



farsouth
networks

Com.X

Hybrid IP PBX / Gateway

Customization Guide

(Com.X 1.2 product release)

Version 1.7, 20 May 2011

Document History

Version	Date	Description of Changes
1	12/03/10	Created.
1.1	04/20/11	Added the full set of customizations available at the time of writing.
1.2	05/20/11	Added automatic upload of music on hold
1.3	06/29/11	Added recording lookup customization
1.4	06/30/11	Added call limit customization
1.5	06/05/11	Added raw CDR via telnet customization
1.6	11/11/07	Added example Yealink XML Browser configuration
1.7	12/03/08	Added SIP provider / trunk profiling script

Table of Contents

1 INTRODUCTION.....	6
1.1 OVERVIEW.....	6
2 CALL BARGE.....	7
2.1 SUMMARY.....	7
2.2 BEHAVIOUR.....	7
2.3 INSTALLATION.....	7
2.4 CONFIGURATION.....	7
3 BLOCKING EXTENSIONS AND PIN SETS.....	8
3.1 SUMMARY.....	8
3.2 BEHAVIOUR.....	8
3.3 INSTALLATION.....	8
3.4 EXECUTION / SCHEDULING.....	8
4 BUDGET TRUNKS.....	10
4.1 SUMMARY.....	10
4.2 BEHAVIOUR.....	10
4.3 INSTALLATION.....	10
4.4 CONFIGURATION.....	10
4.5 EXECUTION / SCHEDULING.....	11
5 CALL BILLABLE SECOND LIMIT.....	12
5.1 SUMMARY.....	12
5.2 BEHAVIOUR.....	12
5.3 INSTALLATION.....	12
5.4 EXECUTION / SCHEDULING.....	12
6 CAUSE CODE REMAPPING.....	13
6.1 SUMMARY.....	13
6.2 BEHAVIOR.....	13
6.3 INSTALLATION.....	13
6.4 CONFIGURION.....	13
6.5 CDR FTP UPLOAD.....	14
6.6 SUMMARY.....	14
6.7 BEHAVIOUR.....	14
6.8 INSTALLATION.....	14
6.9 CONFIGURATION.....	14
6.10 EXECUTION / SCHEDULING.....	14
7 SPYING ON AN EXTENSION (CHANSPY EXTENSION).....	16
7.1 SUMMARY.....	16
7.2 BEHAVIOUR.....	16
7.3 INSTALLATION.....	16

7.4 CONFIGURATION.....	16
7.5 EXECUTION / SCHEDULING.....	16
8 ADDING NAGIOS MONITORING SUPPORT.....	17
8.1 SUMMARY.....	17
8.2 BEHAVIOUR.....	17
8.3 INSTALLATION.....	17
8.4 CONFIGURATION.....	18
8.5 EXECUTION / SCHEDULING.....	19
9 PRI PORT-BASED ROUND-ROBIN WITH FAIL-OVER.....	20
9.1 SUMMARY.....	20
9.2 BEHAVIOUR.....	20
9.3 INSTALLATION.....	20
9.4 CONFIGURATION.....	20
9.5 EXECUTION / SCHEDULING.....	20
10 SCHEDULED RECORDING ARCHIVING TO EXTERNAL DEVICE.....	21
10.1 SUMMARY.....	21
10.2 BEHAVIOUR.....	21
10.3 INSTALLATION.....	21
10.4 CONFIGURATION.....	21
10.5 EXECUTION / SCHEDULING.....	21
11 RECORDING LOOKUP.....	22
11.1 SUMMARY.....	22
11.2 BEHAVIOUR.....	22
11.3 INSTALLATION.....	22
11.4 EXECUTION / SCHEDULING.....	22
12 RING-BACK.....	23
12.1 SUMMARY.....	23
12.2 BEHAVIOUR.....	23
12.3 INSTALLATION.....	23
12.4 CONFIGURATION.....	24
12.5 EXECUTION / SCHEDULING.....	24
13 AUTOMATED MUSIC ON HOLD UPLOAD.....	25
13.1 SUMMARY.....	25
13.2 BEHAVIOUR.....	25
13.3 INSTALLATION.....	25
13.4 TESTING.....	26
14 RAW CDR SERVICE USING THE TELNET PORT.....	27
14.1 SUMMARY.....	27
14.2 BEHAVIOUR.....	27
14.3 INSTALLATION.....	27

14.4 CONFIGURATION.....	27
14.5 EXECUTION.....	27
15 COM.X STATUS QUERIES FROM YEALINK PHONES (YEALINK-XML).....	28
15.1 SUMMARY.....	28
15.2 BEHAVIOUR.....	28
15.3 INSTALLATION.....	28
15.4 CONFIGURATION.....	29
15.5 EXECUTION.....	29
16 PROFILING A SIP PROVIDER / SIP TRUNK.....	30
16.1 SUMMARY.....	30
16.2 BEHAVIOUR.....	30
16.3 INSTALLATION.....	30
16.4 CONFIGURATION.....	30
16.5 EXECUTION.....	30



1 Introduction

1.1 Overview

This document is intended for the Com.X PBX and Gateway administrator and details enhanced functionality for the Com.X by means of customizations of the system.

Warning: These customizations are provided as a service to the administrator and are not directly supported by Far South Networks. These customizations are examples only and should not be deployed in production systems without thorough testing in the lab.

Warning: Implementing these customizations may affect your software warranty as well as limit access to technical support.

All customizations can be downloaded from:

<http://archive.farsouthnet.com/support/customizations>



2 Call barge

2.1 Summary

The call barge feature allows a user of the Com.X system to inject audio into a conversation in progress.

2.2 Behaviour

A barging party could barge into a call in progress and deliver a message to an internal party by dialing *555xxxx, where xxxx is the extension number of the internal party, without the third party hearing him / her. The internal party hears a call waiting tone (analog phone) or a ring chirp (SIP phone) and then the whisper of the barging party.

2.3 Installation

Add the following to /etc/asterisk/extensions_custom.conf:

```
[app-chanspy-custom]
exten => *_555X.,1,Macro(user-callerid,)
exten => *_555X.,n,Answer
exten => *_555X.,n,Set(mydevice=${DB(AMPUSER/${EXTEN:4}/device)})
exten => *_555X.,n,Set(mydial=${DB(DEVICE/${mydevice}/dial)})
exten => *_555X.,n,GotoIf("${mydial}" == "")?invalid
exten => *_555X.,n,Ringing
exten => *_555X.,n,Dial(${mydial},1)
exten => *_555X.,n,ChanSpy(${mydial},wq)
exten => *_555X.,n(hangup),Hangup
exten => *_555X.,n(invalid),Playback(invalid)
exten => *_555X.,n,Hangup
```

Reload the dial-plan:

```
sudo asterisk -rx "dialplan reload"
```

2.4 Configuration

Ensure no other feature codes make use of *555

If the chanspy extension customization is also implemented, change this feature code to something other than 555 in /etc/asterisk/extensions_custom.conf



3 Blocking extensions and pin sets

3.1 Summary

Specific pins and / or extensions can be blocked from placing outbound calls.

3.2 Behaviour

The following changes in behaviour result from the implementation of this customization:

- This feature works only with outbound route pin sets, not with outbound route passwords
- The user is prompted to enter their pin only once (no retries)
- The user must enter the pin after the beep
- CDR recording of the pin used is dependent on the "Record in CDR" flag on the pin set.
- If pins overlap (e.g. 1234 and 12345), only the first pin in the list will always be available, so best avoid this by keeping all pins the same length
- If an extension is blocked, the call will terminate

3.3 Installation

To install pinset blocking, place the contents of block-by-pin in extensions.conf before:

```
#include extensions_override_freepbx.conf
```

To install extension blocking, place the contents of the extensions_custom.conf file into extension_custom.conf

Place auth.sh in /usr/share/asterisk/bin with the following permissions: -rwxr-xr-x

3.4 Execution / scheduling

To use this customization, issue the following commands using the CLI or AMI:

<pin> indicates the user pin

<exten> indicates the calling extension

To block a pin:

```
database put FSN/PINS/<pin> BLOCKED YES
```

To unblock a pin:

```
database del FSN/PINS/<pin> BLOCKED
```



To enable pin bypass for an extension:

```
database put AMPUSER/<exten> pinless NOPASSWD
```

To disable pin bypass for an extension:

```
database del AMPUSER/<exten> pinless
```



4 Budget trunks

4.1 Summary

Upper thresholds for call billable seconds can be configured for outbound trunks, preventing outbound calls on trunks for which the budget has been exceeded. The threshold can be reset using a configurable time period.

4.2 Behaviour

This feature periodically checks the specified trunks and performs a CDR lookup for each, calculating billable seconds that have expired since the date specified. If the threshold specified is exceeded, the trunk is bypassed during call control. If the condition is no longer met (E.g. change to the configuration file or database or threshold counter reset) the trunk is used in outbound routing. Threshold counters reset after the period specified.

4.3 Installation

To enable this customization, wrap the `/etc/asterisk/extensions_custom.conf` as shown in the example `extensions_custom.conf` file in this directory.

Place `checkBudget` and `checkBudgetTrunk` in `/usr/share/asterisk/bin` with the following permissions: `-rwxr-xr-x`

4.4 Configuration

For each trunk, configure an entry in `/etc/asterisk/budgettrunks` of the form:

ID,CHANNEL,DATE,THRESHOLD

ID	The trunk's ID in the dialplan configuration. This can be determined by watching the CLI as a call is placed over the trunk. This customization prints a message of the form: "Pre-dial on trunk ID"
CHANNEL	The trunk's channel. This can be obtained from the <code>dstchannel</code> field in a call placed over the trunk's CDR record in the CDR database
DATE	The date from which to tally billable seconds. If date is of the form 'MONTHLY=day', e.g. 'MONTHLY=06', billable seconds are always tallied from day 'day' of each month, in the example from the 6th of each month.
THRESHOLD	The second threshold for bypassing the trunk

Example `/etc/asterisk/budgettrunks`



```
2,Zap/9-1,MONTHLY=01,10
```

```
3,Zap/9-2,2010-12-06,500
```

4.5 Execution / scheduling

Configure the checkBudget script to be run at an appropriate time post scheduled backups or generation of the files to be uploaded. The script must run as root. In the example, the trunks are checked every five minutes

```
sudo jed /var/spool/cron/crontabs/root
```

```
*/* * * * * /usr/share/asterisk/bin/checkBudget
```

Restart cron in order for the new schedule to take effect:

```
sudo /etc/init.d/cron restart
```



5 Call billable second limit

5.1 Summary

This customization monitors all calls for a billable second threshold, which, if crossed, results in the channel being soft disconnected with an associated entry in `/var/log/asterisk/messages`

5.2 Behaviour

This feature periodically (every 10 seconds) checks all calls active on the system to determine whether the billable duration of the call exceeds the specified `THRESHOLD` duration. The default `THRESHOLD` duration is 1 hour (3600 seconds)

If a call does exceed the limit, the call is terminated using the “channel request hangup” command. As such this customization is subject to the behaviour of “channel request hangup”.

5.3 Installation

Copy the `checkCallLimit` script to `/usr/share/asterisk/bin/` with execute permissions (`chmod +x`)

5.4 Execution / scheduling

The script should be executed as root (using `sudo`) and requires no parameters

Configure the script to be run at an appropriate time (e.g. system start-up, asterisk start-up, etc.)

Please ensure that the script will not be run more than once.



6 Cause code remapping

6.1 Summary

This customization changes the `HANGUPCAUSE` in the Asterisk dialplan, resulting in the next interpreter of cause codes (e.g. an ISDN trunk in a gateway scenario) to report the protocol codes associated with the hangup cause to its peer.

6.2 Behavior

Example mappings of cause codes to protocol codes include:

Basic Rate ISDN maps the code directly as received from the mapping in the "cause" field of the `DISCONNECT` indication.

Primary Rate ISDN maps the code directly as received from the mapping in the "Cause" field of the `RELEASE` message (77)

6.3 Installation

To enable this customization, add the entries in `extensions_custom.conf` file to `/etc/asterisk/extensions_custom.conf`

Update the `/etc/asterisk/extensions.conf` file and change `exten => s,n(theend),Hangup` to `exten => s,n(theend),Hangup({RC})` as in the example `extensions.conf` in this directory .

Replace the `/usr/share/freepbx/modules/core/functions.inc.php` file with the `functions.inc.php` file in this directory.

Regenerate the dialplan using: `sudo /usr/share/asterisk/bin/module_admin --no-announce reload`

6.4 Configurion

Update `extensions_custom.conf` with the necessary re-mapping as per the examples included and reload the asterisk dialplan: `sudo asterisk -rx "dialplan reload"`

Each mapping is conditionally identified as follows:

```
exten => s,n,GotoIf(["${HANGUPCAUSE}" = "NN"]?mapNN)
```

where NN represents the cause code to be mapped.

Each mapping also has a corresponding map label which performs the mapping:

```
exten => s,n(map16),Set(RC=3)
exten => s,n(map16),Goto(final)
```

Note that it is important to `Goto(final)` to skip over the remaining mappings.



6.5 CDR FTP upload

6.6 Summary

An example FTP upload script is provided that allows the last configuration backup, or alternatively a list of files specified to be automatically uploaded to an FTP server.

6.7 Behaviour

The script can be run in one of two modes. MODE=1 copies the last configuration backup found in the backups directory. The script can easily be adapted to also back up CDRs or voicemail in a similar manner. MODE=2 allows an arbitrary list of files to be zipped using tar and gzip and uploaded to the server.

6.8 Installation

To make use of the `ftp_upload` script, enable automatic upload from the Com.X to the FTP server as follows:

Create a file named `.netrc` in `/home/comma` on the Com.X system.

Populate the login details as follows:

```
default
login username
password password
```

Set the correct permissions on the file:

```
sudo chmod 600 .netrc
```

Place the `ftp_upload` script with execute permissions in `/usr/bin/`

```
sudo cp ftp_upload /usr/bin/
sudo chmod +x /usr/bin/ftp_upload
```

6.9 Configuration

Configure all the necessary fields, including the FTP server IP address and port and the destination (upload) directory on the FTP server.

6.10 Execution / scheduling

If mode 1 is selected, the syntax is simply:

```
ftp_upload 1
```

If mode 2 is selected, the list of files is specified as the second argument (the quotes are important)

```
ftp_upload 2 "file1 file2 dir3/file3 etc"
```

configure the script to be run at an appropriate time post scheduled backups or generation of the files to be uploaded. In the example, an upload takes place every day at 23h41

```
sudo jed /var/spool/cron/crontabs/root
41 23 * * * /usr/bin/ftp_upload 1
```



If you would like the script to run as a specific user (e.g. comma) use:

```
41 23 * * * su - comma -c "/usr/bin/ftp_upload 1"
```

Restart cron in order for the new schedule to take effect:

```
sudo /etc/init.d/cron restart
```



7 Spying on an extension (chanspy extension)

7.1 Summary

This customization extends the spy feature as described in the Com.X Administrator's guide to enable spying on a specified extension as opposed to having to cycle through devices.

7.2 Behaviour

The spy behaviour is as specified in the Com.X Administrator's guide.

7.3 Installation

Add the following to `/etc/asterisk/extensions_custom.conf`. The line `Dial({mydial},1)` presents a beep to the spied on party. This might be a legal requirement.

```
[app-chanspy-custom]
exten => _*555X.,1,Macro(user-callerid,)
exten => _*555X.,n,Answer
exten => _*555X.,n,Set(mydevice=${DB(AMPUSER/${EXTEN:4}/device)})
exten => _*555X.,n,Set(mydial=${DB(DEVICE/${mydevice}/dial)})
exten => _*555X.,n,GotoIf("${mydial}" == ""?invalid)
exten => _*555X.,n,Ringing
exten => _*555X.,n,Dial({mydial},1)
exten => _*555X.,n,ChanSpy({mydial},wq)
exten => _*555X.,n(hangup),Hangup
exten => _*555X.,n(invalid),Playback(invalid)
exten => _*555X.,n,Hangup
```

7.4 Configuration

Ensure no other feature codes make use of *555

If the barge customization is also implemented, change its feature code to something other than 555 in `/etc/asterisk/extensions_custom.conf`

7.5 Execution / scheduling

Dial `*555<extension>` to spy, e.g. `*5551001` to spy on extension 1001



8 Adding NAGIOS monitoring support

8.1 Summary

The Com.X can be configured for custom remote monitoring using the open-source NAGIOS monitoring framework. Some example monitoring scripts are included with this customization .

8.2 Behaviour

NAGIOS behaviour is specific to the version installed. Please refer to the NAGIOS documentation.

8.3 Installation

Download nagios core and plugins from <http://www.nagios.org/> and copy onto the target:

```
wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-3.2.3.tar.gz
wget http://prdownloads.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.15.tar.gz
```

Build nagios:

```
sudo aptitude install build-essential
sudo aptitude install libgd2-dev libapache2-mod-php5
sudo -s
/usr/sbin/useradd -m -s /bin/bash nagios
passwd nagios
/usr/sbin/groupadd nagcmd
/usr/sbin/usermod -a -G nagcmd nagios
/usr/sbin/usermod -a -G nagcmd www-data
tar xvfz nagios-3.2.3.tar.gz
cd nagios-3.2.3
./configure --with-command-group=nagcmd
make all
make install
make install-init
make install-config
make install-commandmode
```

Configure nagios by setting the nagiosadmin email address in:

```
jed /usr/local/nagios/etc/objects/contacts.cfg
```

Disable escaping of HTML tags in order to support clickable links in output as well as multi-line output separated by
. Edit /usr/local/nagios/etc/cgi.cfg and set:



```
escape_html_tags=0
```

Configure nagios web services:

```
make install-webconf
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
/etc/init.d/apache2 restart
cd ..
```

Install the plugins:

```
tar xzf nagios-plugins-1.4.15.tar.gz
cd nagios-plugins-1.4.15/
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
make install
```

Initialize the service:

```
ln -s /etc/init.d/nagios /etc/rcS.d/S99nagios
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo apt-get install mailx
/etc/init.d/nagios start
```

Exit root

```
exit
```

Test the service by logging in at `http://<ip>/nagios/` using 'nagiosadmin' and 'password' (the password set above)

In order for nagios to run scripts that require root permissions (e.g. asterisk commands) add the necessary commands to the `/etc/sudoers` file, e.g.:

```
nagios ALL=(ALL) NOPASSWD:/usr/sbin/asterisk
```

8.4 Configuration

Add Com.X specific monitoring services:

```
cd /usr/local/nagios/etc/objects/
sudo touch comx.cfg
```

Edit `/usr/local/nagios/etc/nagios.cfg` and add:

```
# Definitions for monitoring Com.X
cfg_file=/usr/local/nagios/etc/objects/comx.cfg
```

Configure group and permissions:

```
sudo chown nagios:nagios comx.cfg
sudo chmod 664 comx.cfg
```

Now add `localhost` and some services to the comx configuration file:

```
sudo jed comx.cfg
```



```
define command{
    command_name    eth
    command_line    $USER1$/monitor_eth $ARG1$
}

define service{
    use              generic-service    ; Inherit default values from a
template
    host_name        localhost
    service_description    Ethernet interface 0
    check_command    eth!0
}

define service{
    use              generic-service    ; Inherit default values from a
template
    host_name        localhost
    service_description    Ethernet interface 1
    check_command    eth!1
}
```

Create scripts as required, using the `monitor_eth` script provided as an example:

```
sudo chown nagios:nagios monitor_eth
sudo chmod 755 monitor_eth
```

Check the configuration and reload nagios:

```
sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
sudo /etc/init.d/nagios reload
```

8.5 Execution / scheduling

Please refer to the NAGIOS documentation for information on service management, operation and scheduling.



9 PRI port-based round-robin with fail-over

9.1 Summary

This customization is intended for Com.X2 systems, where the dial-plan is optimized and completely custom.

This customization should not be implemented on Com.X systems with GUI capability unless in conjunction with an XCT support engineer.

9.2 Behaviour

This example dialplan provided here is configured for 2 PRI ports, each in its own group, namely g1 and g2.

Each call is placed alternately over the next PRI trunk group, round robin within the group.

If a trunk group cannot place the call, fail-over is to the next trunk group.

9.3 Installation

Add the example `extensions_additional.conf` content to the system's `/etc/asterisk/extensions_additional.conf`

```
[from-trunk]
exten => _X.,1,GotoIf("${DB(FSN/LASTGROUPUSED)}" = "1")?dialg2)
exten => _X.,n(dialg1),Set(DB(FSN/LASTGROUPUSED)=1)
exten => _X.,n,Dial(Zap/r1/${EXTEN})
exten => _X.,n,Dial(Zap/r2/${EXTEN})
exten => _X.,n,Goto(done)
exten => _X.,n(dialg2),Set(DB(FSN/LASTGROUPUSED)=2)
exten => _X.,n,Dial(Zap/r2/${EXTEN})
exten => _X.,n,Dial(Zap/r1/${EXTEN})
exten => _X.,n(done),NoOp()
exten => _X.,n,Hangup
```

9.4 Configuration

Update the dial-plan as required for the system's number of PRIs and fail-over / dialing priorities.

9.5 Execution / scheduling

```
sudo asterisk -vvvr "dialplan reload"
```



10 Scheduled recording archiving to external device

10.1 Summary

This customization provides functionality for a Com.X to automatically move (archive) call recordings to an alternative location (e.g. USB drive) based on a specified criterium.

10.2 Behaviour

The script then lists the recordings in the `WAVDIR` directory, determines the current data and attempts to move all recordings older than today to the `ARCHIVE` directory. The scripts compares copied files with original files to ensure correctness. Any failures are reported in `/var/log/syslog`, and if a recording could not be copied, it is left in the `WAVDIR` directory. If successfully copied, the recording is deleted from the `WAVDIR` directory.

At the end of the script, it checks to see whether the `ARCHIVEDEVICE`'s usage has exceeded `THRESHOLD`, in which case an email is sent to `NOTIFY` and the warning logged in the `syslog`.

10.3 Installation

Place the `auto_archive.sh` script with execute permissions in `/usr/bin/` :

```
sudo cp auto_archive.sh /usr/bin/  
sudo chmod +x /usr/bin/auto_archive.sh
```

Install the bash calculator required to calculate historical dates:

```
sudo aptitude install bc
```

10.4 Configuration

Configure all the necessary fields, including the `NOTIFY` email address, `ARCHIVEDEVICE`, `THRESHOLD`, `ARCHIVE` and `SITE` fields.

10.5 Execution / scheduling

The script is self-contained and can be executed using `/usr/bin/auto_archive.sh`

Then configure the script to be run at an appropriate time post scheduled backups or generation of the files to be uploaded. In the example, an upload takes place every day at 23h41

```
sudo jed /var/spool/cron/crontabs/root  
41 23 * * * /usr/bin/auto_archive.sh
```

Restart cron in order for the new schedule to take effect:

```
sudo /etc/init.d/cron restart
```



11 Recording lookup

11.1 Summary

This customization allows a user to look up the recording file for a recording made in a certain date range, where once of the parties' extension or number is known. This allows fast search of large call recording databases. A utility is also provided to send a recording to an email address

11.2 Behaviour

The `lookup_recordings` script returns, for each entry matching the query, the call date, the unique call ID, as well as the call recording filename in

`/var/spool/asterisk/monitor` should the call have a recording. On successful completion of the script, all recordings listed are placed in `/home/comma/recordings`

The `fetch_recording` script sends the specified recording file to the email specified using `sendmail`. The mail progress can be followed in `/var/log/mail.info`

11.3 Installation

The `lookup_recordings` script requires a 'recordings' directory in the `/home/comma` directory on the Com.X system. You can create this directory as follows:

```
mkdir /home/comma/recordings
```

The `fetch_recording` script requires the installation of `mutt` in order to attach the recording file to an email:

```
sudo aptitude update
sudo aptitude install mutt
```

11.4 Execution / scheduling

To make use of the `lookup_recordings` script, provide the start date and time, and end date and time of the date range in which recordings should be looked up. Also provide an extension (or number) for one of the parties of the calls to be looked up.

The `fetch_recording` script requires the call id returned by the `lookup_recording` script as well as an email address to send the recording to.



12 Ring-back

12.1 Summary

Ring-back can be configured on Com.X systems in order to change the behaviour of calls encountering a busy extension. This allows ring-back to the originator of the call, or reception, depending on the call scenario.

12.2 Behaviour

On any transfer, if both voicemail and follow me for the target extension is disabled, and the target is busy or does not answer, the call should be sent on to the configured operator extension (reception)

Follow me enabled, voicemail enabled -> no answer -> tries follow-me -> still no answer -> follow-me configured destination (usually extension voicemail)

Follow me enabled, voicemail disabled -> no answer -> tries follow-me -> still no answer -> follow-me configured destination

Follow me disabled, voicemail enabled -> no answer -> voicemail

Follow me disabled, voicemail disabled -> no answer -> rings reception -> still no answer -> reception's voicemail

Follow me enabled, voicemail enabled -> busy -> tries follow-me -> still no answer -> follow-me configured destination

Follow me enabled, voicemail disabled -> busy -> tries follow-me -> still no answer -> follow-me configured destination

Follow me disabled, voicemail enabled -> busy -> voicemail

Follow me disabled, voicemail disabled -> internal call -> busy -> ring-back initiated from called to caller

Follow me disabled, voicemail disabled -> external call -> busy -> rings reception -> still no answer -> reception's voicemail

Users need to be informed as to what to expect. With ring-back enabled their phones should 'ring by itself' once available. If the hand-set is picked up, they should hear ringing while the initial called party is being contacted.

If the other party is now on the phone, also does not have follow-me and does not have voicemail enabled, when the handset of the phone on which ring-back was initiated is picked up, the user would hear a busy tone. A second round of ring-back should then be initiated. I.e. once the original caller (now on the phone) ends the call, his phone would ring, and picking it up, the initial called user would be attempted.

12.3 Installation

Make a copy of `/etc/asterisk/extensions.conf` so that your system can easily be reverted in case of failure.

Please replace `/etc/asterisk/extensions.conf` with the file attached and place `ringback.sh` in `/usr/share/commagui/bin/scripts/` with execute permissions.



12.4 Configuration

In `ringback.sh`, configure the number of retries and the delay between retry. Eventually trying to ring back will give up based on these values.

In General settings, configure the Operator extension to reception's extension

12.5 Execution / scheduling

```
sudo /etc/init.d/asterisk restart
```



13 Automated music on hold upload

13.1 Summary

This customization allows MP3 files to be uploaded (manually or scheduled) to a Com.X, to automatically update the music on hold class. This customization supports only upload of MP3 files, not MP4 or WAV files.

13.2 Behaviour

The MP3 files are expected to be placed in a pre-defined location on the Com.X and are converted to WAV on the Com.X and placed in a specific location.

Upload takes immediate affect when the cron script is run. Music on hold already playing to callers changes mid-track and starts playing the new track.

13.3 Installation

On the Com.X:

On the GUI, select Advanced, Music on hold and create a new category called "autoupload"

Copy the `moh_upload` script to `/usr/bin/moh_upload` and ensure it has execute permissions.

Configure the script to be run at an appropriate time. In the example, the script checks for a new file every day at 23h41:

```
sudo jedit /var/spool/cron/crontabs/root
41 23 * * * /usr/bin/moh_upload
```

Restart cron in order for the new schedule to take effect:

```
sudo /etc/init.d/cron restart
```

On the Com.X GUI, select an inbound route or service, and configure the music on hold category to 'autoupload'

Upload takes immediate affect when the cron script is run. Music on hold already playing to callers changes mid-track and starts playing the new track.

On the system that will perform the upload (system A):

Copy the MP3 file to the Com.X in `/tmp/` and call it `upload.mp3` Note: locations and filenames are case sensitive and needs to be specified exactly as in this README:

```
scp <filename.mp3> comma@<ip-of-comx>:/tmp/upload.mp3
```

The upload of the MP3 file could be automated as well by enabling automatic ssh login to Com.X as follows:

Create an SSH key:

```
ssh-keygen
```



(select the default file location and do not enter a password, assumed here to be
~/.ssh/id_rsa.pub)

Copy the ID to the Com.X in order to enable automatic login:

```
ssh-copy-id ~/.ssh/id_rsa.pub comma@<comx-ip>
```

13.4 Testing

After upload of an MP3 file to the Com.X, the process can be tested by placing a call to the system (destination with music class "autoupload") and then manually running:

```
sudo /usr/bin/moh_upload
```

It might be desirable to remove the uploaded MP3 from the system after conversion in cases where cron runs the script frequently. To activate this, uncomments the `rm` line in `moh_upload`



14 Raw CDR service using the telnet port

14.1 Summary

This customization provides the full CDR field list to a TMS system via a TCP service running on the telnet port (23)

14.2 Behaviour

This service, upon connection on TCP port 23, starts from the date specified in `lastdate` and proceeds to deliver all CDR fields from the Com.X CDR database in plain text.

After each record that has been delivered, `lastdate` is updated to the timestamp of the record, ensuring that in the case of service disruption, delivery of records can start from the last successful entry.

If no more records are available, the service keeps the TCP connection open for 10 seconds, after which the connection is terminated.

14.3 Installation

```
sudo aptitude update
sudo aptitude install courier-base
sudo mkdir /var/lib/commatpcdr
sudo touch /var/lib/commatpcdr/lastdate
sudo cp commatpcdr /usr/bin/
sudo cp commatpcdrsrv /usr/bin/
sudo chmod +x /usr/bin/commatpcdr
sudo chmod +x /usr/bin/commatpcdrsrv
sudo cp commatpcdr.init /etc/init.d/commatpcdr
sudo chmod +x /etc/init.d/commatpcdr
sudo update-rc.d commatpcdr defaults
```

14.4 Configuration

Please see the Com.X Administration guide for use of the `lastdate` file to configure the starting point of cdr delivery.

For this customization the name of the file is `/usr/share/commatpcdr/lastdate`

Please note that this customization does not support a `patterns` file and delivers all CDR fields.

14.5 Execution

```
sudo /etc/init.d/commatpcdr start
```



15 Com.X status queries from Yealink phones (yealink-xml)

15.1 Summary

The Yealink T2X range of phones includes an XML browser, which allows for the configuration of comprehensive menu systems that can access any information and effect any action available from a system as published by the system using an XML URI.

In this customization an HTTP XML URI is used to query the Day/Night status of the Com.X system

15.2 Behaviour

Once configured, pressing the soft-key on the Yealink T2X phone initiates an XML file query to the URI specified in the soft-key configuration.

This query results in the Com.X performing some actions to retrieve the information and to present the information in XML format.

Note: Ensure that any XML and script files used to deliver the service on the Com.X has limited visibility (permissions) so as not to compromise system security.

15.3 Installation

Create a directory called `daynight` in `/var/www/` on the Com.X system

Copy the `functions.inc.php` and `daynight.php` files provided with this customization to `/var/www/daynight`

Configure appropriate permissions for both files

`functions.inc.php`

```
<?php
include('/usr/share/asterisk/agi-bin/phpagi-asmanager.php');

function ami($command){
    $amiManager = new AGI_AsteriskManager();
    if (!$res = $amiManager->connect("127.0.0.1", "admin" , "<freepbx-key>") ){
        exit("Error");
    }

    $res = $amiManager->Command($command);
    $amiManager->disconnect();
    return $res;
}

function dayNight(){
```



```
$res=ami("database show daynight <CN>");
$mode="DAY";
foreach ($res as $value){
    $testval=strstr(strstr($value, "<CN>"), "NIGHT");
    if ($testval != ""){
        $mode="NIGHT";
    }
}
$serverip="<serverip>";
echo '<YealinkIPPhoneTextMenu defaultIndex="1" style="numbered" Beep="no"
wrapList="yes" Timeout="70" LockIn="no"><Title wrap="yes">Day/night mode is: '.
$mode.'</Title><MenuItem><Prompt>Refresh</Prompt><URI>http://'.
$serverip.' /daynight/daynight.php</URI></MenuItem>';
echo'    <SoftKey
index="1"><Label>Refresh</Label><URI>SoftKey:Refresh</URI></SoftKey></YealinkIPPhoneTe
xtMenu>';
}
?>
```

daynight.php

```
<?php
include( '/var/www/daynight/functions.inc.php' );
daynight();
?>
```

15.4 Configuration

Edit the functions.inc.php file and replace <serverip> with the IP of the Com.X as seen by the Yealink phone.

Replace <freepbx-key> in functions.inc.php with the secret in /etc/asterisk/manager.d/freepbx.conf

Replace <CN> with the appropriate day/night index (e.g. c0)

Configure a soft-key on the Yealink to 'XML Browser' and point the key to the following URI: <http://<comx-ip>/daynight/daynight.php>

15.5 Execution

Press the softkey on the yealink phone.



16 Profiling a SIP provider / SIP trunk

16.1 Summary

In trouble-shooting call quality and call drop issues over SIP trunks, it is very useful to have snapshots as well as long-term history of call quality and performance statistics (jitter, packet loss, etc.)

This customization provides some of these measures via a simple script interface.

Note: this customization requires asterisk 1.4.37 or later

16.2 Behaviour

When run, the `peer_stats` script either takes a snapshot of the current RTCP statistics for all calls of a given peer, or logs these statistics to a file specified, and continues to do so every 5 seconds (this interval can be changed by editing the scripts and changing the variable `N`)

16.3 Installation

Copy the `peer_stats` script onto the system and make it executable:

```
chmod +x peer_stats
```

16.4 Configuration

Edit the `peer_stats` file and update the interval variable `N` as required.

16.5 Execution

Usage: `peer_stats [peer-name] <output>`

```
./peer_stats neotel
```

If `<output>` file is specified, will profile the peer every `N` seconds and append the results in `<output>`

```
./peer_stats neotel neotelLog
```