# INSTALL PLAYSMS AND SMSTOOLS ON DEBIAN 11

# # install other sources to install php 7.2 on debian 11

# apt -y install gnupg2 apt-transport-https ca-certificates software-properties-common

# wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg

# echo "deb https://packages.sury.org/php/ $(lsb\_release -sc) main" > /etc/apt/sources.list.d/php.list

# apt update && apt install -y apache2 mariadb-server php7.2 php7.2-opcache php7.2-cli php7.2-mysqli php7.2-mysql php7.2-gd php7.2-mbstring php7.2-xml php7.2-curl php7.2-zip php7.2-fpm

# update-alternatives --set php /usr/bin/php7.2

# while if you are on ubuntu 18.04 only:

# apt-get install apache2 mariadb-server php php-cli php-mysql php-gd php-curl php-mbstring php-xml php-zip

# 

# 

# Install playSMS 1.4.6 and smstools 3 (compiing it) on Ubuntu 18.04

# (ubuntu 20.04 and major cannot compile smstools)

[Howto](https://playsms.org/category/howto/)

playSMS version 1.4.6 has been released, and it is the recommended version as it contains fixes to several bugs and critical security vulnerability. This article is howto install playSMS 1.4.6 on Ubuntu 18.04.

I’m using DigitalOcean (DO) service to test the configuration and commands. Create new Droplet in DO account. Click [here](https://m.do.co/c/aeec1cef58b6" \t "_blank) to register on DO if you don’t have an account.

Choose Ubuntu 18.0.4 (currently 18.04.3 LTS) and select at least the cheapest service (USD 5). Create and wait for a minute or two for the SSH to be ready. You can then login via SSH and start playSMS installation.

Login to your CentOS droplet (later we will call droplet as server) using SSH and follow instructions below step by step. Read carefully why you need to do each step correctly. Please pay attention to details.

This article first published in: <https://antonraharja.com/2020/03/20/playsms-1-4-3-on-ubuntu-18-04/>

**1. Prepare Ubuntu**

**1.1. Add Normal User**

In DO you need to login as root first. But it is recommended to not login as root all the time, so we create a new normal Linux user.

As root create a new normal Linux user and set a strong password for it:

adduser playsms

Add user playsms to sudo group:

usermod -a -G sudo playsms

*Q: Can I use other username beside playsms ?*

Yes, you can. Just remember to adjust every reference of playsms in this article into your own chosen username.

**1.2. Copy authorized\_keys**

This is additional and **optional** steps you need to do if you’re login SSH as root using private key instead of password.

THIS

You need to copy root’s authorized\_keys to playsms:

sudo mkdir -p /home/playsms/.ssh

sudo cp /root/.ssh/authorized\_keys /home/playsms/.ssh/

sudo chown -R playsms.playsms /home/playsms

After this you can login SSH as user playsms using the same private key as root.

*Q: Can I use different key for playsms ?*

Yes, you can. Copy the public key (it’s public key, not private key) to playsms`s authorized\_keys and remove root’s public key from it.

**1.3. Enable Ubuntu Firewall**

Allow SSH first:

sudo ufw allow ssh

Enable UFW, activate it and make it starts on boot:

sudo ufw enable

Reload UFW:

sudo ufw reload

As of now only SSH allowed by server, later we will allow http and https. Don’t forget to ufw reload after changing UFW rules.

**1.4. Install mc , zip and unzip**

Yes. Install mc and unzip :) I’m using nano as console text editor, and you might be checking files/folders frequently, for that I think mc helps. But you can always choose not to install it and stick with nano or vi.

You need to install unzip, composer will need it and playSMS will need composer.

Install mc , zip and unzip:

sudo apt update

sudo apt install mc zip unzip

**1.5. Upgrade Server**

Update and upgrade:

sudo apt update

sudo apt upgrade

Most likely after upgrade Ubuntu asks for server reboot, reboot it then:

sudo shutdown -r now

Re-login SSH using user playsms instead of root. Pass this point you need to login to the server as normal user playsms, and you will use sudo when you need to execute commands as root.

**2. Install MySQL Server**

We will use MariaDB as MySQL server.

If you have not logout out from root you need to logout now and re-login as normal user playsms.

Install MySQL server MariaDB:

sudo apt install mariadb-server

Starts MariaDB and enable it:

sudo systemctl start mariadb.service

sudo systemctl enable mariadb.service

Test your MySQL root access:

sudo mysql

You should now logged in to your MySQL server as MySQL user root. Type quit and <Enter> to exit MySQL console.

Note that you cannot login to MariaDB as MySQL user root if you are not Linux user root. Use sudo to access MySQL server as MySQL user root, you won’t be asked for password.

We will not use MySQL user root in playSMS but we will create a new MySQL user just for playSMS database later.

**3. Install Web Server and PHP 7.2**

We will use Apache2 as the web server.

Install Apache2, PHP 7.2 and required PHP modules:

**REMEMBER, IF INSTALL ON DEBIAN 11, pass step XXX**

**Have installed iet php 7.2?!?!?**

# (((apt update && apt install -y apache2 mariadb-server php7.2 php7.2-opcache php7.2-cli php7.2-mysqli php7.2-mysql php7.2-gd php7.2-mbstring php7.2-xml php7.2-curl php7.2-zip)))

sudo apt install apache2 php php-cli php-mysql php-gd php-curl php-mbstring php-xml php-zip

**\*\*\*STEP XXX\*\*\***

Start Apache2 and enable it:

sudo systemctl start apache2.service

sudo systemctl enable apache2.service

a2enconf php7.2-fpm

systemctl restart php7-2-fpm

Allow HTTP and HTTPS:

sudo ufw allow http

sudo ufw allow https

sudo ufw reload

Let’s test the PHP:

cd /var/www/html

sudo nano test.php

<?php

echo "Hello World";

Save test.php and browse the file, you should *Hello World* displayed.

Remove `test.php` after testing:

sudo rm -f /var/www/html/test.php

**4. Supports HTTPS**

HTTPS supports will be added to our web server by requesting, installing and configuring SSL certificate from [Let’s Encrypt](https://letsencrypt.org/" \t "_blank) on Apache2. Let’s Encrypt provides a free SSL certificate for everyone.

**4.1. Setup VirtualHost**

This step is required for getting free SSL certificate for our HTTPS service from Let’s Encrypt.

In this example I will be using dm143.playsms.org domain as my entry in VirtualHost setup. I also have set the DNS to point dm143.playsms.org to my CentOS server’s public IP. Of course you will need your own domain/subdomain and point to your own CentOS server’s public IP.

The example VirtualHost configuration will make Apache serve PHP file for domain  playsms from our regular user (user playsms) Home Directory (/home/playsms/public\_html to be exact).

Prepare user’s Home Directory:

cd /home/playsms

mkdir -p public\_html log

sudo chmod 775 /home/playsms public\_html log

sudo chown playsms.playsms -R /home/playsms

sudo chown www-data.playsms -R /home/playsms/log

ls -l /home/playsms

Create VirtualHost configuration file for domain dm143.playsms.org:

sudo nano /etc/apache2/sites-enabled/ 000-default.conf

|  |  |
| --- | --- |
|  | <VirtualHost \*:80>      ServerName playsmm      DocumentRoot /home/playsms/public\_html      ErrorLog /home/playsms/log/httpd-error.log      CustomLog /home/playsms/log/httpd-accesss.log combined      <Directory /home/playsms/public\_html>          AllowOverride FileInfo AuthConfig Limit Indexes          Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec          Require method GET POST OPTIONS          php\_admin\_value engine On      </Directory>  </VirtualHost> |

Enable it:

sudo systemctl reload apache2.service

Switch user as user **playsms** and test VirtualHost by create a PHP file in /home/playsms/public\_html.

nano /home/playsms/public\_html/test.php

|  |  |
| --- | --- |
|  | <?php  echo "<b>Welcome !!</b>"; |

Save the file and browse this file at your domain, in this example browse http://ip\_of\_your\_playsms\_machine/test.php

You know your VirtualHost is working when you see **Welcome !!** on your browser.

Remove test.php after testing:

rm -f /home/playsms/public\_html/test.php

**4.2. Install certbot (not needed…)**

We will get the SSL certificate from Let’s Encrypt and use certbot to install it on the server.

Install certbot:

sudo apt install python3-certbot-apache

**4.3. Setup SSL Certificate**

Run certbot for Apache:

sudo certbot --apache

Answer questions correctly. You will need to input your email address, choose **A** to Agree with the ToS and last choose **Redirect** (selection no. 2) to completely remove HTTP and just serve HTTPS by redirecting all HTTP requests to HTTPS.

Example of successful SSL certificate request and installation:

Visit [ssllabs.com/ssltest](https://www.ssllabs.com/ssltest/) and submit your domain to test your HTTPS configuration.

**5. Install playSMS (needed!!!)**

Now that we have a working web server with PHP and HTTPS supports, and MySQL server, we can then install playSMS 1.4.6.

From now on you must execute commands as normal Linux user. In this article playSMS will be installed under user playsms as mentioned before.

**5.1. Prepare Directories**

Here are some important directories that need to be ready before playSMS installation:

public\_html and log is already exists and prepared, they are created previously on section 4.1 as part as VirtualHost configuration. So now we need to create the rest and set proper permission.

Then create directories:

cd /home/playsms

mkdir -p bin etc lib src

sudo chmod 775 bin etc lib src

Prepare log files too, this need to be done so that both web server Apache2 and playSMS daemon have write access to playSMS log files:

cd /home/playsms

sudo touch log/audit.log log/playsms.log

sudo chmod 664 log/audit.log log/playsms.log

sudo chown www-data.playsms -R log

ls -l log

**5.2. Check PHP Modules**

Required PHP modules should already be installed if you follow this article from the start, it is on section 3. But before proceeding with playSMS installation you need to make sure that required PHP modules are installed:

php -m

Make sure you see at least curl, gd, mbstring, mysqli and xml on the list. If they are not on the list then please install them, see section 3.

**5.3. Prepare Database**

Create MySQL database that will be used by playSMS:

sudo mysqladmin create playsms

Login as MySQL user root and create a new MySQL user for above database:

sudo mysql

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE USER 'playsms'@'localhost' IDENTIFIED BY 'strongpasswordhere';  GRANT ALL PRIVILEGES ON playsms.\* TO 'playsms'@'localhost';  FLUSH PRIVILEGES;  exit |

Do not copy-paste above SQL commands directly to MySQL console, you must use your own strong password, change the strongpasswordhere with your own strong password.

As of this section you will have a MySQL database named playsms and MySQL normal user playsms with your own strong password which only have access to database playsms.

**5.4. Get playSMS Source Code**

playSMS source code available on Github, you will need git to get them.

Go to src folder:

cd /home/playsms/src

Get playSMS version 1.4.6:

git clone -b 1.4.x --depth=1 https://github.com/antonraharja/playSMS

As of now your playSMS 1.4.6 source code is available at /home/playsms/src/playSMS.

**5.5. Prepare install.conf**

Go to playSMS source code directory, copy install.conf.dist to install.conf and then edit it.

Go to playSMS source code directory:

cd /home/playsms/src/playSMS

Edit install.conf:

cp install.conf.dist install.conf

nano install.conf

These are values I set on install.conf:

|  |  |
| --- | --- |
|  | DBUSER="playsms"  DBPASS="strongpasswordhere"  DBNAME="playsms"  DBHOST="localhost"  DBPORT="3306"  WEBSERVERUSER="www-data"  WEBSERVERGROUP="www-data"  PATHSRC="/home/playsms/src/playSMS"  PATHWEB="/home/playsms/public\_html"  PATHLIB="/home/playsms/lib"  PATHBIN="/home/playsms/bin"  PATHLOG="/home/playsms/log"  PATHCONF="/home/playsms/etc" |

Values need to reflect your server configuration. If you follow this article from the start then above values should be correct, with exception your true database password (DBPASS) of course.

Save install.conf and ready to run install script.

**5.6. Run playSMS Install Script**

playSMS install script will download composer and download packages from repo.packagist.org. After that the script will copy necessary files from playSMS source code to public\_html and bin.

Since theres requirement to be able to download from external site (repo.packagist.org), you have to make sure that external site is working and reachable.

But you can just start the install script, because you’ll know if something not right, for example the script fail to download packages. When that happens you can fix the problem first, like fix your networking setup and perhaps firewall, or simply wait (theres a chance the external site down too), and then go back to re-run the install script.

Just to make sure that networking stuff is right, please see section 1.6.

OK, let’s start the installation:

cd /home/playsms/src/playSMS

./install-playsms.sh

Verify installation:

Press **Y** (you will be asked twice, answer Y both) and proceed the installation.

Successful installation will show that all playSMS daemon is running:

Browse your playSMS, don’t worry if the login page looks broken, it’s because we haven’t configure playSMS to enable HTTPS, we will do that after this. For now, check if you can see playSMS login page.

**5.7. Adjust config.php**

Edit playSMS config.php and adjust some value, or just one part, the HTTPS support.

nano /home/playsms/public\_html/config.php

Inside config.php:

* Search for logstate and set it to 3
* Search for ishttps and set it to true. (if prefer https and not only http)

Daemon result red color on web page: go in this file and modify:

/home/playsms/public\_html/plugin/feature/playsmslog/config.php

From:

**$plugin\_config['playsmslog']['playsmsd']['bin'] = '/home/playsms/bin/playsmsd';**

**$plugin\_config['playsmslog']['playsmsd']['conf'] = '/home/playsms/etc/playsmsd.conf';**

to:

**$plugin\_config['playsmslog']['playsmsd']['bin'] = '/home/playsms/bin/playsmsd';**

**$plugin\_config['playsmslog']['playsmsd']['conf'] = '/home/playsms/etc/playsmsd.conf';**

save, all green now!!!

Edit also this file to remove (for me is +393xxx)

/home/playsms/public\_html/plugin/core/sendsms/fn.php

After this line:

if (is\_array($user)) {

$prefix = ($user['replace\_zero'] ? $user['replace\_zero'] : $core\_config['main']['default\_replace\_zero']);

$local\_length = (int) $user['local\_length'];

\_log('before prefix manipulation:[' . $number . ']', 3, 'sendsms\_manipulate\_prefix');

**//add here//**

**$number = str\_replace('+', '', $number);**

**if (substr($number, 0, 3) == '393') {**

**\_log('my own prefix manipulation not need:[' . $number . ']', 3, 'sendsms\_manipulate\_prefix');**

**} else {**

**\_log('my own prefix manipulation needed changing it:[' . $number . ']', 3, 'sendsms\_manipulate\_prefix');**

**$number = '39' . $number;**

**\_log('my own prefix manipulation needed:[' . $number . ']', 3, 'sendsms\_manipulate\_prefix');**

**}**

**Some trick on cache ecc**

sed -i "s/;request\_terminate\_timeout = 0/request\_terminate\_timeout = 300/" /etc/php/7.2/fpm/pool.d/www.conf

sed -i "s/max\_execution\_time = 30/max\_execution\_time = 60/" /etc/php/7.2/fpm/php.ini

sed -i "s/upload\_max\_filesize = 2M/upload\_max\_filesize = 20M/" /etc/php/7.2/fpm/php.ini

sed -i "s/post\_max\_size = 8M/post\_max\_size = 20M/" /etc/php/7.2/fpm/php.ini

sed -i "s/memory\_limit = 128M/memory\_limit = 512M/" /etc/php/7.2/fpm/php.ini

systemctl enable php7.2-fpm

systemctl restart php7.2-fpm

systemctl reload apache2

Save

**Install Playams service:**

on console write:

**su root**

(enter root password)

**cat >> /etc/systemd/system/playsms.service << EOF**

**[Unit]**

**Description=playsms**

**After=mariadb.service**

**[Service]**

**Type=oneshot**

**RemainAfterExit=yes**

**ExecStart=/home/playsms/bin/playsmsd /home/playsms/etc/playsmsd.conf start**

**ExecStop=/home/playsms/bin/playsmsd /home/playsms/etc/playsmsd.conf stop**

**User=www-data**

**Group=www-data**

**[Install]**

**WantedBy=multi-user.target**

**EOF**

**Enable and execute playsms service:**

chmod 755 /home/playsms/bin/playsmsd

systemctl daemon-reload

systemctl enable playsms

systemctl restart playsms

systemctl status playsms

**5.8. Change Default Password**

Go to your browser, browse the server and login as playSMS administrator, and change the default admin password immediately.

**If you use release > ubuntu 18.04, for example debian 11, follow this istructions and the go to STEP “B”**

Install smstools by command:

sudo apt install smstools, and then

sudo update-rc.d -f smstools disable

sudo update-rc.d -f smstools remove

go on my repository and download as zip the fix:

<https://github.com/pappicio/smstools3-for-debian-11>

unzip and copy all folders/files in your root debian 11 system by winscp tool (<https://winscp.net/download/WinSCP-6.1.1-Setup.exe>)

provide to set for all folders/subfolder/files, 0777 permission and as user/group: www-data, and at end execute:

sudo systemctl daemon-reload

sudo update-rc.d -f sms3 defaults

**Go t step “B” now:**

**5.9 Install smstools last release**

apt-get install build-essential libusb-1.0 libusb-1.0-0-dev build-essential manpages-dev

sudo apt-get update & sudo apt-get install usb-modeswitch

cd /tmp/

wget <http://smstools3.kekekasvi.com/packages/smstools3-3.1.21.tar.gz>

tar -zxf smstools3-3.1.21.tar.gz -C /usr/local/src

ls -l /usr/local/src/

cd /usr/local/src/smstools3/

make

make install

**STEP “B”**

**5.8. compiled smstools, now configure:**

mkdir -p /var/log/sms/stats

mkdir -p /var/spool/sms/ {checked,failed,incoming,outgoing,sent}

mkdir / var/spool/sms/modem1

chown www-data:www-data -R /var/spool/sms

chmod 777 -R /var/spool/ sms

mv /etc/smsd.conf /etc/smsd.conf.dist

cd /tmp

wget -c <https://raw.githubusercontent.com/antonraharja/playSMS/master/contrib/smstools/smsd.conf>

cp smsd.conf /etc/

configure your own, my is:

devices = modem1

loglevel = 7

# logfiles

stats = /var/log/sms/stats

logfile = /var/log/sms/smsd.log

# Default queue directory = /var/spool/sms

outgoing = /var/spool/sms/outgoing

checked = /var/spool/sms/checked

failed = /var/spool/sms/failed

incoming = /var/spool/sms/incoming

sent = /var/spool/sms/sent

delaytime = 2

errorsleeptime = 10

blocktime = 180

autosplit = 3

# Queue configurations

[queues]

modem1 = /var/spool/sms/modem1

[modem1]

device = /dev/ttyUSB1

init = AT^CURC=0

###init2 = AT+CPMS="ME","ME","ME"

#pin = 1234

report = yes

incoming = yes

queues = modem1

# mode = new

smsc = 393770001016

baudrate = 115200 ###19200

memory\_start = 0

decode\_unicode\_text = yes

#cs\_convert = yes

report\_device\_details = no

**...and finally...**

sudo update-rc.d sms3 defaults

sudo reboot

and all works!!!

**Block usb modem on same ttyUSBX for ever:**

|  |  |
| --- | --- |
|  | --------------------------- |
|  | Get deviceid for the dongle |
|  | sudo lsusb |
|  |  |

Get to know properties of the device while it is switched in:

**udevadm info -q all -p $(udevadm info -q path -n /dev/ttyUSB1)**

( if your system is old, try instead with this command: )

( **udevinfo -a -p $(udevinfo -q path -n /dev/ttyUSB0)** )

Find some property that can identify the device (uniquely), for instance "serial"

Create a file called

/etc/udev/rules.d/10-usb-serial

which contains the line:

BUS=="usb", ATTR{serial}=="xxxx", NAME="ttyUSB1"

Note the two equal signs for properties that are tested, and one for that which is assigned to.

OPPURE:

# Persistent paths for dynamic device file

This post is related to [playSMS](http://playsms.org/" \o "playSMS free and open source SMS portal" \t "_blank) and [Kannel](http://kannel.org/" \o "Kannel SMS Gateway" \t "_blank), how to configure Kannel when you know that the device file names aren’t persistent.

### Intro.

When USB GSM modem plugged to a server Linux kernel assigned dynamic device file /dev/ttyUSB\*, such as /dev/ttyUSB0 or /dev/ttyUSB1. For example, USB GSM modem with 2 ports will then be assigned to /dev/ttyUSB0 for port 1 and /dev/ttyUSB1 for port 2.

Problem starts when we unplug the GSM modem and re-plug back afterwards. Linux kernel will then assign different device file to it, was /dev/ttyUSB0 now /dev/ttyUSB2 and was /dev/ttyUSB1 now /dev/ttyUSB3.

### Let’s talk about the problem.

Put your attention to this SMSC configuration part of our Kannel:

## SMSC gsm1

group = smsc

smsc = at

smsc-id = gsm1

modemtype = wavecom

device = /dev/ttyUSB0

log-file = /var/log/kannel/smsc-gsm1.log

log-level = 0

## SMSC gsm2

group = smsc

smsc = at

smsc-id = gsm2

modemtype = wavecom

device = /dev/ttyUSB1

log-file = /var/log/kannel/smsc-gsm2.log

log-level = 0

Note that SMSC ID gsm1 is mapped to /dev/ttyUSB0, and SMSC ID gsm2 is mapped to /dev/ttyUSB1.

When we unplug the GSM modem and re-plug it Linux kernel will assign different device files, was /dev/ttyUSB0 then become /dev/ttyUSB2, was /dev/ttyUSB1 then become /dev/ttyUSB3. This will make your Kannel wrongly configured and stop sending or receiving SMS.

What we would do were to change configuration to use newly assigned device file /dev/ttyUSB2 and /dev/ttyUSB3. But we would have to change it back when the server restarted or we unplugged and re-plugged it again and again. Imagine how unstable our system looked like.

What we want is whenever the server restarted, the modem unplugged and re-plugged, whichever the device files would be, we do not need to change our kannel.conf and restart Kannel.

### Here’s how to get what you want.

With the help of [udev](http://en.wikipedia.org/wiki/Udev" \o "udev" \t "_blank) configuration and a script we can dynamically map device file to a specific, and persistent, path, upon plugging the physical device.

Create **/etc/udev/rules.d/80-ttyusb-map.rules**:

vi /etc/udev/rules.d/80-ttyusb-map.rules

And fill it with this:

ACTION=="add", KERNEL=="ttyUSB[0-9]\*", PROGRAM="/etc/udev/rules.d/ttyusb-map.sh %p", SYMLINK+="gsm%c"

Then create **/etc/udev/rules.d/ttyusb-map.sh**:

touch /etc/udev/rules.d/ttyusb-map.sh

chmod 755 /etc/udev/rules.d/ttyusb-map.sh

vi /etc/udev/rules.d/ttyusb-map.sh

And fill it with this:

#!/usr/bin/perl -w

@items = split("/", $ARGV[0]);

for ($i = 0; $i < @items; $i++) {

    if ($items[$i] =~ m/^usb[0-9]+$/) {

        print $items[$i + 1] . "\n";

        last;

    }

}

That is all.

Now try to plug GSM modem, and then plug it back. We should see that /dev/gsm1-1 symlink to /dev/ttyUSB0 and /dev/gsm2-1 symlink to /dev/ttyUSB1.

See example below:

[anton@srv ~]$ ls -l /dev/gsm\*

lrwxrwxrwx 1 root root 7 Mei  4 15:40 /dev/gsm1-1 -> ttyUSB0

lrwxrwxrwx 1 root root 7 Mei  4 15:40 /dev/gsm2-1 -> ttyUSB1

Those symlinks can be different each time you plug and re-plug the GSM modem, or restart the server, but the name of those device files are persistent.

We can then use /dev/gsm1-1 as our map to physical USB port 1 and /dev/gsm2-1 as our map to physical USB port 2.

Your Kannel configuration would then be like this:

## SMSC gsm1

group = smsc

smsc = at

smsc-id = gsm1

modemtype = wavecom

device = /dev/gsm1-1

log-file = /var/log/kannel/smsc-gsm1.log

log-level = 0

## SMSC gsm2

group = smsc

smsc = at

smsc-id = gsm2

modemtype = wavecom

device = /dev/gsm2-1

log-file = /var/log/kannel/smsc-gsm2.log

log-level = 0

Restart your Kannel and tail SMSC log files, see if Kannel works properly.

tail -f /var/log/kannel/smsc-gsm1.log

tail -f /var/log/kannel/smsc-gsm2.log

**MANAGE ACL per menu limitati per i users/subusers:**

**generico e troppo permissivo.**

inc=core\_sendsms,

inc=feature\_report,

inc=feature\_schedule,

inc=feature\_msgtemplate,

inc=feature\_queuelog,

inc=feature\_credit,

inc=feature\_report&route=user

**per gli USERS:**

**inc=feature\_phonebook,**

**inc=core\_user&route=subuser\_mgmnt,**

**inc=core\_user&route=user\_pref&op=user\_pref,**

**inc=feature\_queuelog&op=queuelog\_list**

**per i subuser:**

inc=core\_sendsms,

inc=feature\_report&route=user,

inc=feature\_schedule,

inc=feature\_msgtemplate,

inc=core\_user&route=user\_pref&op=user\_pref,

inc=feature\_queuelog

**SUBUSERS CON PHONEBOOK:**

inc=feature\_phonebook,

inc=core\_sendsms,

inc=feature\_report&route=user,

inc=feature\_schedule,

inc=feature\_msgtemplate,

inc=core\_user&route=user\_pref&op=user\_pref,

inc=feature\_queuelog

Playsms trick on php-html files/functions

/plugin/core/user/fn.php

// commentando questa funzione si possono aggiungere quanti subuser (addetti all'INVIO SMS) si vuole, con lo stesso numero telefonico (magari quello del centralino COC).

//che è il minimo comun denominatore per vedere la rubrica che crea l'utente del compartimento!!!

Quindi basta aggiungere un contatto nella rubrica tipo “fakenapoliuser” con lo stesso numero telefonico (081000000) e tutti gli utenti addetti all’invio SMS che saranno creati dall’user di turno, avranno lo stesso numero telefonico personale(esempio: 081000000), cosi inserito “fakenapoliuser” in rubrica e nel gruppo NAPOLI, potranno vedere la rubruca!!!

E seguendo questo discorso logico (aggiiungendo un nuovo gruppo con nuovi contatti e nuovi subuser / utente con stesso numero, esempio fakepgnapoli (081111111)) si puo creare quante ribriche si vuole e ognuno vedra solo il feuppo cui fa parte (con il fakenumebr, diciamo!)

// check mobile, must check for duplication only when filled

// if ($ret['status'] && $data['mobile']) {

// if (dba\_isexists(\_DB\_PREF\_ . '\_tblUser', array(

// 'flag\_deleted' => 0,

// 'mobile' => $data['mobile']

// ), 'AND')) {

// if ($data['mobile'] != $existing['mobile']) {

// $ret['error\_string'] = \_('Account with this mobile already exists') . " (" . \_('mobile') . ": " . $data['mobile'] . ")";

// $ret['status'] = false;

// }

// }

// }

}

Per dare tutti i gruppi imitati (ACL) anche ai subusers cambiare le seguenti righe di codice:

in: /plugin/core/user/subuser\_mgmnt.php

//////$option\_acl = \_select('add\_acl\_id', array\_flip(acl\_getallbyuid($user\_config['uid'])));

$option\_acl = \_select('add\_acl\_id', array\_flip(acl\_getall()));

e

in: /plugin/core/user/user\_config.php

if ($user\_edited['status'] == 4) {

$parent\_id = user\_getparentbyuid($user\_edited['uid']);

if ($parent\_id == $user\_config['uid']) {

////////$c\_option\_acl = array\_flip(acl\_getallbyuid($user\_config['uid']));

$c\_option\_acl = array\_flip(acl\_getall());

**FACOLTATIVI (hidden by ACL)**

per rendere readonly il telefono di un subuser (che ricordiamo deve essere fisso e uguale al fake-contatto in ribrica e nel gruppo, senno addio invio sms con ricerca….)

qui: /plugin/core/user/user\_prefs.php

<tr>

<td>{{ Mobile }}</td>

<td><input type=text maxlength=20 name=up\_mobile value="{{ mobile }}" readonly></td>

</tr>

In pratica aggiungere “readonly”

per rendere deadonly la “firma del subuser qui:

in: /plugin/core/user/template/user\_config.html

<tr>

<td>{{ Default message footer }}</td>

<td><input type=text maxlength=30 name=up\_footer value="{{ footer }}" readonly> {{ HINT\_MAX\_ALPHANUMERIC }}

</td>

</tr>

In pratica aggiungere “readonly”

Per rendere la “firma” readonly anche su invio sms qui:

/plugin/core/semdsms/templates/sendsms.html

<label for="msg\_footer">{{ Message footer }}</label>

<p>

<input type="text" name="sms\_footer" id="msg\_footer"

style="width: 100%" value="{{ sms\_footer }}" readonly>

</p>

In pratica aggiungere “readonly”.