_models"."id" FROM "phone_models" WHERE
 ("phone_models"."id" = 4) LIMIT 1
  Oui Load (0.1ms)  SELECT "ouis".* FROM "ouis" WHERE ("ouis"."value" = '000413')
 LIMIT 1
  CACHE (0.0ms)  SELECT "ouis".* FROM "ouis" WHERE ("ouis"."value" = '000413')
 LIMIT 1
  Manufacturer Load (0.1ms)  SELECT "manufacturers".* FROM "manufacturers" WHERE
 ("manufacturers"."id" = 1) LIMIT 1
  PhoneModel Load (0.5ms)  SELECT "phone_models".* FROM "phone_models" WHERE
 ("phone_models"."id" = 4) LIMIT 1
  CACHE (0.0ms)  SELECT "manufacturers".* FROM "manufacturers" WHERE
 ("manufacturers"."id" = 1) LIMIT 1
  AREL (0.4ms)  UPDATE "phones" SET "ip_address" = '10.0.0.55', "updated_at" =
 '2011-01-10 15:20:18.588354' WHERE ("phones"."id" = 7)
Completed 200 OK [...]
```

Das Telefon löschen (`destroy`):

```
debian:~# curl -D - -X DELETE \
  http://localhost:3000/phones/7.xml

HTTP/1.1 200 OK
[...]
```

Geloggt wird dabei:

```
Started DELETE "/phones/7.xml" [...]
  Processing by PhonesController#destroy as XML
  Parameters: {"id"=>"7"}
  Phone Load (0.6ms)  SELECT "phones".* FROM "phones" WHERE ("phones"."id" = 7)
 LIMIT 1
  SipAccount Load (0.3ms)  SELECT "sip_accounts".* FROM "sip_accounts" WHERE
 ("sip_accounts".phone_id = 7) ORDER BY position
  RebootRequest Load (0.2ms)  SELECT "reboot_requests".* FROM "reboot_requests"
 WHERE ("reboot_requests".phone_id = 7) ORDER BY start
  AREL (0.2ms)  DELETE FROM "phones" WHERE ("phones"."id" = 7)
Completed 200 OK [...]
```

Wir sehen daß prinzipiell alle Operationen der API mit einem Tool wie `curl` möglich sind, auch wenn das dann auf der Shell etwas unübersichtlich wird. Schöner geht es wenn man eine fertige Bibliothek benutzt, die dann tatsächlich auch die XML-Daten versteht und ein komfortables Interface bereitstellt, wie z.B. `ActiveResource` in Ruby (*Abschnitt 3.3, „ActiveResource in Ruby“*).

3.3. ActiveResource in Ruby

Die in *Abschnitt 3.1, „API im REST-Stil“* kennengelernte XML-API läßt sich gut aus eigenen Skripten heraus aufrufen – man nennt das auch „konsumieren“ (*consume*). Dazu wird `ActiveResource` genutzt (siehe *http://api.rubyonrails.org/classes/ActiveResource/Base.html*).

Anmerkung

Sie können ein Skript zum Testen einfach auf dem gleichen Rechner anlegen (z.B. in einem neuen Terminal-Fenster) oder auf einem anderen Rechner.

```
debian:~# cd /var/tmp/
debian:/var/tmp# touch api-test.rb
debian:/var/tmp# chmod u+x api-test.rb
debian:/var/tmp# vim api-test.rb
```

Beispiel-Skript das die API verwendet:

```
#! /usr/bin/env ruby

require 'active_resource'

class PhoneModel < ActiveResource::Base
    self.site = "http://localhost:3000/"
    self.element_name = 'phone_model'
end

# alle Telefon-Modelle abrufen:
phone_models = PhoneModel.all
phone_models.each { |pm|
    puts "PhoneModel ID: #{pm.id}, name: #{pm.name}"
}
```

Rufen wir das Skript mal auf:

```
debian:/var/tmp# ./api-test.rb
[...]
PhoneModel ID: 4, name: Snom 360
PhoneModel ID: 5, name: Snom 370
[...]
debian:/var/tmp#
```

Auf dem Server sehen wir dabei im Log wie erwartet folgende Meldung:

```
Started GET "/phone_models.xml" for 127.0.0.1 [...]
  Processing by PhoneModelsController#index as XML
  PhoneModel Load (43.6ms)  SELECT "phone_models".* [...]
Completed 200 OK in 114ms (Views: 45.3ms | ActiveRecord: 43.6ms)
```

Die per API bereitgestellten Ressourcen lassen sich also beispielsweise in eigenen Skripten ganz normal verwenden.

Weitere Parameter und Dokumentation zur Verwendung von ActiveResource finden Sie in der Dokumentation von Ruby on Rails unter folgender URL:

http://api.rubyonrails.org/classes/ActiveResource/Base.html

Kapitel 4. Frontend

#######

Alle für den Betrieb notwendigen Aufgaben lassen sich über das Webfrontend ausführen.

Sobald der Server *gestartet* ist, ist das Frontend über `http://localhost:3000` erreichbar.

4.1. Phone Manufacturers

4.1.1. List all manufacturers

Über diesen Punkt können alle Telefon-Hersteller aufgelistet, bearbeitet und gelöscht werden.

Neue Hersteller können ebenfalls angelegt werden. Bitte beachten Sie für das weitere Vorgehen, dass kein Telefon angelegt werden kann, für das noch kein Hersteller angelegt wurde.

4.1.2. Create new manufacturer

An dieser Stelle können neue Hersteller in die Liste der bekannten Hersteller aufgenommen werden.

Benötigte Informationen zum Anlegen eine neuen Herstellers sind:

1. Name für die interne Verwendung
2. IEEE-Name des Herstellers (*http://de.wikipedia.org/wiki/MAC-Adresse#Herstellerkennungen)*[2]
3. URL des Herstellers (nicht zwingend erforderlich)

4.2. Phone Models

Hier können die unterschiedlichen Gerätetypen verwaltet werden.

4.2.1. List all phone models

Hier werden alle bereits angelegten Telefonmodelle angezeigt.

Modelle können bearbeitet, gelöscht und angelegt werden.

Um ein neues Modell anlegen zu können muss der Hersteller bereits angelegt sein, und Sie benötigen zwingend folgende Informationen:

1. Name des Telefonmodells

Um ein Telefonmodell vollständig anlegen zu könnnen sind folgende Informationen notwendig:

1. Maximale Anzahl der SIP-Accounts (default = 1)
2. Maximale Anzahl der belegbaren Tasten (default = 0)

Warnung

Unter Umständen reicht es nicht ein neues Telfonmodell anzulegen. Der View der die Konfiguration für das Telefon ausgibt muss unter Umständen auch angepasst werden.

4.2.2. Create a new phone model

Hier kann ein neues Telefonmodell direkt angelegt werden.

4.3. Phones

Hier befinden Sie sich am zentralen Punkt des Provisioning-Servers. Von hier starten die meisten alltäglichen Workflows.

4.3.1. List all phones

Alle bereits angelegten Telefone werden hier mit (wenn vorhanden) den zugeordneten SIP-Accounts angezeigt.

Einen neuen SIP-Account legen Sie am einfachsten an indem Sie über das ensprechende Telefon einsteigen. Dadurch haben Sie automatisch alle das Telefon betreffenden Informationen richtig ausgefüllt.

Über "create new sip account" können Sie einen neuen SiIP-Account für das jeweilige Telefon anlegen.

Sie benötigen mindestens folgende Informationen:

1. Name (eindeutige Zuordnung für den Account)
2. User (SIP User-ID)

In den meisten Fällen werden Sie noch zusätzlich diese Informationen benötigen:

1. Auth. user (Benutzer für Authentifizierung beim Registrar)
2. Password (Passwort zugehörig zu Auth. user)
3. Registrar (SIP-Registrar an dem Sie sich anmelden wollen)
4. Registrar port (Standard: 5060)

Folgende Parameter sind optional:

1. Outbound proxy (Proxy für alle ausgehenden Gespräche)
2. Outbound proxy port (zugehöriger Port)
3. Realm (siehe *http://www.voip-info.org/wiki/view/SIP+Authentication*)
4. Screen name (auf dem Telefon angezeigter Name)
5. Display name (beim Angerufenen angezeigter Name)
6. Registration expiry time (Zeitraum nach der eine erneute Registrierung erfolgen muss)
7. DTMF mode (Art der DTMF Übertragung)
8. Remote password (Passwort zur Authentifizierung bei eingehnden Gesprächen)
9. Codecs (erlaubte Sprachcodecs für diesen Account)

4.3.2. Create new phone

Um ein neues Telefon anzulegen benötigen Sie mindestens folgende Informationen:

1. MAC address (MAC-Adresse des Telefons)
2. Phone model (Modell des Telefons)

Optional können Sie folgende Informationen angeben:

1. IP address (IP-Adresse des Telefons)
2. HTTP user (User für Zugriff auf das Webinterface des Telefons)
3. HTTP password (Passwort für Zugriff auf das Webinterface des Telefons

4.3.3. Phone keys

Die auf den einzelnen Telefonen belegbaren Tasten sind nicht direkt zugängliche, da sie immer von einem bereits bestehnden Sip Account und einem Telefon abhängen. Um Tastenbelegungen aufzulisten oder neue Tasten anzulegen gehen Sie bitte folgenden Weg:

- List all phones
- Telefon auswählen (mit bereits angelegtem Sip Account!)

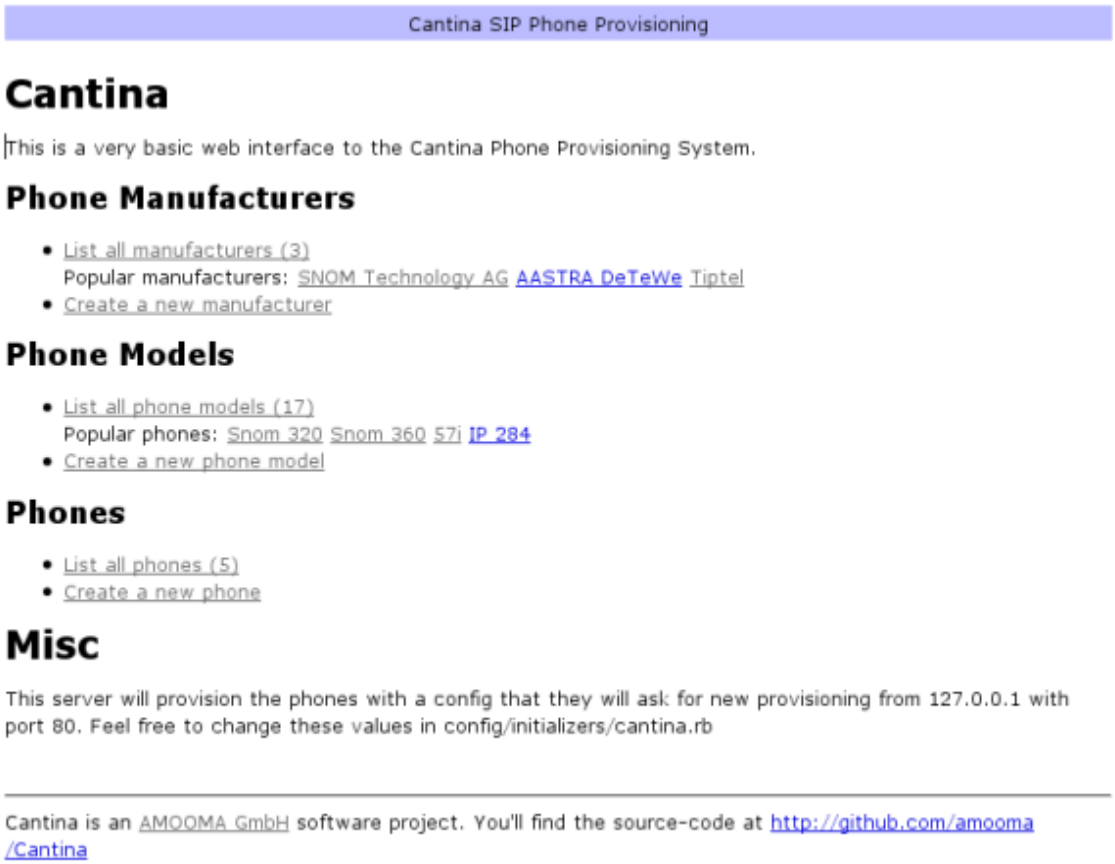
In der Liste der angelegten Sip Accounts finden Sie entweder den Link "Phone Keys" (wenn bereits Tasten angelegt wurden) oder den Link "New phone key" über den Sie eine neue Taste anlegen können.

Um eine neu Taste anlegen zu können benötigen Sie folgende Informationen:

1. Phone model key (welche Taste wollen Sie belegen => Auswahlliste)
2. Phone key function definition (Funktion der Taste z.B Speed dial = Kurzwahl)
3. Value (Wert der Taste. Unser Beispiel 42 = Kurzwahl 42)
4. Label (Beschriftung für Taste. z.B Aastra 57i)

4.3.4. Beispiel

Im folgenden Beispiel werden wir ein neues Telefon anlegen (Snom 370), dem Telefon einen neuen Sip Account zuweisen und eine Kurzwahltaste belegen.



Cantina SIP Phone Provisioning

Cantina

This is a very basic web interface to the Cantina Phone Provisioning System.

Phone Manufacturers

- List all manufacturers (3)
  Popular manufacturers: SNOM Technology AG AASTRA DeTeWe Tiptel
- Create a new manufacturer

Phone Models

- List all phone models (17)
  Popular phones: Snom 320 Snom 360 57i IP 284
- Create a new phone model

Phones

- List all phones (5)
- Create a new phone

Misc

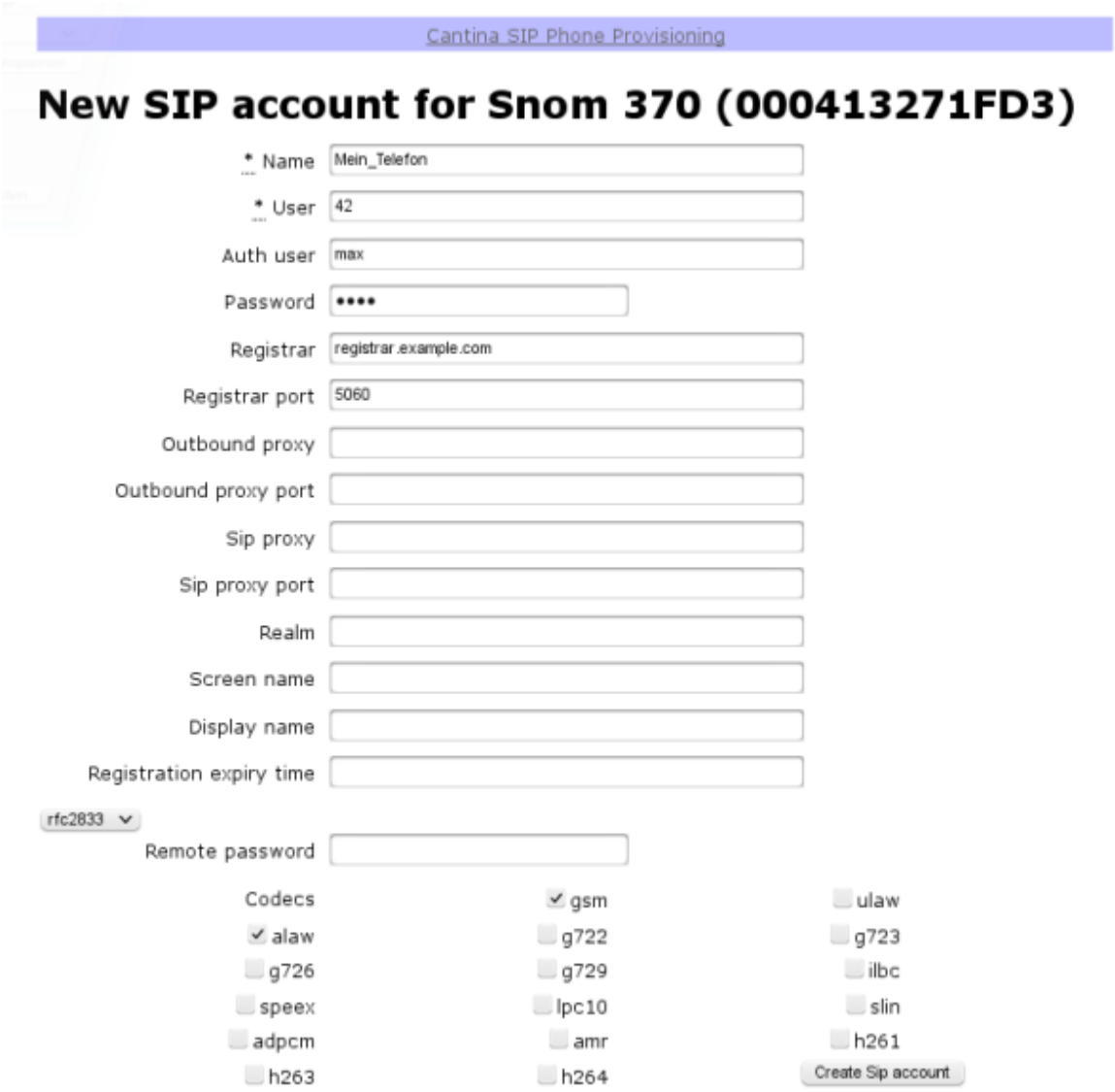
This server will provision the phones with a config that they will ask for new provisioning from 127.0.0.1 with port 80. Feel free to change these values in config/initializers/cantina.rb

---

Cantina is an AMOOMA GmbH software project. You'll find the source-code at http://github.com/amooma/Cantina


Cantina SIP Phone Provisioning
New phone
MAC address
000413271FDB
Phone model
Snom 370
IP address
HTTP user
HTTP password
Create Phone
Back
Cantina is an AMOOMA GmbH software project. You'll find the source-code at http://github.com/amooma/Cantina


Cantina SIP Phone Provisioning
Phone was successfully created.
MAC address: 000413271FD3
Phone model: Snom 370
IP address:
Edit
SIP Accounts
Create a new sip account for this phone.
Name
Phone
MAC Adress
Auth. user
User
Password
Registrar
Phone keys
Cantina is an AMOOMA GmbH software project. You'll find the source-code at http://github.com/amooma/Cantina


Cantina SIP Phone Provisioning
New SIP account for Snom 370 (000413271FD3)
* Name
Mein_Telefon
* User
42
Auth user
max
Password
••••
Registrar
registrar.example.com
Registrar port
5060
Outbound proxy
Outbound proxy port
Sip proxy
Sip proxy port
Realm
Screen name
Display name
Registration expiry time
rfc2833
Remote password
Codecs
gsm
ulaw
alaw
g722
g723
g726
g729
ilbc
speex
lpc10
slin
adpcm
amr
h261
h263
h264
Create Sip account

Cantina SIP Phone Provisioning

Sip account was successfully created.

MAC address: 000413271FD3

Phone model: Snom 370

IP address:

Edit

SIP Accounts

Create a new sip account for this phone.

Name	Phone	MAC Adress	Auth. user	User	Password	Registrar	Phone keys		
Mein_Telefon	Snom 370	000413271FD3	max	42	asda	registrar.example.com	New phone key	Edit	Destroy

Cantina is an AMOOMA GmbH software project. You'll find the source-code at http://github.com/amooma/Cantina

Cantina SIP Phone Provisioning

New phone key

Phone model key

P 001 (fkey[0]) (pos. 0) for Snom 370

Phone key function definition

Speed dial

Value

23

Label

Create Phone key

Cantina is an AMOOMA GmbH software project. You'll find the source-code at http://github.com/amooma/Cantina

Phone key was successfully created.

Phone: 000413271FD3

Name: Mein_Telefon

Auth. user: max

User: 42

Password: asda

Registrar: registrar.example.com

Registrar port: 5060

Outbound proxy:

Outbound proxy port:

SIP proxy:

SIP proxy port:

Realm:

Screen name:

Display name:

Registration expiry time:

DTMF mode: rfc2833

Remote password:

Position: 1

Codecs:

- gsm
- alaw

Create new phone key

Sip account	Phone model key	Phone key function definition	Value	Label			
42@registrar.example.com (Mein_Telefon) (ID 2)	P 001 (fkey[0]) (pos.: 0)	Speed dial	23		Show	Edit	Destroy