

▼ Gemeinschaft4

Authorgroup Stefan Wintermeyer stefan.wintermeyer@amooma.de

Philipp Kempgen philipp.kempgen@amooma.de

Sascha Daniels sascha.daniels@amooma.de

Amooma GmbH <http://www.amooma.de>

▼ Chapter 1: Installation

Nachfolgend wird beschrieben wie zuerst Ruby on Rails installiert wird und dann das eigentliche Gemeinschaft4-System. Die folgende Beschreibung bezieht sich auf die Installation auf einem Debian 5 System. Für ein Debian 6 System steht ein Script zur Verfügung. Eine Genaue Anleitung finden Sie im Kapitel "Installation auf Debian 6".

▼ 1 Installation von Ruby on Rails 3.0 mit RVM auf Debian 5.0

Authorgroup Stefan Wintermeyer stefan.wintermeyer@amooma.de

Philipp Kempgen philipp.kempgen@amooma.de

❖ Installation; von Ruby on Rails; auf Debian 5.0 ❖ Debian 5.0; Installation von Ruby on Rails ❖ Ruby; Installation ❖ Rails; Installation ❖ RVM (Ruby Version Manager) ❖ Ruby Version Manager [see RVM](#)

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 Netzininstallations-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 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
```

Note 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 aktua
Muss 6501kB an Archiven herunterladen. Nach dem Entpacken werden 16,1MB zusätzlich be
[...]
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 aktua
Muss 25,5MB an Archiven herunterladen. Nach dem Entpacken werden 80,1MB zusätzlich be
[...]
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.

Footnote siehe <http://packages.debian.org/lenny-backports/sqlite3>

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].

```

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 aktua
Muss 699kB an Archiven herunterladen. Nach dem Entpacken werden 1659kB zusätzlich bel
[...]
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) ...
[...]

```

▼ 1.2 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 6d48919a9a6984a38a576elec19496d24b2a2696
walk 6d48919a9a6984a38a576elec19496d24b2a2696
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 En
# 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 insta
# If you wish to have the 'pretty colors' again, set 'export rvm_pretty_print_flag=1'
```

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 inst
```

```
  ruby: aptitude install build-essential bison openssl libreadline5 libreadline-dev c
  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 following 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 loa
```

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
```

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 yo

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
```

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](http://irc.freenode.net) 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 'wayneeseguine' 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 knifflig, 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 $HOME/.rvm/scripts/rvm'
xyz@debian:~$ perl -pi -e 's/\[ -z \"$PS1\" \] \&& return/if \[\[ -n \"$PS1\" \]\]'
xyz@debian:~$ echo 'fi' >> .bashrc
xyz@debian:~$
```

Important Jetzt müssen Sie das Terminal schließen und ein neues Terminal öffnen

Footnote `alternativ: exec bash`

, 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 (wayneeseguine@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

Footnote siehe <http://rvm.beginrescueend.com/rubies/list/>

```
:
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*
[...]
```

Note  Interpreter; Ruby  Ruby; Interpreter  Es gibt für Ruby verschiedene Interpreter. Relevant ist hier für uns der ganz normale  MRI (Matz's Ruby Interpreter)  Matz's Ruby Interpreter see  MRI  MRI (*Matz's Ruby Interpreter*)

Footnote siehe [http://de.wikipedia.org/wiki/Ruby_\(Programmiersprache\)#Implementierungen](http://de.wikipedia.org/wiki/Ruby_(Programmiersprache)#Implementierungen)
http://en.wikipedia.org/wiki/Ruby_MRI

, dessen Kern seit Ruby 1.9 die Ruby-VM namens  YARV (Yet Another Ruby VM)  **YARV** (*Yet Another Ruby VM*)

Footnote siehe <http://en.wikipedia.org/wiki/YARV>

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 c

  % 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
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

Footnote 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_(variable))

```
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
```

Note 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.

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

▼ 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 [rails-1.9.2-mit-rvm-debian5](#) durch.

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

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 activessupport-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 activessupport-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
```

▼ 2 Installation von Gemeinschaft4

Authorgroup Amooma GmbH <http://www.amooma.de>

Important Voraussetzung: Installation von Ruby on Rails 3 auf Debian 5 (siehe [rails3-install-debian](#)).

Important In Kombination mit dem Cantina-Provisioning-Dienst als Single-Server-System (beides auf einem Server) ist auch die Installation von Cantina Voraussetzung. Die

Installationsanleitung dazu finden Sie in der Dokumentation zu Cantina. Bitte beachten Sie daß der Cantina-Dienst und der Gemeinschaft4-Dienst dann *nicht* auf dem gleichen Port gestartet werden können und dürfen!

Note Beim normalen Betrieb als Single-Server-System (d.h. alles auf einem Server) ist natürlich nur einmal die Installation von Ruby erforderlich.

Note Informationen zur Installation bzw. Konfiguration des SIP-Proxy (Kamailio) werden von der SIP-Proxy-Steuerungsapplikation („sipproxy“) bereitgestellt. Die Applikation „sipproxy“ ist zuständig für die Steuerung des SIP-Proxy-Dienstes.

Gemeinschaft4 ist verfügbar auf <https://github.com/amooma/Gemeinschaft4> und kann von dort wie folgt per Git installiert werden:

```
debian:~# cd /usr/src
debian:/usr/src# git clone git@github.com:amooma/Gemeinschaft4.git
Cloning into Gemeinschaft4...
[...]
```

2.1 Anlegen der Datenbank

Initial muß die Datenbank des Gemeinschaft4-Servers angelegt werden. Wechseln Sie dazu in das Installations-Verzeichnis (abhängig von Ihrer Installation, beispielsweise /home/gemeinschaft4/):

```
debian:/usr/src# cd /usr/src/Gemeinschaft4/
```

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

```
debian:/usr/src/Gemeinschaft4# rake db:setup
(in /usr/src/Gemeinschaft4)
[...]
```

Die Datenbank ist nun angelegt.

2.2 Gemeinschaft4-Dienst starten

Um den Gemeinschaft4-Dienst zu starten wechseln Sie in das Installations-Verzeichnis (abhängig von Ihrer Installation, beispielsweise /home/gemeinschaft4/):

```
debian:~# cd /usr/src/Gemeinschaft4/
```

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

```
debian:/usr/src/Gemeinschaft4# 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
^C
```

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/Gemeinschaft4# 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.↵
```

▼ Chapter 2: Installation unter Debian Squeeze

Authorgroup Amooma GmbH <http://www.amooma.de>

Sascha Daniels

Amooma GmbH

▼ 1 Installation des Basis Systems

Als Basis System kommt eine Minimal Installation von Debian 6 ("Squeeze") zum Einsatz.

Bitte wählen Sie bei der Paketauswahl alle Pakete ausser "ssh" ab. Das Paket erleichtert vor allem in einer virtuellen Installation die weiteren Schritte.

▼ 2 Installation von Gemeinschaft4

Alle für Gemeinschaft4 benötigten komponenten können über das im github.com Repository vorhandene Installationsscript installiert werden.

Für die Installation des Demo Systems werden alle drei Ruby Applikationen in einem Webserver gestartet. Hierfür kommt lighttpd verwendet. Die einzelnen Applikationen sind weiterhin über die bekannte Ports erreichbar:

- Gemeinschaft4 3000
- Cantina 3001
- sipproxy 3002

Als erstes laden Sie sich bitte die Datei install.sh von <https://github.com/amooma/Gemeinschaft4> herunter und transferrieren Sie die Datei auf den Server (z.B. per scp).

Bitte achten Sie darauf die Datei aus dem richtigen Branch (NICHT master) zu laden. Bitte verwenden Sie immer die aktuellste Version (zum jetzigen Zeitpunkt 1.0). Nachdem Sie install.sh heruntergeladen haben, gehen Sie bitte wie folgt vor:

```
scp install.sh root@GEMEINSCHAFT:/tmp/
ssh root@GEMEINSCHAFT
cd /tmp
chmod +x install.sh
./install.sh
```

Bitte führen Sie install.sh unbedingt als "root" aus, da sonst die globalen gems nicht installiert werden können.



Das Script zum Zeitpunkt der Dokumentation. Bitte IMMER die aktuelle Version verwenden.

```
#!/bin/bash

echo -e "Please enter your github.com username\n"
read USER

echo -e "Please enter your github.com password\n"
read PASS

aptitude update
aptitude -y install curl git-core patch file \
  build-essential bison \
  openssl zlib1g-dev libssl-dev libreadline5-dev libxml2-dev \
  libreadline5-dev libxml2-dev sqlite3 libsqlite3-dev libxslt-dev \
  libfcgi-ruby1.9.1 libfcgi-dev lighttpd libpcre3-dev libyaml-dev \
  nmap

cd /usr/local/src
wget http://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.2-p180.tar.bz2
tar -xvjf ruby-1.9.2-p180.tar.bz2
cd ruby-1.9.2-p180
./configure
make
make install

gem update
gem install rake
gem install bundler

cd /opt
PROJECTS="Gemeinschaft4
Cantina
sipproxy";
for i in $PROJECTS
do
cd /opt

  git clone https://$USER:$PASS@github.com/amooma/$i.git
  cd /opt/$i

  bundle install
```

```

rake db:migrate RAILS_ENV=production
rake db:seed RAILS_ENV=production

cd /opt/$i/public
bundle install --path .
chown -R www-data /opt/$i
done

aptitude -y install gcc flex bison libmysqlclient-dev make \
  libcurl4-openssl-dev libpcre3-dev libpcre++-dev

cd /usr/local/src
git clone git://git.sip-router.org/sip-router kamilio
cd kamilio
git checkout -b 3.1 origin/3.1
make FLAVOUR=kamilio include_modules="dbtext dialplan" cfg
make PREFIX="/opt/kamilio-3.1" FLAVOUR=kamilio include_modules="db_text dialplan" c
make all
make install

cp /etc/lighttpd/lighttpd.conf /etc/lighttpd/lighttpd.conf.DIST
cp /opt/Gemeinschaft4/misc/lighttpd.conf /etc/lighttpd/lighttpd.conf
/etc/init.d/lighttpd restart
cp -r /opt/Gemeinschaft4/misc/kamilio/etc/* /opt/kamilio-3.1/etc/kamilio/
chgrp www-data /opt/kamilio-3.1/etc/kamilio/db_text/subscriber
chgrp www-data /opt/kamilio-3.1/etc/kamilio/db_text/dbaliases
chmod g+rw /opt/kamilio-3.1/etc/kamilio/db_text/subscriber
chmod g+rw /opt/kamilio-3.1/etc/kamilio/db_text/dbaliases
cp /opt/Gemeinschaft4/misc/etc/init.d/kamilio /etc/init.d/
update-rc.d kamilio defaults
/etc/init.d/kamilio start

```

Sobald das Script erfolgreich abgearbeitet wurde, können Sie auf das Webinterface von Gemeinschaft4 über `http://IP_DES_SERVERS:3000` zugreifen.

▼ Chapter 3: Frontend

Authorgroup Amooma GmbH <http://www.amooma.de>

Sascha Daniels

Amooma GmbH

▼ 1 Erste Schritte nach der Installation

Die initiale Setup-Routine leitet Sie automatisch zu den notwendigen Einstellungen. Wir werden in diesem Kapitel die Ersteinrichtung Schritt für Schritt durchgehen.

Beim ersten Aufruf des Gemeinschaft4-Web-Interfaces werden Sie automatisch zum Setup weitergeleitet.

Automatische Weiterleitung zum Setup

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amooma Gemeinschaft4 +

[Sign in](#)

Gemeinschaft4

Setup after installation

Create first user

Anlegen des ersten Administrators

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amooma Gemeinschaft4 +

[Sign in](#)

Gemeinschaft4

Create first User

Username

Password

Password confirmation

First name

Last name

Email

[Back](#)

Anmeldung mit dem gerade angelegten Account

[Sign in](#)

Gemeinschaft4

Sign in

Username

admin

Password

.....

☐ Remember me

[Forgot your password?](#)

Anlegen des ersten SIP-Servers

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Setup after installation

First User already created

[Create first SIP server](#)

[Create first provisioning server](#)

[Create first SIP proxy server](#)

Geben Sie die IP-Adresse des Rechners an auf dem Gemeinschaft4 installiert ist (NICHT 127.0.0.1!). Der Konfigurations-Port ist der Port auf dem der Rails-Server für die Kamailio-Konfiguration läuft.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

New SIP server

Host

192.168.110.66

Configuration port

3001

Create Sip server

[Back](#)

Der Server wurde ohne Fehler in der Datenbank angelegt.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Sip server was successfully created.

Sip server was successfully created.

Host: 192.168.110.66 3001

[Edit](#) | [Back](#)

Im nächsten Schritt muss der erste Provisioning-Server (Cantina) angelegt werden.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Setup after installation

First User already created

First SIP server already configured

[Create first provisioning server](#)

[Create first SIP proxy server](#)

Bitte geben Sie die IP-Adresse des Cantina-Servers an. Der Port ist wieder der Port auf dem der Rails-Server von Cantina läuft.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

New provisioning server

Host

Port

[Create Provisioning server](#)

[Back](#)

Der Provisionig-Server wurde fehlerfrei angelegt.

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amooma Gemeinschaft4 +

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Provisioning server was successfully created.

Provisioning server was successfully created.

Host: 192.168.110.66

Port: 3001

[Edit](#) | [Back](#)

Zuletzt muss noch ein SIP Proxy angelegt werden.

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amooma Gemeinschaft4 +

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Setup after installation

First User already created
First SIP server already configured
First provisioning server already configured
Create first SIP proxy server

Es wird wieder die IP des Servers benötigt. Ein Port für den Konfigurationsserver wird nicht benötigt.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

New SIP proxy

Host

Configuration port

Create Sip proxy

[Back](#)

Der SIP Proxy wurde fehlerfrei angelegt.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Sip proxy was successfully created.

Sip proxy was successfully created.

Host: 192.168.110.66

[Edit](#) | [Back](#)

Um einen SIP Account auf ein Telefon provisionieren zu können, muss zuerst ein auf "Cantina" angelegtes Telefon in Gemeinschaft4 angelegt werden.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

PBX User Manager

This is a basic web interface to GS4.

Users

- [List all users \(1\)](#)
- [Create a new user](#)

SIP accounts

- [List all SIP accounts \(0\)](#)
- [Create a new SIP account](#)

SIP phones

- [List all SIP phones \(0\)](#)
- [Create a new SIP phone](#)

SIP proxies

- [List all SIP proxies \(1\)](#)
- [Create a new SIP proxy](#)

SIP servers

- [List all SIP servers \(1\)](#)
- [Create a new SIP server](#)

Provisioning servers

- [List all provisioning servers \(1\)](#)
- [Create a new provisioning server](#)

Extensions

- [List all extensions \(0\)](#)
- [Create a new extension](#)

Das Telefon kann über die MAC Adresse identifiziert werden.

Signed in as "admin". [Sign out](#)

Gemeinschaft4

New SIP phone

Provisioning server

192.168.110.66:3001 (ID 2) ▾

Phone on prov. server

00156513EC2F ▾

00156513EC2F

000413296887

00085024387A

000413271FDB

000413271FD8

Das Telefon wurde erfolgreich angelegt.

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amoona Gemeinschaft4 +

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Sip phone was successfully created.

Sip phone was successfully created.

Provisioning server: 192.168.110.66:3001 (ID 2)

Phone ID on prov. server: 1

[Edit](#) | [Back](#)

Jetzt kann der erste SIP Account angelegt werden.

PBX User Manager

This is a basic web interface to GS4.

Users

- [List all users \(1\)](#)
- [Create a new user](#)

SIP accounts

- [List all SIP accounts \(0\)](#)
- [Create a new SIP account](#)

SIP phones

- [List all SIP phones \(1\)](#)
- [Create a new SIP phone](#)

SIP proxies

- [List all SIP proxies \(1\)](#)
- [Create a new SIP proxy](#)

SIP servers

- [List all SIP servers \(1\)](#)
- [Create a new SIP server](#)

Provisioning servers

- [List all provisioning servers \(1\)](#)
- [Create a new provisioning server](#)

Extensions

- [List all extensions \(0\)](#)
- [Create a new extension](#)

Das Feld User wird noch nicht benötigt. Das Feld Realm muss im Moment noch leer bleiben. Ein SIP phone muss nur ausgewählt werden, wenn der Account auf ein Telefon provisioniert werden soll. Für ein Softphone z.B. wird das Feld nicht benötigt.

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amooma Gemeinschaft4 +

Signed in as "admin". [Sign out](#)

Gemeinschaft4

New SIP account

User
▼

Auth name
test1

Password
test1

Realm

Phone number
111

SIP server
192.168.110.66 (ID 1) ▼

SIP proxy
192.168.110.66 (ID 1) ▼

SIP phone
(Prov. server 2 phone 1 (ID 1)) ▼

Create Sip account

[Back](#)

Der SIP Account wurde erfolgreich angelegt. Das Telefon wurde auf "Cantina" übertragen und der Account wurde auf "sipproxy" angelegt.

File Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Amooma Gemeinschaft4 +

Signed in as "admin". [Sign out](#)

Gemeinschaft4

Sip account was successfully created.

Sip account was successfully created.

User: -

Auth name: test1

Password: test1

Realm:

Phone number: 111

SIP server: ID 1 (192.168.110.66)

SIP proxy: ID 1 (192.168.110.66)

SIP phone: ID 1 (Prov. server 2 phone 1, ID 1)

Provisioning SIP account: test1@192.168.110.66

[Edit](#) | [Back](#)

Mit den hier gezeigten Schritten sind Sie in der Lage eine beliebige Anzahl von Telefonen anzulegen.

Der Menüpunkt "Extensions" hat im Moment noch keine Funktion. An dieser Stelle können in der Zukunft einem SIP Account eine beliebige Anzahl von weiteren Durchwahlen zugeordnet werden.

Die Dokumentation zu "Cantina" finden Sie im doc Verzeichnis des Projektes.

▼ Chapter 4: Die einzelnen Applikationen

Authorgroup Amooma GmbH <http://www.amooma.de>

Sascha Daniels

Amooma GmbH

▼ 1 Interaktion der Ruby Applikationen

Gemeinschaft4 besteht im Moment aus drei einzelnen ruby Applikationen.

▼ 1.1 Cantina

Cantina ist der zentrale Provisionierungsserver für SIP Telefone. Die Dokumentation zu dieser Applikation finden Sie im doc Verzeichnis des Projekts.

▼ 1.2 sipproxy

sipproxy ist eine Applikation, die eine beliebige Datenbank für die Userdaten und die benötigten Alias Daten für Kamailio befüllt.

▼ 1.2.1 Userdaten

Kamailio benötigt für jedes Telefon, das sich registrieren können soll einen Usernamen und ein Passwort. Diese Daten können sipproxy über einen Restfull Aufruf übergeben werden. Auch wenn sipproxy eine html Schnittstelle zur Verfügung stellt, sollten Änderungen immer nur über das Webinterface von Gemeinschaft4 vorgenommen werden.

▼ 1.2.2 Alias Daten

Per default ist jeder auf Kamailio angelegte User über seinen Usernamen erreichbar. Da ein Username auch alphanummerisch sein kann erschwert das das Wählen über eine Telfontastatur. Aus diesem Grund wird jedem User ein numerischer Alias zugeordnet. In unserem Beispiel erhält der User "test1" den Alias "111".

▼ 1.3 Gemeinschaft4

Gemeinschaft4 ist die zentrale Verwaltungsstelle für die gesamte Telefonanlage. Das Webinterface ist so ausgelegt, dass eine beliebige Anzahl von Cantina und sipproxy Instanzen verwaltet werden können.

In der Datenbank von Gemeinschaft werden alle benötigten Informationen gespeichert. An die weiteren Applikationen werden über Restfull Aufrufe nur die Informationen übermittelt, die für die jeweilige Applikation notwendig sind.

Genaue Informationen über das Zusammenspiel der Applikationen entnehmen Sie bitte der Datei "gemeinschaft4-interaktion.pdf".

▼ Chapter 5: Daten-Modell

Philipp Kempgen

Amooma GmbH

Stefan Wintermeyer

Amooma GmbH

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 Gemeinschaft4 gibt es folgende Daten-Modelle:

Modell	Dateiname
Authentication	<code>authentication.rb</code>
Extension	<code>extension.rb</code>
ProvisioningServer	<code>provisioning_server.rb</code>
SipAccount	<code>sip_account.rb</code>
SipPhone	<code>sip_phone.rb</code>
SipProxy	<code>sip_proxy.rb</code>
SipServer	<code>sip_server.rb</code>
User	<code>user.rb</code>

Note Daneben gibt es noch die Daten-Modelle *Cantina** in den Dateien `cantina_*.rb`. Es handelt sich dabei um als *ActiveResource* eingebundene Daten-Modelle aus dem Cantina-Provisioning-Server. Die Beschreibungen dazu finden sich in der Dokumentation zu Cantina. Diese Daten-Modelle werden dazu verwendet um die XML-API von Cantina anzusprechen und beispielsweise die SIP-Account auf Cantina anzulegen, zu ändern und zu löschen.

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.dot` in Form eines Entity-Relationship-Diagramms.

Footnote Information für Entwickler: Mit dem Befehl `./script/erd` im Wurzelverzeichnis von Gemeinschaft4 lassen sich die Dateien `ERD.dot` und `ERD.pdf` generieren.

Im folgenden wird erklärt was die einzelnen Daten-Modelle enthalten. Wir gehen hier zwecks leichter Nachschlagbarkeit in alphabetischer Reihenfolge vor.

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).

▼ 1 Authentication

Enthält die Authentifizierungen (*authentications*) der Benutzer (*users*). Diese Tabelle wird vom Authentifizierungssystem „Devise“ in Kombination mit „OmniAuth“ verwendet.

Verknüpfungen

● belongs_to  model-user

2 Extension

Enthält die Nebenstellen (*extensions*) der SIP-Konten (*SIP accounts*). Diese werden im momentanen System noch nicht verwendet.

Verknüpfungen

● has_many  model-sip-account

3 ProvisioningServer

Enthält die Provisionierungs-Server (*provisioning servers*), die jeweils eine bestimmte Menge der Telefone provisionieren. Im einfachsten Fall befindet sich in dieser Tabelle nur ein einziger Cantina-Provisionierungs-Server der alle Telefone provisioniert.






Verknüpfungen

● has_many  model-sip-phone

4 SipAccount

Enthält die SIP-Konten (*SIP accounts*) der Benutzer (*users*) und deren Zuweisung auf die SIP-Telefone (*SIP phones*).



Verknüpfungen

- belongs_to  model-user
- belongs_to  model-sip-phone
- belongs_to  model-sip-server
- belongs_to  model-sip-proxy
- belongs_to  model-extension

5 SipPhone

Enthält die SIP-Telefone (*SIP phones*). Ein SIP-Telefon befindet sich jeweils auf einem bestimmten Provisioning-Server. (Im einfachsten Fall gibt es nur genau einen Provisioning-Server der alle Telefone provisioniert.)

Verknüpfungen

- belongs_to  model-provisioning-server
- has_many  model-sip-account

6 SipProxy

Enthält die SIP-Proxies (*SIP proxies*) der SIP-Konten (*SIP accounts*). Im momentanen System muss der SIP Proxy identisch mit dem SIP Server gehalten werden.

Verknüpfungen

● has_many  model-sip-account

▼ 7 SipServer

Enthält die SIP-Server (*SIP servers*) der SIP-Konten (*SIP accounts*).

Verknüpfungen

• has_many  model-sip-account

▼ 8 User

Enthält die Benutzer (*users*), die jeweils ein oder mehrere SIP-Konten (*SIP accounts*) haben. Diese Tabelle wird vom Authentifizierungssystem „Devise“ verwendet.

Verknüpfungen

• has_many  model-sip-account

▼ Architektur, Sicherheitskonzept

Authorgroup Amooma GmbH <http://www.amooma.de>

Die Telefonanlage Gemeinschaft4 verwendet folgende Software-Komponenten in der Architektur:

- Ruby und Ruby on Rails
- FreeSwitch
- Kamilio
- MySQL

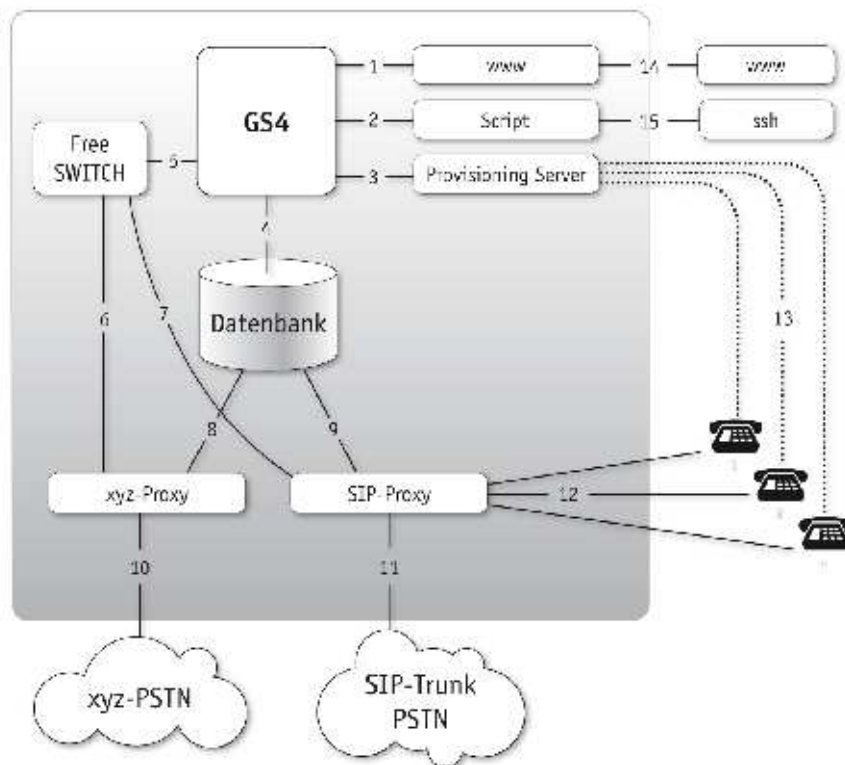
▼ 1 Übersicht

Authorgroup Amooma GmbH <http://www.amooma.de>

Gemeinschaft 4.0 ist modular aufgebaut und sieht von der Struktur eine einfache Skalierung auf mehrere Server und eine saubere Abtrennung der einzelnen Komponenten vor. Selbst wenn diese Möglichkeit der Skalierung für SiVoIP nicht notwendig ist (weil die Installation auf einem Server gemacht wird), birgt sie auch keine Nachteile und erhöht die saubere Abgrenzbarkeit. Durch eine saubere Abgrenzung ist eine bessere Testbarkeit gegeben.

Die folgende Skizze zeigt ein Single-Server-SiVoIP-System auf Basis von Gemeinschaft 4.0:

SiVoIP



Eine Erklärung zu den Kommunikationswegen der einzelnen Komponenten:

1. Kommunikation über ein RESTful Interface. Die Web-GUI kann nicht direkt auf FreeSWITCH oder die Datenbank zugreifen. Sämtliche Kommunikation geschieht über einen RESTful HTTP(S)-Aufruf.
2. Kommunikation über ein RESTful Interface. Die Web-GUI kann nicht direkt auf FreeSWITCH oder die Datenbank zugreifen. Sämtliche Kommunikation geschieht über einen RESTful HTTP(S)-Aufruf.
3. Kommunikation über ein RESTful Interface. Die Web-GUI kann nicht direkt auf FreeSWITCH oder die Datenbank zugreifen. Sämtliche Kommunikation geschieht über einen RESTful HTTP(S)-Aufruf.
4. Die Kommunikation zwischen dem Kern von Gemeinschafts und der Datenbank erfolgt per SQL. Durch die Verwendung von Active-Record als Bibliothek wird sichergestellt, dass nur valides SQL benutzt wird und die Datenbank bei Bedarf gegen eine andere SQL-Datenbank ausgetauscht werden kann.
5. Die Kommunikation zwischen FreeSWITCH und dem Kern von Gemeinschaft 4.0 erfolgt über eine RESTful HTTP(S)-Schnittstelle.
6. Die Kommunikation zwischen FreeSWITCH und jedem vorgeschalteten VoIP-Proxy erfolgt über das entsprechende Protokoll.
7. Die Kommunikation zwischen FreeSWITCH und dem Kamailio SIP-Proxy erfolgt per SIP.
8. Jeder Proxy speichert und liest Daten aus der Datenbank per SQL.
9. Der SIP-Proxy (Kamailio) speichert und liest Daten aus der Datenbank per SQL.
10. Die Kommunikation vom xyz-Proxy mit dem PSTN-Netz oder xyz-Telefonen erfolgt über das xyz-Protokoll. xyz steht hier exemplarisch für alle zukünftigen VoIP-Protokolle.
11. Der SIP-Proxy (Kamailio) spricht per SIP mit dem PSTN (dem Festnetz).
12. Der SIP-Proxy (Kamailio) spricht per SIP mit den Telefonen.

13. Der Provisioning-Server spricht per HTTP(S) oder SFTP mit den Telefonen.
14. Der Web-Browser spricht per HTTP(S) mit dem Web-Server von SiVoIP.
15. Der Administrator kann sich bei entsprechender Konfiguration per SSH auf dem System anmelden, um dort mit den Admin-Skripten zu arbeiten.

Die Grundidee ist es, jedes einzelne Modul von SiVoIP so klar wie möglich abzugrenzen und über eine einheitliche API anzusteuern. So können automatische Tests sicherstellen, dass ein Module korrekt funktioniert, und es wird Entwicklern einfacher gemacht neue Funktionen einzubauen, da sie durch das Testen mit dem Test-Kit sichergehen können, nichts kaputtgemacht zu haben.

2 Sicherheitskonzept

Authorgroup Amooma GmbH <http://www.amooma.de>

In [beschreibung](#) finden Sie eine Übersicht über die grundlegende Funktionsweise von Gemeinschaft4. Hierfür soll ein Sicherheitskonzept erstellt werden.

Das Sicherheitskonzept wird in fünf Phasen erstellt:

- Strukturanalyse
- Schutzbedarfsfestellung
- Modellierung
- Basissicherheitscheck
- Realisierung

Bei dem aktuellen Fortschritt des Gesamtprojektes können lediglich die ersten beiden Phasen der Erstellung des Sicherheitskonzeptes Berücksichtigung finden. Hierzu werden im folgenden diese beiden Phasen genauer betrachtet.

2.1 Strukturanalyse Gemeinschaft 4.0

Authorgroup Amooma GmbH <http://www.amooma.de>

Die Strukturanalyse erfasst den Informationsverbund vollständig und ermittelt alle beteiligten Komponenten. Hierzu gehören die Netzwerkverbindungen, die IT-Systeme und die Anwendungen und Daten.

Gemeinschaft 4.0 ist modular aufgebaut. Die einzelnen Applikationen können auf einem einzigen IT-System betrieben werden. In dieser Form sind keine Netzwerkverbindungen innerhalb des Verbundes erforderlich. Für eine mögliche Skalierung sieht Gemeinschaft 4.0 aber auch die Verteilung der Applikation auf mehrere IT-Systeme und die Trennung einzelner Applikationsbestandteile vor.

Für die Strukturanalyse ist eine Analyse der folgenden Daten erforderlich

- Netzplan
- IT-Systeme
- Anwendungen

Der Netzplan für die Gemeinschaft 4.0 ist sehr einfach aufgebaut und stellt die Verbindungen mit der Außenwelt und den Telefonen dar. Hier werden die folgenden Systeme erkannt:

- Gemeinschaft 4.0 Telefonanlage
- VoIP-Telefongeräte

Die Telefongeräte sind nicht Bestandteil der Gemeinschaft 4.0. Daher werden diese im weiteren nicht betrachtet.

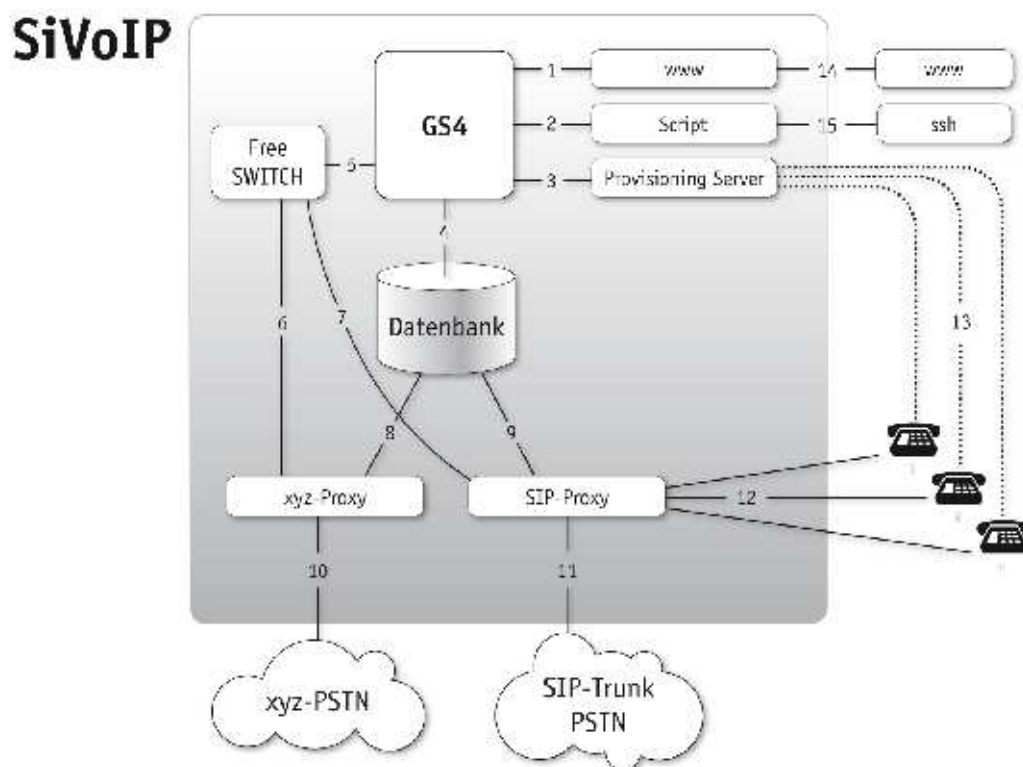
Für die Erfassung der IT-Systeme ist somit nur das System, auf dem die Gemeinschaft 4.0 betrieben wird, relevant. Zum aktuellen Zeitpunkt der Projektierung werden hier die folgenden Parameter angenommen:

- Name: Gemeinschaft4-Server
- Typ: Linux-System
- Standort: Abgeschlossener Serverraum

Auf dem identifizierten IT-System werden mehrere Applikationen für die Telefonanlage betrieben. Diese sind:

- Gemeinschaft-Applikation mit Web-Interface und API
- Gemeinschaft-Provisioning-Server „Cantina“
- Datenbank
- FreeSwitch
- SIP-Proxy

Die folgende Skizze zeigt ein Single-Server-SiVoIP-System auf Basis von Gemeinschaft 4.0:



Die einzelnen Komponenten und die genutzten Kommunikationswege werden im folgenden erläutert:

1. Kommunikation über ein RESTful Interface. Die Web-GUI kann nicht direkt auf FreeSWITCH oder die Datenbank zugreifen. Sämtliche Kommunikation geschieht über einen RESTful HTTP(S)-Aufruf.

2. Kommunikation über ein RESTful Interface. Die Web-GUI kann nicht direkt auf FreeSWITCH oder die Datenbank zugreifen. Sämtliche Kommunikation geschieht über einen RESTful HTTP(S)-Aufruf.
3. Kommunikation über ein RESTful Interface. Die Web-GUI kann nicht direkt auf FreeSWITCH oder die Datenbank zugreifen. Sämtliche Kommunikation geschieht über einen RESTful HTTP(S)-Aufruf.
4. Die Kommunikation zwischen dem Kern von Gemeinschafts und der Datenbank erfolgt per SQL. Durch die Verwendung von Active-Record als Bibliothek wird sichergestellt, dass nur valides SQL benutzt wird und die Datenbank bei Bedarf gegen eine andere SQL-Datenbank ausgetauscht werden kann.
5. Die Kommunikation zwischen FreeSWITCH und dem Kern von Gemeinschaft 4.0 erfolgt über eine RESTful HTTP(S)-Schnittstelle.
6. Die Kommunikation zwischen FreeSWITCH und jedem vorgeschalteten VoIP-Proxy erfolgt über das entsprechende Protokoll.
7. Die Kommunikation zwischen FreeSWITCH und dem Kamailio SIP-Proxy erfolgt per SIP:
8. Jeder Proxy speichert und liest Daten aus der Datenbank per SQL.
9. Der SIP-Proxy (Kamailio) speichert und liest Daten aus der Datenbank per SQL.
10. Die Kommunikation vom xyz-Proxy mit dem PSTN-Netz oder xyz-Telefonen erfolgt über das xyz-Protokoll. xyz steht hier exemplarisch für alle zukünftigen VoIP-Protokolle.
11. Der SIP-Proxy (Kamailio) spricht per SIP mit dem PSTN (dem Festnetz).
12. Der SIP-Proxy (Kamailio) spricht per SIP mit den Telefonen.
13. Der Provisioning-Server spricht per HTTP(S) oder SFTP mit den Telefonen.
14. Der Web-Browser spricht per HTTP(S) mit dem Web-Server von SiVoIP.
15. Der Administrator kann sich bei entsprechender Konfiguration per SSH auf dem System anmelden, um dort mit den Admin-Skripten zu arbeiten.

Grundlage des Design, ist die Abgrenzung der einzelnen Komponenten untereinander. Für die Kommunikation wird eine einheitliche API angeboten und genutzt. Dies erlaubt die Erzeugung automatischer Tests für die Funktionsprüfung der einzelnen Module. Dies garantiert Zukunftssicherheit und die einfache Integration weiterer Funktionen.

▼ 2.2 Schutzbedarfsfeststellung Gemeinschaft 4.0

Authorgroup Amooma GmbH <http://www.amooma.de>

Ziel der Schutzbedarfsfeststellung ist es, den Schutzbedarf für alle Komponenten des Informationsverbundes, hier Gemeinschaft 4.0, zu bestimmen. Da eine Quantifizierung des exakten Schutzbedarfs nicht möglich ist, wird lediglich, wie allgemein üblich, eine qualitative Aussage getroffen und der Schutzbedarf in die drei Kategorien normal, mittel und hoch unterschieden.

Diese Schutzbedarfskategorien werden in Anlehnung an die Dokumentation des BSI wie folgt definiert:

- normal: Die Schadensauswirkungen sind begrenzt und überschaubar.
- hoch: Die Schadensauswirkungen können beträchtlich sein.
- sehr hoch: Die Schadensauswirkungen können ein existentiell bedrohliches, katastrophales Ausmaß erreichen.

Um die Einordnung der Schutzbedarfskategorien zu erleichtern, werden im folgenden mehrere typische

Schadensszenarien (nach BSI) vorgestellt, die bei der Beurteilung berücksichtigt werden sollten:

- Beeinträchtigung des informationellen Selbstbestimmungsrechts und Missbrauch personenbezogener Daten
- Beeinträchtigung in der Aufgabenerfüllung
- Verstoß gegen Verträge, Vorschriften und Gesetze
- Negative Außenwirkung
- Finanzielle Auswirkungen

Im folgenden wird eine erste Abschätzung des Schutzbedarfs für die einzelnen Komponenten und Kommunikationsverbindungen durchgeführt.

Note Die Abschätzung des Schutzbedarfs der Telefonanlage in Bezug auf Verfügbarkeit, Vertraulichkeit und Integrität ist teilweise schwierig, denn unterschiedliche Organisationen/Firmen haben z.T. verschiedene Anforderungen, die sich u.a. aus dem Tätigkeitsfeld ergeben, aus gesetzlichen Bestimmungen und aus betrieblichen Vereinbarungen. So können beispielsweise die mögliche Beeinträchtigung in der Aufgabenerfüllung, die finanziellen Auswirkungen, die negative Außenwirkung oder andere Auswirkungen stark variieren. Es ist daher schwierig eine pauschale und allgemeingültige Analyse durchzuführen.

▼ 2.2.1 Schutzbedarf der einzelnen Komponenten

Übersicht:

▼ Schutzbedarf

Komponente	Personenbezogene Daten	Verfügbarkeit	Vertraulichkeit	Integrität
Datenbank	ja	normal	normal	hoch
Gemeinschaft4	nein	normal	hoch	normal
FreeSwitch	teilweise	hoch	normal	normal
SIP-Proxy	teilweise	hoch	hoch	normal
Prov.-Server	teilweise	normal	normal	normal
Verbindung Telefone	teilweise	hoch	hoch	hoch
Verbindung SIP-Trunk	teilweise	hoch	hoch	hoch
Verbindung ISDN-Trunk	teilweise	hoch	hoch	hoch

Nachfolgend wird erläutert wie die Abschätzung des Schutzbedarfs der einzelnen Komponenten zustandekommt.

▼ 2.2.1.1 Datenbank

Das Datenbank-Management-System (DBMS) ist aus öffentlichen Quellen wiederherstellbar. Die vom DBMS verwaltete Datenbank enthält personenbezogene Daten (Namen, Anruflisten). Ohne sie wäre keine Provisionierung der Endgeräte mehr möglich, jedoch noch eine eingeschränkte Telefonie im Notbetrieb. Den Schutzbedarf der Integrität der Datenbank erachten wir als hoch. Die Liste der Personen und die Liste der Endgeräte sind üblicherweise recht leicht wiederherzustellen (aus LDAP oder einer anderen Datenbank der Organisation). Ein Ausfall der Anruflisten, der von Benutzern definierte Tastenfunktionen auf den

Telefonen, der gesetzte Rufumleitungen sind typischerweise nicht kritisch, können aber von einigen Organisationen bzw. je nach Einsatz-Szenario als relevant bewertet werden. Hier wäre dann entsprechend vom Administrator ein ein- oder besser mehrstufiges Backup-Konzept anzuwenden. Die Backups sollten verschlüsselt sein, denn beispielsweise die Anruflisten können sensible Informationen enthalten.

▼ 2.2.1.2 Gemeinschaft4

Die Applikationslogik von Gemeinschaft4 und die Web-Schnittstellen speichern selbst keine personenbezogene Daten (diese liegen in der Datenbank, die separat betrachtet wird). Ohne Gemeinschaft4 wäre noch eine eingeschränkte Telefonie im Notbetrieb möglich. Die Software ist aus öffentlichen Quellen wiederherstellbar. Da über die Schnittstellen von Gemeinschaft4 (HTTPS) potenziell sensible Daten übertragen werden sollten diese Schnittstellen bzw. die Übertragungsstrecke nach dem derzeitigen Stand der Technik geschützt werden. Ohne Gemeinschaft4 wäre eine Benutzeradministration über das Web-Interface bzw. die XML-API nicht mehr möglich. Nach unserer Erwartung stellt der Umstand daß während eines angenommenen Ausfalls des Benutzer-Managements keine neuen Benutzer angelegt/geändert werden können jedoch eher eine untergeordnete Rolle.

▼ 2.2.1.3 Provisioning-Server

Der Provisioning-Server („Cantina“) dient der Konfiguration der Telefone. Bei einem Ausfall würden die Telefone ihre derzeitige Konfiguration beibehalten, Änderungen könnten nicht mehr automatisiert sondern nur noch manuell vorgenommen werden. Nach unserer Erwartung würde ein vorübergehender Ausfall in den meisten Fällen kein Problem darstellen, da mit den Geräten in ihrer derzeitigen Konfiguration weiterhin telefoniert werden kann. Die Software ist aus öffentlichen Quellen wiederherstellbar. Im weiteren Projekt sind hier noch Anpassungen der Betrachtung der Schutzwürdigkeit erforderlich.

▼ 2.2.1.4 SIP-Proxy

Der SIP-Proxy Kamailio speichert selbst keine direkten personenbezogenen Daten. Er agiert anhand der in der Datenbank hinterlegten SIP-Accounts. Je nach Konfiguration könnten jedoch die SIP-Accounts Rückschlüsse auf bestimmte Personen ermöglichen. Den Schutzbedarf der Verfügbarkeit erachten wir als hoch, denn ohne den SIP-Proxy ist in aller Regel keine Telefonie mehr möglich. Der SIP-Proxy sieht (zwecks Routing) als potenziell sensibel einzustufende Informationen über die SIP-Topologie. Die Software ist aus öffentlichen Quellen wiederherstellbar.

▼ 2.2.1.5 FreeSwitch

FreeSwitch kommt als B2BUA für TK-Anlagen-Funktionen zum Einsatz. Die Software ist aus öffentlichen Quellen wiederherstellbar. Der Schutzbedarf im Hinblick auf die Verfügbarkeit ist hier als etwas weniger hoch als beim SIP-Proxy anzusehen, da ohne FreeSwitch zumindest noch in einem Notbetrieb telefoniert werden kann. Hier ist im weiteren Projekt noch eine nähere Betrachtung erforderlich.

▼ 2.2.1.6 Verbindung Telefone

Die Anbindung der Telefone an den SIP-Proxy und an den Provisioning-Server erfolgt über das lokale Ethernet-Netzwerk (LAN). Der Schutzbedarf der Verfügbarkeit, Vertraulichkeit und Integrität ist als hoch anzusiedeln. Bei einem Ausfall der LAN-Verbindung ist keine Telefonie mehr möglich. Sowohl zum SIP-Proxy als auch zum Provisioning-Server sollen Übertragungen vertraulich erfolgen als auch die Integrität sichergestellt sein. Es ist entscheidend ein Endgerät bzw. einen SIP-Account eindeutig identifizieren zu können, damit beispielsweise bei der Provisionierung sensible Daten nicht an ein anderes Gerät gesendet werden. Die Sicherstellung der Verfügbarkeit der Netzwerk-Infrastruktur liegt in der Verantwortung des Administrators.

Möglichkeiten wie vor einem Angreifer der auf dem Ethernet-Kabel „lauscht“ verborgen werden kann daß überhaupt verschlüsselte Gespräche stattfinden werden hier nicht betrachtet.

▼ 2.2.1.7 Verbindung SIP-Trunk

Die Möglichkeit der Anbindung der TK-Anlage an SIP-Trunks erfolgt in einem späteren Projektschritt.

SIP-Trunks sind beispielsweise denkbar zu einem Telekom-Carrier ins öffentliche Telefonnetz oder zwischen zwei Standorten/Organisationen/Firmen. Verfügbarkeit, Vertraulichkeit und Integrität sind hier normalerweise als hoch anzusehen. Die Sicherstellung der Verfügbarkeit der Netzwerk-Infrastruktur liegt in der Verantwortung des Administrators. Die TK-Anlage kann jedoch über mehrere SIP-Trunks angebunden werden die so konfiguriert werden können daß sie gegenseitig als Fallback dienen.

▼ 2.2.1.8 Verbindung ISDN-Trunk

Der Schutzbedarf eines ISDN-Anschlusses über ein Media-Gateway wird in diesem Projektschritt noch nicht näher betrachtet. Er entspricht voraussichtlich in etwa dem eines SIP-Trunks. Unter Umständen können hier verschlüsselte Verbindungen zum Einsatz kommen, die allerdings das Vorhandensein spezieller Hardware (oder Software) auf beiden Seiten erforderlich machen.

▼ 2.2.1.9 Weitere Anpassungen

Basierend auf dieser ersten Abschätzung erfolgt die weitere Modellierung in der Phase 3 des Sicherheitskonzepts. Dabei ist eine laufende Kontrolle der Ergebnisse der Strukturanalyse und der Schutzbedarfsbestimmung im Projekt erforderlich, da im weiteren Änderungen in der Struktur möglich sind.