XiVO Solutions Documentation

Avencall

Mar 18, 2021

Contents

1 Introduction **3**
- 1.1 XiVO History . . . 3
- 1.2 GPRD . . . 4

2 Getting Started **5**

3 Installation & Upgrade Guide **11**
- 3.1 XiVO Installation & Upgrade . . . 11
- 3.2 XiVOcc Installation & Upgrade . . . 53
- 3.3 XiVO Distributed System . . . 76

4 Administrator's Guide **83**
- 4.1 XiVO Administration . . . 83
- 4.2 XiVOcc Administration . . . 162
- 4.3 Troubleshooting . . . 181

5 IPBX Configuration Guide **195**
- 5.1 Advanced Configuration . . . 195
- 5.2 Boss Secretary Filter . . . 202
- 5.3 Call Completion . . . 204
- 5.4 Call Permissions . . . 207
- 5.5 Call Logs . . . 209
- 5.6 Conference Room . . . 210
- 5.7 CTI Server . . . 213
- 5.8 Display customer informations . . . 217
- 5.9 Devices . . . 220
- 5.10 Directories . . . 222
- 5.11 Directed Pickup . . . 240
- 5.12 Entities . . . 241
- 5.13 Fax . . . 242
- 5.14 Graphics . . . 249
- 5.15 Groups . . . 249
- 5.16 Group Pickup . . . 251
- 5.17 Incall . . . 251
- 5.18 Interconnections . . . 253
- 5.19 Interactive Voice Response . . . 269
- 5.20 Monitoring . . . 279
- 5.21 Music on Hold . . . 279
- 5.22 Paging . . . 281
- 5.23 Parking . . . 282
- 5.24 Phonebook . . . 285

5.25 Provisioning . . . 286
5.26 SCCP Configuration . . . 299
5.27 Schedules . . . 304
5.28 Sound Files . . . 307
5.29 Switchboard . . . 308
5.30 Users . . . 331
5.31 Voicemail . . . 337
5.32 WebRTC . . . 342
5.33 Web Services Access . . . 345

6 Contact Center **349**
6.1 Agents . . . 349
6.2 Queues . . . 351
6.3 Contact Center Management . . . 357
6.4 CC Agent Environment . . . 366
6.5 Call Qualifications . . . 376
6.6 Recording . . . 379
6.7 Callbacks . . . 382
6.8 Profile Management . . . 389
6.9 Skills-Based Routing . . . 391
6.10 Reporting and statistics . . . 397
6.11 Reporting . . . 410

7 XiVO Centralized User Management **415**
7.1 Intended usage and features . . . 415
7.2 Installation . . . 417
7.3 Upgrade . . . 419
7.4 Configuration of Xivos for Centralized Routing . . . 421
7.5 Manual installation . . . 422
7.6 Web interface . . . 423
7.7 REST API . . . 431

8 User's Guide **443**
8.1 UC Assistant . . . 443
8.2 Desktop Applications . . . 454
8.3 WebRTC Environment . . . 460

9 API and SDK **463**
9.1 Unified Communication Framework . . . 463
9.2 Third Party Integration . . . 505
9.3 Recording server REST Api . . . 508
9.4 XiVO REST Api . . . 514
9.5 XiVO Configuration MGT REST Api . . . 569
9.6 Subroutine . . . 578
9.7 Queue logs . . . 581

10 Contributing **583**
10.1 Contributing to the Documentation . . . 583
10.2 Debugging Asterisk . . . 586
10.3 Debugging Daemons . . . 589
10.4 Generate your own prompts . . . 590
10.5 XiVO Guidelines . . . 591
10.6 Network . . . 592
10.7 Debian packaging for XiVO . . . 592
10.8 Profiling Python Programs . . . 593
10.9 Style Guide . . . 594
10.10 Translating XiVO . . . 600
10.11 XiVO Package File Structure . . . 601
10.12 CTI Server . . . 602

10.13 Diagrams . . . 637
10.14 Provisioning . . . 639
10.15 SCCP . . . 648
10.16 Web Interface . . . 651
10.17 Community Documentation . . . 653

11 Release Notes **657**
11.1 Borealis (2018.16) . . . 657
11.2 Borealis Bugfixes Versions . . . 659
11.3 Borealis Intermediate Versions . . . 670

12 Indices and tables **679**

Index **681**

Important: What's new in this version ?

- XiVO Distributed System - Be able to manage large telephony system by horizontaly scaling media servers.
- UC Assistant: volume indication for the speaker and the microphone for WebRTC calls.
- Enhancement on recording feature.
- Debian 9 (stretch) support.

See *Borealis (2018.16)* page for the complete list of **New Features** and **Behavior Changes**.

XiVO solutions developed by Avencall is a suite of PBX applications based on several free existing components including Asterisk and our own developments. This powerful and scalable solution offers a set of features for corporate telephony and call centers to power their business.

You may also have a look at our development blog for technical news about the solution

CHAPTER 1

Introduction

XiVO solutions is a suite of PBX applications developed by Avencall, based on several free existing components including Asterisk Play Akka and Scala. It provides a solution for enterprises who wish use modern communication services (IPBX, Unified Messaging, . . .) api and application to businesses.

It gives especially access to outsourced statistics, real-time supervision screens, third-party CTI integration and recording facilities.

XiVO solutions is free software. Most of its distinctive components, and XiVO solutions as a whole, are distributed under the *GPLv3 license* and or the *LGPLv3 license*..

XiVO solutions documentation is also available as a downloadable HTML, EPUB or PDF file. See the downloads page for a list of available files or use the menu on the lower right.

1.1 XiVO History

XiVO was created in 2005 by Sylvain Boily (Proformatique SARL). The XiVO mark is now owned by Avencall SAS after a merge between Proformatique SARL and Avencall SARL in 2010. The XiVO core team now works for Avencall in Lyon (France) and Prague (Czech Republic)

- XiVO 1.2 was released on February 3, 2012.
- XiVO 13.07 was the last version with Asterisk 1.8.21.0
- XiVO 13.08 includes Asterisk 11.3.0 in May 2013
- XiVO 13.25 is running under Wheezy in December 2013
- XiVO 14.02 is now called XiVO Five to celebrate the editor 5th year
- XiVO 15.13 runs Asterisk 13 in July 2015
- XiVO 15.20 is running under Jessie in January 2016
- XiVO 2016.02 starts the new versioning system including XiVO-CC and XiVO-UC modules in October 2016
- XiVO solutions 2016.04 includes new XiVO assistant, web edition, mobile edition and Desktop edition in December 2016
- XiVO solutions 2017.03 in April 2017 is the first Long Term Support version known as Five
- XiVO solutions 2017.11 in October 2017 is the second Long Term Support release known as Polaris

In April 2017 the 3rd, Avencall released the fist Long Term Support release of XiVO.solution software suite. The main goal is to offer a stable version every 6 months even if the team is still doing small and agile iterations of 3 weeks. This release LTS 2017.03 will be the last XiVO Five.

In October 2017, Avencall released the second LTS release of XiVO.solutions software suite, Polaris: among other things, it includes major improvements in the CC Agent application.

Next release is on the way, will be called XiVO Aldebaran and will be available in April 2018.

1.2 GPRD

XiVO architecture was redesigned to be GPRD compliant by being "privacy by design", which is the GPRD ADN. We have in this respect refined all the features since Polaris to be 100% compliant.

CHAPTER 2

Getting Started

This section will show you how to create a user with a SIP line. This simple use case covers what a lot of people need to start using a phone. You can use these steps for configuring a phone (e.g a softphone, an Analog-to-Digital switch or a SIP phone).

This tutorial doesn't cover how to automatically provision a supported device. For this, consult the *provisionning section.*

We first need to log into the XiVO web interface. The web interface is where you can administer the whole system.

When logged in, you will see a page with all the status information about your system. This page helps you monitor the health of your system and gives you information about your network. Please note the IP address of your server, you will need this information later on when you will configure your device (e.g. phone)

To configure a device for a user, start by navigating to the IPBX menu. Hover over the *Services* tab, a dropdown menu will appear. Click on *IPBX*.

Select the *Users* setting in the left menu.

From here, press on the "plus" sign. A pop up will appear where you can click on *Add.*

We now have the form that will allow us to create a new user. The three most important fields are 'First name', 'Last name' and 'Language'. Fill in the fields and click on *Save* at the bottom. For our example, we will create a used called 'Alice Wonderland'.

Afterwards, click on the "Lines" tab.

Enter a number for your phone. If you click inside the field, you will see the range of numbers you can use. For our example, we will use '1000'.

By default, the selected protocol is SIP, which is what we want for now. Click on Save to create the line.

We now have a user named 'Alice Wonderland' with the phone number '1000'.

Now we need to go get the SIP username and password to configure our phone. Go back to the IPBX menu on the left, and click on 'Lines'.

You will see a line associated with the user we just created. Click on the pencil icon to edit the line.

We can now see the username and password for the SIP line. you can configure your phone using the IP for your server, the username and the password.

Fig. 1: Logging into the XiVO

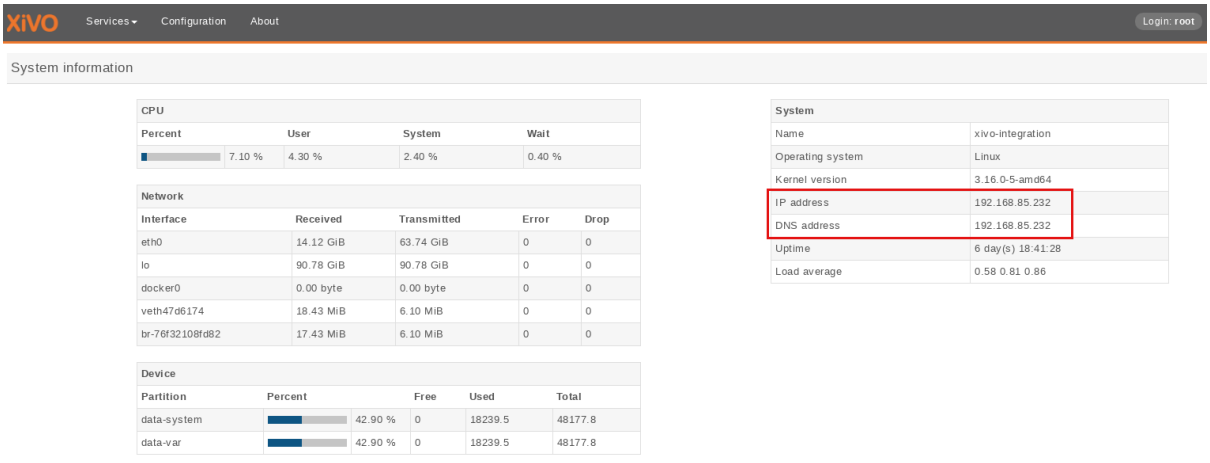


Fig. 2: System informations

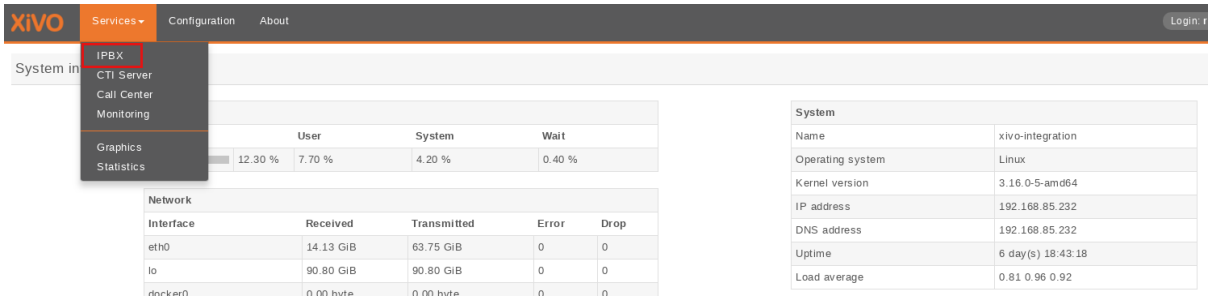


Fig. 3: Menu IPBX

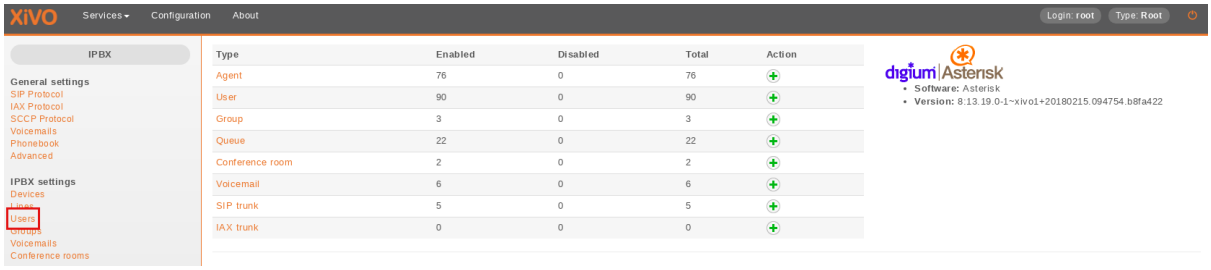


Fig. 4: Users settings

Fig. 5: Adding a new line

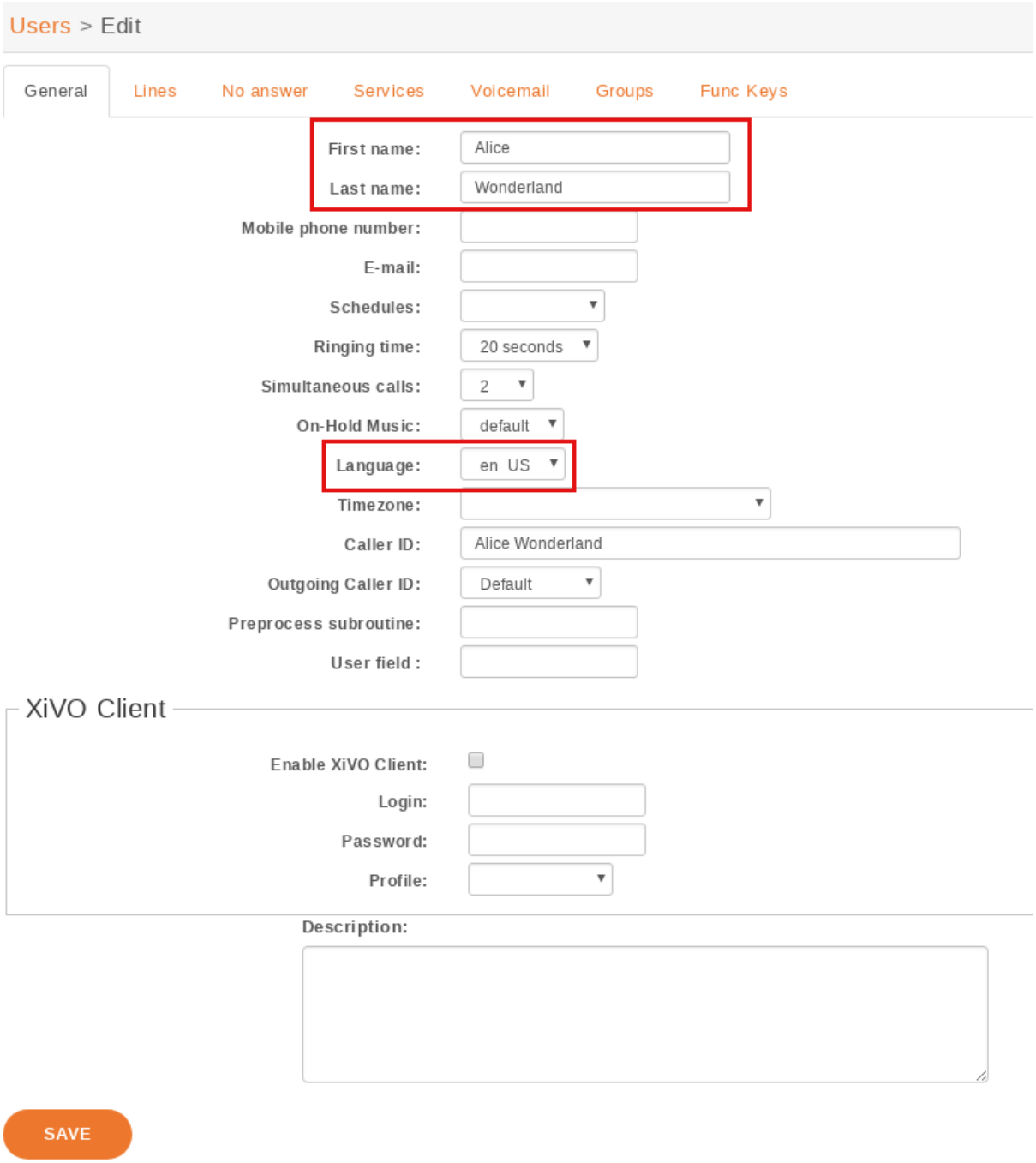


Fig. 6: User information

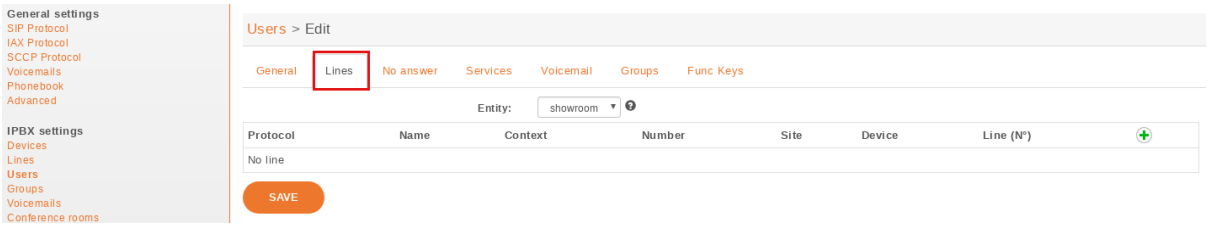


Fig. 7: Lines menu

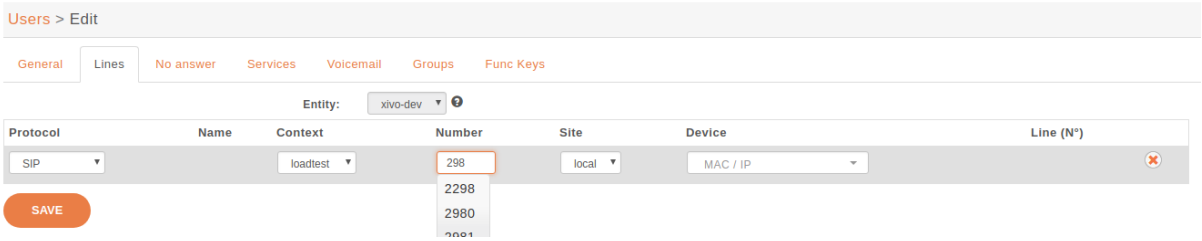


Fig. 8: Line information

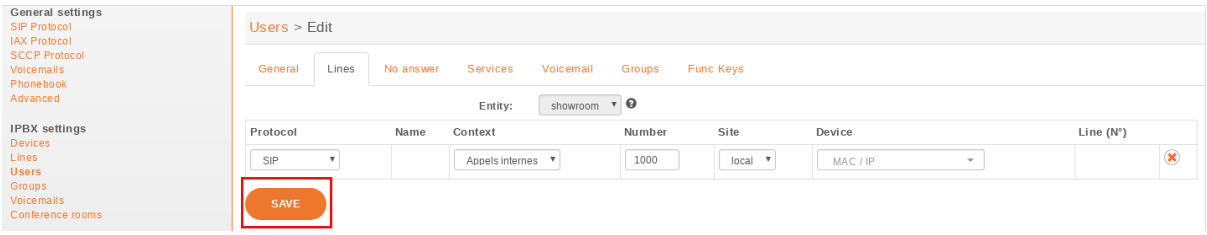


Fig. 9: Save

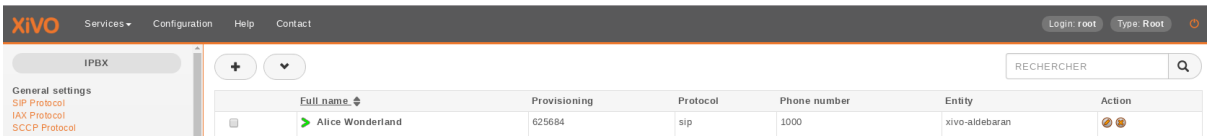


Fig. 10: User added information

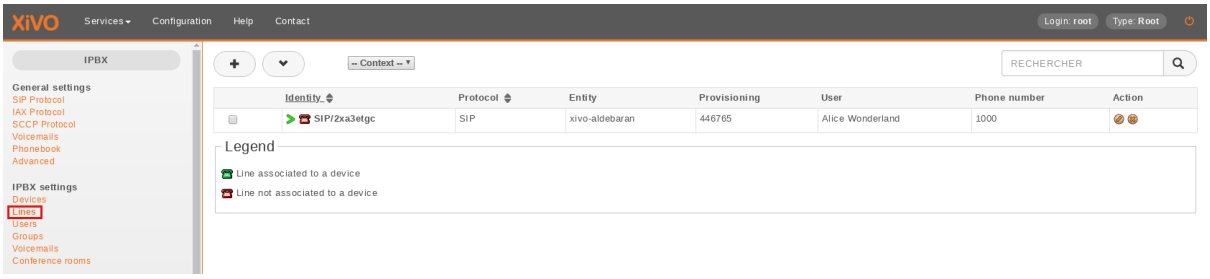


Fig. 11: Lines information

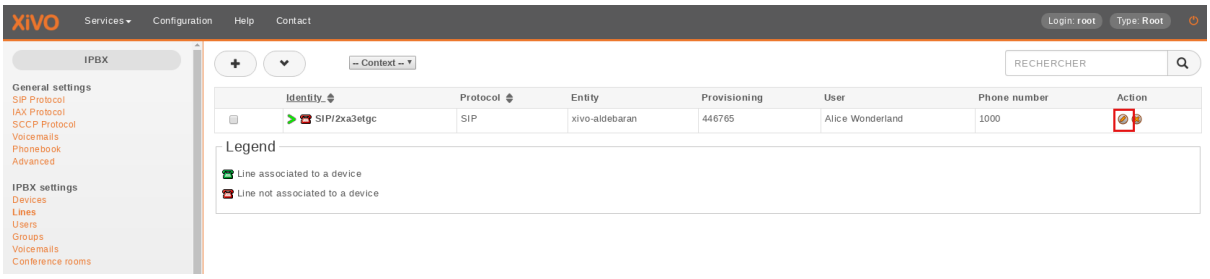


Fig. 12: Edit line

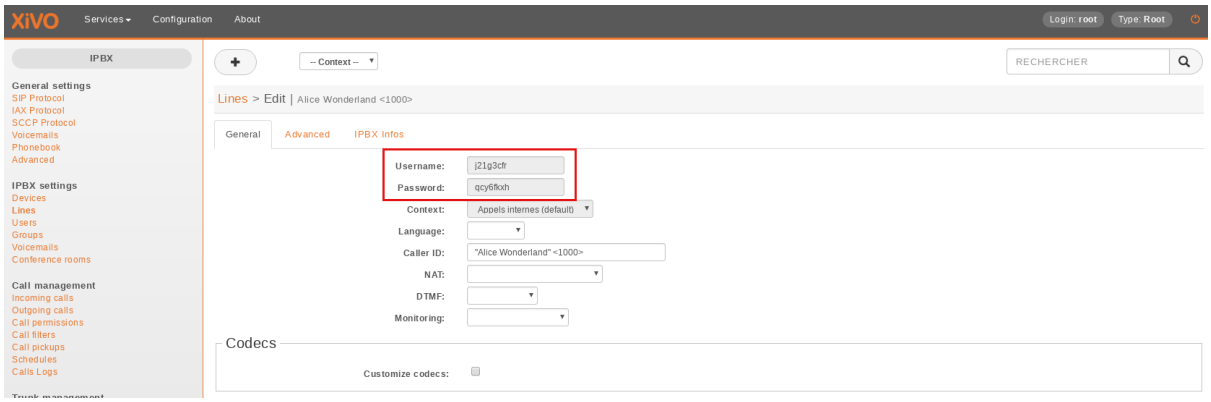


Fig. 13: General line information

CHAPTER 3

Installation & Upgrade Guide

In-depth documentation on installation and deployment of XiVO solution systems.

3.1 XiVO Installation & Upgrade

3.1.1 Installing the System

Please refer to the section *Troubleshooting* if ever you have errors during the installation.

There are two official ways to install XiVO:

- using the official ISO image
- using a minimal Debian installation and the XiVO installation script

XiVO can be installed on both virtual (QEMU/KVM, VirtualBox, ...) and physical machines. That said, since Asterisk is sensitive to timing issues, you might get better results by installing XiVO on real hardware.

Warning: By default XiVO installation will pre-empt network subnets 172.17.0.0/16 and 172.18.1.0/24. If these subnets are already used, some manual steps will be needed to be able to install XiVO. These steps are not described here.

Installing from the ISO image

Note: Our ISO image does not support UEFI system

- Download the ISO image. (latest LTS version) (all versions)
- Boot from the ISO image, select `Install` and follow the instructions. You must select locale `en_US.UTF-8`.
- At the end of the installation, you can continue by running the *configuration wizard.*

During the installation of Debian, only a proxy that supports proxying http/https requests may eventually be entered. Otherwise GPG key of XiVO repository will not be installed and must be added manually:

```
wget http://mirror.xivo.solutions/xivo_current.key -O - | apt-key add -
```

Installing from a minimal Debian installation

XiVO can be installed directly over a **64-bit** Debian **9 Stretch**. When doing so, you are strongly advised to start with a clean and minimal installation of Debian **Stretch**.

The latest installation image for Debian **Stretch** can be found at https://www.debian.org/releases/stretch/debian-installer.

Requirements

The installed Debian must:

- use the architecture `amd64`
- have a default locale `en_US.UTF-8`
- use `ext4` filesystem (for compatibility with docker overlay2 storage driver)
- use legacy network interface naming `eth#`. To change the network interface naming to `eth#` use this procedure:
  - Edit `/etc/default/grub`, find line `GRUB_CMDLINE_LINUX` and set it to `GRUB_CMDLINE_LINUX="net.ifnames=0"`
  - Run `update-grub`
  - Edit interface names in `/etc/network/interfaces`
  - Reboot the machine

Installation

Note: If your server needs a proxy to access Internet, configure the proxy for `apt`, `wget` and `curl` as documented in *Proxy Configuration*.

Once you have your Debian stretch properly installed, download the XiVO installation script and make it executable:

```
wget http://mirror.xivo.solutions/xivo_install.sh
chmod +x xivo_install.sh
```

And run it:

```
./xivo_install.sh -a 2018.16-latest
```

At the end of the installation, you can continue by running the *configuration wizard.*

Alternative versions

The installation script can also be used to install an *archive version* of XiVO (14.18 or later only). For example, if you want to install XiVO 2018.05-latest:

```
./xivo_install.sh -a 2018.05-latest
```

When installing an archive version, note that:

- versions 14.18 to 15.19 of XiVO can only be installed on a Debian 7 (wheezy) system
- the 64-bit versions of XiVO are only available starting from 15.16

You may also install development versions of XiVO with this script. These versions may be unstable and should not be used on a production server. Please refer to the usage of the script:

```
./xivo_install.sh -h
```

Other installation methods

It's also possible to install XiVO by PXE. It is not documented here.

3.1.2 Running the Wizard

After the system installation, you must go through the wizard before being able to use your XiVO. Open your browser and enter your server's IP address in the navigation bar. (For example: http://192.168.1.10)

Language

You first have to select the language you want to use for the wizard.

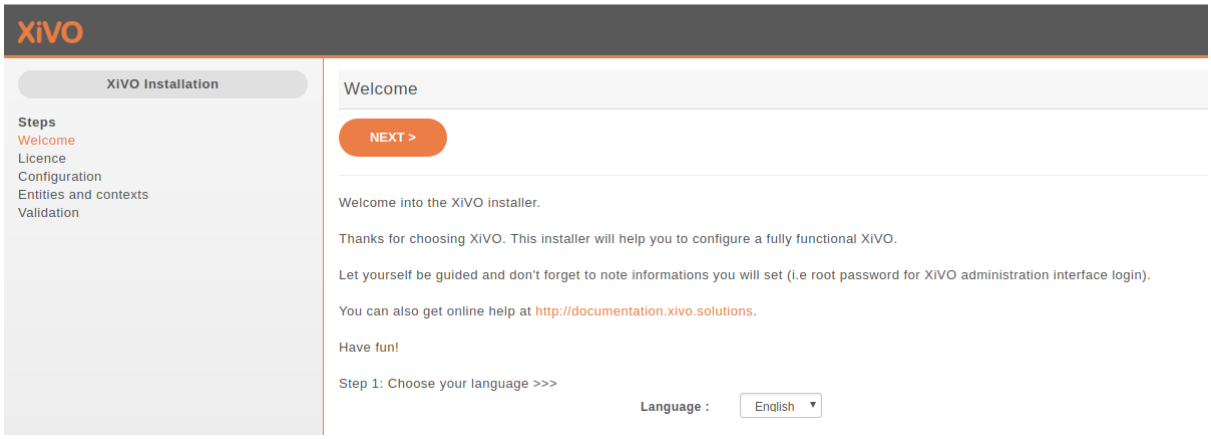


Fig. 1: Select the language

License

You then have to accept the *GPLv3 License* under which XiVO is distributed.

Configuration

1. Enter the hostname (Allowed characters are : `A-Z a-z 0-9 -`)
2. Enter the domain name (Allowed characters are : `A-Z a-z 0-9 - .`)
3. Enter the password for the `root` user of the web interface,
4. Configure the IP address and gateway used by the VoIP interface
5. Modify the DNS server information if needed
6. And finally, choose (or not) to apply the default configuration for France (see: *Default configuration for France*).

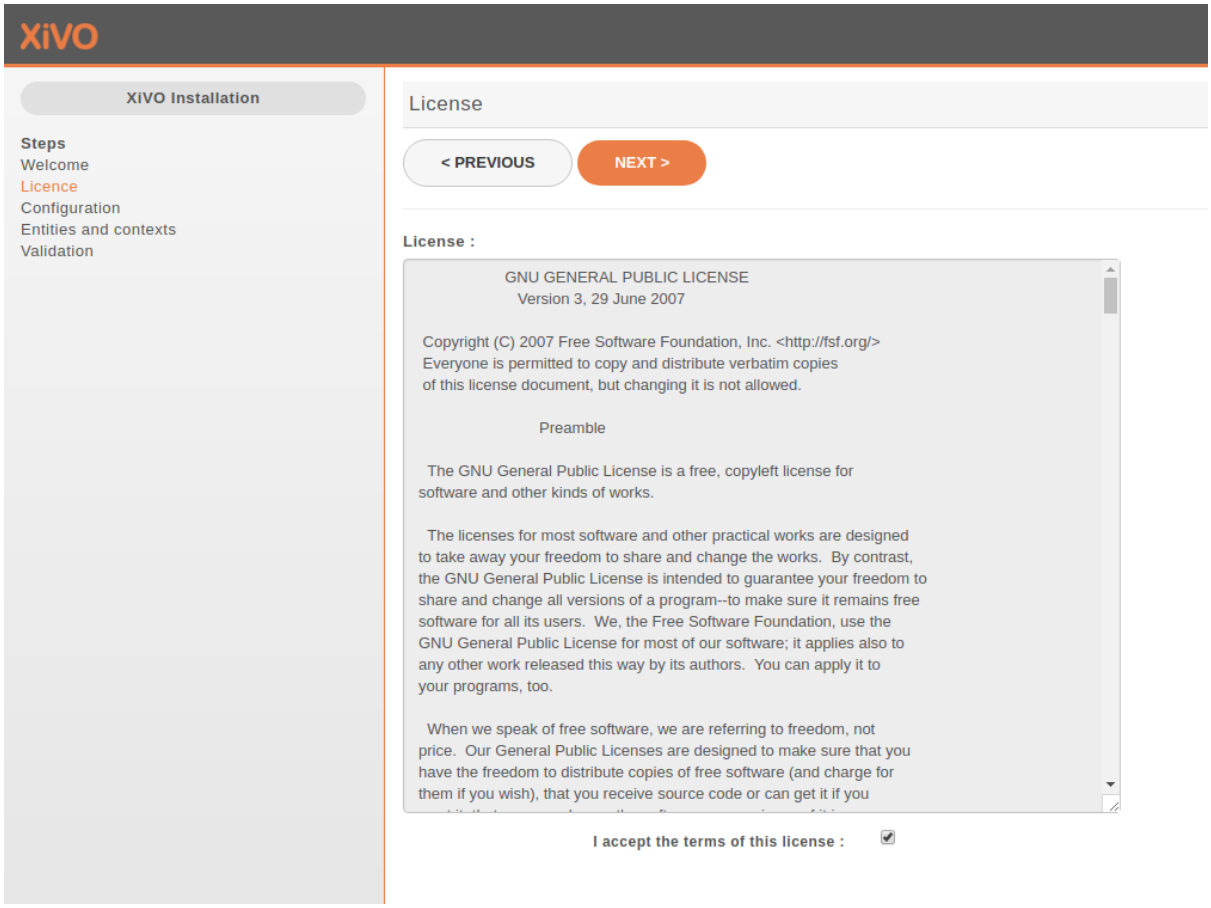


Fig. 2: Accept the license

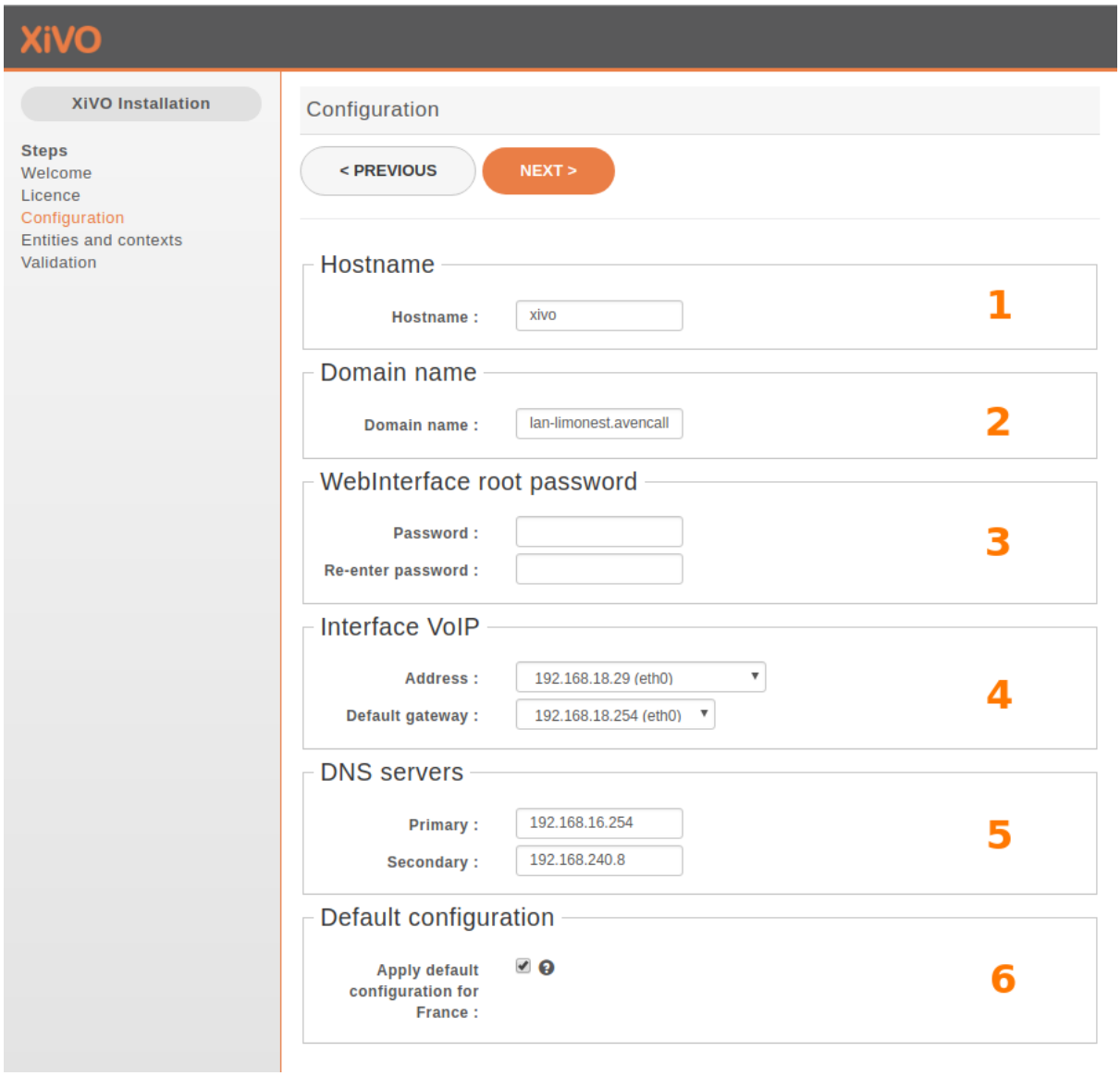


Fig. 3: Basic configuration

Entities and Contexts

Contexts are used for managing various phone numbers that are used by your system.

- The Internal calls context manages extension numbers that can be reached internally
- The Incalls context manages calls coming from outside of your system
- The Outcalls context manages calls going from your system to the outside

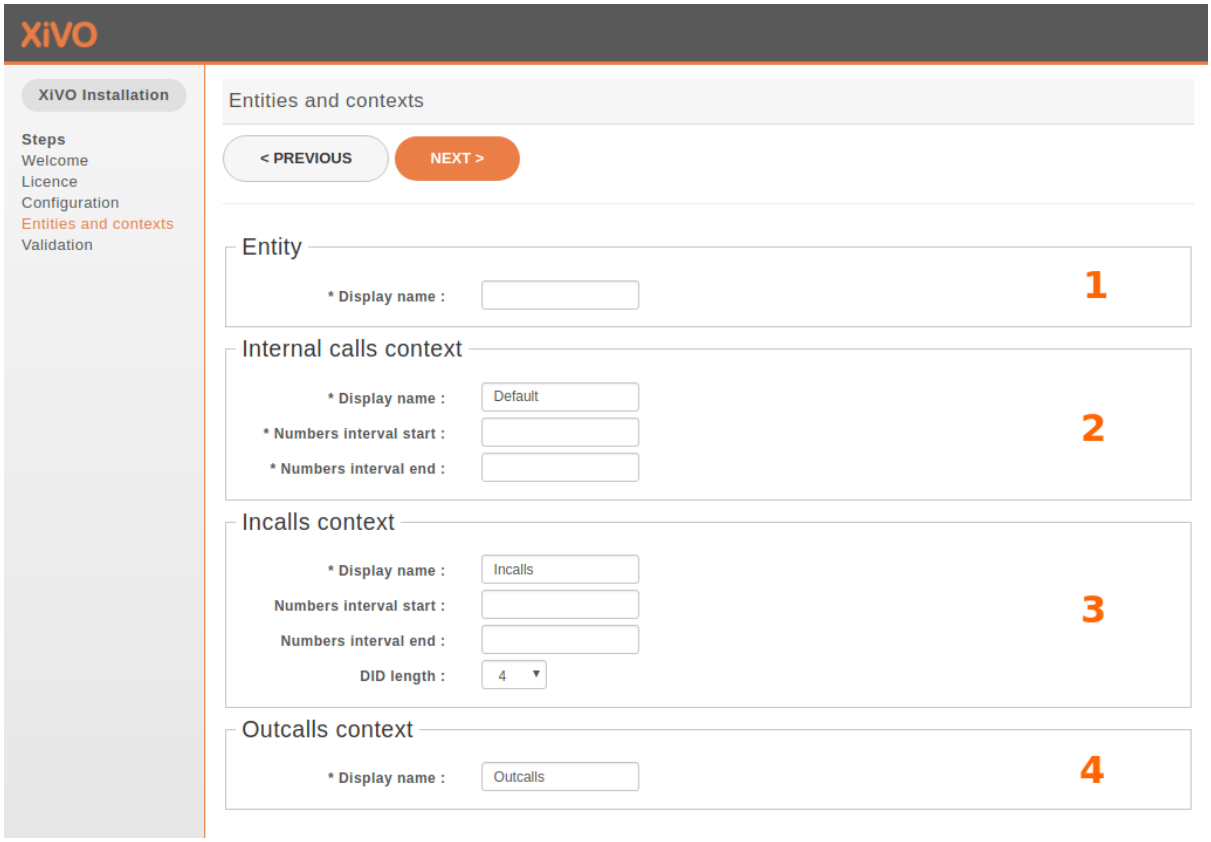


Fig. 4: Entities and Contexts

1. Enter the entity name (e.g. your organization name) (Allowed characters are : `A-Z a-z 0-9 - .`)
2. Enter the number interval for you internal context. The interval will define the users's phone numbers for your system (you can change it afterwards)
3. Enter the DID range and DID length for your system.
4. You may change the name of your outgoing calls context.

Validation

Finally, you can validate your configuration by clicking on the `Validate` button. Note that if you want to change one of the settings you can go backwards in the wizard by clicking on the `Previous` button.

> **Warning:** This is the last time the `root` password will be displayed. Take care to note it.

Congratulations, you now have a fully functional XiVO server.

To start configuring XiVO, see *Getting Started*.

3.1.3 Default configuration for France

Note: This option was introduced in 2017.01 version.

During the wizard you can choose to apply a default configuration for France : see *Wizard configuration step*. This option introduce a set of default parameters that will be useful particularly for a *XiVO PBX* installed in France.

The default parameters configured are listed in the sections below.

Default SIP parameters

In *Services -> IPBX -> General settings -> SIP Protocol* the following parameters are changed:

- for call presentation (in tab *Default*):
  - *Trust the Remote-Party-ID* is set to `Yes`
  - *Send the Remote-Party-ID* is set to `PAI`
- for codecs order (in tab *Signaling*): *G.711 A-law* > *G.722* > *G.729A* > *H.264* is the default order.

Outgoing call rules

A set of default outgoing call rules according to the French numbering plan is set up by default. In *Services -> IPBX -> Call management -> Outgoing calls* two outgoing call rules are defined:

1. *sortants-france*: pattern for french numbering plan numbers,
2. *urgences-france*: pattern for french emergency numbers.

Note: For these outgoing call rules, a 'void' cutomized trunk named 'Local/template_a_changer' is defined. This one must be deleted or modified according to your configuration.

Right call rules

Also, a set of right call is predefined according to the set of outgoing call rules. In *Services -> IPBX -> Call management -> Call permissions* you will find the following preconfigured right call group:

Name[1]	Action	Description
national	Allow	Patterns for national numbers.
urgences	Allow	Patterns for emergency numbers.
mobiles	Allow	Patterns for mobile numbers.
numeros-a-valeur-ajoutee	Allow	Patterns for services numbers.
international	Allow	Patterns for international numbers.
refuser-tout	Allow	Patterns for all.

Default template device

The 'Default config device' template (in *Configuration -> Provisioning -> Template device*) has preconfigured language and time zone.

[1] this name can be used when importing users. See *Call permissions* section in *User import*.

3.1.4 Post Installation

Here are a few configuration options that are commonly changed once the installation is completed. Please note that these changes are optional.

Display called name on internal calls

Note: Configured by default if you checked the *Apply default onfiguration for France* at the wizard time (see *Wizard configuration step*).

When you call internally another phone of the system you would like your phone to display the name of the called person (instead of the dialed number only). To achieve this you must change the following SIP options:

- *Services* → *IPBX* → *General settings* → *SIP Protocol* → *Default*:
  - Trust the Remote-Party-ID: yes,
  - Send the Remote-Party-ID: select `PAI`

Incoming caller number display

The caller ID number on incoming calls depends on what is sent by your operator. You can modify it via the file `/etc/xivo/asterisk/xivo_in_callerid.conf`.

Note: The reverse directory lookup use the caller ID number after it has been modified by `xivo_in_callerid.conf`

Examples:

- If you use a prefix to dial outgoing numbers (like a 0) you should add a 0 to all the `add =` sections,
- You may want to display incoming numbers in E.164 format. For example, you can change the `[national1]` section to:

```
callerid = ^0[1-9]\d{8}$
strip = 1
add = +33
```

To enable the changes you have to restart xivo-agid:

```
service xivo-agid restart
```

Time and date

- Configure your locale and default time zone device template => *Configuration* → *Provisioning* → *Template Device* by editing the default template
- Configure the timezone in => *Services* → *IPBX* → *General settings* → *Advanced* → *Timezone*
- If needed, reconfigure your timezone for the system:

```
dpkg-reconfigure tzdata
```

Codecs

Note: Configured by default if you checked the *Apply default onfiguration for France* at the wizard time (see *Wizard configuration step*).

You should also select default codecs. It obviously depends on the telco links, the country, the phones, the usage, etc. Here is a typical example for Europe (the main goal in this example is to select *only* G.711 A-Law instead of both G.711 A-Law and G.711 µ-Law by default):

- SIP : *Services* → *IPBX* → *General settings* → *SIP Protocol* → *Signaling*:
  - Customize codec : enabled
  - Codec list:

```
G.711 A-Law
G.722
G.729A
H.264
```

3.1.5 Telephony Hardware

This section describes how to configure the telephony hardware on a XiVO server.

Note: Currently XiVO supports only Digium Telephony Interface cards

The configuration process is the following :

Load the correct DAHDI modules

For your Digium card to work properly you must load the appropriate DAHDI kernel module. This is done via the file `/etc/dahdi/modules` and this page will guide you through its configuration.

Know which card is in your server

You can see which cards are detected by issuing the `dahdi_hardware` command:

```
dahdi_hardware
pci:0000:05:0d.0     wcb4xxp-     d161:b410 Digium Wildcard B410P
pci:0000:05:0e.0     wct4xxp-     d161:0205 Wildcard TE205P (4th Gen)
```

This command gives the card name detected and, more importantly, the DAHDI kernel module needed for this card. In the above example you can see that two cards are detected in the system:

- a Digium B410P *which needs* the `wcb4xxp` module
- and a Digium TE205P *which needs* the `wct4xxp` module

Create the configuration file

Now that we know the modules we need, we can create our configuration file:

1. Create the file `/etc/dahdi/modules`:

```
touch /etc/dahdi/modules
```

2. Fill it with the modules name you found with the `dahdi_hardware` command (one module name per line). In our example, your `/etc/dahdi/modules` file should contain the following lines:

```
wcb4xxp
wct4xxp
```

Note: In the `/usr/share/dahdi/modules.sample` file you can find all the modules supported in your XiVO version.

Apply the configuration

To apply the configuration, restart the services:

```
xivo-service restart
```

Next step

Now that you have loaded the correct module for your card you must:

1. check if you need to follow one of the *Specific configuration* sections below,
2. and continue with the next configuration step which is to *configure the echo canceller*.

Specific configuration

This section lists some specific configuration. You should not follow them unless you have a specific need.

TE13x, TE23x, TE43x: E1/T1 selection

With E1/T1 cards you must select the correct *line mode* between:

- E1 : the European standard,
- and T1 : North American standard

For old generation cards (TE12x, TE20x, TE40x series) the *line mode* is selected via a physical jumper.

For new generation cards like TE13x, TE23x, TE43x series the *line mode* is selected by configuration.

If you're configuring one of these **TE13x, T23x, T43x** cards then you **MUST** create a configuration file to set the line mode to E1:

1. Create the file `/etc/modprobe.d/xivo-wcte-linemode.conf`:

```
touch /etc/modprobe.d/xivo-wcte-linemode.conf
```

2. Fill it with the following lines replacing `DAHDI_MODULE_NAME` by the correct module name (`wcte13xp`, `wcte43x` ...):

```
# set the card in E1/T1 mode
options DAHDI_MODULE_NAME default_linemode=e1
```

3. Then, restart the services:

```
xivo-service restart
```

Hardware Echo-cancellation

It is *recommended* to use telephony cards with an hardware echo-canceller module.

Warning: with **TE13x, TE23x and TE43x** cards, you **MUST** install the echo-canceller firmware. Otherwise the card won't work properly.

Know which firmware you need

If you have an hardware echo-canceller module you **have to** install its firmware.

You first need to know which firmware you have to install. The simplest way is to restart dahdi and then to lookup in the dmesg which firmware does DAHDI request at startup:

```
xivo-service restart
dmesg |grep firmware
[5461540.738209] wct4xxp 0000:01:0e.0: firmware: agent aborted loading dahdi-fw-
↪oct6114-064.bin (not found?)
[5461540.738310] wct4xxp 0000:01:0e.0: VPM450: firmware dahdi-fw-oct6114-064.bin␣
↪not available from userspace
```

In the example above you can see that the module `wct4xxp` requested the `dahdi-fw-oct6114-064.bin` firmware file but did not found it. But you now know that you need the `dahdi-fw-oct6114-064.bin` firmware.

Install the firmware

When you know which firmware you need you can install it with `xivo-fetchfw` utility.

1. Use `xivo-fetchfw` to find the name of the package. You can search for `digium` occurrences in the available packages:

   ```
   xivo-fetchfw search digium
   ```

2. Find the package name which matches the firmware file you need. In our example, we need the `dahdi-fw-oct6114-064.bin` file which is supplied by the package named `digium-oct6114-064`:

   ```
   xivo-fetchfw install digium-oct6114-064
   ```

Activate the Hardware Echo-cancellation

Now that you installed hardware echo-canceller firmware you must activate it in `/etc/asterisk/chan_dahdi.conf` file:

```
echocancel = 1
```

Apply the configuration

To apply the configuration, restart the services:

```
xivo-service restart
```

Next step

Now that you have loaded the correct module for your card you must:

1. check if you need to follow one of the *Specific configuration* sections below,
2. and continue with the next configuration step which is to *configure your card* according to the operator links.

Specific configuration

This section describes some specific configuration. You should not follow them unless you have a specific need.

Use the Hardware Echo-canceller for DTMF detection

If you have an hardware echo-canceller you *may* want to use it to detect the DTMF signal (instead of asterisk).

1. Create the file `/etc/modprobe.d/xivo-hwec-dtmf.conf`:

```
touch /etc/modprobe.d/xivo-hwec-dtmf.conf
```

2. Fill it with the following lines replacing `DAHDI_MODULE_NAME` by the correct module name (`wcte13xp`, `wct4xxp` ...):

```
options DAHDI_MODULE_NAME vpmdtmfsupport=1
```

3. Then, restart the services:

```
xivo-service restart
```

Card configuration

Now that you have *loaded the correct DAHDI modules* and *configured the echo canceller* you can proceed with the card configuration. Follow one of the appropriate link below :

BRI card configuration

Verifications

Verify that the `wcb4xxp` module is uncommented in `/etc/dahdi/modules`.

If it wasn't, do again the step *Load the correct DAHDI modules*.

Generate DAHDI configuration

Issue the command:

```
dahdi_genconf
```

Warning: it will erase all existing configuration in `/etc/dahdi/system.conf` and `/etc/asterisk/dahdi-channels.conf` files !

Configure

DAHDI system.conf configuration

First step is to check `/etc/dahdi/system.conf` file:

- check the span numbering,
- if needed change the clock source,

See detailed explanations of this file in the *`/etc/dahdi/system.conf`* section.

Below is **an example** for a typical french BRI line span:

```
# Span 1: B4/0/1 "B4XXP (PCI) Card 0 Span 1" (MASTER) RED
span=1,1,0,ccs,ami
# termtype: te
bchan=1-2
hardhdlc=3
echocanceller=mg2,1-2
```

Asterisk dahdi-channels.conf configuration

Then you have to modify the `/etc/asterisk/dahdi-channels.conf` file:

- remove the unused lines like:

  ```
  context = default
  group = 63
  ```

- change the `context` lines if needed,
- the `signalling` should be one of:
  - `bri_net`
  - `bri_cpe`
  - `bri_net_ptmp`
  - `bri_cpe_ptmp`

See some explanations of this file in the *`/etc/asterisk/dahdi-channels.conf`* section.

Below is **an example** for a typical french BRI line span:

```
; Span 1: B4/0/1 "B4XXP (PCI) Card 0 Span 1" (MASTER) RED
group = 0,11              ; belongs to group 0 and 11
context = from-extern     ; incoming call to this span will be sent in 'from-extern
↪' context
switchtype = euroisdn
signalling = bri_cpe    ; use 'bri_cpe' signalling
channel => 1-2          ; the above configuration applies to channels 1 and 2
```

Next step

Now that you have configured your BRI card:

1. you must check if you need to follow one of the *Specific configuration* sections below,
2. then, if you have another type of card to configure, you can go back to the *configure your card* section,
3. if you have configured all your card you have to configure the *DAHDI interconnections* in the web interface.

Specific configuration

You will find below 3 configurations that we recommend for BRI lines. These configurations were tested on different type of french BRI lines with success.

Note: The pre-requisites are:

- XiVO >= 14.12,
- Use per-port dahdi interconnection (see the *DAHDI interconnections* section)

If you don't know which one to configure we recommend that you try each one after the other in this order:

1. *PTMP without layer1/layer2 persistence*
2. *PTMP with layer1/layer2 persistence*
3. *PTP with layer1/layer2 persistence*

PTMP without layer1/layer2 persistence

In this mode we will configure asterisk and DAHDI:

- to use Point-to-Multipoint (PTMP) signalling,
- and to leave Layer1 and Layer2 DOWN

Follow theses steps to configure:

1. **Before** the line `#include dahdi-channels.conf` add, in file `/etc/asterisk/chan_dahdi.conf`, the following lines:

```
layer1_presence = ignore
layer2_persistence = leave_down
```

2. In the file `/etc/asterisk/dahdi-channels.conf` use `bri_cpe_ptmp` signalling:

```
signalling = bri_cpe_ptmp
```

3. Create the file `/etc/modprobe.d/xivo-wcb4xxp.conf` to deactivate the layer1 persistence:

```
touch /etc/modprobe.d/xivo-wcb4xxp.conf
```

4. Fill it with the following content:

```
options wcb4xxp persistentlayer1=0
```

5. Then, apply the configuration by restarting the services:

```
xivo-service restart
```

Note: Expected behavior:

- The *dahdi show status* command should show the BRI spans in *RED* status if there is no call,
- For outgoing calls the layer1/layer2 should be brought back up by the XiVO (i.e. asterisk/chan_dahdi),
- For incoming calls the layer1/layer2 should be brought back up by the operator,
- You can consider that there is *a problem* only if incoming or outgoing calls are rejected.

PTMP with layer1/layer2 persistence

In this mode we will configure asterisk and DAHDI:

- to use Point-to-Multipoint (PTMP) signalling,
- and to keep Layer1 and Layer2 UP

Follow theses steps to configure:

1. **Before** the line `#include dahdi-channels.conf` add, in file `/etc/asterisk/chan_dahdi.conf`, the following lines:

```
layer1_presence = required
layer2_persistence = keep_up
```

2. In the file `/etc/asterisk/dahdi-channels.conf` use `bri_cpe_ptmp` signalling:

```
signalling = bri_cpe_ptmp
```

3. If it exists, delete the file `/etc/modprobe.d/xivo-wcb4xxp.conf`:

```
rm /etc/modprobe.d/xivo-wcb4xxp.conf
```

4. Then, apply the configuration by restarting the services:

```
xivo-service restart
```

Note: Expected behavior:

- The *dahdi show status* command should show the BRI spans in **OK** status even if there is no call,
- In asterisk CLI you may see the spans going Up/Down/Up : it is *a problem* only if incoming or outgoing calls are rejected.

PTP with layer1/layer2 persistence

In this mode we will configure asterisk and DAHDI:

- to use Point-to-Point (PTP) signalling,
- and use default behavior for Layer1 and Layer2.

Follow theses steps to configure:

1. In file `/etc/asterisk/chan_dahdi.conf` remove all occurrences of `layer1_presence` and `layer2_persistence` options.
2. In the file `/etc/asterisk/dahdi-channels.conf` use `bri_cpe` signalling:

```
signalling = bri_cpe
```

3. If it exists, delete the file `/etc/modprobe.d/xivo-wcb4xxp.conf`:

```
rm /etc/modprobe.d/xivo-wcb4xxp.conf
```

4. Then, apply the configuration by restarting the services:

```
xivo-service restart
```

Note: Expected behavior:

- The *dahdi show status* command should show the BRI spans in **OK** status even if there is no call,
- In asterisk CLI you should not see the spans going Up and Down : if it happens, it is *a problem* only if incoming or outgoing calls are rejected.

PRI card configuration

Verifications

Verify that the correct module is configured in `/etc/dahdi/modules` depending on the card you installed in your server.

If it wasn't, do again the step *Load the correct DAHDI modules*

Warning: *TE13x, TE23x, TE43x* cards :

- these cards need a specific dahdi module configuration. See *TE13x, TE23x, TE43x: E1/T1 selection* paragraph,
- you **MUST** install the correct echo-canceller firmware to be able to use these cards. See *Hardware Echo-cancellation* paragraph.

Generate DAHDI configuration

Issue the command:

```
dahdi_genconf
```

Warning: it will erase all existing configuration in `/etc/dahdi/system.conf` and `/etc/asterisk/dahdi-channels.conf` files !

Configure

DAHDI system.conf configuration

First step is to check `/etc/dahdi/system.conf` file:

- check the span numbering,
- if needed change the clock source,
- usually (at least in France) you should remove the `crc4`

See detailed explanations of this file in the */etc/dahdi/system.conf* section.

Below is **an example** for a typical french PRI line span:

```
# Span 1: TE2/0/1 "T2XXP (PCI) Card 0 Span 1" CCS/HDB3/CRC4 RED
span=1,1,0,ccs,hdb3
# termtype: te
bchan=1-15,17-31
dchan=16
echocanceller=mg2,1-15,17-31
```

Asterisk dahdi-channels.conf configuration

Then you have to modify the `/etc/asterisk/dahdi-channels.conf` file:

- remove the unused lines like:

```
context = default
group = 63
```

- change the `context` lines if needed,
- the `signalling` should be one of:
  - `pri_net`
  - `pri_cpe`

Below is **an example** for a typical french PRI line span:

```
; Span 1: TE2/0/1 "T2XXP (PCI) Card 0 Span 1" CCS/HDB3/CRC4 RED
group = 0,11            ; belongs to group 0 and 11
context = from-extern   ; incoming call to this span will be sent in 'from-extern'␣
↪context
switchtype = euroisdn
signalling = pri_cpe    ; use 'pri_cpe' signalling
channel => 1-15,17-31   ; the above configuration applies to channels 1 to 15 and␣
↪17 to 31
```

Next step

Now that you have configured your PRI card:

1. you must check if you need to follow one of the *Specific configuration* sections below,
2. then, if you have another type of card to configure, you can go back to the *configure your card* section,
3. if you have configured all your card you have to configure the *DAHDI interconnections* in the web interface.

Specific configuration

Multiple PRI cards and sync cable

If you have several PRI cards in your server you should link them with a synchronization cable to share the exact same clock.

To do this, you need to:

- use the coding wheel on the Digium cards to give them an order of recognition in DAHDI/Asterisk (see Digium_telephony_cards_support),
- daisy-chain the cards with a sync cable (see Digium_telephony_cards_support),
- load the DAHDI module with the `timingcable=1` option.

Create `/etc/modprobe.d/xivo-timingcable.conf` file and insert the line:

```
options DAHDI_MODULE_NAME timingcable=1
```

Where `DAHDI_MODULE_NAME` is the DAHDI module name of your card (e.g. wct4xxp for a TE205P).

Analog card configuration

Limitations

- XiVO does not support hardware echocanceller on the TDM400 card. Users of TDM400 card willing to setup an echocanceller will have to use a software echocanceller like OSLEC.

Verifications

Verify that one of the `{wctdm,wctdm24xxp}` module is uncommented in `/etc/dahdi/modules` depending on the card you installed in your server.

If it wasn't, do again the step *Load the correct DAHDI modules*

Note: Analog cards work with card module. You must add the appropriate card module to your analog card. Either:

- an FXS module (for analog equipment - phones, ...),
- an FXO module (for analog line)

Generate DAHDI configuration

Issue the command:

```
dahdi_genconf
```

Warning: it will erase all existing configuration in `/etc/dahdi/system.conf` and `/etc/asterisk/dahdi-channels.conf` files !

Configure

DAHDI system.conf configuration

First step is to check `/etc/dahdi/system.conf` file:

- check the span numbering,

See detailed explanations of this file in the */etc/dahdi/system.conf* section.

Below is **an example** for a typical FXS analog line span:

```
# Span 2: WCTDM/4 "Wildcard TDM400P REV I Board 5"
fxoks=32
echocanceller=mg2,32
```

Asterisk dahdi-channels.conf configuration

Then you have to modify the `/etc/asterisk/dahdi-channels.conf` file:

- remove the unused lines like:

```
context = default
group = 63
```

- change the `context` and `callerid` lines if needed,
- the `signalling` should be one of:
  - `fxo_ks` for **FXS** lines -yes it is the reverse
  - `fxs_ks` for **FXO** lines - yes it is the reverse

Below is **an example** for a typical french PRI line span:

```
; Span 2: WCTDM/4 "Wildcard TDM400P REV I Board 5"
signalling=fxo_ks
callerid="Channel 32" <4032>
mailbox=4032
group=5
context=default
channel => 32
```

Next step

Now that you have configured your PRI card:

1. you must check if you need to follow one of the *Specific configuration* sections below,
2. then, if you have another type of card to configure, you can go back to the *configure your card* section,
3. if you have configured all your card you have to configure the *DAHDI interconnections* in the web interface.

Specific configuration

FXS modules

If you use **FXS** modules you should create the file `/etc/modprobe.d/xivo-tdm` and insert the line:

```
options DAHDI_MODULE_NAME fastringer=1 boostringer=1
```

Where DAHDI_MODULE_NAME is the DAHDI module name of your card (e.g. wctdm for a TDM400P).

FXO modules

If you use **FXO** modules you should create file `/etc/modprobe.d/xivo-tdm`:

```
options DAHDI_MODULE_NAME opermode=FRANCE
```

Where DAHDI_MODULE_NAME is the DAHDI module name of your card (e.g. wctdm for a TDM400P).

Voice Compression Card configuration

Verifications

Verify that the `wctc4xxp` module is uncommented in `/etc/dahdi/modules`.

If it wasn't, do again the step *Load the correct DAHDI modules*.

Configure

To configure the card you have to:

1. Install the card firmware:

```
xivo-fetchfw install digium-tc400m
```

2. Comment out the following line in `/etc/asterisk/modules.conf`:

```
noload = codec_dahdi.so
```

3. Restart asterisk:

```
service asterisk restart
```

Next step

Now that you have configured your Voice Compression card:

1. you must check if you need to follow one of the *Specific configuration* sections below,
2. then, if you have another type of card to configure, you can go back to the *configure your card* section.

Specific configuration

Select the transcoding mode

The Digium TC400 card can be used to transcode:

- 120 G.729a channels,
- 92 G.723.1 channels,
- or 92 G.729a/G.723.1 channels.

Depending on the codec you want to transcode, you can modify the `mode` parameter which can take the following value:

- mode = mixed : this the default value which activates transcoding for 92 channels in G.729a or G.723.1 (5.3 Kbit and 6.3 Kbit)
- mode = g729 : this option activates transcoding for 120 channels in G.729a
- mode = g723 : this option activates transcoding for 92 channels in G.723.1 (5.3 Kbit et 6.3 Kbit)

1. Create the file `/etc/modprobe.d/xivo-transcode.conf`:

```
touch /etc/modprobe.d/xivo-transcode.conf
```

2. And insert the following lines:

```
options wctc4xxp mode=g729
```

3. Apply the configuration by restarting the services:

```
xivo-service restart
```

4. Verify that the card is correctly seen by asterisk with the `transcoder show` CLI command - this command should show the encoders/decoders registered by the TC400 card:

```
*CLI> transcoder show
0/0 encoders/decoders of 120 channels are in use.
```

Apply configuration

If you didn't do it already, you have to restart the services to apply the configuration:

```
xivo-service restart
```

At the end of this page you will also find some general notes and DAHDI.

Notes on configuration files

/etc/dahdi/system.conf

A *span* is created for each card port. Below is an example of a standard E1 port:

```
span=1,1,0,ccs,hdb3
dchan=16
bchan=1-15,17-31
echocanceller=mg2,1-15,17-31
```

Each span has to be declared with the following information:

```
span=<spannum>,<timing>,<LBO>,<framing>,<coding>[,crc4]
```

- `spannum` : corresponds to the span number. It starts to 1 and has to be incremented by 1 at each new span. This number MUST be unique.
- `timing` : describes the how this span will be considered regarding the synchronization :
  - 0 : do not use this span as a synchronization source,
  - 1 : use this span as the primary synchronization source,
  - 2 : use this span as the secondary synchronization source etc.
- `LBO` : 0 (not used)
- `framing` : correct values are `ccs` or `cas`. For ISDN lines, `ccs` is used.
- `coding` : correct values are `hdb3` or `ami`. For example, `hdb3` is used for an E1 (PRI) link, whereas `ami` is used for T0 (french BRI) link.
- `crc4` : this is a framing option for PRI lines. For example it is rarely use in France.

Note that the `dahdi_genconf` command should usually give you the correct parameters (if you correctly set the cards jumper). All these information should be checked with your operator.

/etc/asterisk/chan_dahdi.conf

This file contains the general parameters of the DAHDI channel. It is not generated via the `dahdi_genconf` command.

/etc/asterisk/dahdi-channels.conf

This file contains the parameters of each channel. It is generated via the `dahdi_genconf` command.

Below is an example of span definition:

```
group=0,11
context=from-extern
switchtype = euroisdn
signalling = pri_cpe
channel => 1-15,17-31
```

Note that parameters are read from top to bottom in a last match fashion and are applied to the given channels when it reads a line `channel =>`.

Here the channels 1 to 15 and 17 to 31 (it is a typical E1) are set:

- in groups 0 and 11 (see *DAHDI interconnections*)
- in context `from-extern` : all calls received on these channels will be sent in the context `from-extern`
- and configured with switchtype `euroisdn` and signalling `pri_cpe`

Debug

Check IRQ misses

It's always useful to verify if there isn't any *missed IRQ* problem with the cards.

Check:

```
cat /proc/dahdi/<span number>
```

If the *IRQ misses* counter increments, it's not good:

```
cat /proc/dahdi/1
Span 1: WCTDM/0 "Wildcard TDM800P Board 1" (MASTER)
IRQ misses: 1762187
  1 WCTDM/0/0 FXOKS (In use)
  2 WCTDM/0/1 FXOKS (In use)
  3 WCTDM/0/2 FXOKS (In use)
  4 WCTDM/0/3 FXOKS (In use)
```

Digium gives some hints in their *Knowledge Base* here : http://kb.digium.com/entry/1/63/

PRI Digium cards needs 1000 interruption per seconds. If the system cannot supply them, it increment the IRQ missed counter.

As indicated in Digium *KB* you should avoid shared IRQ with other equipments (like HD or NIC interfaces).

XiVO UC

3.1.6 XiVO UC add-on

This page describes how to install *XiVO UC* on the *XiVO PBX* server and how to use it. By *XiVO UC* we mean a subset of *XiVO CC* application, namely the *Web and Desktop Assistant*.

Prerequisites

Important: Your **XiVO PBX** server **MUST** meet the following requirements:

- OS : Debian 9 (stretch), **64 bits**
- **4 GB of RAM**
- 4-core CPU

- 20 GB of free disk space
- you have a *XiVO PBX* installed in a compatible version (basically the two components XiVO and *XiVO UC* have to be in the *same* version).
- the *XiVO PBX* **is setup** (wizard must be passed) with users, queues and agents, you must be able to place and answer calls.

Warning:

- By default XiVO-UC installation will pre-empt network subnets 172.17.0.0/16 and 172.18.0.0/24. If these subnets are already used, some manual steps will be needed to be able to install XiVO-UC. These steps are not described here.
- After installing the XiVO UC the XiVO PBX Administration will be only available at https://XiVO_PBX_IP/admin

Install process overview

The installation and configuration of *XiVO UC* is handled by the `xivouc-installer` script provided with XiVO. You will be asked few questions during the process:

- the XiVO PBX IP address,
- and whether or not you want to restart *XiVO PBX* by the installer or later

The `xivouc-installer` script will install packages *xivouc* and *xivouc-nginx-cfg*.

Install XiVO UC

On *XiVO PBX*, run XiVO UC installer script:

```
xivouc-installer
```

If you choose to restart *XiVO PBX* later, please do so as soon as possible to apply the modifications made by the installer. Until then, the *XiVO UC* will not be able to connect correctly to the database.

To restart XiVO services manually, run

```
xivo-service restart all
```

After-install steps

After a successful installation, start docker containers using the installed `xivocc-dcomp` script:

```
xivocc-dcomp up -d
```

Note: Please, ensure your server date is correct before starting. If system date differs too much from correct date, you may get an authentication error preventing download of the docker images.

Upgrade

Packages *xivouc-nginx-cfg* and *xivouc* will be upgraded automatically during *XiVO PBX upgrade*.

The XiVO UC upgrade must then be completed by pulling new docker containers and starting them:

```
xivocc-dcomp pull
xivocc-dcomp up -d
```

Using XiVO UC

After this installation you have on your *XiVO PBX* the *XiVO UC*.

You can now use the **XiVO UC* applications* using the XiVO PBX IP address. For example you can access the Web Assistant at :

- https://XIVO_PBX_IP

The XiVO PBX Admin interface is now available at: https://XIVO_PBX_IP/admin

Monitoring

You can monitor and control XiVO UC components from the XiVO web interface (see *Monitoring*).

Uninstallation

Uninstallation consists of these steps:

1. purge the xivouc package: `apt-get purge xivouc`
2. install the default nginx configuration: `apt-get install xivo-nginx-cfg`
3. remove docker network: `docker network rm xivocc_default`

Warning: Do not purge the `xivouc-installer` package! It is required by XiVO.

Upgrading

3.1.7 Upgrade

Upgrading a *XiVO PBX* is done by executing commands through a terminal on the server.

Note: Downgrade is not supported

Overview

The upgrade consists of the following steps:

- switch the version via `xivo-dist` utility
- upgrade via the `xivo-upgrade` utility: it will upgrade the system (Debian packages) and the *XiVO PBX* packages

Warning: The following applies to *XiVO PBX* **>= 2016.03**. For older version, see *Version-specific upgrade procedures* section.

Preparing the upgrade

There are two cases:

1. *Upgrade to another LTS XiVO PBX version*,
2. *Upgrade to the latest Bugfix release* of your current installed LTS version.

Upgrade to another LTS version

To prepare the upgrade you should:

1. Switch the sources to the new XiVO PBX **LTS** version with `xivo-dist`, for example, to switch to Aldebaran LTS version:

```
xivo-dist xivo-aldebaran
```

2. **Read carefully the** *Release Notes* starting from your current version to the version you target (read **even more carefully** the New features and Behavior changes between LTS)
3. **Check** the specific instructions and manual steps *from your current LTS to your targetted LTS* and all intermediate LTS: see *Manual steps for LTS upgrade*
4. **Check also** if you are in a specific setup that requires a *specific procedure* to be followed (e.g. *Upgrading a cluster*).
5. And then upgrade, see *Upgrading*

Upgrade to latest Bugfix release of an LTS version

After the release of an **LTS** *version* (e.g. *Polaris*) we may backport some bugfixes in this version. We will then create a **subversion** (e.g. Polaris **.04**) shipping these bugfixes. These bugfix version does not contain any behavior change.

To upgrade to the **latest subversion** of your current installed *version* you need to:

1. **Read carefully the** *Release Notes* starting from your installed version (e.g. Polaris.00) to the latest bugfix release (e.g. Polaris.04).
2. Verify that the debian sources list corresponds to your *installed LTS* or refix it, for example for Polaris:

```
xivo-dist xivo-polaris
```

3. And then upgrade, see *Upgrading*

Upgrading

Note: About *xivo-upgrade* script usage see *xivo-upgrade script*

After having prepared your upgrade (see above), you can upgrade:

1. When ready, launch the upgrade process. **All XiVO PBX services will be stopped during the process**:

```
xivo-upgrade
```

2. Verify that the docker services were downloaded:

```
xivo-dcomp pull
```

3. Upgrade the docker services:

```
xivo-dcomp up -d --remove-orphans
```

Post Upgrade

When finished:

- Check that all services are running:

```
xivo-service status all
```

- Check that all the docker services are in the correct version. Compare the output of `xivo-dcomp version` with the table in *Release Notes*
- Check that services are correctly working like SIP registration, ISDN link status, internal/incoming/outgoing calls, XiVO Client connections etc.

Manual steps for LTS upgrade

Upgrade Five to Polaris

In this section are listed the manual steps to do when migrating from Five to Polaris.

Before Upgrade

On XiVO PBX

Follow the *Upgrade* page.

On XiVO CC

The *xivo-solutions-VERSION* no longer exists. The `xivocc-installer` package is now located in *xivo-VERSION* distribution. You have to update your source list accordingly.

1. Remove apt source list file:

```
rm /etc/apt/sources.list.d/xivo-solutions.list
```

2. Add new source list file:

```
echo "deb http://mirror.xivo.solutions/debian xivo-polaris main" > /etc/apt/
↪sources.list.d/xivo-dist.list
```

After Upgrade

On XiVO PBX

- Accept new cel.conf: if you are asked by `xivo-upgrade` installer, you must choose to replace the `cel.conf` file or ensure that its content correspond to these *defaults*.
- Finish to remove xivo-ctid-ng and xivo-websocketd:

```
apt-get purge xivo-websocketd xivo-ctid-ng
```

- Add *writetimeout* parameter to the the `/etc/asterisk/manager.d/02-xivocc.conf` file:

```
[xuc]
secret = ...
deny = ...
permit = ...
read = ...
write = ...
writetimeout = 10000
```

- You **MUST** update:
  - *Snom* phones to use plugin version **>=2.2** to be able to use CTI Transfer (UC Assistant or CCAgent),
  - *Yealink* phones to use plugin with **v81** firmware to be able to use CTI Transfer (UC Assistant or CCAgent).
- The WebRTC call limit was raised to 2 (to enable transfers). The simultcalls parameter of a WebRTC user should be set to 2 also.

On XiVO CC

- Update new fingerboard:

```
xivocc-dcomp stop fingerboard
docker rm xivocc_fingerboard_1
xivocc-dcomp up -d
```

- SpagoBI:
  - **Import new reports** as described in *SpagoBI* post install step
  - Then, you should also remove old sample reports:
    - Go to *Reports* menu and delete all reports which are located under **Racine -> Sample**
    - Once all reports are deleted inside these folder you can go to *Functionalities Management* menu as shown in following screenshot:

    - From here, you can delete empty folders by clicking on it
    - At the end you should have a report folder list, that contains only *Rapports* and *Accueil* folder as seen here (unless you have some specific customer report):

Upgrade Polaris to Aldebaran

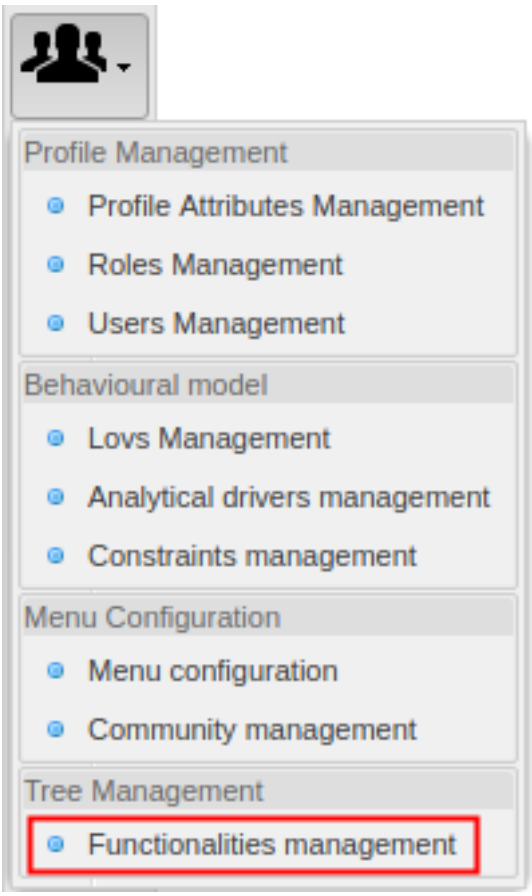
In this section are listed the manual steps to do when migrating from Polaris to Aldebaran.

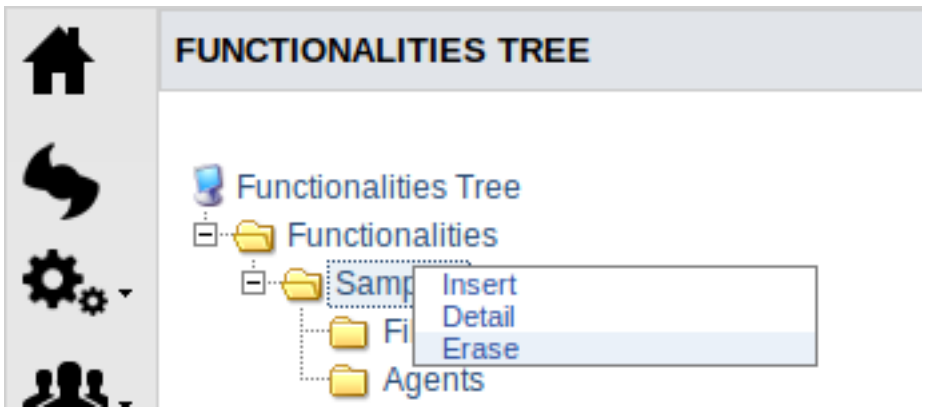
Before Upgrade

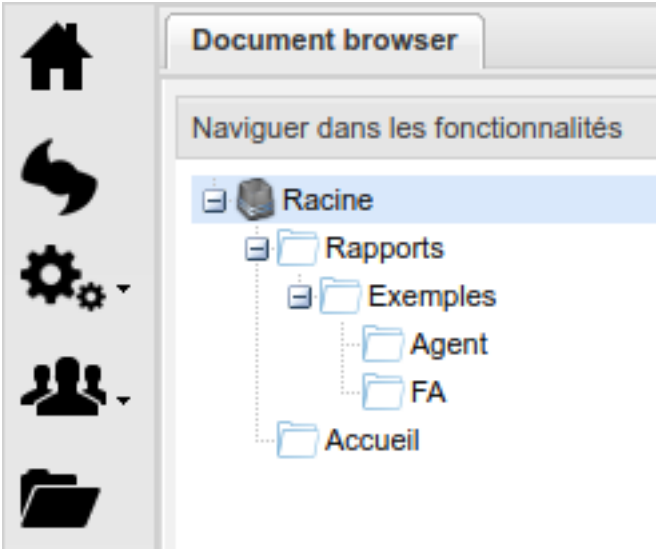
- If docker is installed (on XiVO PBX or XiVO CC), its custom start options may cause the upgrade to fail. Please check that `/etc/systemd/system/docker.service.d` directory does not exist or it is empty. Otherwise see documentation on docker web.

On XiVO PBX

- XiVO Aldebaran is only available in **64-bit** versions. To migrate XiVO from `i386` to `amd64`, you must follow *Migrate XiVO from i386 (32 bits) to amd64 (64 bits)*.


Profile Management
Profile Attributes Management
Roles Management
Users Management
Behavioural model
Lovs Management
Analytical drivers management
Constraints management
Menu Configuration
Menu configuration
Community management
Tree Management
Functionalities management


FUNCTIONALITIES TREE
Functionalities Tree
Functionalities
Insert
Detail
Erase
Agents



- Configuration files and dialplan subroutine for ACD outgoing calls for call blending (See Polaris documentation) was integrated to XiVO. The configuration **must** be manually removed if it exists:

```
sed -i '/generate_agent_skills.py/d' /etc/asterisk/queueskills.conf
rm /usr/local/sbin/generate_agent_skills.py
rm /etc/asterisk/extensions_extra.d/xuc_outcall_acd.conf
```

- If the XiVO was added to XiVO Centralized User Management, the file `routage.conf` **must** be updated.
  - Download the new routage.conf file
  - Replace the old file `/etc/asterisk/extensions_extra.d/routage.conf` by the new one
- Follow the *Upgrade* page.

On XiVO CC

Nothing specific, follow the *Upgrade* page.

After Upgrade

On XiVO PBX

- Outgoing call was removed from redirections.

  You MUST check the xivo-upgrade log with the following command:

```
zgrep MIGRATE_OUTCALL_FWD /var/log/xivo-upgrade.log*
```

  Then you MUST edit each object to reconfigure the changed destination. All Outgoing call redirections can be replaced by an Extension redirection with the appropriate context. For example an outgoing redirection towards "my_outcall (@to-extern)" to number "0123456789" can be replaced by an extension redirection to "0123456789" with context "to-extern"
- Callbacks: callback API URL was changed. It is now prefixed with *configmgt*. If you have any AGI calling the callbacks API (for example *api/1.0/callback_lists*) you MUST add the prefix *configmgt* to it : *configmgt/api/1.0/callback_lists*.
- ConfigMgt service was moved to XiVO PBX. Here's how to migrate the ConfigMgt database:

- On your **XiVO CC** (where the service pgxivocc is run) authorize user root to login with password via ssh,
- Then on your **XiVO CC**, stop the services with `xivocc-dcomp stop`
- Then, on your **XiVO PBX**, run the migration script and follow the instruction:

```
xivo-migrate-configmgt-db
```

- Finally, when migration is complete, relaunch the services on **XiVO CC**:

```
xivocc-dcomp up -d --remove-orphans
```

On XiVO CC

- Update new kibana default panels:

```
xivocc-dcomp stop kibana_volumes
docker rm -v xivocc_kibana_volumes_1
xivocc-dcomp up -d
```

Upgrade Aldebaran to Borealis

In this section is listed the manual steps to do when migrating from Aldebaran to Borealis.

Warning: For XiVO PBX Debian was upgraded to Debian 9 (stretch). Therefore:

- the upgrade to Borealis will take longer than usual
- upgrade from older version than XiVO 15.20 are not supported.

Please read carefully *Debian 9 (stetch) upgrade notes* page.

Warning: **Known Upgrade Limitations:** as of Borealis release these are the known limitations for upgrade. These should be removed during the next bugfix releases of Borealis.

- XiVO PBX / UC / CC is **not installable or upgradable on XFS partition created without** `ftype=1` **option**. If the partition is XFS, you **MUST** check if the option is enabled with the `xfs_info` command.
- Upgrade for **XiVO CC** to Debian 9 **is currently not supported**.

Before Upgrade

Note: If you have a *XiVO CC* or *XiVO UC* you **MUST** follow this upgrade order:

1. Stop *XiVO CC / UC* services,
2. Upgrade *XiVO PBX*
3. Upgrade *XiVO CC / UC*: during the upgrade it will again restart the *XiVO PBX* services (to install the `db_replic` container)

On XiVO PBX

- Debian system will be upgraded to Debian 9 (stretch)

> **Warning:** Please **read carefully** the *Before the upgrade* section in the Debian 9 (stretch) upgrade notes.

- **UC Add-on**: during upgrade, *UC* database (*pgxivocc* container) will be removed.

> **Important:** Thus **all call history** (in *UC Assistant*) **will be lost**.

The history will be computed again starting from the last call processed before the upgrade. If you want to recompute all the calls present in the *XiVO PBX* database you must follow a manual procedure which is not described here.

On XiVO CC

- Before launching upgrade, you **MUST** verify that replication (of tables `cel`, `queue_log`, `callback_request` and `qualification_answers`) was completed. You can use these SQL commands to compare *asterisk* and *xivo_stats* databases:

```
select * from cel order by eventtime desc limit 1;
select * from queue_log order by time desc limit 1;
select * from callback_request order by reference_number desc limit 1;
select * from qualification_answers order by time desc limit 1;
```

After Upgrade

On XiVO PBX

- Debian system will be upgraded to Debian 9 (stretch)

> **Warning:** Please **read carefully** the *After the upgrade* section in the Debian 9 (stretch) upgrade notes.

- Recording: recording for queues was entirely modified
  - The `xivocc-recording` package will be automatically installed or upgraded with XiVO if you **don't** have `xivo-gateway-recording` installed (to record on external server). These two packages are conflicting because they use the same dialplan subroutine names. If you are using gateway recording and your XiVO is capable to record internally, you should replace the old package by running `apt-get install xivocc-recording`. Then follow the *Recording configuration* page. These features are available only with `xivocc-recording`:
    - the *pause recording* feature
    - and the *stop recording upon external transfer* feature

    Gateway recording was also removed from documentation and it is not available in XiVO repository.
  - If you are upgrading from older version than 2017.03.02 with recording installed, you must follow the *XiVOCC Recording upgrade procedure*.
  - In the file `/etc/asterisk/extensions_extra.d/xivocc-recording.conf` the `ipbx_name` was reset to its default value. You need to set it back to its previous value.

  - You MUST edit the queues configuration and (1) remove the subroutine used to start the recording, (2) and replace them by the correct configuration in the queue (see *Recording configuration*).

- Callbacks: callback API URL was changed. It is now prefixed with *configmgt*. If you have any AGI calling the callbacks API (for example *api/1.0/callback_lists*) you MUST add the prefix *configmgt* to it : *configmgt/api/1.0/callback_lists*.
- *HTTPS certificate*: it is required to have `subjectAltName` defined in the HTTPS certificate. If you use the default HTTPS certificate, you must regenerate it. See *Regenerating the default certificate*.
- APT keyring lookup hashtable troubleshooting: this error can appear at the end of the upgrade before starting xivo services:

```
gpg: lookup_hashtable failed: Unknown system error
gpg: trustdb: searching trust record failed: Unknown system error
gpg: Error: The trustdb is corrupted.
gpg: You may try to re-create the trustdb using the commands:
gpg:   cd ~/.gnupg
gpg:   gpg --export-ownertrust > otrust.tmp
gpg:   rm trustdb.gpg
gpg:   gpg --import-ownertrust < otrust.tmp
```

  If this error appears, follow the printed procedure to recreate the lookup hashtable.

On XiVO CC

Nothing specific, follow the *Upgrade* page.

Specific procedures

Upgrading a cluster

Here are the steps for upgrading a cluster, i.e. two XiVO with *High Availability (HA)*:

1. On the master : deactivate the database replication by commenting the cron in `/etc/cron.d/xivo-ha-master`
2. On the slave, deactivate the xivo-check-master-status script cronjob by commenting the line in `/etc/cron.d/xivo-ha-slave`
3. On the slave, start the upgrade:

   ```
   xivo-slave:~$ xivo-upgrade
   ```

4. When the slave has finished, start the upgrade on the master:

   ```
   xivo-master:~$ xivo-upgrade
   ```

5. When done, launch the database replication manually:

   ```
   xivo-master:~$ xivo-master-slave-db-replication <slave ip>
   ```

6. Reactivate the cronjobs (see steps 1 and 2)

Migrate XiVO from `i386` (32 bits) to `amd64` (64 bits)

There is no fully automated method to migrate XiVO from `i386` to `amd64`.

The procedure is:

1. *Upgrade* your `i386` machine to XiVO >= 15.13

2. *Install* a XiVO `amd64` **using the same version as the upgraded XiVO i386**
3. Make a backup of your XiVO `i386` by following the *backup procedure*
4. Copy the backup tarballs to the XiVO `amd64`
5. Restore the backup by following the *restore procedure*

Before starting the services after restoring the backup on the XiVO `amd64`, you should ensure that there won't be a conflict between the two machines, e.g. two DHCP servers on the same broadcast domain, or both XiVO fighting over the same SIP trunk register. You can disable the XiVO `i386` by running:

```
xivo-service stop
```

But be aware the XiVO `i386` will be enabled again after you reboot it.

Asterisk upgrade procedure

Introduction

There are three distributions available for each release (since release 2017.03).

For XiVO archive repositories

```
deb http://mirror.xivo.solutions/archive/ xivo-VERSION-latest main
deb http://mirror.xivo.solutions/archive/ xivo-VERSION-candidate main
deb http://mirror.xivo.solutions/archive/ xivo-VERSION-oldstable main
```

Alike for XiVO production repositories

```
deb http://mirror.xivo.solutions/debian/ xivo-five main
deb http://mirror.xivo.solutions/debian/ xivo-five-candidate main
deb http://mirror.xivo.solutions/debian/ xivo-five-oldstable main
```

- The distribution **latest** contains the current stable version of *XiVO PBX* and Asterisk.
- The distribution **candidate** contains only Asterisk in higher version than the current.
- The distribution **oldstable** contains only Asterisk in the previous stable version.

Example

Repository	Distribution	Section	Content
mirror.xivo.solutions/archive	xivo-2017.03-candidate	main	latest-built-and-tested-version (e.g. asterisk-13.14.0)
mirror.xivo.solutions/archive	xivo-2017.03-latest	main	most-stable-known-version (e.g. asterisk-13.13.1)
mirror.xivo.solutions/archive	xivo-2017.03-previous	main	old-most-stable-known-version (e.g. asterisk-13.10.0)

Upgrade to asterisk candidate

Before integrating a new version of asterisk in *XiVO PBX*, we first build it and deliver it in the `xivo-VERSION-candidate` distribution of our repository.

The goal is to make it available for specific cases (e.g. urgent bugfixes) before shipping it as the default version. This page explains how to upgrade/downgrade to/from this candidate version.

Warning: This is a specific procedure. You should know what you are doing.

Upgrade to candidate version

Warning: This will upgrade your asterisk version and trigger a *restart* of asterisk. You should know what you are doing.

1. Edit the xivo sources list file `/etc/apt/sources.list.d/xivo-dist.list` and add the `xivo-VERSION-candidate` distribution (see **last line entry** below):

```
# xivo-2017.03-latest
deb http://mirror.xivo.solutions/archive/ xivo-VERSION-latest main
# deb-src http://mirror.xivo.solutions/archive/ xivo-VERSION-latest main
deb http://mirror.xivo.solutions/archive/ xivo-VERSION-candidate main
```

2. Update the packages list:

```
apt-get update
```

3. Check the proposed versions with `apt-cache policy asterisk`. For example it will give you the following result:

```
# apt-cache policy asterisk
asterisk:
  Installed: 8:13.13.1-1~xivo9+13.13.1+20170105.165406.d2ff9a5
  Candidate: 8:13.13.1-1~xivo9+13.13.1+20170105.165406.d2ff9a5
  Version table:
     8:13.14.0-1~xivo9+13.13.1+20170105.165406.d2ff9a5 0
        100 http://mirror.xivo.solutions/archive/ xivo-2017.03-candidate/main␣
↪amd64 Packages
 *** 8:13.13.1-1~xivo9+20170321.182632.9ad94c7 0
        500 http://mirror.xivo.solutions/archive/ xivo-2017.03-latest/main␣
↪amd64 Packages
```

4. Install the new version (install also `asterisk-dbg` package if applicable):

```
apt-get -t xivo-VERSION-candidate install asterisk
```

5. Restart the services (if you have a *XiVO CC*, you should restart its services too):

```
xivo-service restart
```

Note: The priority will prevent installing asterisk from `xivo-VERSION-candidate` whenever running **apt-get upgrade** or **xivo-upgrade**.

Downgrade to oldstable version

Warning: This will downgrade your asterisk version and trigger a *restart* of asterisk. You should know what you are doing.

1. Edit the xivo sources list file `/etc/apt/sources.list.d/xivo-dist.list` and add the `xivo-VERSION-oldstable` distribution (see **last line entry** below):

```
# xivo-2017.03-latest
deb http://mirror.xivo.solutions/archive/ xivo-2017.03-latest main
# deb-src http://mirror.xivo.solutions/archive/ xivo-2017.03-latest main
deb http://mirror.xivo.solutions/archive/ xivo-2017.03-oldstable main
```

2. Update the sources list:

```
apt-get update
```

3. Check the proposed versions with `apt-cache policy asterisk`. For example it will give you the following result:

```
# apt-cache policy asterisk
asterisk:
  Installed: 8:13.13.1-1~xivo9+20170321.130355.9ad94c7
  Candidate: 8:13.13.1-1~xivo9+20170321.130355.9ad94c7
  Version table:
 *** 8:13.13.1-1~xivo9+20170321.130355.9ad94c7 1
      100 http://mirror.xivo.solutions/archive/ xivo-2017.03-latest/main amd64␣
↪Packages
     8:13.10.0-1~xivo9+2017.01+master+20170208.085934.3b143b7 0
        1 http://mirror.xivo.solutions/archive/ xivo-2017.03-oldstable/main␣
↪amd64 Packages
```

4. And downgrade the version by giving the version in the **oldstable** distribution of the repository. With the example below (do the same with `asterisk-dbg` if applicable):

```
apt-get install asterisk='8:13.10.0-1~xivo9+2017.01+master+20170208.085934.
↪3b143b7'
```

5. Restart the services (if you have a *XiVO CC*, you should restart its services too):

```
xivo-service restart
```

Return to the current version

Warning: This will return your asterisk version to the current most stable version and trigger a *restart* of asterisk. You should know what you are doing.

1. Change the source lists with `xivo-VERSION-latest`:

```
xivo-dist xivo-2017.03-latest
```

2. Check the proposed versions with `apt-cache policy asterisk`. For example it will give you the following result:

```
# apt-cache policy asterisk
asterisk:
  Installed: 8:13.10.0-1~xivo9+2017.02+master+20170301.144142.3b143b7
  Candidate: 8:13.13.1-1~xivo9+20170321.182632.9ad94c7
  Version table:
     8:13.13.1-1~xivo9+20170321.182632.9ad94c7 0
        500 http://mirror.xivo.solutions/archive/ xivo-2017.03-latest/main␣
↪amd64 Packages
 *** 8:13.10.0-1~xivo9+2017.02+master+20170301.144142.3b143b7 0
        100 /var/lib/dpkg/status
```

3. And install the version. With the example below (do the same with `asterisk-dbg` if applicable):

```
apt-get install asterisk
```

4. Restart the services (if you have a *XiVO CC*, you should restart its services too):

```
xivo-service restart
```

XiVOCC Recording upgrade procedure

Note: Since 2017.03.02, `xivo-recording` and `call-recording-filtering` packages are deprecated and are replaced by package `xivocc-recording`. This page describe the upgrade procedure (for *feature description, see here*).

`xivo-recording` and `call-recording-filtering` packages are deprecated and they were **uninstalled, but not purged** from your **XiVO PBX** during the upgrade.

You now have to follow this manual procedure:

Note: This has to be done on **XiVO PBX**

---

1. Configure `xivocc-recording` package (when ask, **take care** to enter same IP for Recording server and same *XiVO PBX* name as in previous configuration):

```
xivocc-recording-config
```

2. **Update all the different locations** where the old `xivo-incall-recording` or `xivo-outcall-recording` subroutines were called.

   For queues, you must remove the subroutines and activate recording by checkbox, see *Enable recording in the Queue configuration*.

   For other objects, **change them** to call the new `xivocc-incall-recording` or `xivocc-outcall-recording` subroutines:

   1. in file `/etc/xivo/asterisk/xivo_globals.conf`
   2. in Custom dialplan
   3. per object

   See *Enable recording via subroutines* for details.

3. If you made specific recording subroutines you should also compare files `/etc/asterisk/extensions_extra.d/xivo-recording.conf` and `/etc/asterisk/extensions_extra.d/xivocc-recording.conf` and transfer all custom changes to the new `xivocc-recording.conf`.

4. If there are some audio files in the failed directory of previous installation you should move them:

```
mv /var/spool/xivo-recording/failed/*.wav /var/spool/xivocc-recording/failed/
```

5. When you're done, test that recording still works. Test that files are recorded, correctly sent to Recording server (see `/var/log/xivocc-recording/replication.log`), correctly displayed in the Recording server interface,

6. If it works correctly, you should **purge** deprecated packages with (**take care**, it will remove the package and all associated configuration files):

```
apt-get purge xivo-recording call-recording-filtering
```

Debian 9 (stretch) Upgrade Notes

Debian was upgraded to Debian 9 (stretch) in XiVO 2018.14 release.

Warning: Upgrade from versions XiVO 15.19 or earlier are not supported. You MUST first upgrade to at least XiVO 15.20 or, more recommended, to XiVO Five before upgrading to XiVO Borealis.

Warning: Docker storage driver changed from `aufs` to `overlay2`. To not suffer data loss you must follow a manual procedure which is not documented yet. Therefore:

- XiVO PBX / UC / CC is **not installable or upgradable on XFS partition created without** `ftype=1` **option**. If the partition is XFS, you **MUST** check if the option is enabled with the `xfs_info` command.
- Upgrade for **XiVO CC** to Debian 9 **is currently not supported**.

Before the upgrade

- It is not possible to upgrade from XiVO 15.19 or earlier. You first need to upgrade to XiVO Five.
- **Make sure you have sufficient space for the upgrade**. You should have more than 2GiB available in the filesystem that holds the `/var` and `/` directories.
- Note that the upgrade will take longer than usual because of all the system upgrade.
- You **MUST** deactivate all non-xivo apt sources list:
  - in directory */etc/apt/sources.list.d/* you should only have the files `xivo-dist.list` and (from Aldebaran) `docker.list`.
  - you **MUST** suffix all other files with *.save* to deactivate them.
- You **MUST** check the Debian sources list are correct: the file `/etc/apt/sources.list` must contain the following and only the following:

```
deb http://ftp.fr.debian.org/debian/ jessie main
deb-src http://ftp.fr.debian.org/debian/ jessie main

deb http://security.debian.org/ jessie/updates main
deb-src http://security.debian.org/ jessie/updates main

# jessie-updates, previously known as 'volatile'
deb http://ftp.fr.debian.org/debian/ jessie-updates main
deb-src http://ftp.fr.debian.org/debian/ jessie-updates main
```

- You may want to clean your system before upgrading:
  - Remove package that were automatically installed and are not needed anymore:

    ```
    apt-get autoremove --purge
    ```

  - Purge removed packages. You can see the list of packages in this state by running `dpkg -l | awk '/^rc/ { print $2 }'` and purge all of them with:

    ```
    apt-get purge $(dpkg -l | awk '/^rc/ { print $2 }')
    ```

  - Remove `.dpkg-old`, `.dpkg-dist` and `.dpkg-new` files from previous upgrade. You can see a list of these files by running:

    ```
    find /etc -name '*.dpkg-old' -o -name '*.dpkg-dist' -o -name '*.dpkg-new'
    ```

After the upgrade

- After having check your network configuration and the grub configuration, you **MUST** *reboot your system*. It is necessary for the upgrade to the Linux kernel to be effective.
- Check that customization to your configuration files is still effective.

  During the upgrade, new version of configuration files are going to be installed, and these might override your local customization. For example, the vim package provides a new `/etc/vim/vimrc` file. If you have customized this file, after the upgrade you'll have both a `/etc/vim/vimrc` and `/etc/vim/vimrc.dpkg-old` file, the former containing the new version of the file shipped by the vim package while the later is your customized version. You should merge back your customization into the new file, then delete the `.dpkg-old` file.

  You can see a list of affected files by running `find /etc -name '*.dpkg-old'`. If some files shows up that you didn't modify by yourself, you can ignore them.
- Purge removed packages. You can see the list of packages in this state by running `dpkg -l | awk '/^rc/ { print $2 }'` and purge all of them with:

```
apt-get purge $(dpkg -l | awk '/^rc/ { print $2 }')
```

Changes

Here's a non-exhaustive list of changes that comes with XiVO on Debian 9:

- Network utility: the tools from the `net-tools` package are no longer part of new installations by default. They are replaced by the `iproute2` toolset. For a complete summary of the net-tools commands with their iproute2 equivalent see the Official Debian 9 Release notes related chapter. Here are 4 examples:
  - Instead of arp, use `ip n` (short for `ip neighbor`)
  - Instead of ifconfig, use `ip a` (short for `ip addr`)
  - Instead of netstat, use `ss`
  - Instead of route, use `ip r` (short for `ip route`)
- Docker recommended storage driver is `overlay2` and the usage of `aufs` storage driver was deprecated in Debian 9 (stretch). Thus we recommend to use `overlay2` as storage driver and our upgrade procedure will try to stick to this choice if possible. (Note that, via a manual procedure and installation of extra packages, it is still possible to use `aufs` storage driver but we do not recommend it).

External Links

Official Debian 9 Release Notes

Version-specific upgrade procedures

Note: If your *XiVO PBX* is **below 2016.03** you have first to *Switch to xivo.solutions* mirrors.

Switch to xivo.solutions

Note: If you are in *XiVO PBX* **below 2016.03** you should first switch to xivo.solutions mirrors.

In order to do that follow the following procedure:

1. Download the `switch-to-xivo-solutions` script:

```
wget http://mirror.xivo.solutions/debian/tools/migration-tools/switch-to-xivo-
↪solutions.sh
chmod +x ./switch-to-xivo-solutions.sh
```

2. Execute the script:

```
./switch-to-xivo-solutions.sh

...

Your XiVO has been switched to xivo.solutions successfully.
Votre XiVO a été basculé vers xivo.solutions avec succès.
```

3. Update the sources list:

```
apt-get update
```

Other Version Specific Procedures

Several version specific procedure are listed here.

Upgrading from XiVO 14.11 and before

When upgrading from XiVO 14.11 or earlier, you must do the following, before the normal upgrade:

```
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
```

Upgrading from XiVO 14.01, 14.02, 14.03, 14.04 installed from the ISO

In those versions, xivo-upgrade keeps XiVO on the same version. You must do the following, before the normal upgrade:

```
echo "deb http://mirror.xivo.solutions/debian/ xivo-five main" > /etc/apt/sources.
↪list.d/xivo-upgrade.list \
&& apt-get update \
&& apt-get install xivo-fai \
&& rm /etc/apt/sources.list.d/xivo-upgrade.list \
&& apt-get update
```

Upgrading from XiVO 13.24 and before

When upgrading from XiVO 13.24 or earlier, you must do the following, before the normal upgrade:

1. Ensure that the file `/etc/apt/sources.list` is *not* configured on `archive.debian.org`. Instead, it must be configured with a non-archive mirror, but still on the `squeeze` distribution, even if it is not present on this mirror. For example:

```
deb http://ftp.us.debian.org/debian squeeze main
```

2. Add `archive.debian.org` in another file:

```
cat > /etc/apt/sources.list.d/squeeze-archive.list <<EOF
deb http://archive.debian.org/debian/ squeeze main
EOF
```

And after the upgrade:

```
rm /etc/apt/sources.list.d/squeeze-archive.list
```

Upgrading from XiVO 13.03 and before

When upgrading from XiVO 13.03 or earlier, you must do the following, before the normal upgrade:

```
wget http://mirror.xivo.solutions/xivo_current.key -O - | apt-key add -
```

Upgrading from XiVO 12.13 and before

When upgrading from XiVO 12.13 or earlier, you must do the following, before the normal upgrade:

```
apt-get update
apt-get install debian-archive-keyring
```

Upgrading from XiVO 1.2.1 and before

Upgrading from 1.2.0 or 1.2.1 requires a special procedure before executing `xivo-upgrade`:

```
apt-get update
apt-get install xivo-upgrade
/usr/bin/xivo-upgrade
```

Upgrading to/from an archive version

Upgrade involving archive version of XiVO

Introduction

What is an archive version?

An archive version refers to a XiVO installation whose version is frozen: you can't upgrade it until you manually change the upgrade server.

What is the point?

Using archive versions enable you to upgrade your XiVO to a specific version, in case you don't want to upgrade to the latest (which is not recommended, but sometimes necessary). You will then be able to upgrade your newer archive version to the latest version or to an even newer archive version.

Prerequisites

Warning: These procedures are *complementary* to the upgrade procedure listed in *Version-specific upgrade procedures*. You must follow the version-specific procedure *before* running the following procedures.

Archive package names

Archive packages are named as follow:

XiVO version	Archive package name
1.2 to 1.2.12	pf-fai-xivo-1.2-skaro-1.2.1
12.14 to 13.24	xivo-fai-skaro-13.04
13.25 to 14.17	xivo-fai-14.06
14.18+	*packages removed*

Upgrade from an archive to the latest version

Archive version < 13.25:

```
apt-get update
apt-get install {xivo-fai,xivo-fai-skaro}/squeeze-xivo-skaro-$(cat /usr/share/pf-
↪xivo/XIVO-VERSION)
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
xivo-upgrade
```

Archive version >= 13.25 and < 14.18:

```
apt-get update
apt-get install xivo-fai
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
xivo-upgrade
```

Archive version >= 14.18:

```
xivo-dist xivo-five
xivo-upgrade
```

As a result, xivo-upgrade will upgrade XiVO to the latest stable version.

Upgrade from an older non-archive version to a newer archive version

Non-archive version means any "normal" way of installing XiVO (ISO install, script install over pre-installed Debian, xivo-upgrade).

Downgrades are not supported: you can only upgrade to a greater version.

We only support upgrades to archive versions >= 13.25, e.g. you can upgrade a 12.16 to 14.16, but not 12.16 to 13.16

Current version before 14.18 (here 13.25)

```
apt-get install xivo-fai-13.25
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
```

You are now considered in an archived version, see the section *Upgrade from an older archive version to a newer archive version* below.

Current version after 14.18

```
xivo-dist xivo-15.12
apt-get update
apt-get install xivo-upgrade/xivo-15.12
xivo-upgrade
```

Upgrade from an older archive version to a newer archive version

Downgrades are not supported: you can only upgrade to a greater version.

We only support upgrades to archive versions >= 13.25, e.g. you can upgrade a 12.16 to 14.16, but not 12.16 to 13.16

1.2 - 13.24 to 13.25 - 14.17 (here 1.2.3 to 14.16)

```
cat > /etc/apt/sources.list.d/squeeze-archive.list <<EOF
deb http://archive.debian.org/debian/ squeeze main
EOF

apt-get update
apt-get install {xivo-fai,xivo-fai-skaro}/squeeze-xivo-skaro-1.2.3
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
apt-get update
apt-get install xivo-fai-14.16
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
apt-get update
apt-get install xivo-upgrade/xivo-14.16

cat > /etc/apt/preferences.d/50-xivo-14.16.pref <<EOF
Package: *
Pin: release a=xivo-five
Pin-Priority: -10

Package: *
Pin: release a=xivo-14.16
Pin-Priority: 700
EOF

xivo-upgrade
rm /etc/apt/preferences.d/50-xivo-14.16.pref
rm /etc/apt/sources.list.d/squeeze-archive.list
apt-get update
```

13.25 - 14.16 to 13.25 - 14.17 (here 13.25 to 14.16)

```
apt-get update
apt-get install xivo-fai
apt-get purge xivo-fai-13.25
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
apt-get update
apt-get install xivo-fai-14.16
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
apt-get update
apt-get install xivo-upgrade/xivo-14.16
```

(continues on next page)

(continued from previous page)

```
cat > /etc/apt/preferences.d/50-xivo-five.pref <<EOF
Package: *
Pin: release a=xivo-five
Pin-Priority: -10
EOF

xivo-upgrade
rm /etc/apt/preferences.d/50-xivo-five.pref
```

13.25 - 14.17 to 14.18+ (here 14.05 to 15.11)

```
apt-get update
apt-get install xivo-fai
sed -i 's/xivo\.fr/xivo.solutions/g' /etc/apt/sources.list.d/*.list
apt-get update
apt-get install xivo-dist
xivo-dist xivo-15.11
apt-get purge 'xivo-fai*'
apt-get update
apt-get install xivo-upgrade/xivo-15.11
xivo-upgrade
```

14.18+ to 14.19+ (here 14.18 to 15.12)

```
xivo-dist xivo-15.12
apt-get update
apt-get install xivo-upgrade/xivo-15.12
xivo-upgrade
```

Upgrade Notes

See *Release Notes* for version specific informations.

3.2 XiVOcc Installation & Upgrade

The XiVO-CC software suite is made of several independent components. Depending on your system size, they can be installed on separate virtual or physical machines. In this section, we will explain how to install these components on a single machine.

Important: Before installing XiVO CC, study carefully the *Architecture & Flows* diagram.

3.2.1 Installation

This page describes how to install the *XiVO CC*.

It describes the installation with the debian package of the whole *XiVO CC*.

Note: As a reference, the manual installation page is here *Manual configuration and installation*.

Warning:

- the wizard **MUST** be passed on the *XiVO PBX*
- *XiVO PBX* will be reconfigured during the installation and must be restarted. You may accept the automatic restart during the installation or you need to restart it manually later before starting the docker containers.
- If you configure HA on XiVO, you have to re-configure postgres to accept connection of XiVO CC - see *PostgreSQL configuration section*
- By default XiVO CC installation will pre-empt network subnet 172.17.0.0/16 If this subnet is already used, some manual steps will be needed to be able to install XiVO CC. These steps are not described here.

Overview

The following components will be installed :

- XuC : outsourced CTI server providing telephony events, statistics and commands through a WebSocket
- XuC Management : supervision web pages based on the XuC
- Pack Reporting : statistic summaries stored in a PostgreSQL database
- Totem Support : near-real time statistics based on ElasticSearch
- SpagoBI : BI suite with default statistic reports based on the Pack Reporting
- Recording Server : web server allowing to search recorded conversations
- Xuc Rights Management : permission provider used by XuC and Recording Server to manage the user rights

Prerequisites

We will assume your **XiVO CC** server meets the following requirements:

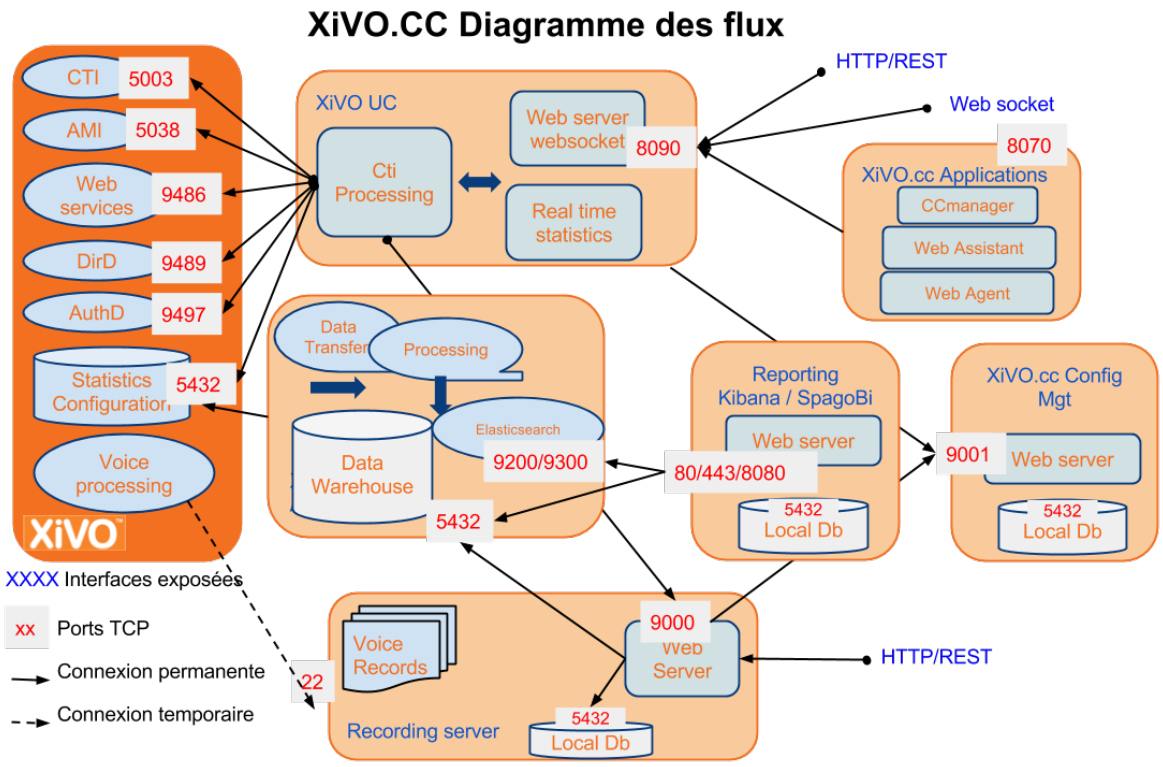
- OS : Debian 9 (stretch), 64 bits.
- you have a *XiVO PBX* installed in a compatible version (basically the two components XiVO and *XiVO CC* have to be in the *same* version).
- the *XiVO PBX* is reachable on the network (ping and ssh between *XiVO CC* and *XiVO PBX* must be possible).
- the *XiVO PBX* **is setup** (wizard must be passed) with users, queues and agents, you must be able to place and answer calls.

For the rest of this page, we will make the following assumptions :

- the *XiVO PBX* has the IP 192.168.0.1
- some data (incoming calls, internal calls etc.) might be available on XiVO (otherwise, you will not see *anything* in the *check-list* below).
- the *XiVO CC* server has the IP 192.168.0.2

Architecture & Flows

This diagram is very important and shows the architecture between the different components inside XiVO CC and also interactions with XiVO PBX components.



XiVO PBX Restrictions and Limitations

XiVO PBX enables a wide range of configuration, XiVO-CC is tested and validated with a number of restriction concerning configurations of *XiVO PBX*:

General Configuration

- Do not activate Contexts Separation in *xivo-ctid* Configuration
- Users deactivation is not supported

Queue Configuration

- Queue ringing strategy should not be *Ring All*
- Do not use pause on one queue or a subset of queues status, only pause or ready on all queues
- Do not activate Call a member already on (*Asterisk ringinuse*) on xivo queue advanced configuration

User And Agent Configuration

- All users and queues have to be in the same context
- Agent and Supervisors profiles should use the same Presence Group
- Agents and Phones should be in the same context for mobile agents
- Agents must not have a password in XiVO agent configuration page
- All users must have the supervision on the XiVO (IPBX-Users-Edit-Services-Enable supervision checked)
- When and agent is disassociated from its user, xuc server has to be restarted.

- We strongly advise to not delete any user or agent to keep reporting available for them. Even so when an agent is deleted, xuc server has to be restarted,

Install from repository

The installation and configuration of *XiVO CC* (with its *XiVO PBX* part) is handled by the *xivocc-installer* package which is available in the repository.

Install process overview

Note: If your server needs a proxy to access Internet, configure the proxy for `apt`, `wget` and `curl` as documented in *Proxy Configuration*.

The install process consists of two parts:

1. The first part is to manually run the `xivocc_install.sh` script to install the dependencies (ntp, docker, docker-compose...) and which will trigger the *XiVO CC* installation.
2. The second part is to install the extra package for the recording.

The installation is automatic and you will be asked few questions during the process:

- Before copying the authentication keys, you will be prompted for the *XiVO PBX* root password.
- Enter IP addresses of *XiVO PBX* and *XiVO CC*.
- *XiVO PBX* must restart, the question will prompt you to restart during the process or to restart later.

Launch install script

Note: To be run on the *XiVO CC* server

Once you have your Debian stretch properly installed, download the *XiVO CC* installation script and make it executable:

```
wget http://mirror.xivo.solutions/xivocc_install.sh
chmod +x xivocc_install.sh
```

Running the script will install the *XiVO CC* components via the `xivocc-installer` package. It is required to restart *XiVO PBX* during or after the setup process. The installer will ask whether you wish to restart *XiVO PBX* later.

Warning:

- Wizard **MUST** be passed on the *XiVO PBX*.
- *XiVO PBX* services will need to be restarted. The installer will ask whether you wish to restart *XiVO PBX* during or after the setup process.

Also, check that you have following information:

- *XiVO PBX* root password;
- OpenSSH `PermitRootLogin` set to `yes` (you could revert to `no` after installation of XivoCC);
- *XiVO PBX*'s IP address;

- *XiVO CC* DNS name or IP address (the one visible *by XiVO PBX*);
- Number of weeks to keep statistics;
- Number of weeks to keep recordings (beware of disk space);

The number of weeks to keep statistics **must be higher** than the number of weeks to keep recordings. Recording purging is based on the statistic data, so the statistic data must not be removed before purging recordings.

Launch installation:

```
./xivocc_install.sh -a 2018.16-latest
```

After-install steps

Configure ntp server

The *XiVO CC* server and the *XiVO PBX* server must be synchronized to the same NTP source.

Recomended configuration : you should configure the NTP server of the *XiVO CC* server towards the *XiVO PBX*. In our example it means to add the following line in the file `/etc/ntp.conf`:

```
server 192.168.0.1 iburst
```

Launch the services

Note: Please, ensure your server date is correct before starting. If system date differs too much from correct date, you may get an authentication error preventing download of the docker images.

After a successful installation, start docker containers using the installed `xivocc-dcomp` script:

```
xivocc-dcomp up -d
```

To restart XiVO services, on *XiVO PBX* server run

```
xivo-service restart all
```

Reinstallation

To reinstall the package, it is required to run `apt-get purge xivocc-installer` then `apt-get install xivocc-installer`. This will re-run the configuration of the package, download the docker compose template and setup *XiVO PBX*.

Purging the package will also **remove** the *xuc* and *stats* users from the *XiVO PBX* database.

Known Issues

To avoid uninstallation problems:

- please use the following command to uninstall `apt-get purge xivocc-installer`
- if the process is aborted, it will break the installation. Then run `apt-get purge` and `apt-get install` again

Checking Installed Version

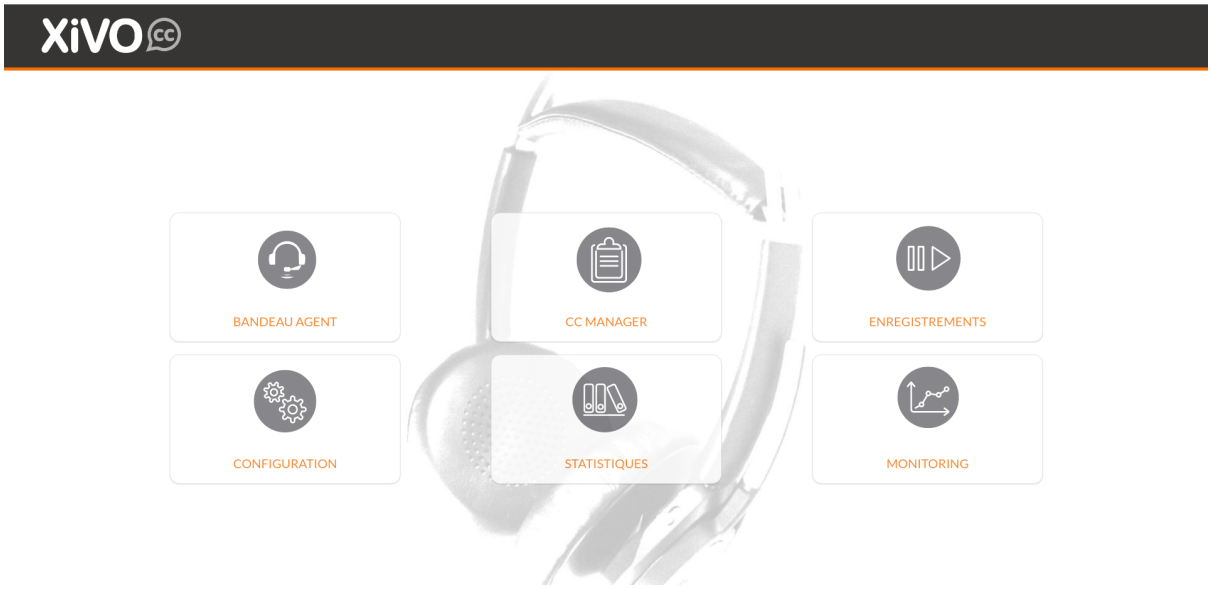
Version of the running docker containers can be displayed by typing (see *Show containers and images versions* for other commands):

```
xivocc-dcomp version
```

Component version can also be found in the log files and on the web pages for web components.

Using XivoCC

The various applications are available on the following addresses:



- Xuc-related applications: http://192.168.0.2:8070/
- SpagoBI: http://192.168.0.2:9500/
- Config Management: http://<XiVO IP Address>:9100/configmgt/
- Recording server: http://192.168.0.2:9400/
- Kibana: http://192.168.0.2/

Post Installation

User Configuration

You should configure users and their rights in the Configuration manager http://<XiVO IP Address>:9100/configmgt/ (default user avencall/superpass).

Warning: If you change the cti login username in xivo configuration, user has to be recreated with apropriate rights in configuration manager.

SpagoBI

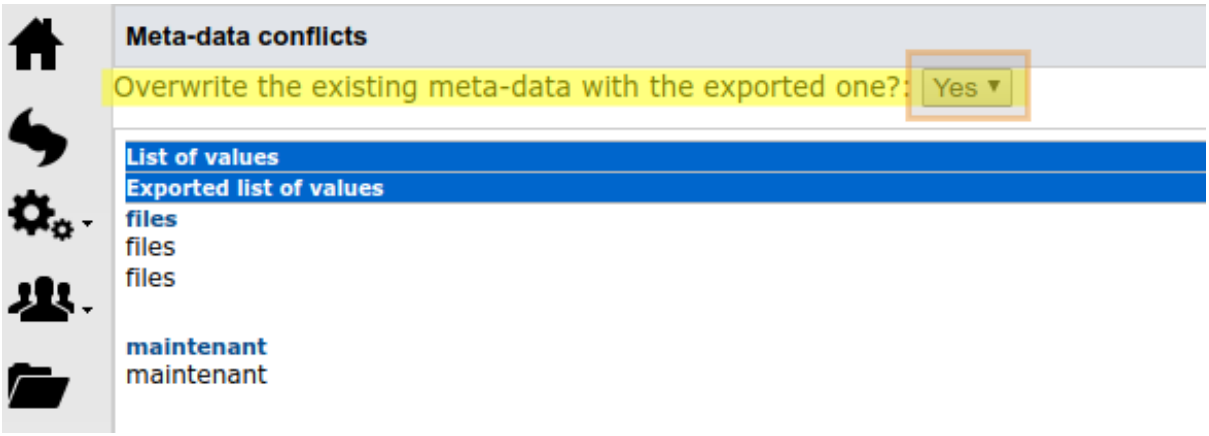
To configure SpagoBI, go to http://192.168.0.2:9500/SpagoBI (by default login: biadmin, password: biadmin).

Update default language

1. Go to " Resources" > "Configuration management"
2. In the "Select Category" field, chose "LANGUAGE_SUPPORTED"
3. change value of the label "SPAGOBI.LANGUAGE_SUPPORTED.LANGUAGE.default" in your language : fr,FR , en,US , . . .

Upload Sample Reports

1. Download the sample reports from https://gitlab.com/xivocc/sample_reports/-/raw/master/spagobi/samples_from_borealis_v23.zip
2. Import zip file in SpagoBI:
    1. Goto "Repository Management" -> "Import/Export"
    2. Click on "Browse/Choose your file" and choose the previous `samples_from_borealis_vxx.zip` downloaded file
    3. Click on "Import" icon
    4. Click next with default options until you are asked to override metadata, set **Yes** as shown in screen below



You can now browse the sample reports in *Document->Rapports->Exemples*.

Use the database status report to check if replication and reporting generation is working :

Totem Panels

Data replication can take some time if there are a lot of data in xivo cel and queue log tables. You may check xivo-db-replication log files (/var/log/xivocc/xivo-db-replication.log).

Preconfigured panels are available on http://@IP/kibana/#/dashboard/file/queues.json et http://@IP/kibana/#/dashboard/file/agents.json to be able to save this panels in elasticsearch database you have to sign on on request user admin/Kibana

Post Installation Check List

- All components are running : xivocc-dcomp ps
- Xuc internal database is synchronized with xivo check status page with http://xivoccserver:8090/

Période:	Du 01/01/2017 au 09/10/2017

Granularité	Heure

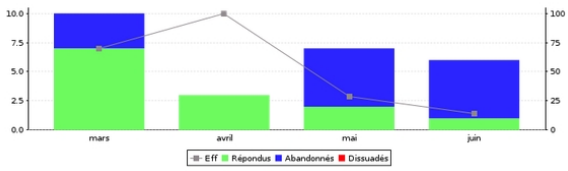
T1	15
T2	20

Légende :

Appels de files d'attente
- *Présentés: Nombres total d'appels présentés en période ouvrée
- *Reçus HNO: Nombres total d'appels reçus en période non ouvrée
- *Répondus: Nombre d'appels répondus
- *TMC: Temps moyen de communication (Temps de communication total / Appels Répondus)
- *Dissuadés: Nombre d'appels dissuadés (Saturé, Aucuns Agents Loggués, diversion,Temps d'Attente Maximun)
- *Raccrochés en FA: Nombre d'appels abandonnés en File d'attente
- *Abandonnés (en moins de T1,entre T1 et T2, en plus de T2) : Nombre d'appels abandonnés en File d'attente
- *Répondus (en moins de T1,entre T1 et T2,en plus de T2): Nombre d'appels répondus en File d'attente en moins de T1
- *TMA: Temps moyen d'attente (Temps d'attente Total / Présentés)
- *Transférés: Nombre d'appels transférés
- *Mis en garde: Nombres d'appels total mis en garde
- *QOS(T1): Qualité de service en fonction de T1 (Appels répondus en moins T1/ Appels répondus)
- *TA: Taux Accueil (Appels répondus / Appels Présentés)

commerce

Mois	Présentés	Répondus	Reçus HNO	TMC	Dissuadés	Raccrochés en file d'attente	Abandonnés en moins de T1	Abandonnés entre T1 et T2	Abandonnés en plus de T2	Répondus en moins de T1	Répondus entre T1 et T2	Répondus en plus de T2	TMA	Transférés	Mis en garde	QoS (T1)	TA
mars	10	7	0	00:00:04	0	3	2	1	0	4	0	3	00:00:13	0	3	57.14 %	70 %
avril	3	3	0	00:00:02	0	0	0	0	0	2	0	1	00:00:08	0	0	66.67 %	100 %
mai	7	2	0	00:00:03	0	5	5	0	0	2	0	0	00:00:02	0	0	100 %	28.57 %
juin	7	1	0	00:00:09	0	5	5	0	0	1	0	0	00:00:00	0	1	100 %	14.29 %
Total	**27**	**13**	**0**	**-**	**0**	**13**	**12**	**1**	**0**	**9**	**0**	**4**	**-**	**0**	**4**	**69.23 %**	**48.15 %**



Etat de la base de données

Dernier CEL

id	time
24143704	2016-06-28 14:29:39.850582

Dernier Queue log

id	time
4243356	2016-06-28 14:29:15.513830

Queue specific

time	queue ref	nb offered
6/28/16 2:00 PM		null

Queue periodic

time	queue	total
6/28/16 2:00 PM	cnr	1

Agent periodic

time	agent	login time
6/28/16 2:00 PM	482	00:14:59

Agent specific

time	agent num	nb offered
6/28/16 2:00 PM	429	null

Agent queue specific

time	agent num	queue ref
6/28/16 2:00 PM	525	

Call data

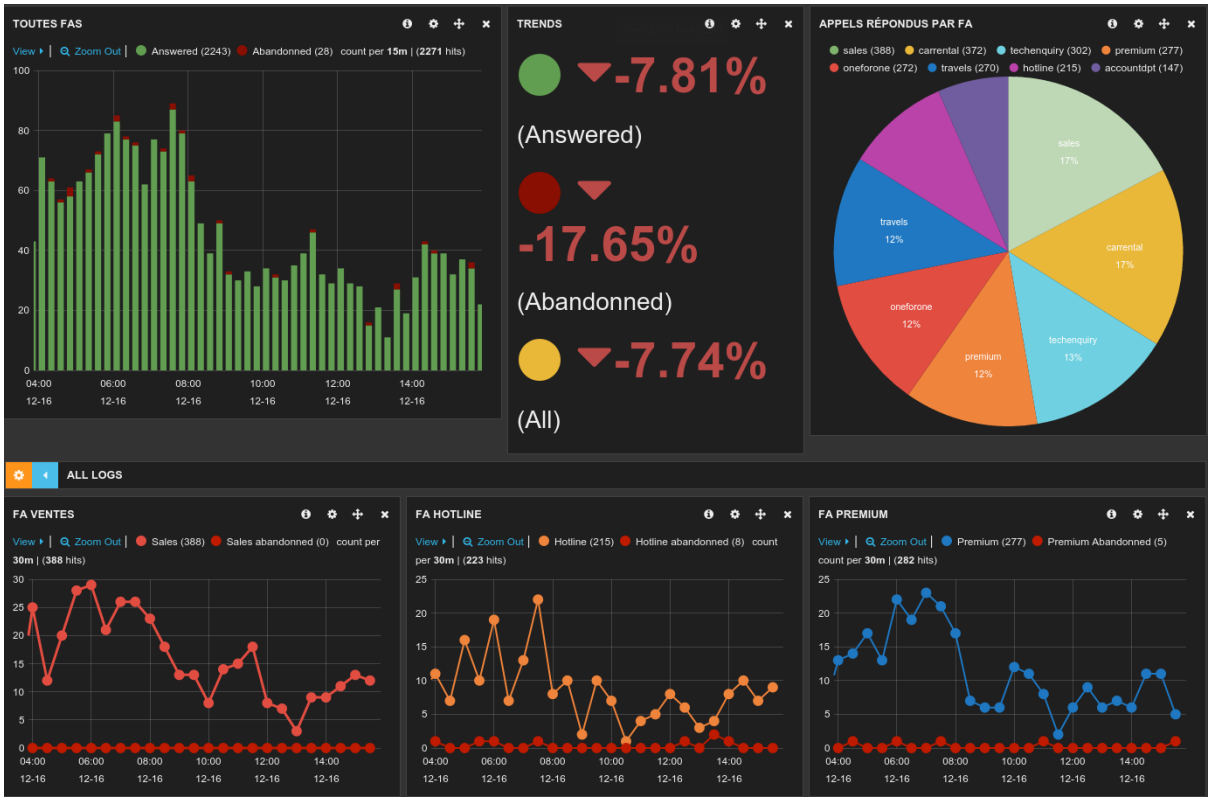
start time	src num	dst num	status	uniqueid
6/28/16 2:29 PM	489	496	answer	1467116971.417999
6/28/16 2:29 PM	397	391	answer	1467116957.417997

Call on queue

queue time	queue r	agent n	status
6/28/16 2:29 PM		257	answered
6/28/16 2:28 PM		406	answered

Configuration object

Agents	Queues	Agent groups	Extensions
369	69	11	707



- CCManager is running, log a user and check if you can see and manage queues : http://xivoccserver:8070/ccmanager
- Check database replication status using spagobi system report http://xivoccserver:9500/SpagoBI
- Check elasticsearch database status (totem panels) http://xivoccserver:9200/queuelogs/_status
- Check that you can listen to recordings http://xivoccserver:9400/
- Check totem panels http://192.168.85.102/kibana

Warning: Make sure to have few calls made in your XiVO, despite you will not see **anything** in totem or spagobi.

Recording

This feature **needs additional configuration steps on XiVO PBX**, see:

1. *Recording configuration*,
2. and (optionally) *Recording filtering configuration*.

3.2.2 Components Configuration

Important: When reading this section, keep in mind the *Architecture & Flows* diagram.

Overview

This section describes files installed by *xivocc-installer*. Some of these files may be modified for these reasons:

- Components customization
- Multi-server installation

XiVO CC system configuration is stored in the `/etc/docker/compose` directory. All files ale configured by the *xivocc-installer* and don't need to be edited. The directory contains these files:

- `docker-xivocc.yml`, called *compose file*, defines XiVO CC components and their configuration. It uses variables defined in the `.env` file.
- `.env` file assigns values to variables used in the *compose file*. But this file is being generated by *xivocc-dcomp* script and **should not be edited directly**.
- `factory.env` file stores XiVO CC version number and distribution and **should not be edited**.
- `custom.env` file may be used for component customization and multi-server installation.

Compose File

`docker-xivocc.yml`

Sections

- The main headers in the compose file (without indent) are container names.
- `image` - links to container image url on https://hub.docker.com/r/ page.
- `ports` - exposes container internal ports to host in format HOST:CONTAINER.
- `volumes_from` - mounts all of the volumes from another container
- `environment` - adds environment variables into container system
- `extra_hosts` - adds host address to /etc/hosts
- `links` - adds another container's host address to /etc/hosts

See detailed documentation on docker web.

Variables

Compose file contains more kinds of variables:

- Variable in the `environment` section without assignment is replaced by the same variable from `.env` file. If it's not defined or assigned in the `.env` file, it doesn't appear in the container system.
- Variable in the `environment` section with assigned value overrides value defined in the `.env` file.
- Variable defined in the `links` section can be assigned to variable in the `environment` section.
- Variable inside `${ }` block is replaced by value defined in `.env` file and can be used anywhere.

Variables in the factory.env file

This file should not be edited. XiVO CC version matches version of xivocc-installer and the other distributions are only for testing purposes.

- `XIVOCC_TAG` - XiVO CC version number
- `XIVOCC_DIST` - XiVO CC distribution

Variables in the custom.env file

All variables listed below can be edited or added to this file. But variables which are assigned inside compose file must be first unassigned from there.

List of variables shared across containers

Main components addresses and ports:

- `XIVO_HOST`
- `XUC_HOST`
- `XUC_PORT`

Customizable addresses and ports for multi-server installation:

- `CONFIG_MGT_HOST`
- `CONFIG_MGT_PORT`
- `RECORDING_SERVER_HOST`
- `RECORDING_SERVER_PORT`

Other:

- `JAVA_OPTS` - allocates Java memory and sets other Java options. If you want to use different values for containers, they must be assigned inside compose file. Or you can define new variable for this purpose - e.g.: `JAVA_OPTS=JAVA_OPTS_XUC` and set the value in the custom.env file.

List of docker containers and their variables

- pgxivocc
  - `POSTGRES_PASSWORD`
- postgresvols
- xivo_replic
  - `ELASTICTTL`
  - `DISABLEELASTIC`
- xivo_stats
- pack_reporting
  - `WEEKS_TO_KEEP`
- elasticsearch
- kibana_volumes
- fingerboard
- nginx
- spagobi
- recording_rsync
- config_mgt
- recording_server
  - `RECORDING_WEEKS_TO_KEEP`
- xuc

  - XIVO_AMI_SECRET
  - XIVO_AMI_PORT
  - XIVO_CTI_VERSION
  - API_eventUrl
  - XIVO_DIRD_PROFILE
  - SECURED_KRB5_PRINCIPAL
- xucmgt
  - SHOW_RECORDING_CONTROLS
  - SHOW_CALLBACKS
  - SHOW_QUEUE_CONTROLS
  - THIRD_PARTY_URL
  - ENFORCE_MANAGER_SECURITY
  - USE_SSO
- timezone
- xivocclogs

3.2.3 Phone Integration

XUC based web applications like agent interface or UC Assistant integrates buttons for phone control. This section details necessary configuration, supported phones and limitations.

Note: The VoIP VLAN network have to be accessible by the xivocc xuc server

Required configuration

The following steps are not required if you updated the Provisioning plugins.

Polycom phones

Warning: This is required only for plugins:

- xivo-polycom-4.0.9 version below v1.9
- xivo-polycom-5.4.3 version below v1.8

To enable phone control buttons on web interfaces you must update the basic template of Polycom phones:

- go to the plugin directory: */var/lib/xivo-provd/plugins/xivo-polycom-VERSION*
- copy the default template from *templates/base.tpl* to *var/templates/*
- then you must update *app.push* parameters in the else section **(do not replace switchboard settings)** as follows:

```
apps.push.messageType="5"
apps.push.username="guest"
apps.push.password="guest"
```

Snom phones

For transfer to work on Polaris version you must have plugins with version v2.2 or above.

Yealink phones

Warning: This is required only for plugins xivo-yealink-v80 below v1.31

To enable phone control buttons on web interfaces you must update the basic template of Yealink phones:

- go to the plugin directory: */var/lib/xivo-provd/plugins/xivo-yealink-VERSION*
- copy the default template from *templates/base.tpl* to *var/templates/*
- enable sip notify even for non switchboard profiles **(do not replace switchboard settings)**

```
{% if XX_options['switchboard'] -%}
push_xml.sip_notify = 1
call_waiting.enable = 0
{% else -%}
push_xml.sip_notify = 1
call_waiting.enable = 1
{% endif %}
```

Update Device Configuration

- to update device configuration you must run **`xivo-provd-cli -c 'devices.using_plugin("xivo-polycom-VERSION").reconfigure()'`**
- and finally you must resynchronize the device: **`xivo-provd-cli -c 'devices.using_plugin("xivo-polycom-VERSION").synchronize()'`**
- refer to provisioning documentation for more details
- if the phone synchronization fails check if the phone uses the version of the plugin you have updated, you can use **`xivo-provd-cli -c 'devices.find()'`**

Known limitations

Phone integration with Agent and Web / Desktop application has these limitations:

Transfer

- If the second call was initiated from Agent / Assistant and the called user rejected the call, the first call will stay hold until it is manually resumed
- If the second call was initiated from the phone, the transfer must be also completed from the phone. It can't be completed from Agent / Assistant.
- You cannot complete a transfer initiated from the Agent / Assistant by hanging up.

Conference with Yealink / Polycom

- Conference can't be created from Agent or Web / Desktop Assistant

3.2.4 XiVOcc Installation Troubleshooting

In order for the XiVOcc components to be fully functional, some customizations need to be done on the XiVOcc and the XiVO PBX.

This page can help to check that all the correct customization have been done by the installation package.

For the rest of this page we well make the following assumptions: - XiVO PBX has the IP 192.168.0.1 - XiVO CC has the IP 192.168.0.2

Important: Refer to the *Architecture & Flows* diagram.

Check XiVOcc Configuration

Check the prerequisites

- the OS must be Debian 9 (stretch), 64 bit,
- Docker must be installed,
- Docker-compose must be installed,
- the XiVO PBX must be reachable on the network.

Check ntp installation

The XiVO CC server and the XiVO server must be synchronized to the same source NTP source.

Check Logrotate configuration

A file `/etc/logrotate.hourly/docker-container` must be present which should log rotate files `/var/lib/docker/containers/*/*.log`

You can test it with `logrotate -fv /etc/logrotate.hourly/docker-container`. You should get some output and a new log file with suffix [CONTAINER ID]-json.log.1 should be created. This file is compressed in next rotation cycle.

Check Docker compose

- No alias for docker-compose should be defined. The following command should return "OK":

```
alias |grep -E 'docker-compose|dcomp' || echo "OK"
```

- The version of the docker images in the file `/etc/docker/compose/docker-xivocc.yml` must be in the form `${XIVOCC_TAG}.${XIVOCC_DIST}` and these variables must be set in the `/etc/docker/compose/factory.env` file:

```
...
xivo_stats:
    image: xivoxc/xivo-full-stats:${XIVOCC_TAG}.${XIVOCC_DIST}
...

xuc:
    image: xivoxc/xuc:${XIVOCC_TAG}.${XIVOCC_DIST}
...
```

Check the services

The list of the services launched should look like :

```
# xivocc-dcomp ps
           Name                         Command               State
↪         Ports
-------------------------------------------------------------------------------------
↪-----------------------------------
xivocc_elasticsearch_1      /docker-entrypoint.sh elas ...   Up       0.0.0.0:9200-
↪>9200/tcp, 0.0.0.0:9300->9300/tcp
xivocc_fingerboard_1        /bin/sh -c /usr/bin/tail - ...   Up
xivocc_kibana_volumes_1     /bin/sh -c /usr/bin/tail - ...   Up
xivocc_nginx_1              nginx -g daemon off;             Up       443/tcp, 0.0.
↪0.0:80->80/tcp
xivocc_pack_reporting_1     /bin/sh -c echo            ...   Up
xivocc_pgxivocc_1           /docker-entrypoint.sh postgres   Up       0.0.0.0:5443-
↪>5432/tcp
xivocc_postgresvols_1       /bin/bash                        Exit 0
xivocc_recording_server_1   bin/recording-server-docker      Up       0.0.0.0:9400-
↪>9000/tcp
xivocc_reporting_rsync_1    /usr/local/sbin/run-rsync.sh     Up       0.0.0.0:873->
↪873/tcp
xivocc_spagobi_1            /bin/sh -c /root/start.sh        Up       0.0.0.0:9500-
↪>8080/tcp
xivocc_timezone_1           /bin/bash                        Exit 0
xivocc_xivo_replic_1        /usr/local/bin/start.sh /o ...   Up
xivocc_xivo_stats_1         /usr/local/bin/start.sh /o ...   Up
xivocc_xivocclogs_1         /bin/bash                        Exit 0
xivocc_xuc_1                bin/xuc_docker                   Up       0.0.0.0:8090-
↪>9000/tcp
xivocc_xucmgt_1             bin/xucmgt_docker                Up       0.0.0.0:8070-
↪>9000/tcp
```

Check the XiVO PBX

Check PostgreSQL configuration

- Postgresql has to be configured to listen on all interfaces. See *listen_addresses* in file `/etc/postgresql/9.4/main/postgresql.conf`.
- Connection from the XiVO CC for user asterisk must be authorized. See file `/etc/postgresql/9.1/main/pg_hba.conf` which must contain a line:

```
host asterisk all 192.168.0.2/32 md5
```

- A user *stats* must exists. Use command `\dg` in psql.

Check AMI configuration

- A *xuc* user must be configured in the file `/etc/asterisk/manager.d/02-xivocc.conf`
- The command:

```
asterisk -rx "manager show user xuc"
```

must show the user.

CEL Configuration

The correct events must be activated in the file `/etc/asterisk/cel.conf`:

```
[general]
enable = yes
apps = dial,park,queue
events = APP_START,CHAN_START,CHAN_END,ANSWER,HANGUP,BRIDGE_ENTER,BRIDGE_EXIT,USER_
↪DEFINED,LINKEDID_END,HOLD,UNHOLD,BLINDTRANSFER,ATTENDEDTRANSFER

[manager]
enabled = yes
```

Check CTI configuration

In *Services* → *IPBX* → *Users* a user the must be created with the following parameters:

- CTI login : xuc
- CTI password : 0000
- Profile supervisor

Check WS configuration

In *Configuration* → *Web Services Access* a user must be created with the following parameters :

- Login : xivows
- Password : xivows
- Host : 192.168.0.2

Check ACD configuration

In *Services* → *Ipbx* → *Advanced configuration* make sure `Multiqueues call stats sharing` is checked.

Check the phone integration

Verify that the phone configuration where customized as detailed in *Required configuration for phone integration*.

Check the recording

The package `xivocc-recording` must be installed on XiVO PBX (see *Recording*) and configured (see *Recording configuration*).

For specific installations

3.2.5 Manual configuration and installation

This section describes the manual installation of the XiVO CC components. In most cases you **SHOULD NOT** follow this page, and install the XiVO CC components via the *xivocc-installer* package (see *Installation*).

Important: You **SHOULD NOT** follow this page to install XiVO CC. We leave this page here :

- to document how to install only a subset of the XiVO CC components (since it is not currently possible via the *xivocc-installer* package).
- to help with reconfiguring XiVO for XiVO CC after it has been *restored from backup*
- as a reference

Note: Since XiVO PBX 2017.06 some parts of the installation were moved from *xivocc-installer* to installation of XiVO PBX.

Prerequisites

We will assume your *XiVO CC* server meets the following requirements:

- OS : Debian 9 (stretch), 64 bit
- the latest stable version of Docker is installed
- the latest stable version of Docker-compose is installed
- the XiVO PBX is reachable on the network
- the XiVO PBX is setup with users, queues and agents, you must be able to place and answer calls.

Note : Install only stable version of docker and docker compose.

We will make the following assumptions :

- the *XiVO PBX* has the IP 192.168.0.1
- some data (incoming calls, internal calls etc.) might be available on XiVO (otherwise, you will not see *anything* in the *Post Installation Check List*).
- the *XiVO CC* server has the IP 192.168.0.2

XiVO PBX configuration

PostgreSQL configuration

Firstly, allow access to PostgreSQL from the outside. Edit `/etc/postgresql/9.4/main/postgresql.conf`:

```
listen_addresses = '*'
```

Add this line to `/etc/postgresql/9.4/main/pg_hba.conf`:

```
host asterisk all 192.168.0.2/32 md5
```

Create a user *stats* with read permissions :

```
sudo -u postgres psql asterisk << EOF
CREATE USER stats WITH PASSWORD 'stats';
GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO stats;
EOF
```

And run `xivo-service restart all` to apply these modifications.

AMI configuration

- Add file `/etc/asterisk/manager.d/02-xivocc.conf` directory with the following content, replacing *X.X.X.X* by your xucserver IP address :

```
[xuc]
secret = xucpass
deny=0.0.0.0/0.0.0.0
permit=X.X.X.X/255.255.255.255
read = system,call,log,verbose,command,agent,user,dtmf,originate,dialplan
write = system,call,log,verbose,command,agent,user,dtmf,originate,dialplan
writetimeout = 10000
```

- And reload the AMI :

```
asterisk -rx "manager reload"
asterisk -rx "manager show user xuc" and check your if previous configuration is␣
↪displayed.
```

CEL Configuration

- Replace content of file `/etc/asterisk/cel.conf` by the following :

```
[general]
enable = yes
apps = dial,park,queue
events = APP_START,CHAN_START,CHAN_END,ANSWER,HANGUP,BRIDGE_ENTER,BRIDGE_EXIT,USER_
↪DEFINED,LINKEDID_END,HOLD,UNHOLD,BLINDTRANSFER,ATTENDEDTRANSFER

[manager]
enabled = yes
```

- and reload the cel module in Asterisk :

```
asterisk -rx "module reload cel"
```

Customizations in the web interface

- Create a user *xuc* in *Services* -> *IPBX* -> *Users* with the following parameters:
- CTI login : xuc
- CTI password : 0000
- profil supervisor
- Create a Web Services user in *Configuration* -> *Web Services Access* with the following parameters :
- Login : xivows
- Password : xivows
- Host : 192.168.0.2

Make sure **Multiqueues call stats sharing** is enabled in *Services* -> *IPBX* -> *Advanced configuration* tab.

Phone integration

Do not forget to follow configuration steps detailed in *Required configuration for phone integration*.

Recording

This feature **needs additional configuration steps on XiVO PBX**, see:

1. *Recording configuration*,
2. and (optionally) *Recording filtering configuration*.

XiVO CC configuration

Now we switch to the installation of the XiVO CC server.

Install ntp server

```
apt-get install ntp
```

XiVO CC server and *XiVO PBX* server must be synchronized to the same source.

Enable Docker LogRotate

Docker container log output to /dev/stdout and /dev/stderr. The Docker container log file is saved in /var/lib/docker/containers/[CONTAINER ID]/[CONTAINER_ID]-json.log.

Create a new Logrotate config file for your Docker containers in the Logrotate folder /etc/logrotate.d/docker-container.

```
/var/lib/docker/containers/*/*.log {
  rotate 7
  daily
  compress
  missingok
  delaycompress
  copytruncate
}
```

You can test it with logrotate -fv /etc/logrotate.d/docker-container. You should get some output and a new log file with suffix [CONTAINER ID]-json.log.1 should be created. This file is compressed in next rotation cycle.

Retrieve the configuration script and launch it:

Containers installation

```
wget  https://gitlab.com/xivo.solutions/packaging/raw/master/install/install-
↪docker-xivocc.sh
bash install-docker-xivocc.sh
```

During the installation, you will be asked for :

- the XiVO IP address (e.g. 192.168.0.1)
- the number of weeks to keep for the statistics
- the number of weeks to keep for the recording files
- the external IP of the machine (i.e. the adress used afterwards for http URLs)

The number of weeks to keep statistics **must be higher** than the number of weeks to keep recordings. Recording purging is based on the statistic data, so the statistic data must not be removed before purging recordings.

Create the following alias in your .bashrc file:

```
vi ~/.bashrc
alias dcomp='docker-compose -p xivocc -f /etc/docker/compose/docker-xivocc.yml'
```

Containers modification

The yml file `/etc/docker/compose/docker-xivocc.yml` should have the correct tag version for each imeage.

Check also that the **XIVO_CTI_VERSION** is correct for the xuc container.

```
xivo_replic :
  image: xivoxc/xivo-db-replication:2016.03.latest

xivo_stats :
  image: xivoxc/xivo-full-stats:2016.03.latest

pack_reporting:
  image: xivoxc/pack-reporting:2016.03.latest

config_mgt:
  image: xivoxc/config-mgt:2016.03.latest

recording_server:
  image: xivoxc/recording-server:2016.03.latest

xuc:
  image: xivoxc/xuc:2016.03.latest

  environment:
  - XIVO_CTI_VERSION=2.1

xucmgt:
  image: xivoxc/xucmgt:2016.03.latest
```

Starting XivoCC

Then you can launch the XiVO CC with the following command :

```
dcomp up -d
```

List XivoCC services :

```
# dcomp ps
           Name                         Command               State
↪         Ports
-------------------------------------------------------------------------------------
↪-----------------------------------
xivocc_elasticsearch_1      /docker-entrypoint.sh elas ...   Up       0.0.0.0:9200-
↪>9200/tcp, 0.0.0.0:9300->9300/tcp
xivocc_fingerboard_1        /bin/sh -c /usr/bin/tail - ...   Up
xivocc_kibana_volumes_1     /bin/sh -c /usr/bin/tail - ...   Up
xivocc_nginx_1              nginx -g daemon off;             Up       443/tcp, 0.0.
↪0.0:80->80/tcp
xivocc_pack_reporting_1     /bin/sh -c echo            ...   Up
xivocc_pgxivocc_1           /docker-entrypoint.sh postgres   Up       0.0.0.0:5443-
↪>5432/tcp
xivocc_postgresvols_1       /bin/bash                        Exit 0
xivocc_recording_server_1   bin/recording-server-docker      Up       0.0.0.0:9400-
↪>9000/tcp
```

(continues on next page)

(continued from previous page)

```
xivocc_reporting_rsync_1     /usr/local/sbin/run-rsync.sh      Up        0.0.0.0:873->
↪873/tcp
xivocc_spagobi_1             /bin/sh -c /root/start.sh         Up        0.0.0.0:9500-
↪>8080/tcp
xivocc_timezone_1            /bin/bash                         Exit 0
xivocc_xivo_replic_1         /usr/local/bin/start.sh /o ...    Up
xivocc_xivo_stats_1          /usr/local/bin/start.sh /o ...    Up
xivocc_xivocclogs_1          /bin/bash                         Exit 0
xivocc_xuc_1                 bin/xuc_docker                    Up        0.0.0.0:8090-
↪>9000/tcp
xivocc_xucmgt_1              bin/xucmgt_docker                 Up        0.0.0.0:8070-
↪>9000/tcp
```

Upgrading

3.2.6 Upgrade

Upgrading a *XiVO CC* is done by executing commands through a terminal on the server.

Note: Downgrade is not supported

Overview

The upgrade consists of the following steps:

- switch/verify the version in the debian sources list
- update of the `xivocc-installer` package
- update of the Docker images

Warning: This upgrade procedure applies only to XiVO CC installed via the `xivocc-installer` package.

Preparing the upgrade

There are two cases:

1. *Upgrade to another LTS XiVO CC version*,
2. *Upgrade to the latest Bugfix release* of your current installed LTS version.

Upgrade to another LTS version

To upgrade to another XiVO Solution **LTS**:

1. Switch the debian sources to the targetted **LTS** version (it should be located in the file `/etc/apt/sources.list.d/xivo-dist.list`). For example, to switch to Aldebaran LTS version:

   ```
   deb http://mirror.xivo.solutions/debian/ xivo-aldebaran main
   ```

2. **Read carefully the** *Release Notes* starting from your current version to the version you target (read **even more carefully** the New features and Behavior changes between LTS)
3. **Check** the specific instructions and manual steps *from your current LTS to your targetted LTS* and all intermediate LTS: see *Manual steps for LTS upgrade*

4. **Check also** if you are in a specific setup that requires a *specific procedure*
5. And then upgrade, see *Upgrading*

Upgrade to latest Bugfix release of an LTS version

Important: For version older than Five (2017.03), see XiVO Five documentation

After the release of a *version* (e.g. *Polaris (2017.11)*) we may backport some bugfixes in this version. We will then create a **subversion** (e.g. Polaris **.04** (2017.11 **.04**)) shipping these bugfixes. These bugfix version does not contain any behavior change.

To upgrade to the **latest subversion** of your current installed *version* you need to:

1. **Read carefully the** *Release Notes* starting from your installed version (e.g. Polaris.00) to the latest bugfix release (e.g. Polaris.04).
2. Verify that the debian sources list corresponds to your *installed LTS* (it should be located in the file `/etc/apt/sources.list.d/xivo-dist.list`)
3. Verify that the `/etc/docker/compose/factory.env` file has
   - `XIVOCC_TAG=VERSION` (where `VERSION` is your current installed *version* - e.g. *2017.11*)
   - and `XIVOCC_DIST=latest`
4. And then upgrade, see *Upgrading*

Upgrading

After having prepared your upgrade (see above), you can upgrade:

1. When you have checked the sources.list you can upgrade with the following commands:

```
apt-get update
apt-get install xivocc-installer
```

2. If there is any change, you should accept the new `docker-compose.yml` file. Then compare it with the old `docker-compose.yml.dpkg-old` file and report in the new any specific configuration.
3. Then download the new docker images:

```
xivocc-dcomp pull
```

4. And run the new containers (**Corresponding XiVO CC services will be restarted**):

```
xivocc-dcomp up -d --remove-orphans
```

Note: Please, ensure your server date is correct before starting. If system date differs too much from correct date, you may get an authentication error preventing download of the docker images.

Post Upgrade

When finished:

- Check your upgrade through *Post Installation Check List*.
- Check that all the services are in the correct version. Compare the output of `xivocc-dcomp version` with the table in *Release Notes*

Manual steps for LTS upgrade

See *Manual steps for LTS upgrade* in XiVO Upgrade page.

Specific procedures

Old Pack Reporting Upgrade Procedures

These notes include upgrade procedures for old versions of the **Pack reporting**, before **XivoCC** starts and before it was packaged with Docker. In those cases, run the following command to find the installed version of the pack reporting:

```
dpkg -l|grep pack-reporting
```

From version < 1.6

- data retention time will be lost during upgrade : save it and write it back in */etc/xivo-reporting-db.conf*
- the upgrade is likely to be long if there is a lot of data in *queue_log*. Purge old data out of this table if possible in order to accelerate the upgrade
- at the end of the upgrade, run *apt-get autoremove* (deletion of xivo-stat, xivo-libdao and xivo-lib-python)

From version < 1.8

- XiVO in version < 14.08 is not supported anymore
- if it is required, the upgrade of the XiVO must be done before the upgrade of the pack reporting, and no call must be performed between the two upgrades

From a version using Debian packaging to a version using Docker

- **Beware**: this will require a migration of the original PostgreSQL database to the Dockerised one. For this you need to have free disk space : the amount of free disk space must equal the size of */var/lib/postgresql*. This check must be performed after docker images have been pulled.
- Run the following commands:

```
apt-get update
service xivo-db-replication stop
service xivo-full-stats stopsource/releasenotes/index.rst
apt-get install pack-reporting xivo-full-stats xivo-reporting-db xivo-db-
↪replication db-utils
service xivo-db-replication stop
service xivo-full-stats stop
```

- Install docker, docker-compose and xivocc-installer
- Open *docker-xivocc.yml* and remove sections recording_rsync, config_mgt, recording_server, xuc, xucmgt
- Run *xivocc-dcomp pull*
- CHECK THE FREE DISK SPACE. The next command will migrate the database. This may take several hours.

```
sudo -u postgres pg_dump --format c xivo_stats | docker exec -i xivocc_pgxivocc_1␣
↪pg_restore -U postgres -d xivo_stats
```

- Start by *xivocc-dcomp up -d*

From a dockerized version before callbacks

- Run the following commands:

```
docker exec -ti compose_pgxivocc_1 psql -U postgres -c 'CREATE EXTENSION IF NOT␣
↪EXISTS "uuid-ossp"' xivo_stats
docker exec -ti compose_pgxivocc_1 psql -U postgres -c 'CREATE EXTENSION IF NOT␣
↪EXISTS "uuid-ossp"' xuc_rights
```

Upgrade notes

See *Release Notes* for version specific informations.

3.3 XiVO Distributed System

3.3.1 Installing XDS

The XDS architecture has the following components:

- XiVO
- Media Server (MDS) (one or more)

An XDS needs also, to interconnect with a provider, a:

- a XiVO Gateway (XGW) (one or more) to interconnect with provider(s),
- a Reporting Server for the centralized call history,
- a CTI Server for the UC features.

This page will guide you through:

1. the configuration of the XiVO (see *XiVO Configuration* section)
2. the installation and configuration of the MDS (see *Media Server Configuration* section)
3. the configuration of the XGW (see *XiVO Gateway Configuration* section)
4. and the configuration of the CC (Reporting and CTI Server) (see *XiVO CC Configuration* section)

Requirements

Before starting you need to have 3 servers. Here's a table summarizing what we are installing. Replace the IP by those you chose.

Server	server1	server2	server3	server4
Role	XiVO	XiVO Gateway	Media Server	Reporting/CTI Server
Name	mds0	xgw1	mds1	cc
IP Data	10.32.0.1	10.32.0.201	10.32.0.101	10.32.0.9
IP VoIP	10.32.5.1	10.32.5.201	10.32.5.101	10.32.5.9

XiVO Configuration

On *server1*:

- install XiVO (see *Installing the System*).
- pass the Wizard

AMI configuration

Note: Once a *media server is defined* in webi, Reporting Server will use the VoIP interface for AMI connection to all media servers **and XiVO**. Here we authorize it on XIVO.

If XiVO CC is installed, do the following steps before adding media server. Otherwise you can first define media servers and do these steps right after XiVO CC installation, but before starting it to prevent problems with fail2ban.

1. Edit **existing** file `/etc/asterisk/manager.d/02-xivocc.conf` to add permission for Reporting Server:
   - `permit` to authorize the VoIP IP of the Reporting Server. E.g.:

```
...
deny=0.0.0.0/0.0.0.0
permit=10.32.5.9/255.255.255.255
permit=10.32.0.9/255.255.255.255
...
```

2. Apply the configuration:

```
asterisk -rx 'manager reload'
```

Define Media Servers

Note: Here we define our Media Servers (MDS) names and VoIP IP address.

In XiVO webi,

1. go to *Configuration -> Management -> Media Servers*
2. Add a line per Media Server (MDS) (below an example for mds1):
   1. *Name*: mdsX (e.g. mds1)
   2. *Displayed Name*: Media Server X (e.g. Media Server 1)
   3. *IP VoIP*: <VoIP IP of mdsX> (e.g. 10.32.5.101) - *note:* the VoIP streams between XiVO and mdsX will go through this IP

Define Media Servers for Provisionning

Note: Here we configure the Media Servers (MDS) for the phones.

In XiVO webi

1. go to *Configuration -> Provisioning -> Template Line*
2. Create a template line per MDS (below the example for mds1):
   1. *Unique name*: <mdsX> (e.g. mds1) - *note:* it **must be** the same name as the one defined in section *Define Media Servers*
   2. *Displayed Name*: <Media Server X> (e.g. Media Server 1)
   3. *Registrar Main*: <VoIP IP of mdsX> (e.g. 10.32.5.101)
   4. *Proxy Main*: <VoIP IP of mdsX> (e.g. 10.32.5.101)

Create SIP peers for XDS intra-routing

Note: These SIP peers will be used for call routing between XiVO and MDS.

In XiVO webi:

1. go to *Services -> IPBX -> Trunk Management -> SIP Protocol*
2. create one peer for mds0:
    1. *Name*: default - *note:* it **must be** default
    2. *Authentication username*: default
    3. *Password*: <pick a password>
    4. *IP Addressing Type*: Static with <VoIP IP of mds0> (e.g. 10.32.5.1)
    5. *Connection type*: Peer
    6. *Context*: default
3. create one peer **per mds**:
    1. *Name*: <mdsX> (e.g. mds1) - *note:* it **must be** the same name as the one defined in section *Define Media Servers*
    2. *Authentication username*: <mdsX> (e.g. mds1)
    3. *Password*: <pick a password>
    4. *IP Addressing Type*: Static with <VoIP IP of mdsX> (e.g. 10.32.5.101)
    5. *Connection type*: Peer
    6. *Context*: default

Create SIP trunks towards XiVO Gateway

Note: This SIP peer and user will be used for call routing between XDS (XiVO and MDS) and XGW.

In XiVO Webi:

1. go to *Services -> IPBX -> Trunk Management -> SIP Protocol*
2. create one peer **per xgw** (this peer will handle outgoing call from XDS to the provider via the XGW):
    1. *Name*: <xds_out-xgwX> (e.g. xds_out-xgw1)
    2. *Authentication username*: <xds_out-xgwX> (e.g. xds_out-xgw1)
    3. *Password*: <pick a password>
    4. *IP Addressing Type*: Static with <VoIP IP of xgwX> (e.g. 10.32.5.201)
    5. *Connection type*: Peer
    6. *Context*: default
    7. in tab *Advanced*
        1. *From User*: <xds_out-xgwX> (e.g. xds_out-xgw1)
3. create a user (this user will handle incoming call from all XGW to the XDS via XiVO (mds0)):
    1. *Name*: xds_in-mds0
    2. *Authentication username*: xds_in-mds0

3. *Password*: <pick a password>
4. *Connection type*: **User**
5. *Context*: from-extern

Create Outgoing Call Rule

Note: This outgoing call rule will handle outgoing call from XDS to XGW.

In XiVO Webi:

1. go to *Services -> IPBX -> Call Management -> Outgoing calls*
2. create an outgoing call rule for XDS:
    1. Trunks: <xds_out-xgwX> (e.g. xds_out-xgw1)
    2. Extentions: X.
    3. Callerid: <main DID of the system>

System

Note: The MDS need to connect to XiVO database.

In file `/etc/postgresql/9.4/main/postgresql.conf` change parameter:

```
listen_addresses = 'localhost'
```

to:

```
listen_addresses = '*'
```

In file `/etc/postgresql/9.4/main/pg_hba.conf` add an authorization **per mds** to connect to the db (here you could use the IP Data of mdsX):

```
host  asterisk    all 10.32.4.201/32  md5
```

And restart the services:

```
xivo-service restart all
```

Media Server Configuration

Requirements

On *server3* install a **Debian 9** with:

- `amd64` architecture,
- `en_US.UTF-8` locale,
- `ext4` filesystem
- a hostname correctly set (files `/etc/hosts` and `/etc/hostname` must be coherent).

Before installing the MDS you **have to** have added:

- the MDS to the XiVO configuration (see *Define Media Servers* section)
- the MDS IP address to the postgresql configuration (see *System* section)

Installation

Important: The MDS installer will ask you:

- the XiVO Data IP Address
- the Media Server you're installing (taken from the Media Server you declared at step *Define Media Servers*)
- the Media Server Data IP
- the Reporting Server IP

To install the MDS, download the XiVO installation script:

```
wget http://mirror.xivo.solutions/mds_install.sh
chmod +x mds_install.sh
```

and run it:

Important: Use `-a` switch to chose **the same version** as your XiVO (mds0)

```
./mds_install.sh -a 2018.16-latest
```

When prompted:

- give the IP of XiVO (mds0): <XiVO Data IP> (e.g. 10.32.0.2)
- select the MDS you're installing: <mdsX> (e.g. mds1)
- enter the MDS Data IP: <mdsX Data IP> (e.g. 10.32.0.101)
- and finally the Reporting Server Data IP: <reporting Data IP) (e.g. 10.32.0.5)

Configuration

To finalize the MDS configuration you have to:

1. In file `/etc/docker/mds/custom.env`, add the following environment variable:
   - `XIVO_VOIP_HOST` with the VoIP IP address of the XiVO (e.g. 10.32.5.1)
   - `MDS_PEERNAME` with the name of the SIP peer of this MDS (e.g. mds1) - *note* as created in step *Create SIP peers for XDS intra-routing*
   - `MDS_PEERSECRET` with the password of the SIP peer of this MDS - *note* as created in step *Create SIP peers for XDS intra-routing*
   - e.g.:

     ```
     cat >> /etc/docker/mds/custom.env << EOF
     XIVO_VOIP_HOST=10.32.5.1
     MDS_PEERNAME=mds1
     MDS_PEERSECRET=secret
     EOF
     ```

2. Create file `/etc/asterisk/manager.d/02-xuc.conf` to add permission for Xuc Server to connect with:

- `secret` must be the same as the secret for xuc user on XiVO,
- `permit` to authorize the VoIP IP of the Reporting Server
- e.g.:

```
cat > /etc/asterisk/manager.d/02-xuc.conf << EOF
[xuc]
secret = muq6IWgNU1Z
deny=0.0.0.0/0.0.0.0
permit=10.32.5.9/255.255.255.255
read = system,call,log,verbose,command,agent,user,dtmf,originate,
↪dialplan
write = system,call,log,verbose,command,agent,user,dtmf,originate,
↪dialplan
writetimeout = 10000
EOF
```

3. Restart the services:

```
xivo-service restart all
```

XiVO Gateway Configuration

On *server2*:

- install XiVO (see *Installing the System*).
- pass the Wizard

Create SIP trunks for XDS

Note: These SIP peer/user will be used for call routing between XGW and XDS (XiVO and MDS).

In XiVO GW Webi:

1. go to *Services -> IPBX -> Trunk Management -> SIP Protocol*
2. create one user (this user will handle calls from the XDS to the provider via XGW) - *note:* this user must match the peer defined in section *Create SIP trunks towards XiVO Gateway*
    1. *Name*: <xds_out-xgwX> (e.g. xds_out-xgw1)
    2. *Authentication username*: <xds_out-xgwX> (e.g. xds_out-xgw1)
    3. *Password*: <pick a password> (same as in section *Create SIP trunks towards XiVO Gateway*)
    4. *Connection type*: **User**
    5. *Context*: default
3. create a peer (this peer will handle calls from the provider towards the XDS (via mds0) through this XGW) - *note:* this peer must match the user defined in section *Create SIP trunks towards XiVO Gateway*
    1. *Name*: xds_in-mds0
    2. *Authentication username*: xds_in-mds0
    3. *Password*: <pick a password>
    4. *IP Addressing Type*: Static with <VoIP IP of **mds0**> (e.g. 10.32.5.1)
    5. *Connection type*: Peer
    6. *Context*: default

7. in tab *Advanced*
    1. *From User*: xds_in-mds0

Create the Provider Trunk

Add on the XGW the trunk towards your provider (it can be an ISDN or SIP trunk).

Create Call Routing on the XGW

You then need to create the call routing between the provider and the XDS knowing that:

- calls coming from the XDS towards provider will come in via the trunk `xds_out-xgwX` (e.g. xds_out-xgw1)
- and calls coming from the provider towards the XDS have to go out via the trunk `xds_in-mds0`

XiVO CC Configuration

On *server4*:

- install a XiVO CC (see *Installation*)
- configure it
- before starting it, change the *AMI configuration* on XiVO.

Known Limitations

Agent states after XUC restart

Restarting XUC server with active calls in XDS environment will result in having some agents in incorrect state. Please see the note in *restarting* XUC server with active calls.

CHAPTER 4

Administrator's Guide

In-depth documentation on administration of XiVO solution systems.

4.1 XiVO Administration

4.1.1 System

DHCP Server

XiVO includes a DHCP server used for assisting in the provisioning of phones and other devices. (See *Basic Configuration* for the basic setup). This section contains additional notes on how to configure more advanced options that may be helpful when integrating the server with different VOIP subnets.

Activating DHCP on another interface

DHCP Server can be activated through the XiVO Web Interface *Configuration* → *Network* → *DHCP* :

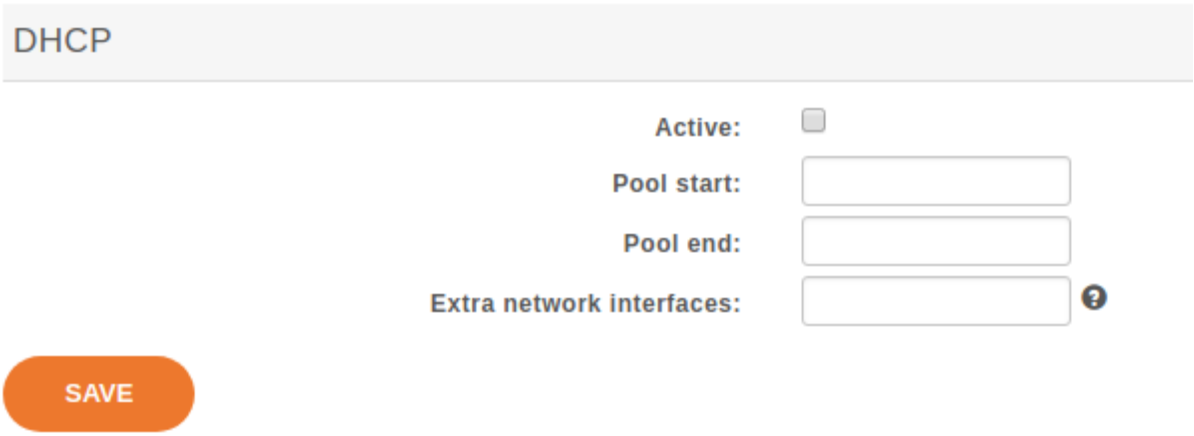


Fig. 1: *Configuration* → *Network* → *DHCP*

By default, it will only answer to DHCP requests coming from the VoIP subnet (defined in the *Configuration* → *Network* → *Interfaces* section). If you need to activate DHCP on an other interface, you have to fill in the *Extra network interfaces* field with the interface name , for example : `eth0`

After saving your modifications, click on *Apply system configuration* so that the new settings can take effect.

Changing default DHCP gateway

By default, the XiVO DHCP server uses the XiVO's IP address as the routing address. To change this you must create a custom-template:

1. Create a custom template for the `dhcpd_subnet.conf.head` file:

```
mkdir -p /etc/xivo/custom-templates/dhcp/etc/dhcp/
cd /etc/xivo/custom-templates/dhcp/etc/dhcp/
cp /usr/share/xivo-config/templates/dhcp/etc/dhcp/dhcpd_subnet.conf.head .
```

2. Edit the custom template:

```
vim dhcpd_subnet.conf.head
```

3. In the file, replace the string `#XIVO_NET4_IP#` by the routing address of your VoIP network, for example:

```
option routers 192.168.2.254;
```

4. Re-generate the dhcp configuration:

```
xivo-update-config
```

DHCP server should have been restarted and should now use the new routing address.

Configuring DHCP server to serve unknown hosts

By default, the XiVO DHCP server serves only known hosts. That is:

- either hosts which MAC address prefix (the OUI) is known
- or hosts which Vendor Identifier is known

Known OUIs and Vendor Class Identifiers are declared in `/etc/dhcp/dhcpd_update/*` files.

If you want your XiVO DHCP server to serve also unknown hosts (like PCs) follow these instructions:

1. Create a custom template for the `dhcpd_subnet.conf.tail` file:

```
mkdir -p /etc/xivo/custom-templates/dhcp/etc/dhcp/
cd /etc/xivo/custom-templates/dhcp/etc/dhcp/
cp /usr/share/xivo-config/templates/dhcp/etc/dhcp/dhcpd_subnet.conf.tail .
```

2. Edit the custom template:

```
vim dhcpd_subnet.conf.tail
```

3. And add the following line at the head of the file:

```
allow unknown-clients;
```

4. Re-generate the dhcp configuration:

```
xivo-update-config
```

DHCP server should have been restarted and should now serve all network equipments.

DHCP-Relay

If your telephony devices aren't located on the same site and the same broadcast domain as the XiVO DHCP server, you will have to add the option *DHCP Relay* to the site's router. This parameter will allow the DHCP requests from distant devices to be transmitted to the IP address you specify as DHCP Relay.

Warning: Please make sure that the IP address used as DHCP Relay is the same as one of XiVO's interfaces, and that this interface is configured to listen to DHCP requests (as decribed in previous part). Also verify that routing is configured between the distant router and the choosen interface, otherwise DHCP requests will never reach the XiVO server.

Configuring DHCP server for other subnets

This section describes how to configure XiVO to serve other subnets that the VOIP subnet. As you can't use the Web Interface to declare other subnets (for example to address DATA subnet, or a VOIP subnet that isn't on the same site that XiVO server), you'll have to do the following configuration on the Command Line Interface.

Creating "extra subnet" configuration files

First thing to do is to create a directory and to copy into it the configuration files:

```
mkdir /etc/dhcp/dhcpd_sites/
cp /etc/dhcp/dhcpd_subnet.conf /etc/dhcp/dhcpd_sites/dhcpd_siteXXX.conf
cp /etc/dhcp/dhcpd_subnet.conf /etc/dhcp/dhcpd_sites/dhcpd_lanDATA.conf
```

Note: In this case we'll create 2 files for 2 differents subnets. You can change the name of the files, and create as many files as you want in the folder `/etc/dhcp/dhcpd_sites/`. Just adapt this procedure by changing the name of the file in the different links.

After creating one or several files in `/etc/dhcp/dhcpd_sites/`, you have to edit the file `/etc/dhcp/dhcpd_extra.conf` and add the according include statement like:

```
include "/etc/dhcp/dhcpd_sites/dhcpd_siteXXX.conf";
include "/etc/dhcp/dhcpd_sites/dhcpd_lanDATA.conf";
```

Adjusting Options of the DHCP server

Once you have created the subnet in the DHCP server, you must edit each configuration file (the files in `/etc/dhcp/dhcpd_sites/`) and modify the different parameters. In section **subnet**, write the IP subnet and change the following options (underlined fields in the example):

```
subnet 172.30.8.0 netmask 255.255.255.0 {
```

- subnet-mask:

  ```
  option subnet-mask 255.255.255.0;
  ```

- broadcast-address:

  ```
  option broadcast-address 172.30.8.255;
  ```

- routers (specify the IP address of the router that will be the default gateway of the site):

```
option routers 172.30.8.1;
```

In section **pool**, modify the options:

```
pool {
```

- log (add the name of the site or of the subnet):

```
log(concat("[", binary-to-ascii(16, 8, ":", hardware), "] POOL VoIP Site XXX
↪"));
```

- range (it will define the range of IP address the DHCP server can use to address the devices of that subnet):

```
range 172.30.8.10 172.30.8.200;
```

Warning: XiVO only answers to DHCP requests from supported devices. In case of you need to address other equipment, use the option *allow unknown-clients;* in the `/etc/dhcp/dhcpd_sites/` files

- If you have checked the "DHCP integration" (See *Advanced Configuration* for the basic setup) in provisionning configuration, you will also **MUST** add the parameter below: (**Otherwise provd won't be able to route the devices to the correct plugins**)

```
on commit {
            execute("dxtorc",
                    "commit",
                    binary-to-ascii(10, 8, ".", leased-address),
                    binary-to-ascii(16, 8, ":", suffix(hardware, 6)),
                    pick-first-value(concat("060", binary-to-ascii(16, 8, ".",␣
↪option vendor-class-identifier)), "")
                  );
        }
```

At this point, you can apply the changes of the DHCP server with the command:

```
service isc-dhcp-server restart
```

After that, XiVO will start to serve the DHCP requests of the devices located on other sites or other subnets than the VOIP subnet. You will see in `/var/log/daemon.log` all the DHCP requests received and how they are handled by XiVO.

Mail

This section describes how to configure the mail server shipped with XiVO (Postfix) and the way XiVO handles mails.

In *Configuration* → *Network* → *Mail*, the following options can be configured:

- *Domain Name messaging* : the server's displayed domain. Will appear in "Received" mail headers.
- *Source address of the server* : domain part of headers "Return-Path" and "From".
- *Relay SMTP* and *FallBack relay SMTP* : relay mail servers.
- *Rewriting shipping addresses* : Canonical address Rewriting. See Postfix canonical documentation for more info.

Warning: Postfix, the mail server shipped with XiVO, should be stopped on an installed XiVO with no valid and reachable DNS servers configured. If Postfix is not stopped, messages will bounce in queues and could end up affecting core pbx features.

If you need to disable Postfix here is how you should do it:

```
systemctl stop postfix
systemctl disable postfix
```

If you ever need to enable Postfix again:

```
systemctl enable postfix
systemctl start postfix
```

Alternatively, you can empty Postfix's queues by issuging the following commands on the XiVO server:

```
postsuper -d ALL
```

Network

This section describes how to configure additional network devices that may be used to better accomodate more complex network infrastructures. Network interfaces are managed in the XiVO web interface via the page *Configuration → Network → Interfaces*.

XiVO offers 2 types of interfaces: *VoIP* and *Data*. The *VoIP* interface is used by the DHCP server, provisioning server, and phone devices connected to your XiVO. These services will use the information provided on the *VoIP* interface for their configuration. For example, the DHCP server will only listen on the VoIP interface by default.

To change these settings, you must either create a new interface or edit an existing one and change its type. When adding a new *VoIP* interface, the type of the old one will automatically be changed to *Data*.

Configuring a physical interface

In this example, we'll add and configure the **eth1** network interface on our XiVO.

First, we see there's already an unconfigured network interface named **eth1** on our system:

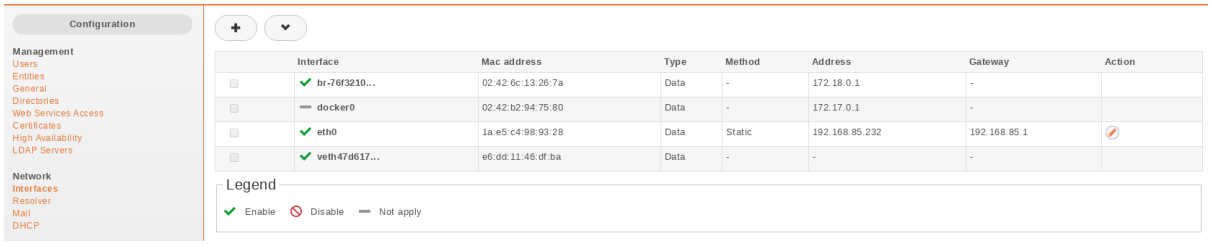


Fig. 2: *Configuration → Network → Interfaces*

To add and configure it, we click on the small plus button next to it, and we get to this page:

In our case, since we want to configure this interface with static information (i.e. not via DHCP), we fill the following fields:

Note that since our **eth0** network interface already has a default gateway, we do not enter information in the `Default gateway` field for our **eth1** interface.

Once the changes have been saved, the action **Apply network configuration** will appear in bold. This action must be clicked in order for the changes to take effect.

Adding a VLAN interface

In this example, the XiVO already has 2 network interfaces configured:

Listing the network interfaces

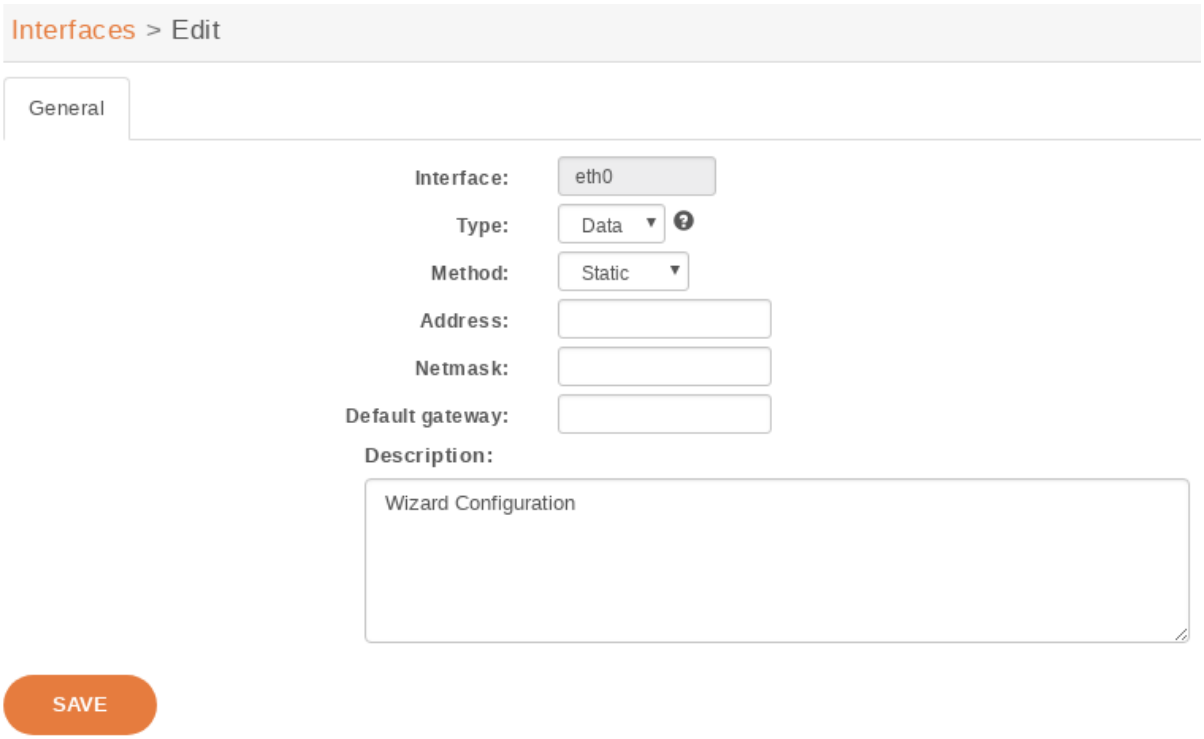


Fig. 3: *Configuration → Network → Interfaces → eth1 → Add*

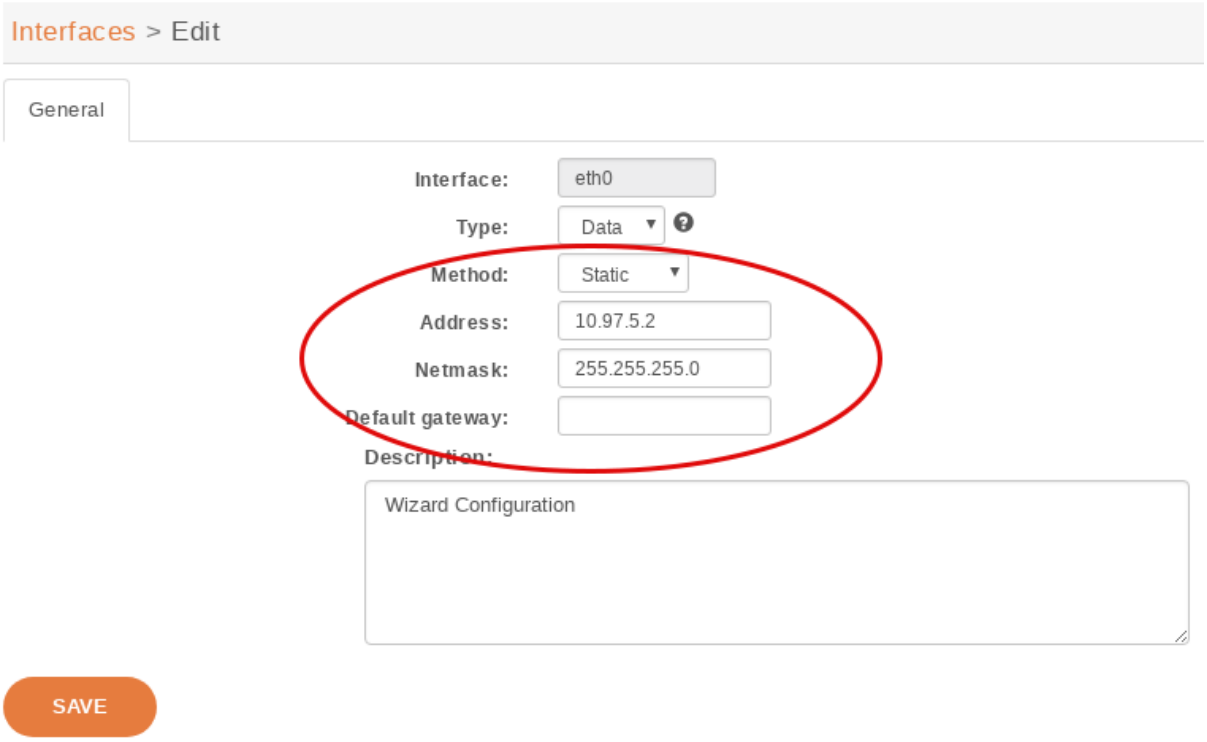


Fig. 4: *Configuration → Network → Interfaces → eth1 → Add*

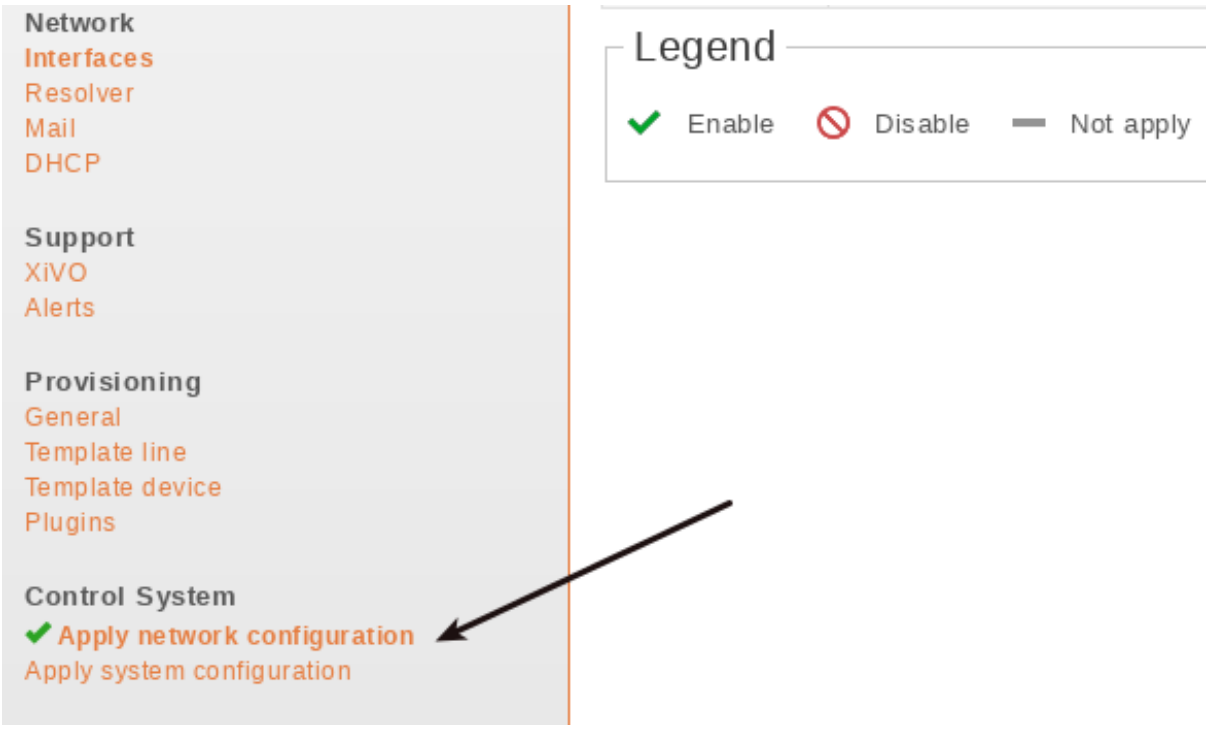


Fig. 5: Apply after modify interface

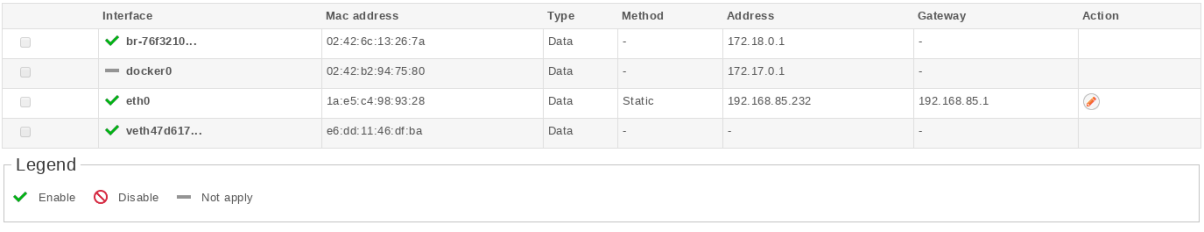


Fig. 6: *Configuration → Network → Interfaces*

To add and configure a new VLAN interface, we click on the small plus button in the top right corner,

Fig. 7: *Configuration → Network → Interfaces → Add button*

and we get to this page:

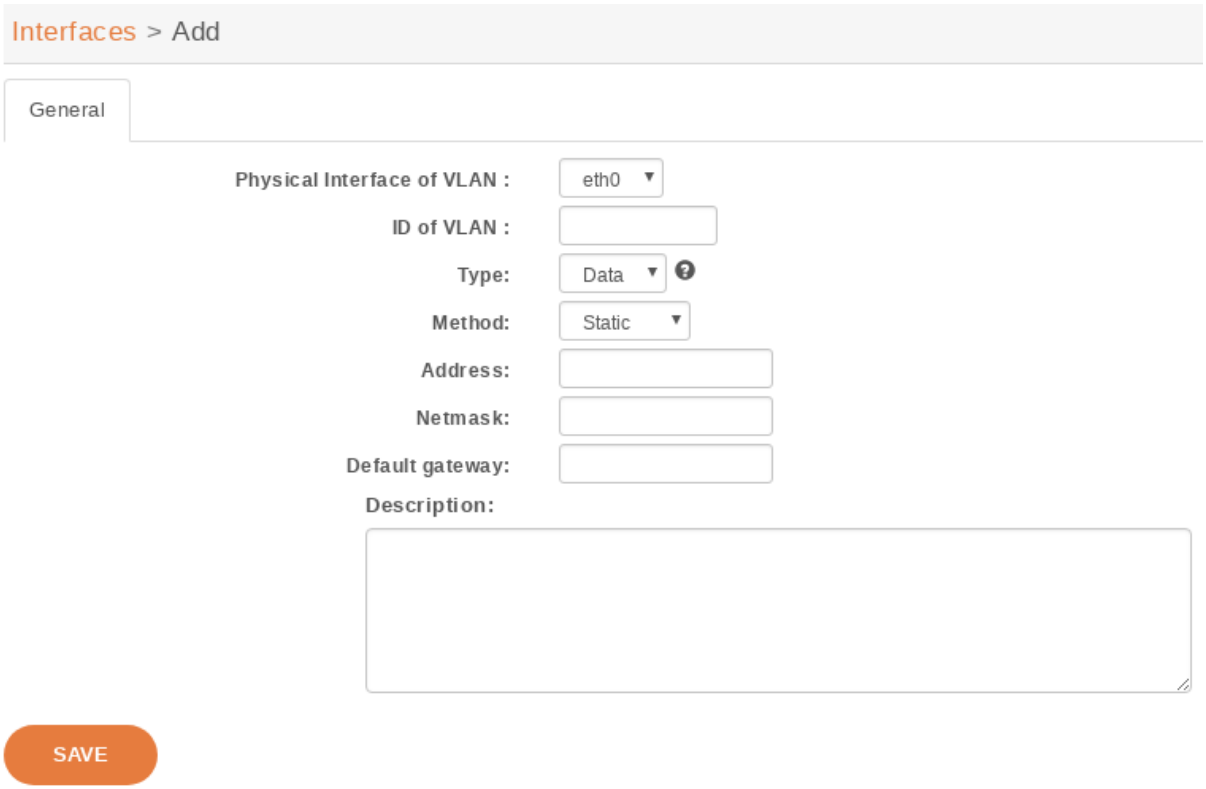


Fig. 8: *Configuration → Network → Interfaces → Add*

In our case, since we want to configure this interface with static information:

Click on **Save** list the network interfaces:

- The new virtual interface has been successfully created.

Note: Do not forget after you finish the configuration of the network to apply it with the button: **Apply network configuration**

After applying the network configuration:

Add static network routes

Static routes cannot be added via the web interface. However, you may add static routes to your XiVO by following following the steps described below. This procedure will ensure that your static routes are applied at startup (i.e. each time the network interface goes up).

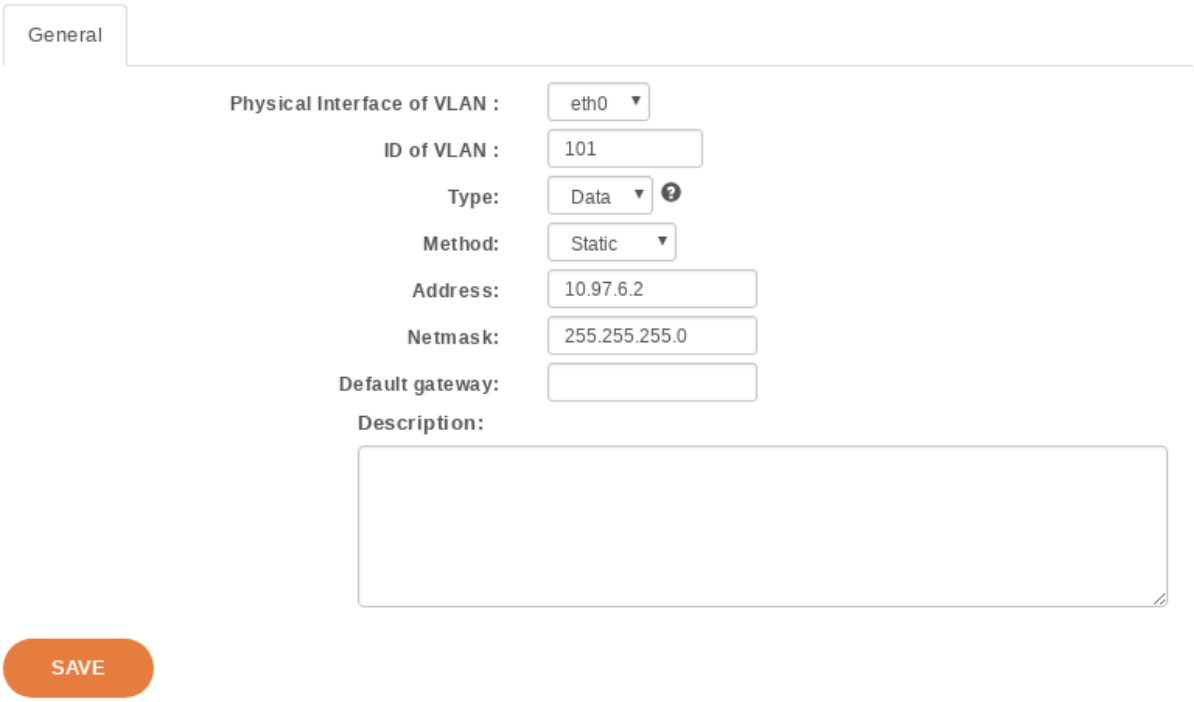


Fig. 9: *Configuration → Network → Interfaces → Add*

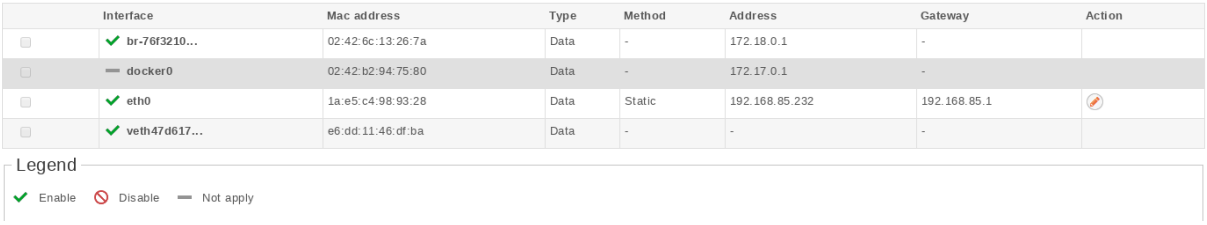


Fig. 10: *Configuration → Network → Interfaces*

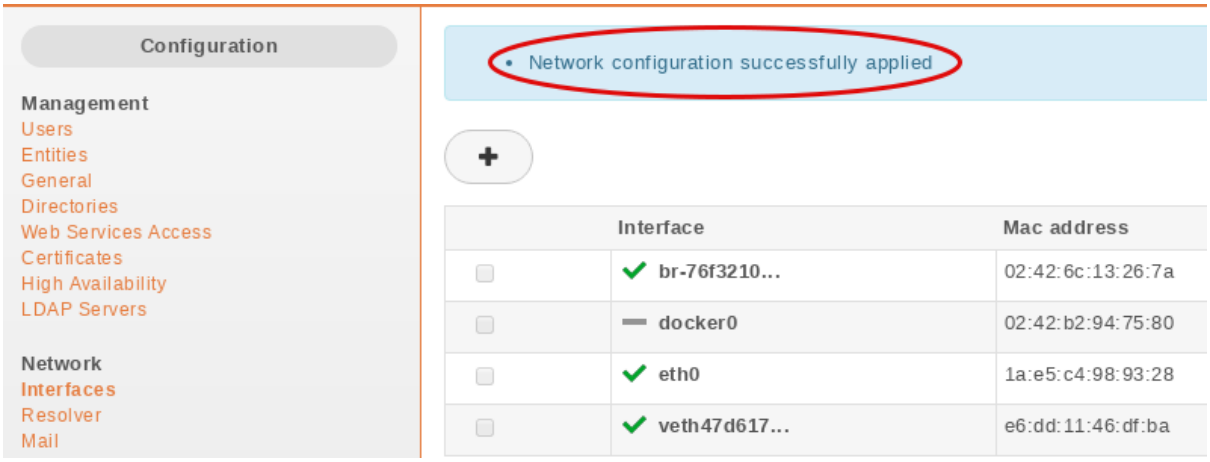


Fig. 11: Network configuration successfully apply

1. Create the file `/etc/network/if-up.d/xivo-routes`:

```
touch /etc/network/if-up.d/xivo-routes
chmod 755 /etc/network/if-up.d/xivo-routes
```

2. Insert the following content:

```
#!/bin/sh

if [ "${IFACE}" = "<network interface>" ]; then
    ip route add <destination> via <gateway>
    ip route add <destination> via <gateway>
fi
```

3. Fields <network interface>, <destination> and <gateway> should be replaced by your specific configuration. For example, if you want to add a route for 192.168.50.128/25 via 192.168.17.254 which should be added when eth0 goes up:

```
#!/bin/sh

if [ "${IFACE}" = "eth0.2" ]; then
    ip route add 192.168.50.128/25 via 192.168.17.254
fi
```

Note: The above check is to ensure that the route will be applied only if the correct interface goes up. This check should contain the actual name of the interface (i.e. *eth0* or *eth0.2* or *eth1* or ...). Otherwise the route won't be set up in every cases.

Change interface MTU

Warning: Manually changing the MTU is risky. Please only proceed if you are aware of what you are doing.

These steps describe how to change the MTU:

```
#. Create the file :file:`/etc/network/if-up.d/xivo-mtu`::
```

touch /etc/network/if-up.d/xivo-mtu chmod 755 /etc/network/if-up.d/xivo-mtu

1. Insert the following content:

```
#!/bin/sh

# Set MTU per iface
if [ "${IFACE}" = "<data interface>" ]; then
    ip link set ${IFACE} mtu <data mtu>
elif [ "${IFACE}" = "<voip interface>" ]; then
    ip link set ${IFACE} mtu <voip mtu>
fi
```

2. Change the <*data interface*> to the name of your interface (e.g. eth0), and the <*data mtu*> to the new MTU (e.g. 1492),
3. Change the <*voip interface*> to the name of your interface (e.g. eth1), and the <*voip mtu*> to the new MTU (e.g. 1488)

Note: In the above example you can set a different MTU per interface. If you don't need a per-interface MTU you can simply write:

```
#!/bin/sh

ip link set ${IFACE} mtu <my mtu>
```

Backup

Periodic backup

A backup of the database and the data are launched every day with a logrotate task. It is run at 06:25 a.m. and backups are kept for 7 days.

Logrotate task:

```
/etc/logrotate.d/xivo-backup
```

Logrotate cron:

```
/etc/cron.daily/logrotate
```

Retrieve the backup

You can retrieve the backup from the web-interface in *Services* → *IPBX* → *IPBX Configuration* → *Backup Files* page.

Otherwise, with shell access, you can retrieve them in `/var/backups/xivo`. In this directory you will find `db.tgz` and `data.tgz` files for the database and data backups.

Backup scripts:

```
/usr/sbin/xivo-backup
```

Backup location:

```
/var/backups/xivo
```

What is actually backed-up?

Data

Here is the list of folders and files that are backed-up:

- `/etc/asterisk/`
- `/etc/consul/`
- `/etc/crontab`
- `/etc/dahdi/`
- `/etc/dhcp/` This will overwrite the network configuration when the backup is restored
- `/etc/hostname` This will overwrite the network configuration when the backup is restored
- `/etc/hosts` This will overwrite the network configuration when the backup is restored
- `/etc/ldap/`
- `/etc/network/if-up.d/xivo-routes`
- `/etc/network/interfaces` This includes the host IP address / netmask and will overwrite the network configuration when the backup is restored
- `/etc/ntp.conf`

- /etc/profile.d/xivo_uuid.sh
- /etc/resolv.conf This will overwrite the network configuration when the backup is restored
- /etc/ssl/
- /etc/systemd/
- /etc/wanpipe/
- /etc/xivo-agentd/
- /etc/xivo-agid/
- /etc/xivo-amid/
- /etc/xivo-auth/
- /etc/xivo-call-logd/
- /etc/xivo-confd/
- /etc/xivo-confgend-client/
- /etc/xivo-ctid/
- /etc/xivo-dird/
- /etc/xivo-dird-phoned/
- /etc/xivo-dxtora/
- /etc/xivo-purge-db/
- /etc/xivo/
- /etc/xivo-xuc.conf
- /usr/local/bin/
- /usr/local/sbin/
- /usr/share/xivo/XIVO-VERSION
- /var/lib/asterisk/
- /var/lib/consul/
- /var/lib/xivo-provd/
- /var/lib/xivo/
- /var/log/asterisk/
- /var/spool/asterisk/
- /var/spool/cron/crontabs/
- /etc/docker/
- /etc/fail2ban/

The following files/folders are excluded from this backup:

- folders:
  - /var/lib/consul/checks
  - /var/lib/consul/raft
  - /var/lib/consul/serf
  - /var/lib/consul/services
  - /var/lib/xivo-provd/plugins/*/var/cache/*
  - /var/spool/asterisk/monitor/

  - /var/spool/asterisk/meetme/
- files
  - /var/lib/xivo-provd/plugins/xivo-polycom*/var/tftpboot/*.ld
- log files, coredump files
- audio recordings
- and, files greater than 10 MiB or folders containing more than 100 files if they belong to one of these folders:
  - /var/lib/xivo/sounds/
  - /var/lib/asterisk/sounds/custom/
  - /var/lib/asterisk/moh/
  - /var/spool/asterisk/voicemail/
  - /var/spool/asterisk/monitor/

Database

The database asterisk from PostgreSQL is backed up. This include almost everything that is configured via the web interface.

Creating backup files manually

Warning: A backup file may take a lot of space on the disk. You should check the free space on the partition before creating one.

Database

You can manually create a *database* backup file named db-manual.tgz in /var/tmp by issuing the following commands:

```
xivo-backup db /var/tmp/db-manual
```

Files

You can manually create a *data* backup file named data-manual.tgz in /var/tmp by issuing the following commands:

```
xivo-backup data /var/tmp/data-manual
```

Restore

Introduction

A backup of both the configuration files and the database used by a XiVO installation is done automatically every day. These backups are created in the /var/backups/xivo directory and are kept for 7 days.

Important Warning

A XiVO backup includes the entirety of the original machine's network configuration : it WILL overwrite any present network settings when you restore it. Remember to change those settings back if required before restarting network services or the machine itself, especially if you do not have physical or console access!

Limitations

- You must restore a backup on the **same version** of XiVO that was backed up (though the architecture – `i386` or `amd64` – may differ)
- You must restore a backup on a machine with the **same hostname and IP address**
- Be aware that this procedure applies **only to XiVO >= 14.08**
- XiVO CC configuration files are not backed up. Follow *Manual XiVO PBX configuration* to restore them.

Before Restoring the System

Warning: Before restoring a XiVO on a fresh install you have to setup XiVO using the wizard (see *Running the Wizard* section).

Stop monit and all the xivo services:

```
xivo-service stop
```

If you want to restore XiVO < 2017.06 that was configured for XiVO CC, you must create PostgreSQL user *stats* before restoring the database. See *Creating user stats*.

Restoring System Files

System files are stored in the data.tgz file located in the `/var/backups/xivo` directory.

This file contains for example, voicemail files, musics, voice guides, phone sets firmwares, provisioning server configuration database.

To restore the file

```
tar xvfp /var/backups/xivo/data.tgz -C /
```

Restoring the Database

Warning:

- This will destroy all the current data in your database.
- You have to check the free space on your system partition before extracting the backups.

Database backups are created as `db.tgz` files in the `/var/backups/xivo` directory. These tarballs contains a dump of the database used in XiVO.

In this example, we'll restore the database from a backup file named `db.tgz` placed in the home directory of root.

First, extract the content of the `db.tgz` file into the `/var/tmp` directory and go inside the newly created directory:

```
tar xvf db.tgz -C /var/tmp
cd /var/tmp/pg-backup
```

Drop the asterisk database and restore it with the one from the backup:

```
sudo -u postgres dropdb asterisk
sudo -u postgres pg_restore -C -d postgres asterisk-*.dump
```

To finalize the restore, see *After Restoring The System*.

Troubleshooting

When restoring the database, if you encounter problems related to the system locale, see *PostgreSQL localization errors*.

Alternative: Restoring and Keeping System Configuration

System configuration like network interfaces is stored in the database. It is possible to keep this configuration and only restore xivo data.

Rename the asterisk database to asterisk_previous:

```
sudo -u postgres psql -c 'ALTER DATABASE asterisk RENAME TO asterisk_previous'
```

Restore the asterisk database from the backup:

```
sudo -u postgres pg_restore -C -d postgres asterisk-*.dump
```

Restore the system configuration tables from the asterisk_previous database:

```
sudo -u postgres pg_dump -c -t dhcp -t netiface -t resolvconf asterisk_previous |␣
↪sudo -u postgres psql asterisk
```

Drop the asterisk_previous database:

```
sudo -u postgres dropdb asterisk_previous
```

Warning: Restoring the data.tgz file also restores system files such as host hostname, network interfaces, etc. You will need to reapply the network configuration if you restore the data.tgz file.

After Restoring The System

Resynchronize the xivo-auth keys:

```
xivo-update-keys
```

Update systemd runtime configuration:

```
source /etc/profile.d/xivo_uuid.sh
systemctl set-environment XIVO_UUID=$XIVO_UUID
systemctl daemon-reload
```

Restart the services you stopped in the first step:

```
xivo-service start
```

You may also reboot the system. Remember that the network settings were overwritten by the backed up settings, check and fix if necessary before rebooting!

CLI Tools

XiVO comes with a collection of console (CLI) tools to help administer the server.

xivo-dist

xivo-dist is the xivo repository sources manager. It is used to switch between distributions (production, development, release candidate, archived version). Example use cases :

- switch to production repository : `xivo-dist xivo-five`
- switch to development repository : `xivo-dist xivo-dev`
- switch to release candidate repository : `xivo-dist xivo-rc`
- switch to an archived version's repository (here 14.18) : `xivo-dist xivo-14.18`

HTTPS certificate

X.509 certificates are used to authorize and secure communications with the server. They are mainly used for HTTPS, but can also be used for SIPS, CTIS, WSS, etc.

There are two categories of certificates in XiVO:

- the default certificate, used for HTTPS in the web interface, REST APIs and WebSockets
- the certificates created and managed via the web interface

This article is about the former. For the latter, see *Telephony certificates*.

Default certificate

XiVO uses HTTPS where possible. The certificates are generated at install time (or during the upgrade to 15.12+). The main certificate is placed in `/usr/share/xivo-certs/server.crt`.

However, this certificate is self-signed, and HTTP clients (browser or REST API client) will complain about this default certificate because it is not signed by a trusted Certification Authority (CA).

The default certificate is untrusted

To make the HTTP client accept this certificate, you have two choices:

- configure your HTTP client to trust the self-signed XiVO certificate by adding a new trusted CA. The CA certificate (or bundle) is the file `/usr/share/xivo-certs/server.crt`.
- replace the self-signed certificate with your own trusted certificate.

Regenerating the default certificate

> **Warning:** If you use your own certificate, you should NOT replace it by the default certificate.

1. Regenerate the default certificate by this command:

```
openssl req -x509 -sha256 -nodes -days 3650 -newkey rsa:2048 \
  -config "/usr/share/xivo-config/x509/openssl-x509.conf" \
  -keyout "/usr/share/xivo-certs/server.key" \
  -out "/usr/share/xivo-certs/server.crt"
```

2. Change ownership and permissions:

```
chown root:www-data "/usr/share/xivo-certs/server.key" "/usr/share/xivo-certs/
↪server.crt"
chmod 640 "/usr/share/xivo-certs/server.key" "/usr/share/xivo-certs/server.crt"
```

3. Restart all XiVO services by running `xivo-service restart all`.

Use your own certificate

For this, follow these steps:

1. Replace the following files with your own private key/certificate pair:
   - Private key: `/usr/share/xivo-certs/server.key`
   - Certificate: `/usr/share/xivo-certs/server.crt`
2. Change the hostname of XiVO for each XiVO component: the different processes of XiVO heavily use HTTPS for internal communication, and for these connection to establish successfully, all hostnames used must match the Common Name (CN) of your certificate. Basically, you must replace all occurrences of `localhost` (the default hostname) with your CN in the *configuration of the XiVO services*. For example:

```
mkdir /etc/xivo/custom
cat > /etc/xivo/custom/custom-certificate.yml << EOF
consul:
  host: xivo.example.com
auth:
  host: xivo.example.com
confd:
  host: xivo.example.com
dird:
  host: xivo.example.com
ajam:
  host: xivo.example.com
agentd:
  host: xivo.example.com
EOF
for config_dir in /etc/xivo-*/conf.d/ ; do
    ln -s "/etc/xivo/custom/custom-certificate.yml" "$config_dir/010-custom-
↪certificate.yml"
done
```

3. Also, you must replace `localhost`, in the definition of your directories in the web interface under *Configuration* → *Directories*, by the hostname matching the CN of your certificate.
4. Then, when done, you must re-save, the CTI Directories definition:
   - Go to *Services* → *CTI Server* → *Directories* → *Definitions*
   - Edit each directory to re-select the new URI
   - And save it
5. If your certificate is not self-signed, and you obtained it from a third-party CA that is trusted by your system, you must enable the system-based certificate verification. By default, certificate verification is set to consider `/usr/share/xivo-certs/server.crt` as the only CA certificate.

First you need to install the debian `ca-certificates` package:

```
apt-get install ca-certificates
```

If one of the CA (or intermediate CA) of your certificate is not present in the CA shipped by the `ca-certificates` package you will need to add it manually:

- Create the following dir if not present:

  ```
  mkdir /usr/local/share/ca-certificates/
  ```

- Copy inside this directory the certificate of the missing CA in a `.crt` file
- And finally upload ca-certificates configuration:

  ```
  update-ca-certificates
  ```

Then to activate the certificat verification, the options are the following:

- Consul: `verify: True`
- Other XiVO services: `verify_certificate: True`

The procedure is the same as 2. with more configuration for each service. For example:

```
cat > /etc/xivo/custom/custom-certificate.yml << EOF
consul:
  host: xivo.example.com
  verify: True
auth:
  host: xivo.example.com
  verify_certificate: True
dird:
  host: xivo.example.com
  verify_certificate: True
...
```

Setting `verify_certificate` to `False` will disable the certificate verification, but the connection will still be encrypted. This is pretty safe as long as XiVO services stay on the same machine, however, this is dangerous when XiVO services are separated by an untrusted network, such as the Internet.

6. Ensure your CN resolves to a valid IP address with:
   - a DNS entry
   - and an entry in `/etc/hosts` resolving your CN to 127.0.0.1. Note that `/etc/hosts` will be rewritten occasionally by xivo-sysconfd. To make the change persistent, you can:
     1. modify `/usr/share/xivo-sysconfd/templates/resolvconf/hosts` instead (which will be rewritten when xivo-sysconfd is upgraded...)
     2. then add a script in `/usr/share/xivo-upgrade/pre-start.d` to re-apply the modification to `/usr/share/xivo-sysconfd/templates/resolvconf/hosts` after each `xivo-upgrade`.
7. Your X.509 certificate must have `subjectAltName` defined. See the example at cacert.org or detailed documentation at ietf.org.
8. Restart all XiVO services:

   ```
   xivo-service restart all
   ```

Configuration Files

This section describes some of the XiVO configuration files.

Configuration priority

Usually, the configuration is read from two locations: a configuration file `config.yml` and a configuration directory `conf.d`.

Files in the `conf.d` extra configuration directory:

- are used in alphabetical order and the first one has priority
- are ignored when their name starts with a dot
- are ignored when their name does not end with `.yml`

For example:

`.01-critical.yml`:

```
log_level: critical
```

`02-error.yml.dpkg-old`:

```
log_level: error
```

`10-debug.yml`:

```
log_level: debug
```

`20-nodebug.yml`:

```
log_level: info
```

The value that will be used for `log_level` will be `debug` since:

- `10-debug.yml` comes before `20-nodebug.yml` in the alphabetical order.
- `.01-critical.yml` starts with a dot so is ignored
- `02-error.yml.dpkg-old` does not end with `.yml` so is ignored

File configuration structure

Configuration files for every service running on a XiVO server will respect these rules:

- Default configuration directory in `/etc/xivo-{service}/conf.d` (e.g. `/etc/xivo-agentd/conf.d/`)
- Default configuration file in `/etc/xivo-{service}/config.yml` (e.g. `/etc/xivo-agentd/config.yml`)

The files `/etc/xivo-{service}/config.yml` should not be modified because **they will be overridden during upgrades**. However, they may be used as examples for creating additional configuration files as long as they respect the *Configuration priority*. Any exceptions to these rules are documented below.

xivo-agentd

- Default configuration directory: `/etc/xivo-agentd/conf.d`
- Default configuration file: `/etc/xivo-agentd/config.yml`

xivo-amid

- Default configuration directory: `/etc/xivo-amid/conf.d`
- Default configuration file: `/etc/xivo-amid/config.yml`

xivo-auth

- Default configuration directory: `/etc/xivo-auth/conf.d`
- Default configuration file: `/etc/xivo-auth/config.yml`

xivo-confgend

- Default configuration directory: `/etc/xivo-confgend/conf.d`
- Default configuration file: `/etc/xivo-confgend/config.yml`
- Default templates directory: `/etc/xivo-confgend/templates`

xivo-ctid

- Default configuration directory: `/etc/xivo-ctid/conf.d`
- Default configuration file: `/etc/xivo-ctid/config.yml`

xivo-dao

- Default configuration directory: `/etc/xivo-dao/conf.d`
- Default configuration file: `/etc/xivo-dao/config.yml`

This configuration is read by many XiVO programs in order to connect to the Postgres database of XiVO.

xivo-dird-phoned

- Default configuration directory: `/etc/xivo-dird-phoned/conf.d`
- Default configuration file: `/etc/xivo-dird-phoned/config.yml`

xivo_ring.conf

- Path: `/etc/xivo/asterisk/xivo_ring.conf`
- Purpose: This file can be used to change the ringtone played by the phone depending on the origin of the call.

Warning: Note that this feature has not been tested for all phones and all call flows. This page describes how you can customize this file but does not intend to list all validated call flows or phones.

This file `xivo_ring.conf` consists of :

- profiles of configuration (some examples for different brands are already included: `[aastra]`, `[snom]` etc.)
- one section named `[number]` where you apply the profile to an extension or a context etc.

Here is the process you should follow if you want to use/customize this feature :

1. Create a new profile, e.g.:

```
[myprofile-aastra]
```

2. Change the `phonetype` accordingly, in our example:

```
[myprofile-aastra]
phonetype = aastra
```

3. Chose the ringtone for the different type of calls (note that the ringtone names are brand-specific):

```
[myprofile-aastra]
phonetype = aastra
intern = <Bellcore-dr1>
group = <Bellcore-dr2>
```

4. Apply your profile, in the section `[number]`

- to a given list of extensions (e.g. 1001 and 1002):

```
1001@default = myprofile-aastra
1002@default = myprofile-aastra
```

- or to a whole context (e.g. default):

```
@default = myprofile-aastra
```

5. Restart `xivo-agid` service:

```
service xivo-agid restart
```

ipbx.ini

- Path: `/etc/xivo/web-interface/ipbx.ini`
- Purpose: This file specifies various configuration options and paths related to Asterisk and used by the web interface.

Here is a partial glimpse of what can be configured in file `ipbx.ini` :

1. Enable/Disable modification of SIP line username and password:

```
[user]
readonly-idpwd = "true"
```

When editing a SIP line, the username and password fields cannot be modified via the web interface. Set this option to false to enable the modification of both fields. This option is set to "true" by default.

Warning: This feature is not fully tested. It should be used only when absolutely necessary and with great care.

Consul

The default consul installation in XiVO uses the configuration file in `/etc/consul/xivo/*.json`. All files in this directory are installed with the package and *should not* be modified by the administrator. To use a different configuration, the adminstrator can add it's own configuration file at another location and set the new configuration directory by creating a systemd unit drop-in file in the `/etc/systemd/system/consul.service.d` directory.

The default installation generates a master token that can be retrieved in `/var/lib/consul/master_token`. This master token will not be used if a new configuration is supplied.

Variables

The following environment variables can be overridden in a systemd unit drop-in file:

- `CONFIG_DIR`: the consul configuration directory
- `WAIT_FOR_LEADER`: should the "start" action wait for a leader ?

Example, in `/etc/systemd/system/consul.service.d/custom.conf`:

```
[Service]
Environment=CONFIG_DIR=/etc/consul/agent
Environment=WAIT_FOR_LEADER=no
```

Agent mode

It is possible to run consul on another host and have the local consul node run as an agent only.

To get this kind of setup up and running, you will need to follow the following steps.

Downloading Consul

For a 32 bits system

```
wget --no-check-certificate https://releases.hashicorp.com/consul/0.5.2/consul_0.5.
↪2_linux_386.zip
```

For a 64 bits system

```
wget --no-check-certificate https://releases.hashicorp.com/consul/0.5.2/consul_0.5.
↪2_linux_amd64.zip
```

Installing Consul on a new host

```
unzip consul_0.5.2_linux_386.zip
```

Or

```
unzip consul_0.5.2_linux_amd64.zip
```

```
mv consul /usr/bin/consul
mkdir -p /etc/consul/xivo
mkdir -p /var/lib/consul
adduser --quiet --system --group --no-create-home \
        --home /var/lib/consul consul
```

Copying the consul configuration from the XiVO to a new host

On the new consul host, modify `/etc/consul/xivo/config.json` to include to following lines.

```
"bind_addr": "0.0.0.0",
"client_addr": "0.0.0.0",
"advertise_addr": "<consul-host>"
```

```
# on the consul host
scp root@<xivo-host>:/lib/systemd/system/consul.service /lib/systemd/system
systemctl daemon-reload
scp -r root@<xivo-host>:/etc/consul /etc
scp -r root@<xivo-host>:/usr/share/xivo-certs /usr/share
consul agent -data-dir /var/lib/consul -config-dir /etc/consul/xivo/
```

Note: To start consul with the systemd unit file, you may need to change owner and group (consul:consul) for all files inside `/etc/consul`, `/usr/share/xivo-certs` and `/var/lib/consul`

Adding the agent configuration

Create the file `/etc/consul/agent/config.json` with the following content

```
{
    "acl_datacenter": "<node_name>",
    "datacenter": "xivo",
    "server": false,
    "bind_addr": "0.0.0.0",
    "advertise_addr": "<xivo_address>",
    "client_addr": "127.0.0.1",
    "bootstrap": false,
    "rejoin_after_leave": true,
    "data_dir": "/var/lib/consul",
    "enable_syslog": true,
    "disable_update_check": true,
    "log_level": "INFO",
    "ports": {
        "dns": -1,
        "http": -1,
        "https": 8500
    },
    "retry_join": [
        "<remote_host>"
    ],
    "cert_file": "/usr/share/xivo-certs/server.crt",
    "key_file": "/usr/share/xivo-certs/server.key"
}
```

- `node_name`: Arbitrary name to give this node, `xivo-paris` for example.
- `remote_host`: IP address of your new consul. Be sure the host is accessible from your XiVO and check the firewall. See the documentation *here*.
- `xivo_address`: IP address of your xivo.

This file should be owned by consul user.

```
chown -R consul:consul /etc/consul/agent
```

Enabling the agent configuration

Add or modify `/etc/systemd/system/consul.service.d/custom.conf` to include the following lines:

```
[Service]
Environment=CONFIG_DIR=/etc/consul/agent
```

Restart your consul server.

```
service consul restart
```

Updating the consul section of xivo-ctid

Add a file in `/etc/xivo-ctid/conf.d/remote_consul.yml` with the following content

```
rest_api:
  http:
    listen: 0.0.0.0

service_discovery:
  advertise_address: <xivo-ctid-host>
  check_url: http://<xivo-ctid-host>:9495/0.1/infos
```

- `xivo-ctid-host`: Hostname to reach xivo-ctid

Log Files

Every XiVO service has its own log file, placed in `/var/log`.

asterisk

The Asterisk log files are managed by logrotate.

It's configuration files `/etc/logrotate.d/asterisk` and `/etc/asterisk/logger.conf`

The message log level is enabled by default in `logger.conf` and contains notices, warnings and errors. The full log entry is commented in `logger.conf` and should only be enabled when verbose debugging is required. Using this option in production would produce VERY large log files.

- Files location: `/var/log/asterisk/*`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-agentd

- File location: `/var/log/xivo-agentd.log`
- Rotate configuration: `/etc/logrotate.d/xivo-agentd`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-agid

- File location: `/var/log/xivo-agid.log`
- Rotate configuration: `/etc/logrotate.d/xivo-agid`
- Number of archived files: 15

- Rotation frequence: Daily

xivo-amid

- File location: `/var/log/xivo-amid.log`
- Rotate configuration: `/etc/logrotate.d/xivo-amid`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-auth

- File location: `/var/log/xivo-auth.log`
- Rotate configuration: `/etc/logrotate.d/xivo-auth`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-call-logd

- File location: `/var/log/xivo-call-logd.log`
- Rotate configuration: `/etc/logrotate.d/xivo-call-logd`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-confd

- File location: `/var/log/xivo-confd.log`
- Rotate configuration: `/etc/logrotate.d/xivo-confd`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-confgend

The xivo-confgend daemon output is sent to the file specified with the `--logfile` parameter when launched with twistd.

The file location can be changed by customizing the xivo-confgend.service unit file.

- File location: `/var/log/xivo-confgend.log`
- Rotate configuration: `/etc/logrotate.d/xivo-confgend`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-ctid

- File location: `/var/log/xivo-ctid.log`
- Rotate configuration: `/etc/logrotate.d/xivo-ctid`
- Number of archived log files: 15
- Rotation frequence: Daily

xivo-dird

- File location: `/var/log/xivo-dird.log`
- Rotate configuration: `/etc/logrotate.d/xivo-dird`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-dird-phoned

- File location: `/var/log/xivo-dird-phoned.log`
- Rotate configuration: `/etc/logrotate.d/xivo-dird-phoned`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-dxtora

- File location: `/var/log/xivo-dxtora.log`
- Rotate configuration: `/etc/logrotate.d/xivo-dxtora`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-provd

- File location: `/var/log/xivo-provd.log`
- Rotate configuration: `/etc/logrotate.d/xivo-provd`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-purge-db

- File location: `/var/log/xivo-purge-db.log`
- Rotate configuration: `/etc/logrotate.d/xivo-purge-db`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-stat

- File location: `/var/log/xivo-stat.log`
- Rotate configuration: `/etc/logrotate.d/xivo-stat`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-sysconfd

- File location: `/var/log/xivo-sysconfd.log`
- Rotate configuration: `/etc/logrotate.d/xivo-sysconfd`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-upgrade

- File location: `/var/log/xivo-upgrade.log`
- Rotate configuration: `/etc/logrotate.d/xivo-upgrade`
- Number of archived files: 15
- Rotation frequence: Daily

xivo-web-interface

- File location: `/var/log/xivo-web-interface/*.log`
- Rotate configuration: `/etc/logrotate.d/xivo-web-interface`
- Number of archived files: 21
- Rotation frequence: Daily

Nginx

XiVO use nginx as a web server and reverse proxy.

In its default configuration, the nginx server listens on port TCP/80 and TCP/443 and allows these services to be used:

- web interface (xivo-web-interface)
- API documentation (xivo-swagger-doc)

Starting from XiVO 16.13, an administrator can easily modify the configuration to allow additional services to be used (e.g. xivo-auth or xivo-confd).

To do so, an administrator only has to create a symbolic link inside the `/etc/nginx/locations/http-enabled` directory to the corresponding file in the `/etc/nginx/locations/http-available` directory, and then reload nginx with `systemctl reload nginx`. A similar operation must be done for HTTPS.

For example, to enable all the available services:

```
ln -sf /etc/nginx/locations/http-available/* /etc/nginx/locations/http-enabled
ln -sf /etc/nginx/locations/https-available/* /etc/nginx/locations/https-enabled
systemctl reload nginx
```

To disable all the services other than the web interface:

```
rm /etc/nginx/locations/http-enabled/* /etc/nginx/locations/https-enabled/*
ln -s /etc/nginx/locations/http-available/xivo-web-interface /etc/nginx/locations/
↪http-enabled
ln -s /etc/nginx/locations/https-available/xivo-web-interface /etc/nginx/locations/
↪https-enabled
systemctl reload nginx
```

NTP

XiVO has a NTP server, that must be synchronized to a reference server. This can be a public one or customized for specific target networking architecture. XiVO's NTP server is used by default as NTP server for the devices time reference.

Usage

Show NTP service status:

```
service ntp status
```

Stop NTP service:

```
service ntp stop
```

Start NTP service:

```
service ntp start
```

Restart NTP service:

```
service ntp restart
```

Show NTP synchronization status:

```
ntpq -p
```

Configuring NTP service

1. Edit `/etc/ntp.conf`
2. Give your NTP reference servers:

```
server 192.168.0.1                          # LAN existing NTP Server
server 0.debian.pool.ntp.org iburst dynamic  # default in ntp.conf
server 1.debian.pool.ntp.org iburst dynamic  # default in ntp.conf
```

3. If no reference server to synchronize to, add this to synchronize locally:

```
server 127.127.1.0              # local clock (LCL)
fudge 127.127.1.0 stratum 10    # LCL is not very reliable
```

4. Restart NTP service

5. Check NTP synchronization status.

Warning: If #5 shows that NTP doesn't use NTP configuration in `/etc/ntp.conf`, maybe have you done a `dhclient` for one of your network interface and the dhcp server that gave the IP address also gave a NTP server address. Thus you might check if the file `/var/lib/ntp/ntp.conf.dhcp` exists, if yes, this is used for NTP configuration prior to `/etc/ntp.conf`. Remove it and restart NTP, check NTP synchronization status, then it should work.

Proxy Configuration

If you use XiVO behind an HTTP proxy, you must do a couple of manipulations for it to work correctly.

Warning: We do not recomend to use `http_proxy` environment variable. It may break some services. Instead you should configure the proxy on a per service basis as described below.

System Applications

Installation and upgrade operations use different tools for which the proxy must be configured if any.

apt

Important: This is needed because apt is used for installation and upgrade

Create the `/etc/apt/apt.conf.d/90proxy` file with the following content:

```
Acquire::http::Proxy "http://domain\username:password@proxyip:proxyport";
```

curl

Important: This is needed because `curl` is used during installation and upgrade

Create the `~/.curlrc` file with the following content:

```
proxy = http://proxyip:proxyport
proxy-user = "username:password"
```

docker

Important: This is needed because docker images will be downloaded during installation or upgrade

When upgrading or installing XiVO it will attempt to download docker images. For the proxy configuration, you need to create a systemd configuration file. See Docker documentation: https://docs.docker.com/config/daemon/systemd/#httphttps-proxy

wget

Important: This step is needed because this tool is used by xivo-upgrade script and install scripts

Create the `~/.wgetrc` file with the following content:

```
use_proxy=yes
http_proxy=http://username:password@proxyip:proxyport
```

XiVO Services

Several XiVO services needs also the proxy to be configured, if any.

dhcp-update

Important: This is needed if you use the DHCP server of the XiVO. Otherwise the DHCP configuration won't be correct. It must be set before the wizard is run.

Proxy information is set via the `/etc/xivo/dhcpd-update.conf` file.

Edit the file and look for the `[proxy]` section.

provd

Note: This is needed to download plugins

Proxy information is set via the *Configuration* → *Provisioning* → *General* page.

xivo-fetchfw

Note: This is needed to download firmwares

Proxy information is set via the `/etc/xivo/xivo-fetchfw.conf` file.

Edit the file and look for the `[proxy]` section.

Service Discovery

Overview

XiVO uses consul for service discovery. When a daemon is started, it registers itself on the configured consul node.

Consul template may be used to generate the configuration files for each daemons that requires the availability of another service. Consul template can also be used to reload the appropriate service.

Service Authentication

XiVO services expose more and more resources through REST API, but they also ensure that the access is restricted to the authorized programs. For this, we use an *authentication daemon* who delivers authorizations via tokens.

Call flow

Here is the call flow to access a REST resource of a XiVO service:

1. Create a username/password (also called service_id/service_key) with the right *ACLs*, via *Web Services Access*.
2. *Create a token* with these credentials and the backend *xivo-service*.
3. *Use this token* to access the REST resource defined by the *ACL*.

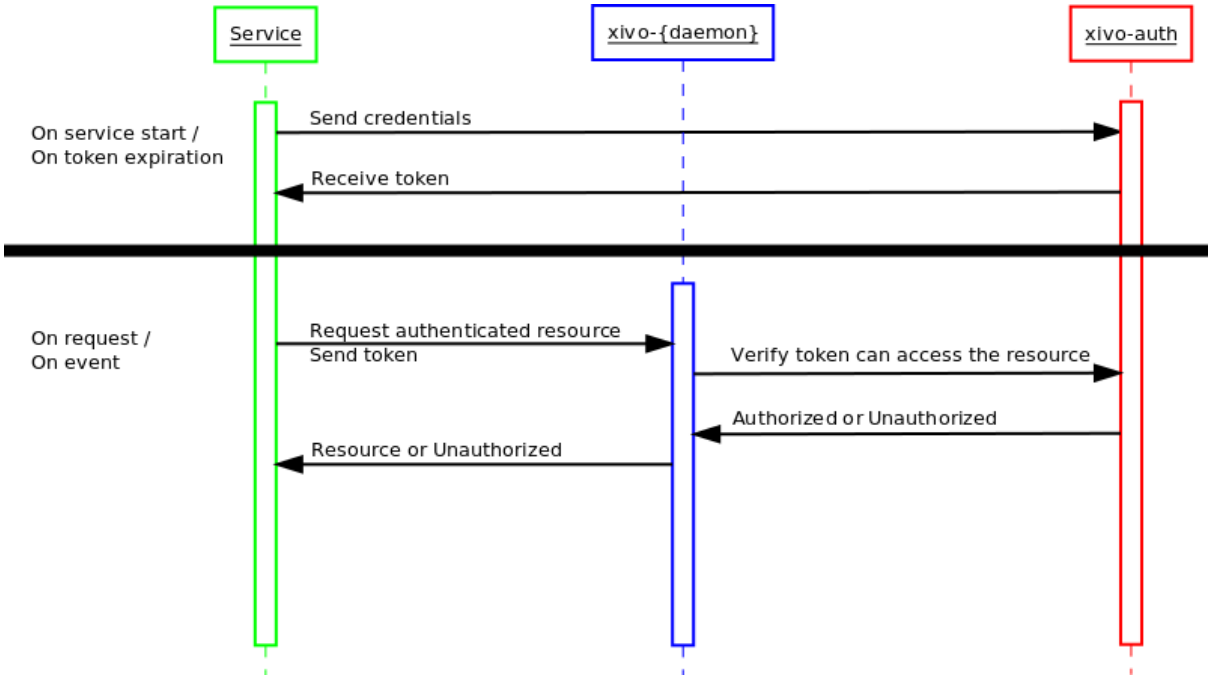


Fig. 12: Call flow of service authentication

Service Service who needs to access a REST resource.

xivo-{daemon} Server that exposes a REST resource. This resource must have an attached ACL.

xivo-auth Server that authenticates the *Service* and validates the required ACL with the token.

XiVO services directly use this system to communicate with each other, as you can see in their Web Services Access.

xivo-auth

xivo-auth is a scalable, extendable and configurable authentication service. It uses an HTTP interface to emit tokens to users who can then use those tokens to identify and authenticate themselves with other services compatible with xivo-auth.

xivo-auth HTTP API Changelog

2018.04

- Added static tokens for internal service authentication

16.02

- POST `/0.1/token`, field `expiration`: only integers are accepted, floats are now invalid.
- Experimental backend `ldap_user_voicemail` has been removed.
- New backend `ldap_user` has been added.

15.19

- POST `/0.1/token` do not accept anymore argument `backend_args`

15.17

- New backend `ldap_user_voicemail` has been added. **WARNING** this backend is **EXPERIMENTAL**.

15.16

- HEAD and GET now take a new `scope` query string argument to check ACLs
- Backend interface method `get_acls` is now named `get_consul_acls`
- Backend interface method `get_acls` now returns a list of ACLs
- HEAD and GET can now return a `403` if an ACL access is denied

15.15

- POST `/0.1/token` accept new argument `backend_args`
- Signature of backend method `get_ids()` has a new argument `args`
- New method `get_acls` for backend has been added
- New backend `service` has been added

xivo-auth Developer's Guide

Architecture

xivo-auth contains 4 major components, an HTTP interface, a celery worker, authentication backends and a consul client. All operations are made through the HTTP interface, tokens are stored by consul as well as the persistence for some of the data attached to tokens. The celery worker is used to schedule tasks that outlive the lifetime of the xivo-auth process. Backends are used to test if a supplied username/password combination is valid and provide the xivo-user-uuid.

xivo-auth is made of the following modules and packages.

plugins

the plugin package contains the xivo-auth backends that are packaged with xivo-auth.

http

The http module is the implementation of the HTTP interface.

- Validate parameters
- Calls the backend the check the user authentication
- Forward instructions to the *token_manager*
- Handle exceptions and return the appropriate status_code

controller

The controller is the plumbin of xivo-auth, it has no business logic.

- Start the HTTP application
- Start the celery worker
- Load all enabled plugins
- Instanciate the token_manager

token

The token modules contains the business logic of xivo-auth.

- Creates and delete tokens
- Creates ACLs for XiVO
- Schedule token expiration
- Read/write token data to consul

tasks

The tasks module contains implementation of celery tasks that are executed by the worker.

- Called by the celery worker
- Forwards instructions to the *token manager*

extension

This is a place holder for a global variable for the celery app. It will be removed and should not be used.

Other modules that should not need documentation are *helpers*, *config*, *interfaces*

Plugins

xivo-auth is meant to be easy to extend. This section describes how to add features to xivo-auth.

Backends

xivo-auth allows its administrator to configure one or many sources of authentication. Implementing a new kind of authentication is quite simple.

1. Create a python module implementing the backend interface.
2. Install the python module with an entry point *xivo_auth.backends*

An example backend implementation is available here.

Internal services

To simplify authentication of internal services without renewing the Token we have added a command line tool *xivo-auth-static-token-manager* to manage static tokens. Currently you can get a static token, which is created automatically during the installation. The same tool can be used to create a new one if needed. The static token is created with default acl: *confd.users.read*.

Stock Plugins Documentation

Backends Plugins

XiVO Admin

Backend name: `xivo_admin`

Purpose: Authenticate a XiVO administrator. The login/password is configured in *Configuration* → *Management* → *Users*.

XiVO Service

Backend name: `xivo_service`

Purpose: Authenticate a XiVO *Web Services Access*. The login/password is configured in *Configuration* → *Management* → *Web Service Access*.

XiVO User

Backend name: `xivo_user`

Purpose: Authenticate a XiVO user. The login/password is configured in *IPBX* → *Services* → *PBX Settings* → *Users* in the CTI client section.

LDAP

Backend name: `ldap_user`

Purpose: Authenticate with an LDAP user.

For example, with the given configuration:

```
ldap:
    uri: ldap://example.org
    bind_dn: cn=xivo,dc=example,dc=org
    bind_password: bindpass
    user_base_dn: ou=people,dc=example,dc=org
```

(continues on next page)

(continued from previous page)

```
    user_login_attribute: uid
    user_email_attribute: mail
```

When an authentication request is received for username `alice` and password `userpass`, the backend will:

1. Connect to the LDAP server at example.org
2. Do an LDAP "bind" operation with bind DN `cn=xivo,dc=example,dc=org` and password `bindpass`
3. Do an LDAP "search" operation to find an LDAP user matching `alice`, using:
   - the base DN `ou=people,dc=example,dc=org`
   - the filter `(uid=alice)`
   - a SUBTREE scope
4. If the search returns exactly 1 LDAP user, do an LDAP "bind" operation with the user's DN and the password `userpass`
5. If the LDAP "bind" operation is successful, search in XiVO a user with an email matching the `mail` attribute of the LDAP user
6. If a XiVO user is found, success

To use an anonymous bind instead, the following configuration would be used:

```
ldap:
    uri: ldap://example.org
    bind_anonymous: True
    user_base_dn: ou=people,dc=example,dc=org
    user_login_attribute: uid
    user_email_attribute: mail
```

The backend can also works in a "no search" mode, for example with the following configuration:

```
ldap:
    uri: ldap://example.org
    user_base_dn: ou=people,dc=example,dc=org
    user_login_attribute: uid
    user_email_attribute: mail
```

When the server receives the same authentication request as above, it will directly do an LDAP "bind" operation with the DN `uid=alice,ou=people,dc=example,dc=org` and password `userpass`, and continue at step 5.

Note: User's email and voicemail's email are two separate things. This plugin only use the user's email.

Configuration

`uri` the URI of the LDAP server. Can only contain the scheme, host and port of an LDAP URL.

`user_base_dn` the base dn of the user

`user_login_attribute` the attribute to login a user

`user_email_attribute` (optional) the attribute to match with the XiVO user's email (default: mail)

`bind_dn` (optional) the bind DN for searching for the user DN.

`bind_password` (optional) the bind password for searching for the user DN.

`bind_anonymous` (optional) use anonymous bind for searching for the user DN (default: false)

Usage

xivo-auth is used through HTTP requests, using HTTPS. Its default port is 9497. As a user, the most common operation is to get a new token. This is done with the POST method.

Alice retrieves a token using her username/password:

```
$ # Alice creates a new token, using the xivo_user backend, expiring in 10 minutes
$ curl -k -X POST -H 'Content-Type: application/json' -u 'alice:s3cre7' "https://
↪localhost:9497/0.1/token" -d '{"backend": "xivo_user", "expiration": 600}';echo
{"data": {"issued_at": "2015-06-05T10:16:58.557553", "token": "1823c1ee-6c6a-0cdc-
↪d869-964a7f08a744", "auth_id": "63f3dc3c-865d-419e-bec2-e18c4b118224", "xivo_
↪user_uuid": "63f3dc3c-865d-419e-bec2-e18c4b118224", "expires_at": "2015-06-
↪05T11:16:58.557595"}}
```

In this example Alice used here XiVO CTI client login `alice` and password `s3cre7`. The authentication source is determined by the *backend* in the POST data.

Alice could also have specified an expiration time on her POST request. The expiration value is the number of seconds before the token expires.

After retrieving her token, Alice can query other services that use xivo-auth and send her token to those service. Those services can then use this token on Alice's behalf to access her personal storage.

If Alice wants to revoke her token before its expiration:

```
$ curl -k -X DELETE -H 'Content-Type: application/json' "https://localhost:9497/0.
↪1/token/1823c1ee-6c6a-0cdc-d869-964a7f08a744"
```

See *Service Authentication* for details about the authentication process.

Usage for services using xivo-auth

A service that requires authentication and identification can use xivo-auth to externalise the burden of authentication. The new service can then accept a token as part of its operations to authenticate the user using the service.

Once a service receives a token from one of its user, it will need to check the validity of that token. There are 2 forms of verification, one that only checks if the token is valid and the other returns information about this token's session if it is valid.

Checking if a token is valid:

```
$ curl -k -i -X HEAD -H 'Content-Type: application/json' "https://localhost:9497/0.
↪1/token/1823c1ee-6c6a-0cdc-d869-964a7f08a744"
HTTP/1.1 204 NO CONTENT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Date: Fri, 05 Jun 2015 14:49:50 GMT
Server: pcm-dev-0

$ # get more information about this token
$ curl -k -X GET -H 'Content-Type: application/json' "https://localhost:9497/0.1/
↪token/1823c1ee-6c6a-0cdc-d869-964a7f08a744";echo
{"data": {"issued_at": "2015-06-05T10:16:58.557553", "token": "1823c1ee-6c6a-0cdc-
↪d869-964a7f08a744", "auth_id": "63f3dc3c-865d-419e-bec2-e18c4b118224", "xivo_
↪user_uuid": "63f3dc3c-865d-419e-bec2-e18c4b118224", "expires_at": "2015-06-
↪05T11:16:58.557595"}}
```

Launching xivo-auth

```
usage: xivo-auth [-h] [-c CONFIG_FILE] [-u USER] [-d] [-f] [-l LOG_LEVEL]

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILE, --config-file CONFIG_FILE
                        The path to the config file
  -u USER, --user USER  User to run the daemon
  -d, --debug           Log debug messages
  -f, --foreground      Foreground, don't daemonize
  -l LOG_LEVEL, --log-level LOG_LEVEL
                        Logs messages with LOG_LEVEL details. Must be one of:
                        critical, error, warning, info, debug. Default: None
```

HTTP API Reference

See also the *xivo-auth HTTP API Changelog*.

Development

See *xivo-auth Developer's Guide*.

xivo-confd

xivo-confd is a HTTP server that provides a RESTful API service for configuring and managing basic resources on a XiVO server.

Developer's Guide (xivo-confd)

xivo-confd resources are organised through a plugin mechanism. There are 2 main plugin categories:

Resource plugins A plugin that manages a resource (e.g. users, extensions, voicemails, etc). A resource plugin exposes the 4 basic CRUD operations (Create, Read, Update, Delete) in order to operate on a resource in a RESTful manner.

Association plugins A plugin for associating or dissociating 2 resources (e.g a user and a line). An association plugin exposes an HTTP action for associating (either `POST` or `PUT`) and another for dissociating (`DELETE`)

The following diagram outlines the most important parts of a plugin:

Resource Class that receives and handles HTTP requests. Resources use flask-restful for handling requests.

There are 2 kinds of resources: *ListResource* for root URLs and *ItemResource* for URLs that have an ID. *ListResource* will handle creating a resource (`POST`) and searching through a list of available resources (`GET`). *ItemResource* handles fetching a single item (`GET`), updating (`PUT`) and deleting (`DELETE`).

Service Class that handles business logic for a resource, such as what to do in order to get, create, update, or delete a resource. *Service* classes do not manipulate data directly. Instead, they coordinate what to do via other objects.

There are 2 kinds of services: *CRUDService* for basic CRUD operations in *Resource plugins*, and *AssociationService* for association/dissociation operations in *Association plugins*.

Dao Data Access Object. Knows how to get data and how to manipulate it, such as SQL queries, files, etc.

Notifier Sends events after an operation has completed. An event will be sent in a messaging queue for each CRUD operation. Certain resources also need to send events to other daemons in order to reload some configuration data. (i.e. asterisk needs to reload the dialplan when an extension is updated)

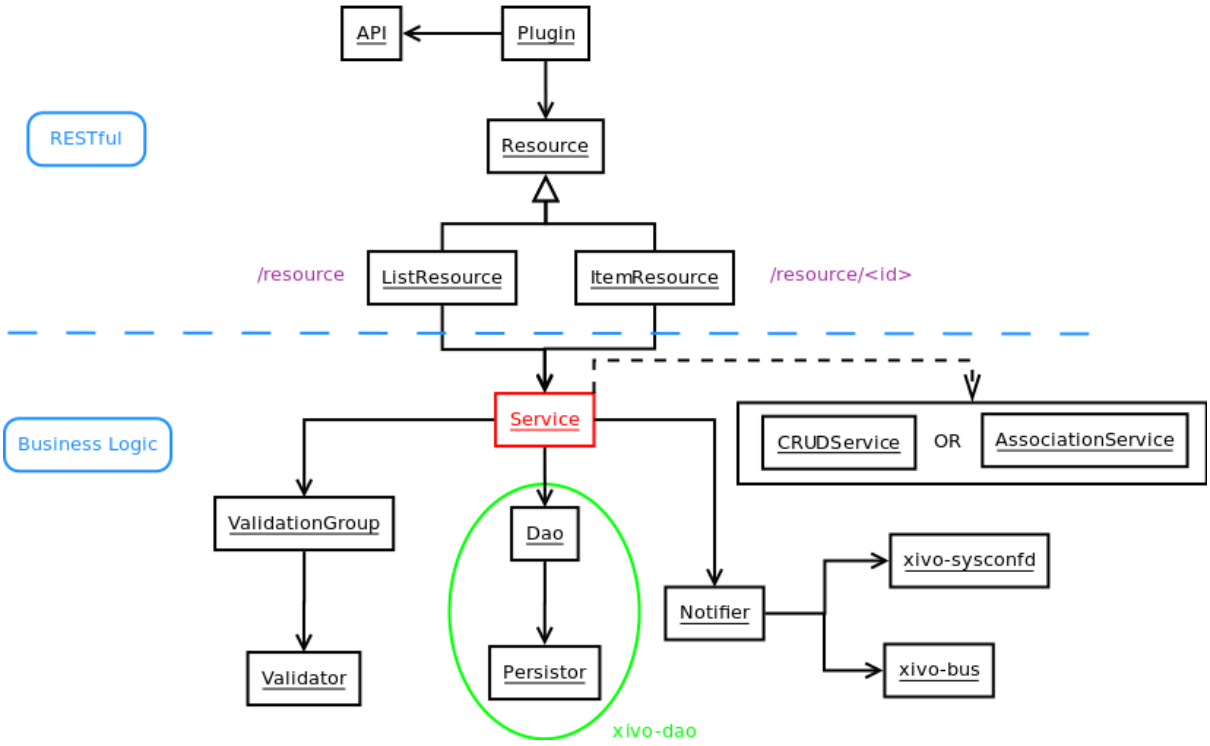


Fig. 13: Plugin architecture of xivo-confd

Validator Makes sure that a resource's data does not contain any errors before doing something with it. A *Validator* can be used for validating input data or business rules.

XiVO confgend

xivo-confgend is a configuration file generator. It is mainly used to generate the Asterisk configuration files.

XiVO confgend developer's guide

xivo-confgend uses drivers to implement the logic required to generate configuration files. It uses stevedore to do the driver instantiation and discovery.

Plugins in xivo-confgend use setuptools' entry points. That means that installing a new plugin to xivo-confgend requires an entry point in the plugin's *setup.py*.

Drivers

Driver plugin are classes that are used to generate the content of a configuration file.

The implementation of a plugin should have the following properties.

1. It's `__init__` method should take one argument
2. It should have a `generate` method which will return the content of the file
3. A setup.py adding an entry point

The `__init__` method argument is the content of the configuration of xivo-confgend. This allows the driver implementor to add values to the configuration in `/etc/xivo-confgend/conf.d/*.yml` and these values will be available in the driver.

The generate method has no argument, the configuration provided to the `__init__` should be sufficient for most cases. `generate` is called within a `scoped_session` of xivo-dao, allowing the usage of xivo-dao without prior setup in the driver.

The namespaces used for entry points in xivo-confgend have the following form:

> xivo_confgend.<resource>.<filename>

as an example, a generator for sip.conf would have the following namespace:

> xivo_confgend.asterisk.sip.conf

Example

Here is a typical `setup.py`:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Copyright 2016 by Avencall
# SPDX-License-Identifier: GPL-3.0+

from setuptools import setup
from setuptools import find_packages


setup(
    name='XiVO confgend driversample',
    version='0.0.1',

    description='An example driver',

    packages=find_packages(),

    entry_points={
        'xivo_confgend.asterisk.sip.conf': [
            'my_driver = src.driver:MyDriver',
        ],
    }
)
```

With the following package structure:

```
.
├── setup.py
└── src
    └── driver.py
```

`driver.py`:

```
# -*- coding: utf-8 -*-
# Copyright 2016 by Avencall
# SPDX-License-Identifier: GPL-3.0+


class MyDriver(object):

    def __init__(self, config):
        self._config = config

    def generate(self):
        return 'Hello World!'
```

To enable this plugin, you need to:

1. Install the plugin with:

```
python setup.py install
```

2. Create a config file in `/etc/xivo-confgend/conf.d`:

```
plugins:
  asterisk.sip.conf: my_driver
```

3. Restart xivo-confgend:

```
systemctl restart xivo-confgend
```

XiVO dird

xivo-dird is the directory server for XiVO. It offers a simple REST interface to query all directories that are configured. xivo-dird is extendable with plugins.

xivo-dird changelog

15.20

- The ldap plugins *ldap_network_timeout* default value has been incremented from 0.1 to 0.3 seconds

15.19

- Added the `voicemail` type in *Views* configuration
- Removed reverse endpoints in REST API:
  - GET `/0.1/directories/reverse/<profile>/me`

15.18

- Added reverse endpoints in REST API:
  - GET `/0.1/directories/reverse/<profile>/<xivo_user_uuid>`
  - GET `/0.1/directories/reverse/<profile>/me`

15.17

- Added directories endpoints in REST API:
  - GET `/0.1/directories/input/<profile>/aastra`
  - GET `/0.1/directories/lookup/<profile>/aastra`
  - GET `/0.1/directories/input/<profile>/polycom`
  - GET `/0.1/directories/lookup/<profile>/polycom`
  - GET `/0.1/directories/input/<profile>/snom`
  - GET `/0.1/directories/lookup/<profile>/snom`
  - GET `/0.1/directories/lookup/<profile>/thomson`
  - GET `/0.1/directories/lookup/<profile>/yealink`

15.16

- Added more cisco endpoints in REST API:
  - GET `/0.1/directories/input/<profile>/cisco`
- Endpoint `/0.1/directories/lookup/<profile>/cisco` accepts a new `limit` and `offset` query string arguments.

15.15

- Added cisco endpoints in REST API:
  - GET `/0.1/directories/menu/<profile>/cisco`
  - GET `/0.1/directories/lookup/<profile>/cisco`

15.14

- Added more personal contacts endpoints in REST API:
  - GET `/0.1/personal/<contact_id>`
  - PUT `/0.1/personal/<contact_id>`
  - POST `/0.1/personal/import`
  - DELETE `/0.1/personal`
- Endpoint `/0.1/personal` accepts a new `format` query string argument.

15.13

- Added personal contacts endpoints in REST API:
  - GET `/0.1/directories/personal/<profile>`
  - GET `/0.1/personal`
  - POST `/0.1/personal`
  - DELETE `/0.1/personal/<contact_id>`
- Signature of backend method `list()` has a new argument `args`
- Argument `args` for backend methods `list()` and `search()` has a new key `token_infos`
- Argument `args` for backend method `load()` has a new key `main_config`
- Methods `__call__()` and `lookup()` of service plugin `lookup` take a new `token_infos` argument

15.12

- Added authentication on all REST API endpoints
- Service plugins receive the whole configuration, rather than only their own section

XiVO dird configuration

There are three sources of configuration for xivo-dird:

- the *command line options*
- the main configuration file
- the sources configuration directory

The command-line options have priority over the main configuration file options.

Main Configuration File

Default location: `/etc/xivo-dird/config.yml`. Format: YAML

The default location may be overwritten by the command line options.

Here's an example of the main configuration file:

```
debug: False
foreground: False
log_filename: /var/log/xivo-dird.log
log_level: info
pid_filename: /var/run/xivo-dird/xivo-dird.pid
source_config_dir: /etc/xivo-dird/sources.d
user: www-data

rest_api:
    wsgi_socket: /var/run/xivo-dird/xivo-dird.sock

enabled_plugins:
   backends:
       - csv
       - ldap
       - phonebook
   services:
       - lookup
   views:
       - cisco_view
       - default_json

views:
    displays:
        switchboard_display:
            -
                title: Firstname
                default: Unknown
                field: firstname
                type: name
            -
                title: Lastname
                default: Unknown
                field: lastname
                type: name
        default_display:
            -
                title: Firstname
                field: fn
                type: name
            -
                title: Location
                default: Canada
```

(continues on next page)

(continued from previous page)

```
                field: country
            -
                title: Number
                field: number
                type: number
    displays_phone:
        default:
            name:
                - display_name
            number:
                -
                    field:
                        - phone
                -
                    field:
                        - phone_mobile
                    name_format: "{name} (Mobile)"
     profile_to_display:
         default: default_display
         switchboard: switchboard_display
     profile_to_display_phone:
         default: default

services:
    lookup:
        default:
            sources:
                - my_csv
                - ldap_quebec
            timeout: 0.5
        switchboard:
            sources:
                - my_csv
                - xivo_phonebook
                - ldap_quebec
            timeout: 1

sources:
    my_source:
        name: my_source
        type: ldap
        ldap_option1: value
        ldap_option2: value
        ...
```

Root section

debug Enable log debug messages. Overrides `log_level`. Default: `False`.

foreground Foreground, don't daemonize. Default: `False`.

log_filename File to write logs to. Default: `/var/log/xivo-dird.log`.

log_level Logs messages with LOG_LEVEL details. Must be one of: `critical`, `error`, `warning`, `info`, `debug`. Default: `info`.

pid_filename File used as lock to avoid multiple xivo-dird instances. Default: `/var/run/xivo-dird/xivo-dird.pid`.

source_config_dir The directory from which sources configuration are read. See *Sources Configuration*. Default: `/etc/xivo-dird/sources.d`.

user The owner of the process. Default: `www-data`.

enabled_plugins section

This sections controls which plugins are to be loaded at xivo-dird startup. All plugin types must have at least one plugin enabled, or xivo-dird will not start. For back-end plugins, sources using a back-end plugin that is not enabled will be ignored.

views section

displays A dictionary describing the content of each display. The key is the display's name, and the value are the display's content.

The display content is a list of fields. Each field is a dictionary with the following keys:

- title: The label of the field
- default: The default value of the field
- type: An arbitrary identifier of the field. May be used by consumers to identify the field without matching the label. For meaningful values inside XiVO, see *Integration of XiVO dird with the rest of XiVO*.
- field: the key of the data from the source that will be used for this field.

The display may be used by a plugin view to configure which fields are to be presented to the consumer.

displays_phone A dictionary describing the content of phone-related displays. Like `displays`, the key is the display's name and the value is the display's content. These displays are used by phone-related view plugins, like the `cisco_view` plugin.

The display content contains 2 keys, `name` and `number`.

The value of the `name` key is a list of source result fields. For a given source result, the first field that will return a non-empty value will be used as the display name on the phone. For example, if `name` is configured with `["display_name", "name"]` and you have a source result with fields `{"display_name": "", "name": "Bob"}`, then "Bob" will be displayed on the phone.

The value of the `number` key is a list of number item. Each item is composed of a dictionary containing at least a `field` key, and optionally a `name_format` key. For example, if you have the following number configuration:

```
name:
    - display_name
number:
    -
        field:
            - phone
    -
        field:
            - phone_mobile
        name_format: "{name} (Mobile)"
```

and you have a source result `{"display_name": "Bob", "phone": "101", "phone_mobile": "102"}`, then 2 results will be displayed on your phone:

1. "Bob", with number "101"
2. "Bob (Mobile)", with number "102"

The `name_format` value is a python format string. There's two substitution variables available, `{name}` and `{number}`.

profile_to_display A dictionary associating a profile to a display. It allows xivo-dird to use the right display when a consumer makes a query with a profile. The key is the profile name and the value is the display name.

profile_to_display_phone: A dictionary associating a profile to a phone display. This is similar to `profile_to_display`, but only used by phone-related view plugins.

services section

This section is a dictionary whose keys are the service plugin name and values are the configuration of that service. Hence the content of the value is dependent of the service plugin. See the documentation of the service plugin (*Stock Plugins Documentation*).

sources section

This section is a dictionary whose keys are the source name and values are the configuration for that source. See the *Sources Configuration* section for more details about source configuration.

Sources Configuration

There are two ways to configure sources:

- in the sources section of the main configuration
- in files of a directory, one file for each source:
  - Default directory location `/etc/xivo-dird/sources.d`
  - Files format: YAML
  - File names are ignored
  - Each file listed in this directory will be read and used to create a data source for xivo-dird.

Here is an example of a CSV source configuration in its own file:

```
type: csv
name: my_contacts_in_a_csv_file
file: /usr/local/share/my_contacts.csv
unique_column: id
searched_columns:
    - fn
    - ln
format_columns:
    name: "{fn} {ln}"
    number: "{num}"
```

This is strictly equivalent in the main configuration file:

```
sources:
    my_contacts_in_a_csv_file:
        type: csv
        name: my_contacts_in_a_csv_file
        file: /usr/local/share/my_contacts.csv
        unique_column: id
        searched_columns:
            - fn
            - ln
        source_to_display_columns:
            ln: lastname
            fn: firstname
            num: number
```

type the type of the source. It must be the same than the name of one of the enabled back-end plugins.

name is the name of this given configuration. The name is used to associate the source to profiles. The value is arbitrary, but it must be unique across all sources.

> **Warning:** Changing the name of the source will make all favorites in that source disappear. There is currently no tool to help you migrate favorites between source names, so choose your source names carefully.

The other options are dependent on the source type (the back-end used). See the documentation of the back-end plugin (*Stock Plugins Documentation*). However, the following keys should be present in all source configurations:

first_matched_columns (optional) the columns used for the reverse lookup. Any column having the search term will be a reverse lookup result.

format_columns (optional) a mapping between result fields and a format string. The new key will be added to the result, if this name already exists in the result, it will be replaced with the new value. The syntax is a python format string. See https://docs.python.org/2/library/string.html#formatspec for a complete reference.

searched_columns (optional) the columns used for the lookup. Any column containing the search term substring will be a lookup result.

unique_column (optional) This column is what makes an entry unique in this source. The `unique_column` is used to build the `uid` that is passed to the list method to fetch a list of results by unique ids. This is necessary for listing and identifying favorites.

XiVO dird developer's guide

The XiVO dird architecture uses plugins as extension points for most of its job. It uses stevedore to do the plugin instantiation and discovery and ABC classes to define the required interface.

Plugins in xivo-dird use setuptools' entry points. That means that installing a new plugin to xivo-dird requires an entry point in the plugin's setup.py. Each entry point's *namespace* is documented in the appropriate documentation section. These entry points allow xivo-dird to be able to discover and load extensions packaged with xivo-dird or installed separately.

Each kind of plugin does a specific job. There are three kinds of plugins in dird.

1. *Back-End*
2. *Service*
3. *View*

All plugins are instantiated by the core. The core then keeps a catalogue of loaded extensions that can be supplied to other extensions.

The following setup.py shows an example of a python library that add a plugin of each kind to xivo-dird:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from setuptools import setup
from setuptools import find_packages


setup(
    name='XiVO dird plugin sample',
    version='0.0.1',

    description='An example program',

    packages=find_packages(),
```

(continues on next page)

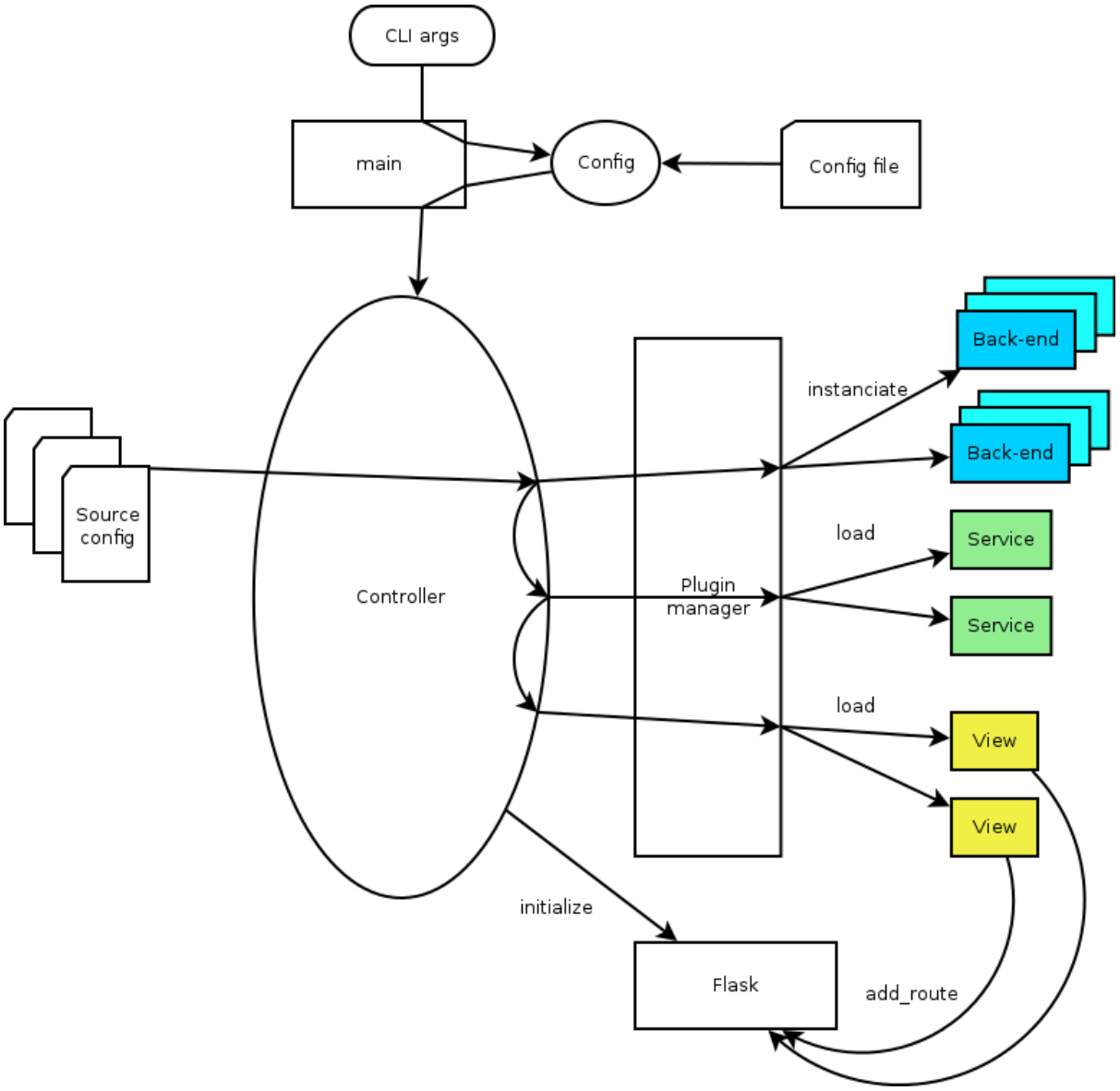


Fig. 14: xivo-dird startup flow

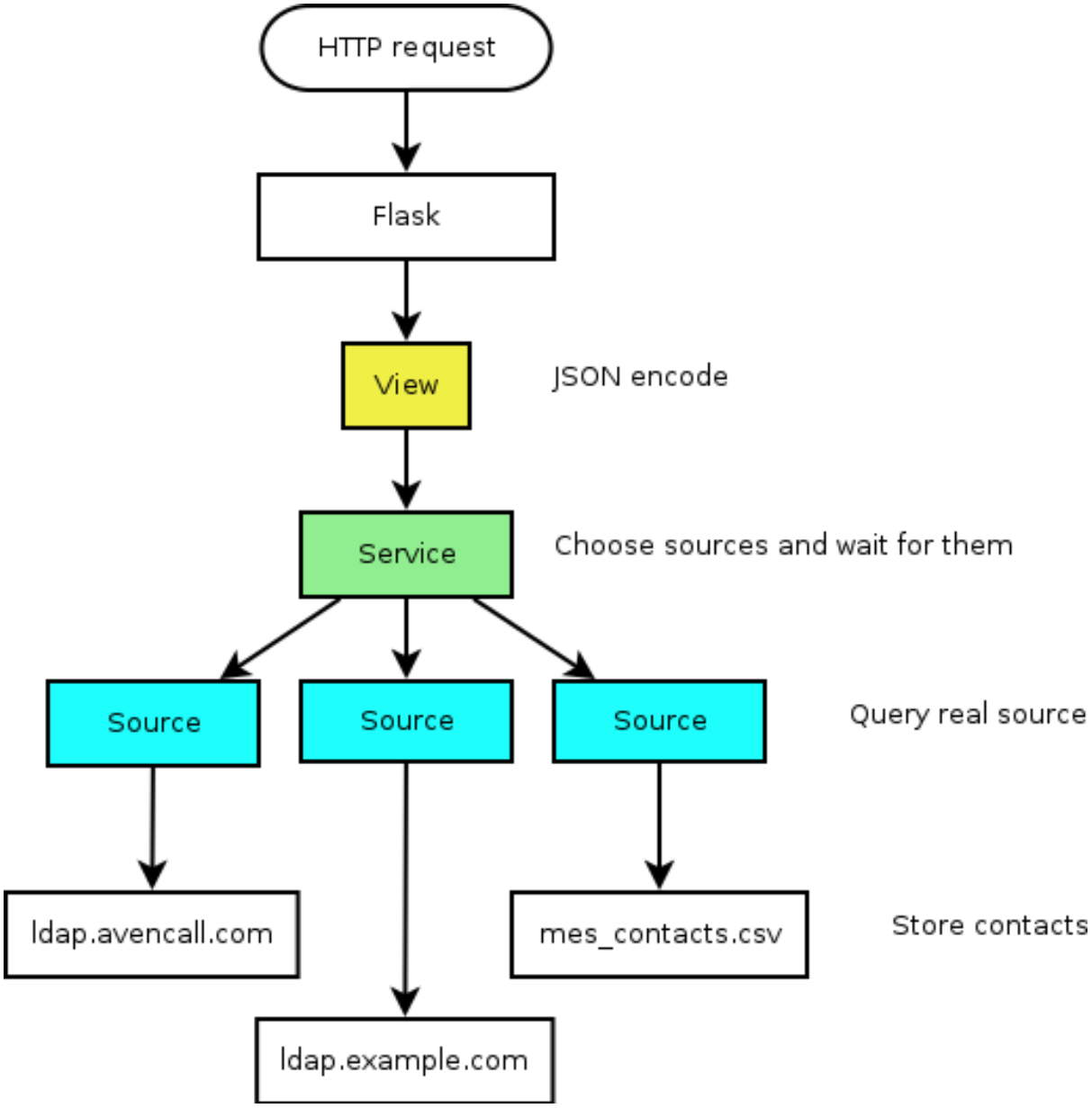


Fig. 15: xivo-dird HTTP query

(continued from previous page)

```
    entry_points={
        'xivo_dird.services': [
            'my_service = dummy:DummyServicePlugin',
        ],
        'xivo_dird.backends': [
            'my_backend = dummy:DummyBackend',
        ],
        'xivo_dird.views': [
            'my_view = dummy:DummyView',
        ],
    }
)
```

Back-End

Back-ends are used to query directories. Each back-end implements a way to query a given directory. Each instance of a given back-end is called a source. Sources are used by the services to get results from each configured directory.

Given one LDAP back-end, I can configure a source from the LDAP at alpha.example.com and another source from the other LDAP at beta.example.com. Both of these sources use the LDAP back-end.

Implementation details

- Namespace: `xivo_dird.backends`
- Abstract source plugin: BaseSourcePlugin
- Methods:
  - `name`: the name of the source, typically retrieved from the configuration injected to `load()`
  - `load(args)`: set up resources used by the plugin, depending on the config. `args` is a dictionary containing:
    - key `config`: the source configuration for this instance of the back-end
    - key `main_config`: the whole configuration of xivo-dird
  - `unload()`: free resources used by the plugin.
  - `search(term, args)`: The search method returns a list of dictionary.
    - Empty values should be `None`, instead of empty string.
    - `args` is a dictionary containing:
      - key `token_infos`: data associated to the authentication token (see *xivo-auth*)
  - `first_match(term, args)`: The first_match method returns a dictionary.
    - Empty values should be `None`, instead of empty string.
    - `args` is a dictionary containing:
      - key `token_infos`: data associated to the authentication token (see *xivo-auth*)
  - `list(uids, args)`: The list method returns a list of dictionary from a list of uids. Each uid is a string identifying a contact within the source.
    - `args` is a dictionary containing:
      - key `token_infos`: data associated to the authentication token (see *xivo-auth*)

See *Sources Configuration*. The implementation of the back-end should take these values into account and return results accordingly.

Example

The following example add a backend that will return random names and number.

`dummy.py`:

```
# -*- coding: utf-8 -*-

import logging

logger = logging.getLogger(__name__)

class DummyBackendPlugin(object):

    def name(self):
        return 'my_local_dummy'

    def load(self, args):
        logger.info('dummy backend loaded')

    def unload(self):
        logger.info('dummy backend unloaded')

    def search(self, term, args):
        nb_results = random.randint(1, 20)
        return _random_list(nb_results)

    def list(self, unique_ids):
        return _random_list(len(unique_ids))

    def _random_list(self, nb_results):
        columns = ['Firstname', 'Lastname', 'Number']
        return [_random_entry(columns) for _ in xrange(nb_results)]

    def _random_entry(self, columns):
        random_stuff = [_random_string() for _ in xrange(len(columns))]
        return dict(zip(columns, random_stuff))

    def _random_string(self):
        return ''.join(random.choice(string.lowercase) for _ in xrange(5))
```

Service

Service plugins add new functionality to the dird server. These functionalities are available to views. When loaded, a service plugin receives its configuration and a dictionary of available sources.

Some service examples that come to mind include:

- A lookup service to search through all configured sources.
- A reverse lookup service to search through all configured sources and return a specific field of the first matching result.

Implementation details

- Namespace: `xivo_dird.services`

- Abstract service plugin: BaseServicePlugin
- Methods:
  - `load(args)`: set up resources used by the plugin, depending on the config. `args` is a dictionary containing:
    - key `config`: the whole configuration file in dict form
    - key `sources`: a dictionary of source names to sources

    `load` must return the service object, which is any kind of python object.
  - `unload()`: free resources used by the plugin.

Example

The following example adds a service that will return an empty list when used.

`dummy.py`:

```
# -*- coding: utf-8 -*-

import logging

from xivo_dird import BaseServicePlugin

logger = logging.getLogger(__name__)

class DummyServicePlugin(BaseServicePlugin):
    """
    This plugin is responsible fow instantiating and returning the
    DummyService. It manages its life time and should take care of
    its cleanup if necessary
    """

    def load(self, args):
        """
        Ignores all provided arguments and instantiate a DummyService that
        is returned to the core
        """
        logger.info('dummy loaded')
        self._service = DummyService()
        return self._service

    def unload(self):
        logger.info('dummy unloaded')


class DummyService(object):
    """
    A very dumb service that will return an empty list every time it is used
    """

    def list(self):
        """
        This function must be called explicitly from the view, `list` is not a
        special method name for xivo-dird
        """
        return []
```

View

View plugins add new routes to the HTTP application in xivo-dird, in particular the REST API of xivo-dird: they define the URLs to which xivo-dird will respond and the formatting of data received and sent through those URLs.

For example, we can define a REST API formatted in JSON with one view and the same API formatted in XML with another view. Supporting the directory function of a phone is generally a matter of adding a new view for the format that the phone consumes.

Implementation details

- Namespace: `xivo_dird.views`
- Abstract view plugin: BaseViewPlugin
- Methods:
  - `load(args)`: set up resources used by the plugin, depending on the config. Typically, register routes on Flask. Those routes would typically call a service. `args` is a dictionary containing:
    - key `config`: the section of the configuration file for all views in dict form
    - key `services`: a dictionary of services, indexed by name, which may be called from a route
    - key `http_app`: the Flask application instance
    - key `rest_api`: a Flask-RestFul Api instance
  - `unload()`: free resources used by the plugin.

Example

The following example adds a simple view: `GET /0.1/directories/ping` answers `{"message": "pong"}`.

`dummy.py`:

```
# -*- coding: utf-8 -*-

import logging

from flask_restful import Resource

logger = logging.getLogger(__name__)


class PingViewPlugin(object):

    name = 'ping'

    def __init__(self):
        logger.debug('dummy view created')

    def load(self, args):
        logger.debug('dummy view args: %s', args)

        args['rest_api'].add_resource(PingView, '/0.1/directories/ping')

    def unload(self):
        logger.debug('dummy view unloaded')


```

(continues on next page)

(continued from previous page)

```
class PingView(Resource):
    """
    Simple API using Flask-Restful: GET /0.1/directories/ping answers "pong"
    """

    def get(self):
        return {'message': 'pong'}
```

Stock Plugins Documentation

View Plugins

default_json

View name: default_json

Purpose: present directory entries in JSON format.

headers

View name: headers

Purpose: List headers that will be available in results from `default_json` view.

personal_view

View name: personal_view

Purpose: Expose REST API to manage personal contacts (create, delete, list).

phonebook_view

View name: phonebook_view

Purpose: Expose REST API to manage xivo-dird's internal phonebooks.

aastra_view

View name: aastra_view

Purpose: Expose REST API to search in configured directories for Aastra phone.

cisco_view

View name: cisco_view

Purpose: Expose REST API to search in configured directories for Cisco phone (see CiscoIP-Phone_XML_Objects).

polycom_view

View name: polycom_view

Purpose: Expose REST API to search in configured directories for Polycom phone.

snom_view

View name: snom_view

Purpose: Expose REST API to search in configured directories for Snom phone.

thomson_view

View name: thomson_view

Purpose: Expose REST API to search in configured directories for Thomson phone.

yealink_view

View name: yealink_view

Purpose: Expose REST API to search in configured directories for Yealink phone.

Service Plugins

lookup

Service name: lookup

Purpose: Search through multiple data sources, looking for entries matching a word.

Configuration

Example (excerpt from the main configuration file):

```
services:
    lookup:
        default:
            sources:
                - my_csv
            timeout: 0.5
```

The configuration is a dictionary whose keys are profile names and values are configuration specific to that profile.

For each profile, the configuration keys are:

sources The list of source names that are to be used for the lookup

timeout The maximum waiting time for an answer from any source. Results from sources that take longer to answer are ignored. Default: no timeout.

favorites

Service name: favorites

Purpose: Mark/unmark contacts as favorites and get the list of all favorites.

personal

Service name: personal

Purpose: Add, delete, list personal contacts of users.

phonebook

Service name: phonebook

Purpose: Add, delete, list phonebooks and phonebook contacts.

Configuration

Example (excerpt from the main configuration file):

```
services:
    favorites:
        default:
            sources:
                - my_csv
            timeout: 0.5
```

The configuration is a dictionary whose keys are profile names and values are configuration specific to that profile.

For each profile, the configuration keys are:

sources The list of source names that are to be used for the lookup

timeout The maximum waiting time for an answer from any source. Results from sources that take longer to answer are ignored. Default: no timeout.

reverse

Service name: reverse

Purpose: Search through multiple data sources, looking for the first entry matching an extension.

Configuration

Example:

```
services:
    reverse:
        default:
            sources:
                - my_csv
            timeout: 1
```

The configuration is a dictionary whose keys are profile names and values are configuration specific to that profile.

For each profile, the configuration keys are:

sources The list of source names that are to be used for the reverse lookup

timeout The maximum waiting time for an answer from any source. Results from sources that take longer to answer are ignored. Default: 1.

Back-end Configuration

This sections completes the *Sources Configuration* section.

csv

Back-end name: csv

Purpose: read directory entries from a CSV file.

Limitations:

- the CSV delimiter is not configurable (currently: `,` (comma)).

Configuration

Example (a file inside `source_config_dir`):

```
type: csv
name: my_csv
file: /var/tmp/test.csv
unique_column: id
searched_columns:
    - fn
    - ln
first_matched_columns:
    - num
format_columns:
    lastname: "{ln}"
    firstname: "{fn}"
    number: "{num}"
```

With the CSV file:

```
id,fn,ln,num
1,Alice,Abrams,55553783147
2,Bob,Benito,5551354958
3,Charles,Curie,5553132479
```

file the absolute path to the CSV file

CSV web service

Back-end name: csv_ws

Purpose: search using a web service that returns CSV formatted results.

Given the following configuration, *xivo-dird* would call "https://example.com:8000/ws-phonebook?firstname=alice&lastname=alice" for a lookup for the term "alice".

Configuration

Example (a file inside `source_config_dir`):

```
type: csv_ws
name: a_csv_web_service
lookup_url: "https://example.com:8000/ws-phonebook"
list_url: "https://example.com:8000/ws-phonebook"
```

(continues on next page)

(continued from previous page)

```
verify_certificate: False
searched_columns:
  - firstname
  - lastname
first_matched_columns:
    - exten
delimiter: ","
timeout: 16
unique_column: id
format_columns:
    number: "{exten}"
```

lookup_url the URL used for directory searches.

list_url (optional) the URL used to list all available entries. This URL is used to retrieve favorites.

verify_certificate (optional) whether the SSL cert will be verified. A CA_BUNDLE path can also be provided. Defaults to True.

delimiter (optional) the field delimiter in the CSV result. Default: ‘,’

timeout (optional) the number of seconds before the lookup on the web service is aborted. Default: 10.

dird_phonebook

back-end name: dird_phonebook

Purpose: search the xivo-dird’s internal phonebooks

Configuration:

```
type: dird_phonebook
name: phonebook
db_uri: 'postgresql://asterisk:proformatique@localhost/asterisk'
tenant: default
phonebook_id: 42
phonebook_name: main
first_matched_columns:
  - number
searched_columns:
  - firstname
  - lastname
format_columns:
    name: "{firstname} {lastname}"
```

db_uri the URI of the DB used by xivo-dird to store the phonebook.

tenant the tenant of the phonebook to query.

phonebook_name the *name* of the phonebook used by this source.

phonebook_id (deprecated, use phonebook_name) the *id* of the phonebook used by this source.

ldap

Back-end name: ldap

Purpose: search directory entries from an LDAP server.

Configuration

Example (a file inside `source_config_dir`):

```
type: ldap
name: my_ldap
ldap_uri: ldap://example.org
ldap_base_dn: ou=people,dc=example,dc=org
ldap_username: cn=admin,dc=example,dc=org
ldap_password: foobar
ldap_custom_filter: (l=québec)
unique_column: entryUUID
searched_columns:
    - cn
first_matched_columns:
    - telephoneNumber
format_columns:
    firstname: "{givenName}"
    lastname: "{sn}"
    number: "{telephoneNumber}"
```

ldap_uri the URI of the LDAP server. Can only contains the scheme, host and port part of an LDAP URL.

ldap_base_dn the DN of the entry at which to start the search

ldap_username (optional) the user's DN to use when performing a "simple" bind.

Default to an empty string.

When both ldap_username and ldap_password are empty, an anonymous bind is performed.

ldap_password (optional) the password to use when performing a "simple" bind.

Default to an empty string.

ldap_custom_filter (optional) the custom filter is used to add more criteria to the filter generated by the back end.

Example:

- ldap_custom_filter: (l=québec)
- searched_columns: [cn,st]

will result in the following filter being used for searches. `(&(l=québec)(|(cn=*%Q*)(st=*%Q*)))`

If only the custom filter is to be used, leave the `searched_columns` field empty.

This must be a valid LDAP filter, where the string `%Q` will be replaced by the (escaped) search term when performing a search.

Example: `(&(o=ACME)(cn=*%Q*))`

ldap_network_timeout (optional) the maximum time, in second, that an LDAP network operation can take. If it takes more time than that, no result is returned.

Defaults to 0.3.

ldap_timeout (optional) the maximum time, in second, that an LDAP operation can take.

Defaults to 1.0.

unique_column (optional) the column that contains a unique identifier of the entry. This is necessary for listing and identifying favorites.

For OpenLDAP, you should set this option to "entryUUID".

For Active Directory, you should set this option to "objectGUID" and also set the "unique_column_format" option to "binary_uuid".

unique_column_format (optional) the unique column's type returned by the queried LDAP server. Valid values are "string" or "binary_uuid".

Defaults to "string".

phonebook

Back-end name: phonebook

Purpose: search directory entries from a XiVO *phone book*.

Configuration

Example (a file inside `source_config_dir`):

```
type: phonebook
name: my_phonebook
phonebook_url: https://example.org/service/ipbx/json.php/restricted/pbx_services/
↪phonebook
phonebook_username: admin
phonebook_password: foobar
first_matched_columns:
    - phonebooknumber.office.number
    - phonebooknumber.mobile.number
format_columns:
    firstname: "{phonebook.firstname}"
    lastname: "{phonebook.lastname}"
    number: "{phonebooknumber.office.number}"
```

phonebook_url (optional) the phonebook's URL.

Default to `http://localhost/service/ipbx/json.php/private/pbx_services/phonebook`.

The URL to use differs depending on if you are accessing the phone book locally or remotely:

- Local: `http://localhost/service/ipbx/json.php/private/pbx_services/phonebook`
- Remote: `https://example.org/service/ipbx/json.php/restricted/pbx_services/phonebook`

phonebook_username (optional) the username to use in HTTP requests.

No HTTP authentication is tried when phonebook_username or phonebook_password are empty.

phonebook_password (optional) the password to use in HTTP requests.

phonebook_timeout (optional) the HTTP request timeout, in seconds.

Defaults to 1.0.

To be able to access the phone book of a remote XiVO, you must create a web services access user (*Configuration -> Web Services Access*) on the remote XiVO.

personal

Back-end name: personal

Purpose: search directory entries among users' personal contacts

You should only have one source of type `personal`, because only one will be used to list personal contacts. The `personal` backend needs a working Consul installation. This backend works with the personal service, which allows users to add personal contacts.

The complete list of fields is in *Personal contacts*.

Configuration

Example (a file inside `source_config_dir`):

```
type: personal
name: personal
first_matched_columns:
    - number
format_columns:
    firstname: "{firstname}"
    lastname: "{lastname}"
    number: "{number}"
```

`unique_column` is not configurable, its value is always `id`.

xivo

Back-end name: xivo

Purpose: add users from a XiVO (may be remote) as directory entries

Configuration

Example (a file inside `source_config_dir`):

```
type: xivo
name: my_xivo
confd_config:
    https: True
    host: xivo.example.com
    port: 9486
    version: 1.1
    username: admin
    password: password
    timeout: 3
unique_column: id
first_matched_columns:
    - exten
searched_columns:
    - firstname
    - lastname
format_columns:
    number: "{exten}"
    mobile: "{mobile_phone_number}"
```

confd_config:host the hostname of the XiVO (more precisely, of the xivo-confd service)

confd_config:port the port of the xivo-confd service (usually 9486)

confd_config:version the version of the xivo-confd API (should be 1.1)

Integration of XiVO dird with the rest of XiVO

Configuration values

Views

In the directory displays (also in the *main configuration file* of xivo-dird, in the `views` section), the following keys are interpreted and displayed in xlet people of the XiVO Client:

`title` The `title` will be shown as a header for the column

`type`

- `agent`: the field value will be ignored and replaced by an icon showing the status of the agent assigned to the contact (e.g. green icon for logged agent, red icon for unlogged agent, ...)
- `callable`: a dropdown action on the `number` field will be added to call the field value.
- `email`: a dropdown action on the `number` field will be added to send an email to the field value.
- `favorite`: the boolean field value will be replaced by an icon showing if the status is favorite (yellow star filled) or not (yellow star empty).
- `name`: a decoration will be added to the field value (typically a color dot) showing the presence status of the contact (e.g. Disconnected, Available, Away, ...)
- `number`: only one number type can be defined per profile. The field value will be:
  - added a decoration (typically a color dot) showing the status of the phone of the contact (e.g. Offline, Ringing, Talking, ...)
  - replaced with a button to call the contact with your phone when using the mouse
- `personal`: the boolean field value will be used to show a deletion action for the contact
- `voicemail`: the voicemail number of the contact

Personal contacts

Here are the list of available attributes of a personal contact:

- `id`
- `company`
- `email`
- `fax`
- `firstname`
- `lastname`
- `mobile`
- `number`

To be able to edit and delete personal contacts, you need a column of type *personal* in your display.

Adding the *personal* column to your display

In the web interface under *Services* → *CTI Server* → *Directories* → *Display filters*.

1. Edit the filter on which you which to enable favorites.
2. Add a column with the type *personal* and display format *personal*.

Favorites

Enabling favorites in the XiVO client.

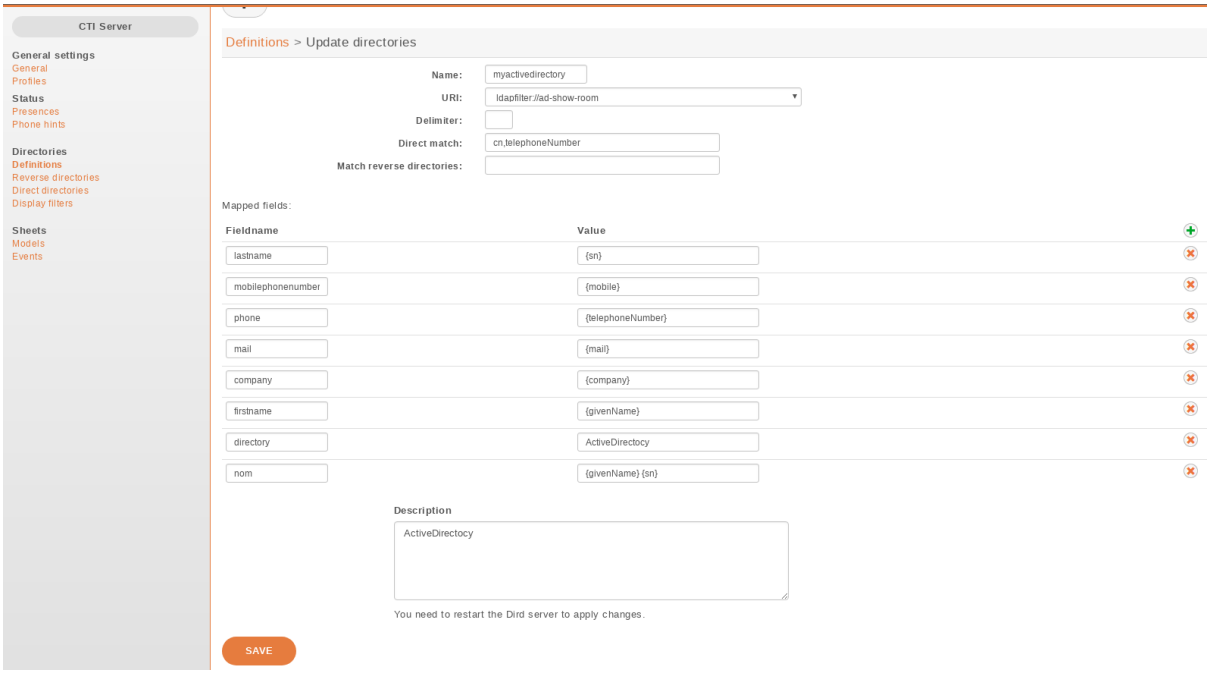
- Add a *unique_column* to your sources.
- Add a *favorite* column to your display

Adding a *unique_column* to your sources

The web interface does not allow the administrator to specify the *unique_column* and *unique_column_format*. To add these configuration options, add a file to */etc/xivo-dird/sources.d* containing *the same name* than the directory definition and all missing fields.

Example:

Given an *ldap* directory source using Active Directory named `myactivedirectory`:



Add a file `/etc/xivo-dird/sources.d/myactivedirectory.yml` with the following content to enable favorites on this source.

```
name: myactivedirectory  # the same name than the directory definition
unique_column: objectGUID
unique_column_format: binary_uuid
```

Adding the *favorite* column to your display

In the web interface under *Services* → *CTI Server* → *Directories* → *Display filters*.

1. Edit the filter on which you which to enable favorites.
2. Add a column with the type *favorite* and display format *favorite*.

Customizing sources

Some configuration options are not available in the web interface. To add configuration to a source that is configured in the web interface, create a file in */etc/xivo-dird/sources.d/* with the key *name* matching your web interface

configuration and add all missing fields.

Example:

adding a timeout configuration to a CSV web service source

```
name: my_csv_web_service
timeout: 16
```

Launching xivo-dird

```
usage: xivo-dird [-h] [-c CONFIG_FILE] [-d] [-f] [-l LOG_LEVEL] [-u USER]

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILE, --config-file CONFIG_FILE
                        The path where is the config file. Default: /etc/xivo-dird/
↪config.yml
  -d, --debug           Log debug messages. Overrides log_level. Default:
                        False
  -f, --foreground      Foreground, don't daemonize. Default: False
  -l LOG_LEVEL, --log-level LOG_LEVEL
                        Logs messages with LOG_LEVEL details. Must be one of:
                        critical, error, warning, info, debug. Default: info
  -u USER, --user USER  The owner of the process.
```

Terminology

Back-end

A back-end is a connector to query a specific type of directory, e.g. one back-end to query LDAP servers, another back-end to query CSV files, etc.

Source

A source is an instance of a back-end. One backend may be used multiples times to query multiple directories of the same type. For example, I could have the customer-csv and the employee-csv sources, each using the CSV back-end, but reading a different file.

Plugins

A plugin is an extension point in xivo-dird. It is a way to add or modify the functionality of xivo-dird. There are currently three types of plugins:

- Back-ends to query different types of directories (LDAP, CSV, etc.)
- Services to provide different directory actions (lookup, reverse lookup, etc.)
- Views to expose directory results in different formats (JSON, XML, etc.)

API

See http://MY_XIVO/api, section XiVO Dird.

XiVO dird phoned

xivo-dird-phoned is an interface to use directory service with phone. It offers a simple REST interface to authenticate a phone and search result from *XiVO dird*.

Usage

xivo-dird-phoned is used through HTTP requests, using HTTP and HTTPS. Its default port is 9498 and 9499. As a user, the common operation is to search through directory from a phone. The phone need to send 2 informations:

- *xivo_user_uuid*: The XiVO user uuid that the phone is associated. It's used to search through personal contacts (see *personal*).
- *profile*: The profile that the user is associated. It's used to format results as configured.

Note: Since most phones dont't support HTTPS, a small protection is to configure authorized_subnets in *Configuration Files* or in *Services* → *General settings* → *Phonebook* → *Hosts*

Launching xivo-dird-phoned

On command line, type `xivo-dird-phoned -h` to see how to use it.

Purge Logs

Keeping records of personal communications for long periods may be subject to local legislation, to avoid personal data retention. Also, keeping too many records may become resource intensive for the server. To ease the removal of such records, `xivo-purge-db` is a process that removes old log entries from the database. This allows keeping records for a maximum period and deleting older ones.

By default, xivo-purge-db removes all logs older than a year (365 days). xivo-purge-db is run nightly.

Note: Please check the laws applicable to your country and modify `days_to_keep` (see below) in the configuration file accordingly.

Tables Purged

The following features are impacted by xivo-purge-db:

- *Call Logs*
- *Switchboard Statistics*

More technically, the tables purged by `xivo-purge-db` are:

- `call_log`
- `cel`
- `queue_log`
- `stat_agent_periodic`
- `stat_call_on_queue`
- `stat_queue_periodic`
- `stat_switchboard_queue`

Configuration File

We recommend to override the setting `days_to_keep` from `/etc/xivo-purge-db/config.yml` in a new file in `/etc/xivo-purge-db/conf.d/`.

> **Warning:** Setting `days_to_keep` to 0 will NOT disable `xivo-purge-db`, and will remove ALL logs from your system.

See *Configuration priority* and `/etc/xivo-purge-db/config.yml` for more details.

Manual Purge

It is possible to purge logs manually. To do so, log on to the target XiVO server and run:

```
xivo-purge-db
```

You can specify the number of days of logs to keep. For example, to purge entries older than 365 days:

```
xivo-purge-db -d 365
```

Usage of `xivo-purge-db`:

```
usage: xivo-purge-db [-h] [-d DAYS_TO_KEEP]

optional arguments:
  -h, --help            show this help message and exit
  -d DAYS_TO_KEEP, --days_to_keep DAYS_TO_KEEP
                        Number of days data will be kept in tables
```

Maintenance

After an execution of `xivo-purge-db`, postgresql's Autovacuum Daemon should perform a VACUUM ANALYZE automatically (after 1 minute). This command marks memory as reusable but does not actually free disk space, which is fine if your disk is not getting full. In the case when `xivo-purge-db` hasn't run for a long time (e.g. upgrading to 15.11 or when *days_to_keep* is decreased), some administrator may want to perform a VACUUM FULL to recover disk space.

> **Warning:** VACUUM FULL will require a service interruption. This may take several hours depending on the size of purged database.

You need to:

```
$ xivo-service stop
$ sudo -u postgres psql asterisk -c "VACUUM (FULL)"
$ xivo-service start
```

Archive Plugins

In the case you want to keep archives of the logs removed by xivo-purge-db, you may install plugins to xivo-purge-db that will be run before the purge.

XiVO does not provide any archive plugin. You will need to develop plugins for your own need. If you want to share your plugins, please open a pull request.

Archive Plugins (for Developers)

Each plugin is a Python callable (function or class constructor), that takes a dictionary of configuration as argument. The keys of this dictionary are the keys taken from the configuration file. This allows you to add plugin-specific configuration in `/etc/xivo-purge-db/conf.d/`.

There is an example plugin in the xivo-purge-db git repo.

Example

Archive name: sample

Purpose: demonstrate how to create your own archive plugin.

Activate Plugin

Each plugin needs to be explicitly enabled in the configuration of `xivo-purge-db`. Here is an example of file added in `/etc/xivo-purge-db/conf.d/`:

```
enabled_plugins:
    archives:
        - sample
```

sample.py

The following example will be save a file in `/tmp/xivo_purge_db.sample` with the following content:

```
Save tables before purge. 365 days to keep!
```

```
 sample_file = '/tmp/xivo_purge_db.sample'

def sample_plugin(config):
    with open(sample_file, 'w') as output:
        output.write('Save tables before purge. {0} days to keep!'.format(config[
↪'days_to_keep']))
```

Install sample plugin

The following `setup.py` shows an example of a python library that adds a plugin to xivo-purge-db:

```
 #!/usr/bin/env python
 # -*- coding: utf-8 -*-

 from setuptools import setup
 from setuptools import find_packages


 setup(
     name='xivo-purge-db-sample-plugin',
     version='0.0.1',

     description='An example program',
     packages=find_packages(),
     entry_points={
         'xivo_purge_db.archives': [
```

(continues on next page)

(continued from previous page)

```
            'sample = xivo_purge_db_sample.sample:sample_plugin',
        ],
    }
)
```

XiVO service

XiVO has many running services. To restart the whole stack, the *xivo-service* command can be used to make sure the service is restarted in the right order.

Usage

Show all services status:

```
xivo-service status
```

Stop XiVO services:

```
xivo-service stop
```

Start XiVO services:

```
xivo-service start
```

Restart XiVO services:

```
xivo-service restart
```

The commands above will only act upon XiVO services. Appending an argument `all` will also act upon `nginx` and `postgresql`. Example:

```
xivo-service restart all
```

UDP port 5060 will be closed while services are restarting.

xivo-upgrade script

Usage

Note:

- You can't use xivo-upgrade if you have not run the wizard yet
- Upgrading from a version prior to *XiVO PBX* 1.2 is not supported.
- When upgrading XiVO, you **must** also upgrade **all** associated XiVO Clients. There is currently no retro-compatibility on older *XiVO PBX* Client versions.

This script will update *XiVO PBX* and restart all services.

There are 2 options you can pass to xivo-upgrade:

- `-d` to only download packages without installing them. **This will still upgrade the package containing xivo-upgrade and xivo-service**.
- `-f` to force upgrade, without asking for user confirmation

`xivo-upgrade` uses the following environment variables:

- `XIVO_CONFD_PORT` to set the port used to query the *HTTP API of xivo-confd* (default is 9486)

Troubleshooting

Postgresql

When upgrading XiVO, if you encounter problems related to the system locale, see *PostgreSQL localization errors*.

xivo-upgrade

If xivo-upgrade fails or aborts in mid-process, the system might end up in a faulty condition. If in doubt, run the following command to check the current state of xivo's firewall rules:

```
iptables -nvL
```

If, among others, it displays something like the following line (notice the DROP and 5060):

```
0     0 DROP       udp  --  *      *       0.0.0.0/0            0.0.0.0/0           ␣
↪ udp dpt:5060
```

Then your XiVO will not be able to register any SIP phones. In this case, you must delete the DROP rules with the following command:

```
iptables -D INPUT -p udp --dport 5060 -j DROP
```

Repeat this command until no more unwanted rules are left.

XiVO sysconfd

xivo-sysconfd is the system configuration server for XiVO. It does quite a few different things; here's a non exhaustive list:

- configuring network (interfaces, hostname, DNS)
- configuring high availability
- staring/stopping/restarting services
- reloading asterisk configuration
- sending some events to components (xivo-agentd, xivo-agid and xivo-ctid)

Configuration File

Default location: `/etc/xivo/sysconfd.conf`. Format: INI.

The default location may be overwritten by the command line options.

Here's an example of the configuration file:

```
[general]
xivo_config_path = /etc/xivo
templates_path = /usr/share/xivo-sysconfd/templates
custom_templates_path = /etc/xivo/sysconfd/custom-templates
backup_path = /var/backups/xivo-sysconfd

```

(continues on next page)

(continued from previous page)

```
[resolvconf]
hostname_file = /etc/hostname
hostname_update_cmd = /etc/init.d/hostname.sh start
hosts_file = /etc/hosts
resolvconf_file = /etc/resolv.conf

[network]
interfaces_file = /etc/network/interfaces

[wizard]
templates_path = /usr/share/xivo-config/templates
custom_templates_path = /etc/xivo/custom-templates

[commonconf]
commonconf_file = /etc/xivo/common.conf
commonconf_generate_cmd = /usr/sbin/xivo-create-config
commonconf_update_cmd = /usr/sbin/xivo-update-config
commonconf_monit = /usr/sbin/xivo-monitoring-update

[openssl]
certsdir = /var/lib/xivo/certificates

[monit]
monit_checks_dir = /usr/share/xivo-monitoring/checks
monit_conf_dir = /etc/monit/conf.d

[request_handlers]
synchronous = false

[bus]
username = guest
password = guest
host = localhost
port = 5672
exchange_name = xivo
exchange_type = topic
exchange_durable = true
```

request_handlers section

synchronous If this option is true, when xivo-sysconfd receives a request to reload the dialplan for example, it will wait for the dialplan reload to complete before replying to the request.

When this option is false, xivo-sysconfd reply to the request immediately.

By default, this option is set to false to speed up some operations (for example, editing a user from the web interface or from xivo-confd), but this means that there will be a small delay (up to a few seconds in the worst case) between the time you create your user and the time you can dial successfully its extension.

4.1.2 High Availability (HA)

The HA (High Availability) solution in XiVO makes it possible to maintain basic telephony function whether your main XiVO server is running or not. When running a XiVO HA cluster, users are guaranteed to never experience a downtime of more than 5 minutes of their basic telephony service.

The HA solution in XiVO is based on a 2-nodes "master and slave" architecture. In the normal situation, both the master and slave nodes are running in parallel, the slave acting as a "hot standby", and all the telephony services are provided by the master node. If the master fails or must be shutdown for maintenance, then the telephony devices automatically communicate with the slave node instead of the master one. Once the master

is up again, the telephony devices failback to the master node. Both the failover and the failback operation are done automatically, i.e. without any user intervention, although an administrator might want to run some manual operations after failback as to, for example, make sure any voicemail messages that were left on the slave are copied back to the master.

Prerequisites

The HA in XiVO only works with telephony devices (i.e. phones) that support the notion of a primary and backup telephony server.

- Phones must be able to reach the master and the slave
- Master and Slave nodes must be in the same subnet
- If firewalling, the master must be allowed to join the slave on ports 22 and 5432
- If firewalling, the slave must be allowed to join the master with an ICMP ping
- Trunk registration timeout (`expiry`) should be less than 300 seconds (5 minutes)
- The slave must have **no** provisioning plugins installed.

The HA solution is guaranteed to work correctly with the following devices.

Quick Summary

- You need two configured XiVO (wizard passed)
- Configure one XiVO as a master -> setup the slave address (VoIP interface)
- Restart services (xivo-service restart) on master
- Configure the other XiVO as a slave -> setup the master address (VoIP interface)
- Configure file synchronization by runnning the script `xivo-sync -i` on the master
- Start configuration synchronization by running the script `xivo-master-slave-db-replication <slave_ip>` on the master
- Resynchronize all your devices
- Configure the XiVO Clients

That's it, you now have a HA configuration, and every hour all the configuration done on the master will be reported to the slave.

Configuration Details

First thing to do is to *install 2 XiVO*.

Important: When you upgrade a node of your cluster, you must also upgrade the other so that they both are running the same version of XiVO. Otherwise, the replication might not work properly.

Warning: If your XiVO is interconnected with a XiVO CC, you **MUST** re-configure postres to accept connections from XiVO CC - see *PostgreSQL configuration section*

You must configure the HA in the Web interface (*Configuration* → *Management* → *High Availability* page).

You can configure the master and slave in whatever order you want.

You must also run `xivo-sync -i` on the master to setup file synchronization. Running `xivo-sync -i` will create a passwordless SSH key on the master, stored under the `/root/.ssh` directory, and will add it to the `/root/.ssh/authorized_keys` file on the slave.

Note: If you want to try the ssh logging as advised by the ssh-copy-id script, you must select the new key to be used by ssh: `ssh -i /root/.ssh/xivo_id_rsa root@<slave_ip>`

The following directories will then be rsync'ed every hour:

- /etc/asterisk/extensions_extra.d
- /etc/xivo/asterisk
- /var/lib/asterisk/agi-bin
- /var/lib/asterisk/moh
- /var/lib/xivo/certificates
- /var/lib/xivo/sounds/acd
- /var/lib/xivo/sounds/playback

Warning: When the HA is configured, some changes will be automatically made to the configuration of XiVO.

SIP expiry value on master and slave will be automatically updated:

- min: 3 minutes
- max: 5 minutes
- default: 4 minutes

The provisioning server configuration will be automatically updated in order to allow phones to switch from XiVO power failure.

Warning: Do not change these values when the HA is configured, as this may cause problems. These values will be reset to blank when the HA is disabled.

Important: For the telephony devices to take the new proxy/registrar settings into account, you must *resynchronize the devices* or restart them manually.

Disable node

Default status of HIGH AVAILABILITY (HA) is disabled:

Warning: You should not disable an HA node in production as it will break the configuration and restart some services.

Important: You have to restart services (xivo-service restart) once the master node is disabled.

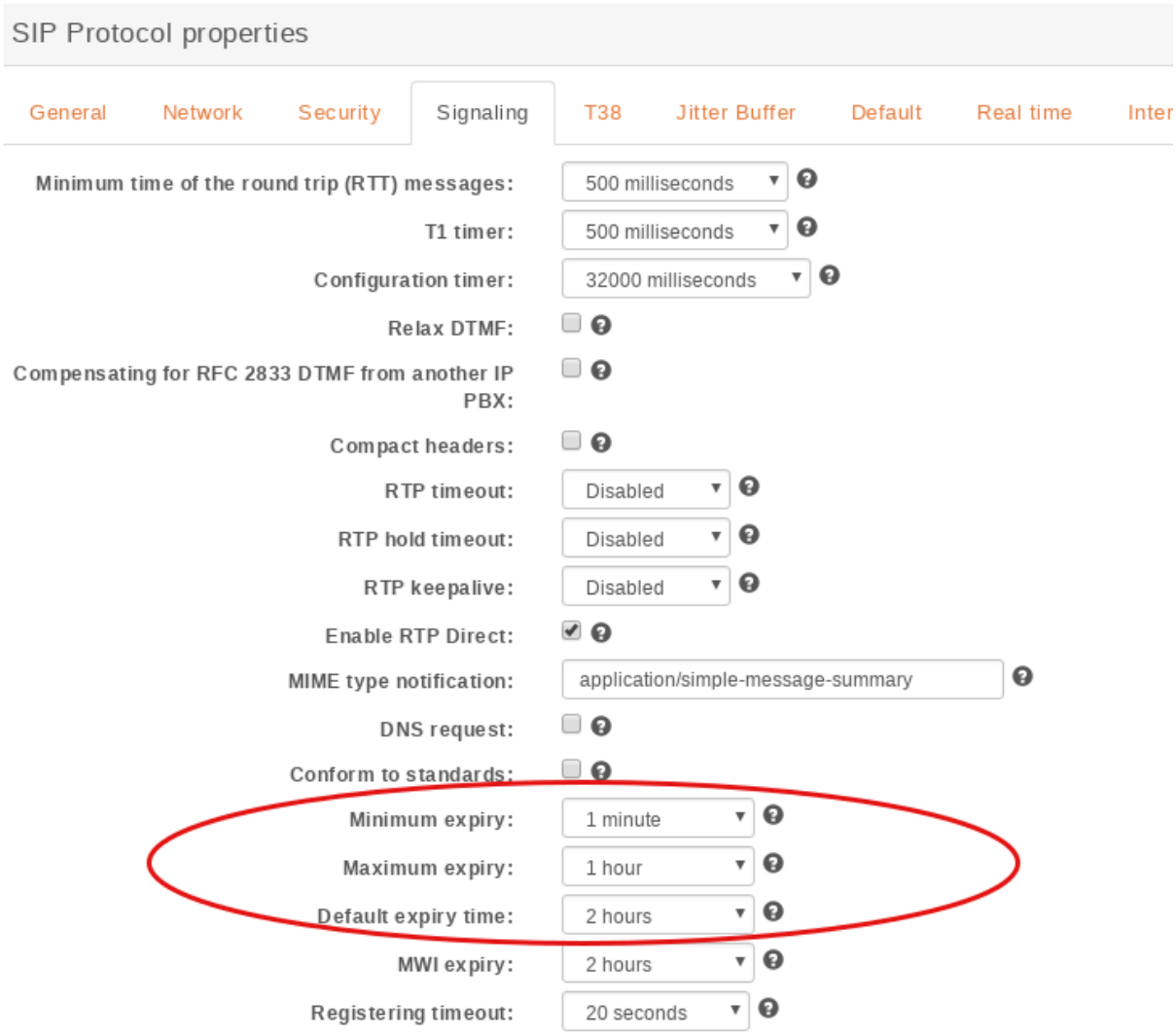


Fig. 16: *Services → IPBX → General Settings → SIP Protocol*

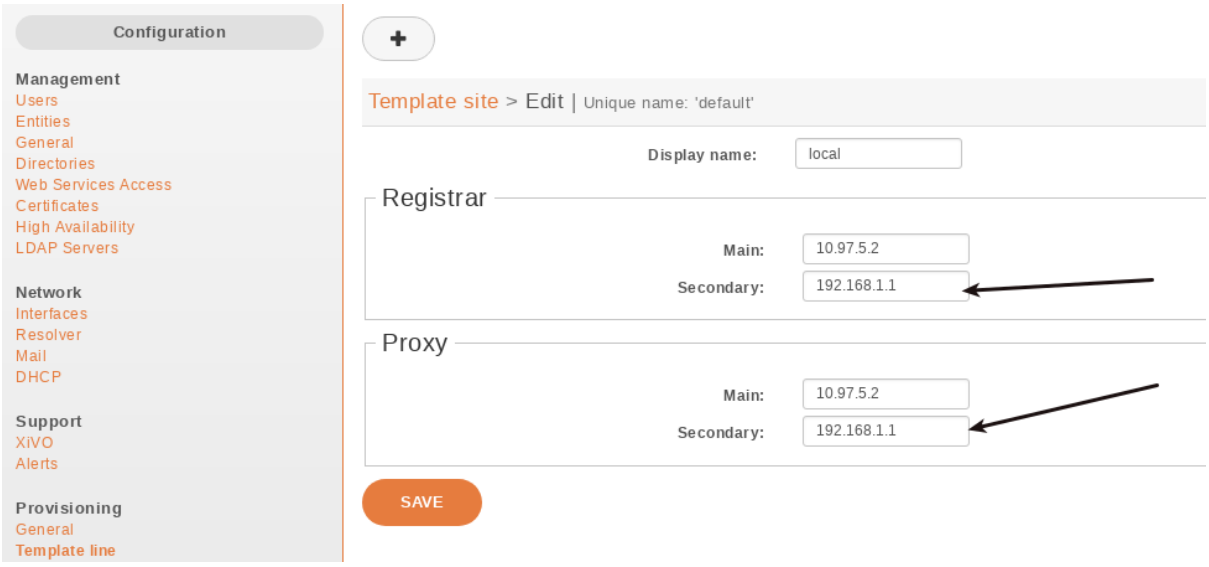


Fig. 17: *Configuration → Provisioning → Template Line → Edit default*

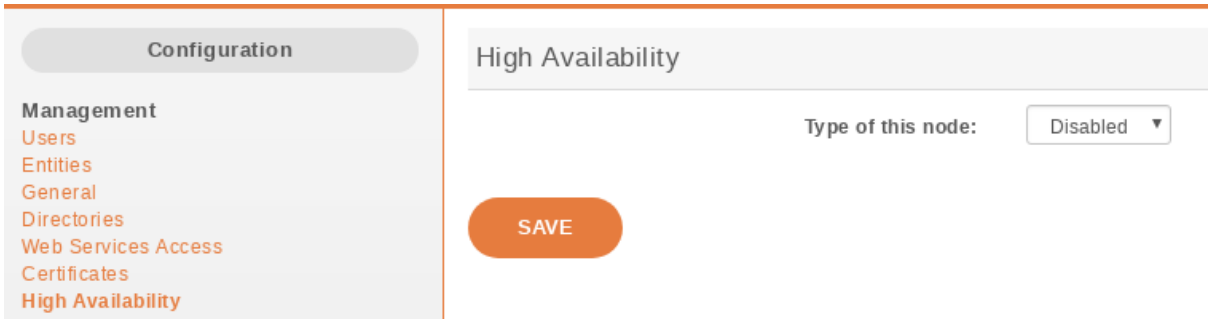


Fig. 18: HA Dashboard Disabled (default state)

Master node

In choosing the method `Master` you must enter the IP address **of the VoIP interface** of the slave node.

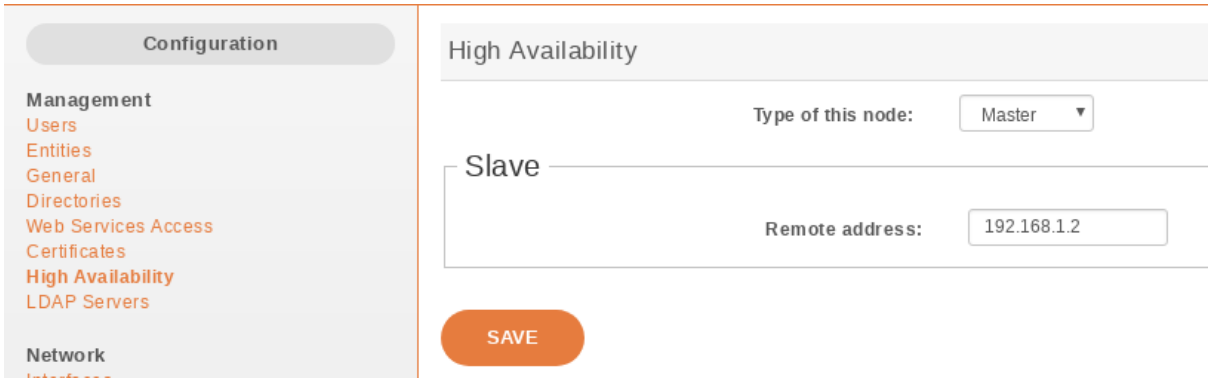


Fig. 19: HA Dashboard Master

Important: You have to restart all services (xivo-service restart) once the master node is configured.

Slave node

In choosing the method `Slave` you must enter the IP address **of the VoIP interface** of the master node.

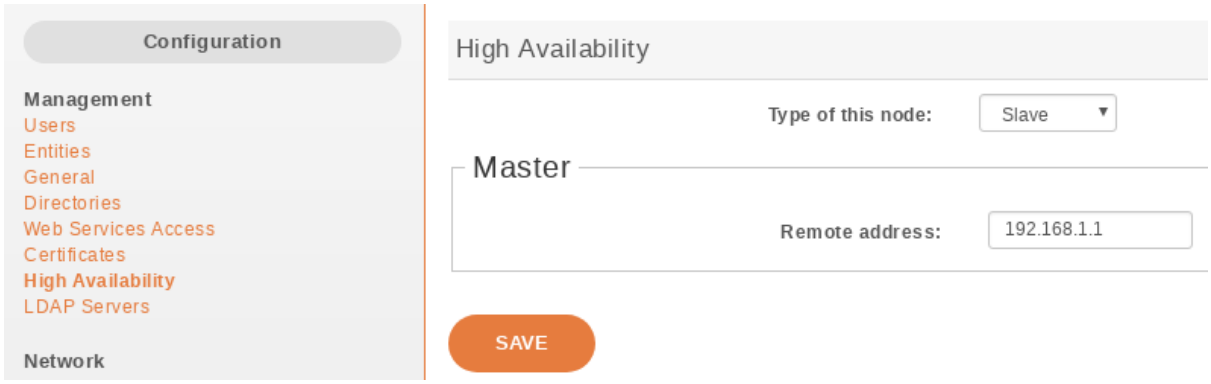


Fig. 20: HA Dashboard Slave

Replication Configuration

Once master slave configuration is completed, XiVO configuration is replicated from the master node to the slave every hour (:00).

Replication can be started manually by running the replication scripts on the master:

```
xivo-master-slave-db-replication <slave_ip>
xivo-sync
```

The replication does not copy the full XiVO configuration of the master. Notably, these are excluded:

- All the network configuration **except DHCP configuration** (i.e. everything under the *Configuration* → *Network* → *{Interfaces, Resolver, Mail}* sections)
- All the support configuration (i.e. everything under the *Configuration* → *Support* section)
- Call logs
- Call center statistics
- Certificates
- HA settings
- Provisioning configuration
- Voicemail messages

Less importantly, these are also excluded:

- Queue logs
- CELs

XiVO Client

You have to enter the master and slave address in the `Connection` tab of the XiVO Client configuration :

The main server is the master node and the backup server is the slave node.

When connecting the XiVO Client with the main server down, the login screen will hang for 3 seconds before connecting to the backup server.

Internals

4 scripts are used to manage services and data replication.

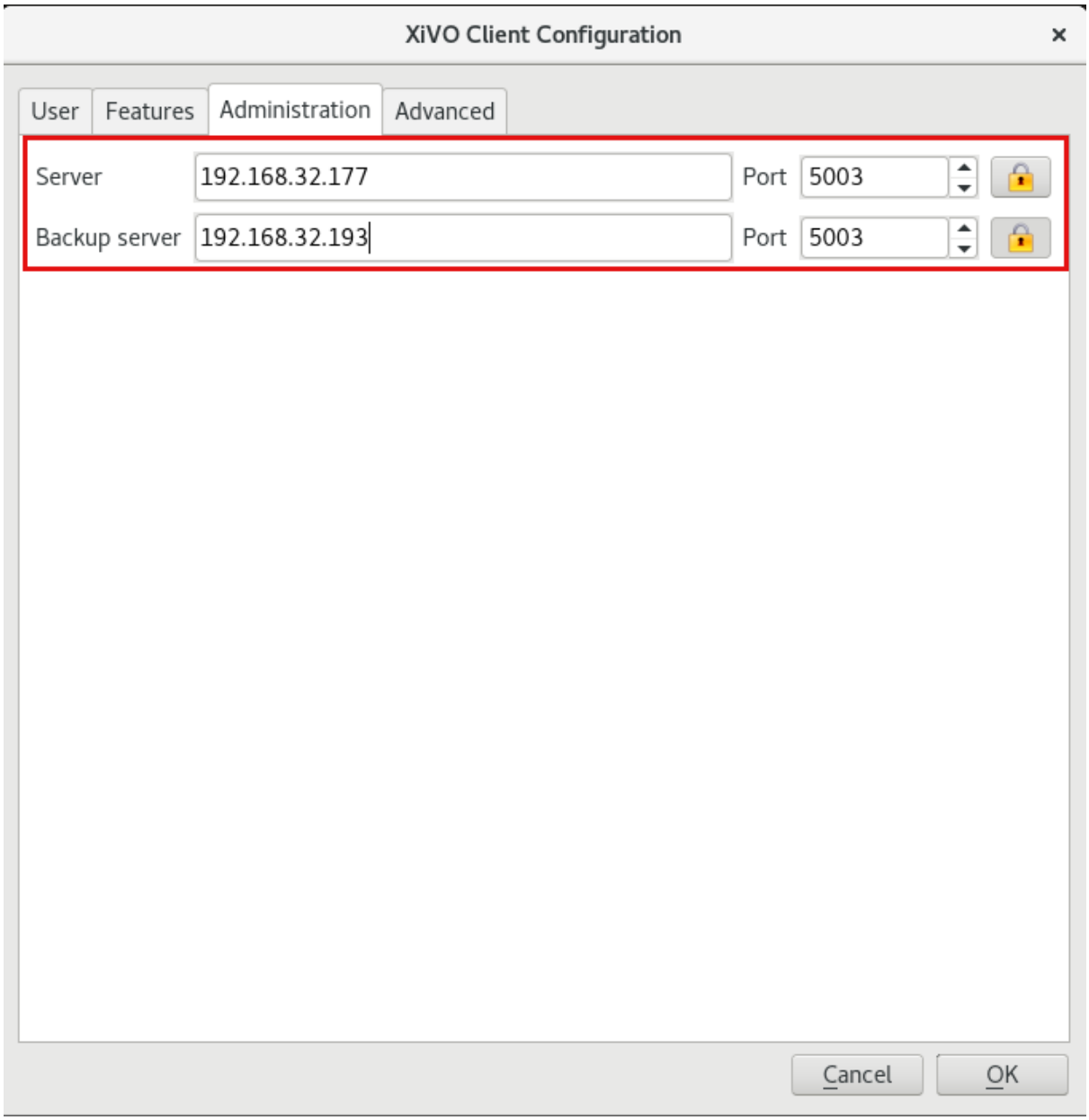
- `xivo-master-slave-db-replication <slave_ip>` is used on the master to replicate the master's data on the slave server. It runs on the master.
- `xivo-manage-slave-services {start,stop}` is used on the slave to start, stop monit and asterisk. The services won't be restarted after an upgrade or restart.
- `xivo-check-master-status <master_ip>` is used to check the status of the master and enable or disable services accordingly.
- `xivo-sync` is used to sync directories from master to slave.

Limitations

Architecture:

- Since DHCP parameters are replicated, Master and Slave node MUST be on the same VoIP network.

When the master node is down, some features are not available and some behave a bit differently. This includes:


XiVO Client Configuration
User
Features
Administration
Advanced
Server
192.168.32.177
Port
5003
Backup server
192.168.32.193
Port
5003
Cancel
OK

- Call history / call records are not recorded.
- Voicemail messages saved on the master node are not available.
- Custom voicemail greetings recorded on the master node are not available.
- Phone provisioning is disabled, i.e. a phone will always keep the same configuration, even after restarting it.
- Phone remote directory is not accessible, because provisioned IP address points to the master.

Note that, on failover and on failback:

- DND, call forwards, call filtering, ..., statuses may be lost if changed recently.
- If you are connected as an agent, then you might need to reconnect as an agent when the master goes down. Since it's hard to know when the master goes down, if your CTI client disconnects and you can't reconnect it, then it's a sign the master might be down.

Additionally, only on failback:

- Voicemail messages are not copied from the slave to the master, i.e. if someone left a message on your voicemail when the master was down, you won't be able to consult it once the master is up again.
- More generally, custom sounds are not copied back. This includes recordings.

Here's the list of limitations that are more relevant on an administrator standpoint:

- The master status is up or down, there's no middle status. This mean that if Asterisk is crashed the XiVO is still up and the failover will NOT happen.

Berofos Integration

Berofos Integration

XiVO offers the possibility to integrate a berofos failover switch within a HA cluster.

This is useful if you have one or more ISDN lines (i.e. T1/E1 or T0 lines) that you want to use whatever the state of your XiVO HA cluster. To use a berofos within your XiVO HA installation, you need to properly configure both your berofos and your XiVOs, then the berofos will automatically switch your ISDN lines from your master node to your slave node if your master goes down, and vice-versa when it comes back up.

You can also use a Berofos failover switch to secure the ISDN provider lines when installing a XiVO in front of an existing PBX. The goal of this configuration is to mitigate the consequences of an outage of the XiVO : with this equipment the ISDN provider links could be switched to the PBX directly if the XiVO goes down.

XiVO **does not offer natively** the possibility to configure Berofos in this failover mode. The *Berofos Integration with PBX* section describes a workaround.

Installation and Configuration

Master Configuration

There is nothing to be done on the master node.

Slave Configuration

First, install the bntools package:

```
apt-get install bntools
```

This will make the `bnfos` command available.

You can then connect your berofos to your network and power it on. By default, the berofos will try to get an IP address via DHCP. If it is not able to get such address from a DHCP server, it will take the 192.168.0.2/24 IP address.

Note: The DHCP server on XiVO does not offer IP addresses to berofos devices by default.

Next step is to create the `/etc/bnfos.conf` file via the following command:

```
bnfos --scan -x
```

If no berofos device is detected using this last command, you'll have to explicitly specify the IP address of the berofos via the -h option:

```
bnfos --scan -x -h <berofos ip>
```

At this stage, your `/etc/bnfos.conf` file should contains something like this:

```
[fos1]
mac = 00:19:32:00:12:1D
host = 10.34.1.50
#login = <user>:<password>
```

It is advised to configure your berofos with a static IP address. You first need to put your berofos into *flash mode* :

- press and hold the black button next to the power button,
- power on your berofos,
- release the black button when the red LEDs of port D start blinking.

Then, you can issue the following command, by first replacing the network configuration with your one:

```
bnfos --netconf -f fos1 -i 10.34.1.20 -n 255.255.255.0 -g 10.34.1.1 -d 0
```

Note:

- `-i` is the IP address
- `-n` is the netmask
- `-g` is the gateway
- `-d 0` is to disable DHCP

You can then update your berofos firmware to version 1.53:

```
wget http://www.beronet.com/downloads/berofos/bnfos_v153.bin
bnfos --flash bnfos_v153.bin -f fos1
```

Once this is done, you'll have to reboot your berofos in operationnal mode (that is in normal mode).

Then you must rewrite the `/etc/bnfos.conf` (mainly if you changed the IP address):

```
bnfos --scan -x -h <berofos ip>
```

Now that your berofos has proper network configuration and an up to date firmware, you might want to set a password on your berofos:

```
bnfos --set apwd=<password> -f fos1
bnfos --set pwd=1 -f fos1
```

You must then edit the `/etc/bnfos.conf` and replace the login line to something like:

```
login = admin:<password>
```

Next, configure your berofos for it to work correctly with the XiVO HA:

```
bnfos --set wdog=0 -f fos1
bnfos --set wdogdef=0 -f fos1
bnfos --set scenario=0 -f fos1
bnfos --set mode=1 -f fos1
bnfos --set modedef=1 -f fos1
```

This, among other things, disable the watchdog. The switching from one relay mode to the other will be done by the XiVO slave node once it detects the master node is down, and vice-versa.

Finally, you can make sure everything works fine by running the xivo-berofos command:

```
xivo-berofos master
```

The green LEDs on your berofos should be lighted on ports A and B.

Connection

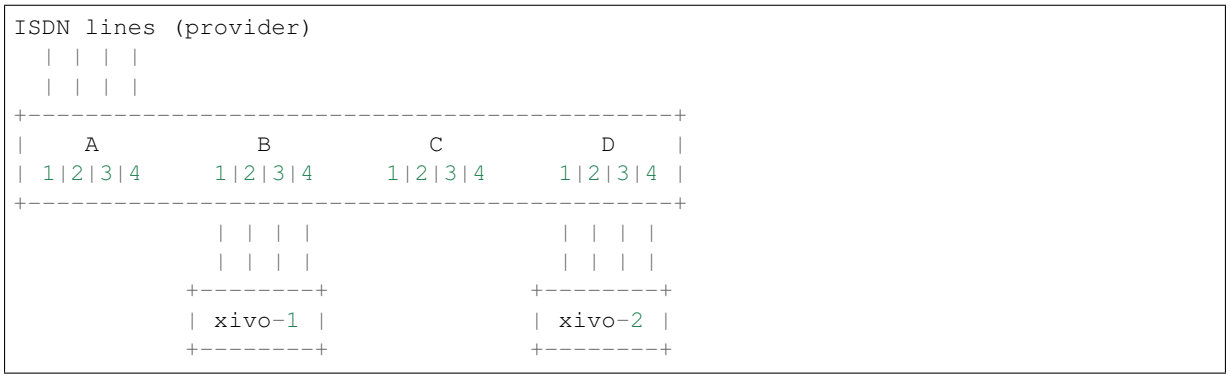
Two XiVOs

Here's how to connect the ISDN lines between your berofos with:

- two XiVOs in high availability

In this configuration you can protect **up two 4** ISDN lines. If more than 4 ISDN lines to protect, you must set up a *Multiple berofos* configuration.

Here's an example with 4 ISDN lines coming from your telephony provider:



Two XiVOs and one PBX

Here's how to connect your berofos with:

- two XiVOs in high availability,
- one PBX.

In this configuration you can protect **up two 2** ISDN lines. If more than 2 ISDN lines to protect, you must set up a *Multiple berofos* configuration.

Logical view:

```
                +--------+                                  +-----+
-- Provider ----| xivo-1 | -- ISDN Interconnection  --| PBX | -- Phones
                +--------+                                  +-----+
                  | xivo-2 |
                  +--------+
```

This example shows the case where there are 2 ISDN lines coming from your telephony provider:

```
ISDN lines (provider)
  | |
  | |
+-------------------------------------------------------+
|    A               B            C            D        |
| 1|2|3|4         1|2   3|4      1|2|3|4     1|2   3|4 |
+-------------------------------------------------------+
      | |     CPE | |   | | NET          CPE | |   | | NET
      | |   spans | |   | | spans      spans | |   | | spans
      | |       +----------+             +------------+
      | |       |  xivo-1  |             |   xivo-2   |
      | |       +----------+             +------------+
      | |
      | |
    +------+
    | PBX  |
    +------+
```

One XiVO and one PBX

This case is not currently supported. You'll find a workaround in the *Berofos Integration with PBX* section.

Multiple berofos

It's possible to use more than 1 berofos with XiVO.

For each supplementary berofos you want to use, you must first configure it properly like you did for the first one. The only difference is that you need to add a berofos declaration to the `/etc/bnfos.conf` file instead of creating/overwriting the file. Here's an example of a valid config file for 2 berofos:

```
[fos1]
mac = 00:19:32:00:12:1D
host = 10.100.0.201
login = admin:foobar

[fos2]
mac = 00:11:22:33:44:55
host = 10.100.0.202
login = admin:barfoo
```

Warning: berofos name must follow the pattern `fosX` where X is a number starting with 1, then 2, etc. The `bnfos` tool won't work properly if it's not the case.

Operation

When your XiVO switch the relay mode of your berofos, it logs the event in the `/var/log/syslog` file.

Default mode

Note that when the berofos is off, the A and D ports are connected together. This behavior is not customizable.

Uninstallation

It is important to remove the `/etc/bnfos.conf` file on the slave node when you don't want to use anymore your berofos with your XiVOs.

Reset the Berofos

You can reset the berofos configuration :

1. Power on the berofos,
2. When red and green LEDs are still lit, press & hold the black button,
3. Release it when the red LEDs of the D port start blinking fast
4. Reboot the beronet, it should have lost its configuration.

External links

- berofos user manual

Troubleshooting

When replicating the database between master and slave, if you encounter problems related to the system locale, see *PostgreSQL localization errors*.

4.2 XiVOcc Administration

4.2.1 System Configuration

ACD Outgoing Calls For Call Blending

Use the following only if you want to use "Least Recent" call distribution strategy and that outbound agent calls have to be taken into account by the distribution strategy.

By default, when your agents process incoming and outgoing calls, the call distribution will not take into account agents which are in outgoing calls in the *least recent* call strategy and at the end of an outgoing call there is no wrapup. So an agent can be distributed just after an outgoing calls even if another agent is free for a longer time, because the outgoing call is not taken into account by the distribution strategy.

You will find below how to improve that.

XiVO-CC agent can make outgoing calls through an outgoing queue. This brings the statistics and supervision visualization for outgoing ACD calls. However, some special configuration steps are required. The outgoing calls must be dialed from CC Agent application to use this feature.

Configuration Steps

- You need to create an outgoing queue with
  - in tab *General*:
    - *Name*: starting with 'out', e.g. outbound,
    - *Number*: some number
    - *Music On-Hold*: None
    - *Preprocess subroutine*: xuc_outcall_acd
  - in tab *Application*:
    - *Ringing Time*: 0
    - *Ring instead of On-Hold music*: activated
- Agent will have to be logged on this queue

How to check correct configuration

Check if agent is logged in the outbound queue:

```
jyl-rennes*CLI> queue show outbound
outbound has 0 calls (max unlimited) in 'ringall' strategy (0s holdtime, 0s␣
↪talktime), W:0, C:0, A:0, SL:0.0% within 0s
Members:
  Agent/2500 (Local/id-19@agentcallback from SIP/ihvbur) (ringinuse disabled)␣
↪(dynamic) (Not in use) (skills: agent-19) has taken no calls yet
No Callers
```

Check the skills attached to the agent by displaying it's agent group:

```
jyl-rennes*CLI> queue show skills groups agent-19
Skill group 'agent-19':
  - agent_19       : 100
  - agent_no_2500  : 100
  - genagent       : 100
```

If the agent dials an outbound call of more than 6 digits (default) you should see the internal queue statistics updated. The agent's state should be "(in call)" for ongoing call and when the call ends, the number of taken calls should be incremented:

```
jyl-rennes*CLI> queue show outbound
outbound has 0 calls (max unlimited) in 'ringall' strategy (0s holdtime, 34s␣
↪talktime), W:0, C:1, A:0, SL:100.0% within 0s
   Members:
      Agent/2500 (Local/id-19@agentcallback from SIP/ihvbur) (ringinuse disabled)␣
↪(dynamic) (Not in use) (skills: agent-19) has taken 1 calls (last was 1 secs ago)
   No Callers
```

Once done, calls requested by an agent through the Cti.js with more than 6 digits are routed via the outgoing queue. You can change the number of digits using the parameter xuc.outboundLength in the XuC's configuration.

Ldap Authentication

Xuc

Configure LDAP authent for CCmanager, UC Assistant and CC Agent

You need to include in the `docker-xivocc.yml` file a link to a specific configuration file by adding in xuc section a specific volume and an environment variable to specify the alternate config file location

```
xuc:
  image: ...

  environment:
  - ...
  - CONFIG_FILE=/conf/xuc.conf

  volumes:
  - /etc/docker/xuc:/conf
```

Edit in `/etc/docker/xuc/` folder a configuration file named `xuc.conf` to add ldap configuration (empty by default)

```
include "application.conf"

authentication {
  ldap {
    managerDN = "uid=company,ou=people,dc=company,dc=com"       # user with read␣
↪rights on the whole LDAP
    managerPassword = "xxxxxxxxxx"                              # password for this␣
↪user
    url = "ldap://ldap.company.com:389"                         # ldap URI
    searchBase = "ou=people,dc=company,dc=com"                  # ldap entry to use␣
↪as search base
    userSearchFilter = "uid=%s"                                 # filter to use to␣
↪search users by login, using a string pattern
  }
}
```

Recreate the container : `xivocc-dcomp up -d xuc`

Kerberos Authentication

To enable Kerberos authentication and single sign on feature, you need to have an existing Kerberos infrastructure with a Key Distribution Center and a Ticket Granting Service. You need to be able to create a service, construct a kerberos server configuration and export a keytab to perform the following configuration. This service must be on the kerberos realm used by your users and must match the dns name of the server hosting the XUC server (or the nginx reverse proxy server if you use one). For example, assuming you have a realm named `MYDOMAIN`, you can create a service named `HTTP/xuc.mydomain` and a dns entry for `xuc.mydomain` pointing the server hosting the XUC.

Warning: The created domain name must be trusted by the user's browser.

Prerequisites

- Create a service for the XUC host, for example:

```
addprinc HTTP/xuc.mydomain
```

- Export the keytab file, for example:

```
ktadd -k xuc.keytab HTTP/xuc.mydomain
```

Warning: The bash commands detailed here are for demonstration only and needs to be adapted to your specific environment. It shows how to create a service for the XUC Server named HTTP/xuc, associated to the example realm `mydomain`.

Only the following encryption types are supported by XiVOCC:

- aes256-cts-hmac-sha1-96
- arcfour-hmac
- des3-cbc-sha1
- des-cbc-crc

XiVOCC Configuration

- Copy the previously generated `xuc.keytab` keytab file to the server hosting the XUC docker container, for example: `/etc/docker/kerberos/xuc.keytab`.
- Create or edit the file `/etc/krb5.conf` on the server hosting the XUC docker container and change settings according to your kerberos environment. For example, the file may contain (name and ip addresses must match your kerberos environment):

```
[libdefaults]
    default_realm = MYDOMAIN

[realms]
    MYDOMAIN = {
        kdc = 172.17.0.14
        admin_server = 172.17.0.14
    }
```

- Edit the docker compose file `/etc/docker/compose/docker-xivocc.yml` to add the following configuration in the xuc section (file name, service name, password may differ on your setup):

```
xuc:
  # ...
  environment:
  - JAVA_OPTS=-Dsecured.krb5.principal=HTTP/xuc.mydomain -Dsecured.krb5.
↪password=xuc -Dsecured.krb5.keyTab=/etc/kerberos/xuc.keytab

  # ...

  volumes:
  - /etc/docker/kerberos:/etc/kerberos
  - /etc/krb5.conf:/etc/krb5.conf
```

- Enable `Single Sign On` on the Agent, Manager, Web and Desktop application interface. Change the value of the following environment variables in the */etc/docker/compose/custom.env*:

```
# ...
USE_SSO=true
XUC_HOST=xuc.mydomain
# ...
```

Browser configuration

The created domain name must be trusted by the user's browser.

For Chrome (windows):

- Internet Option : Add domain with protocol to the list of trusted sites : http://xuc.mydomain (and/or https://xuc.mydomain).

Warning: Kerberos authentication on Chrome is only available on Microsoft Windows.

For Firefox:

- Go to `about:config`
- add domain (without protocol) to the `network.negotiate-auth.delegation-uris` entry (ie. `xuc.mydomain`).
- add domain (without protocol) to the `network.negotiate-auth.trusted-uris` entry (ie. `xuc.mydomain`).

CAS SSO Authentication

To enable CAS authentication and single sign on feature, you need to have an existing CAS infrastructure. You need to be able to create a service for the XiVOCC environment.

Warning: The CAS authentication server must be accessible from the user and the server hosting the XiVOCC containers. CAS Server users' username must match the XiVO username to allow login on the XiVOCC applications. The CAS server must support at least CAS Protocol version 2.0.

XiVOCC Configuration

- Edit the docker compose file `/etc/docker/compose/docker-xivocc.yml` to add the following configuration in the xuc section (use your CAS server URL instead of `https://cas-server.example.org/cas` and set `CAS_LOGOUT_ENABLE` to `true` if you want to logout from CAS when logging out from the application):

```
xucmgt:
  # ...
  environment:
  - CAS_SERVER_URL=https://cas-server.example.org/cas
  - ...

  # ...

xuc:
  # ...
  environment:
  - CAS_SERVER_URL=https://cas-server.example.org/cas
  - CAS_LOGOUT_ENABLE=false
  - ...

  # ...
```

- Recreate and start the XiVOCC environment:

```
xivocc-dcomp up -d
```

Install trusted certificate for nginx

To install a trusted certificate for the nginx reverse proxy instead of the self signed certificate, follow the following instructions:

- in directory `/etc/docker/nginx/ssl` replace content of files `xivoxc.crt`, `xivoxc.csr` and `xivoxc.key` while keeping filenames unchanged
- restart nginx container by command `xivocc-dcomp restart nginx`.

Warning: When configuring the certificate, please ensure the certificate chain is complete, especially for the XiVO Mobile Assistant. You can check the server certificate chain by using the following web site https://www.ssllabs.com/ssltest/analyze.html which will warn you if there is an error with the certificate (`Chain issues - Incomplete`)

What is a complete certificate chain

When a client application (browser or mobile application) checks a certificate for a web site, it checks the received certificate is issued by a known certificate authority and matches the web site domain name. But sometimes, the certificate is not issued by a root certificate authority but by an intermediate authority.

Here is an example of a such a certificate chain:

```
GeoTrust Global CA
|--> RapidSSL SHA256 - CA - G3
     |--> *.company.com
```

The possible problem here is that even if the browser knows the root authority, it is unaware of the intermediate one. The solution is to create a bundle of the complete certificate chain by concatenating the certificates of all parties (root, intermediate & site). Please see http://nginx.org/en/docs/http/configuring_https_servers.html#chains for more information.

Mobile Assitant

If using an HTTPS connection for the XiVO Mobile Assistant, you must use a trusted certificate with a complete certification chain, see *Install trusted certificate for nginx*.

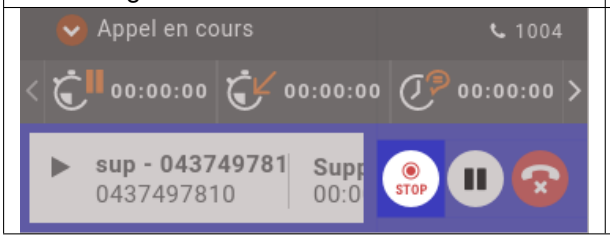
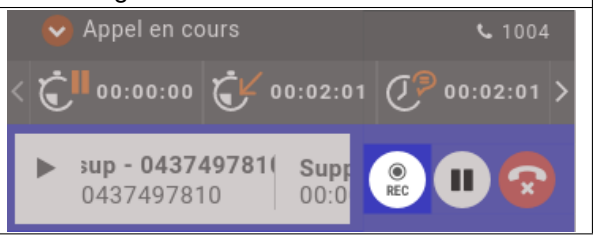
4.2.2 XiVOcc Applications Configuration

This section covers specific configuration parameters for the different application of *XiVO CC*.

CC Agent configuration

Recording

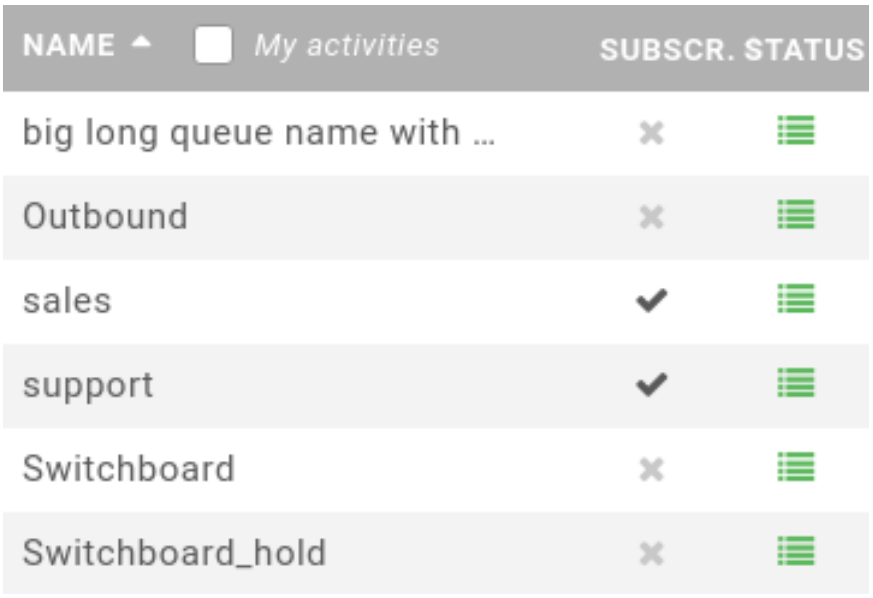
Recording can be paused or started by an agent.

Recording On	Recording Paused
	

This feature can be disabled by changing `showRecordingControls` option in file `application.conf`. You can also set the environnment variable `SHOW_RECORDING_CONTROLS` to false for your xucmgt container in `/etc/docker/compose/custom.env` file. When disabled the recording status is not displayed any more

Activity (Queue) control

By using the `showQueueControls` option in application.conf, you may allow an agent to enter or leave an activity. You can also use `SHOW_QUEUE_CONTROLS` environment variable in `/etc/docker/compose/custom.env` file.



On hold notification

You can configure `notifyOnHold` (in seconds)) in application.conf, this option once set, will trigger a popup and a system notification to user if he has a call on hold for a long time. You can also use `NOTIFY_ON_HOLD` environment variable in `/etc/docker/compose/custom.env` file.

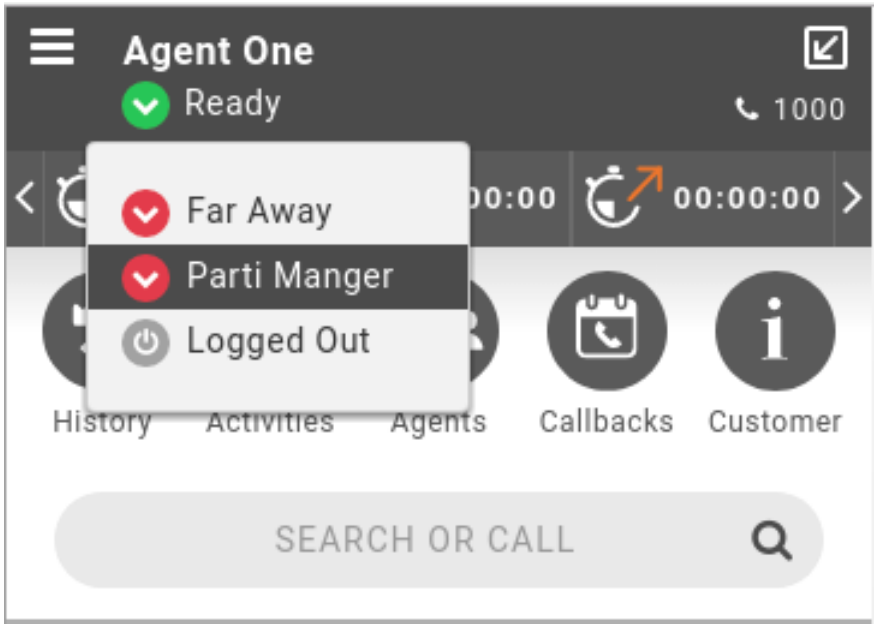
Pause Cause and Status

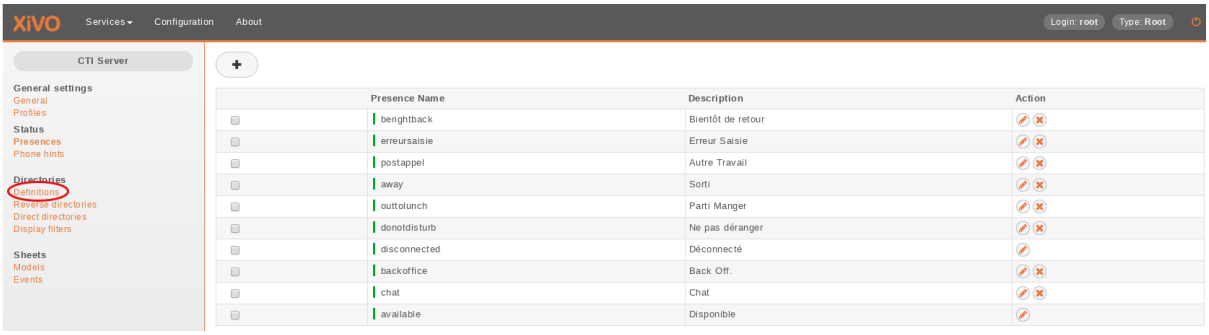
You can configure XiVO to have the following scenario:

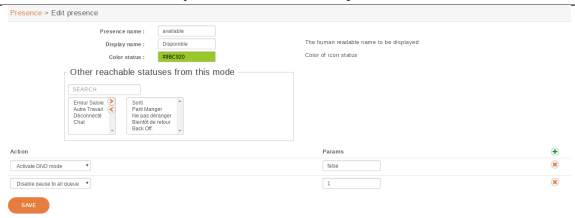
- The agent person leaves temporarily his office (lunch, break, . . .)
- He sets his presence in the CCAgent to the according state
- The agent will be automatically set in pause and his phone will not ring from queues
- He comes back to his office and set his presence to 'Available'
- The pause will be automatically cancelled

By default the pause action from the agent cannot be specified with a specific cause such as Lunch Time, or Tea Time. To be able to use a specific cause, you will have to define new Presences in the cti server configuration.

You define presences with action **Activate pause to all queue** to **true**, for not ready causes, and you must have one presence defined with an action **Disable pause to all queue** to be able to go back to not ready. When this presences are defined, you must restart the xuc server to be able to use them in ccagent, these presences will also be automatically available in *CCmanager* and new real time counters will be calculated.


Agent One
Ready
1000
Far Away
Parti Manger
Logged Out
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL


XiVO
Services
Configuration
About
Login: root
Type: Root
CTI Server
General settings
General
Profiles
Status
Presences
Phone hints
Directories
Definitions
Reverse directories
Direct directories
Display filters
Sheets
Models
Events
Presence Name
Description
Action
berightback
Bientôt de retour
erreursaisie
Erreur Saisie
postappel
Autre Travail
away
Sorti
outtolunch
Parti Manger
donotdisturb
Ne pas déranger
disconnected
Déconnecté
backoffice
Back Off.
chat
Chat
available
Disponible

Presence from ready to pause	Presence from pause to ready
	

Screen Popup

It is possible to display customer information in an external web application using Xivo *sheet* mecanism.

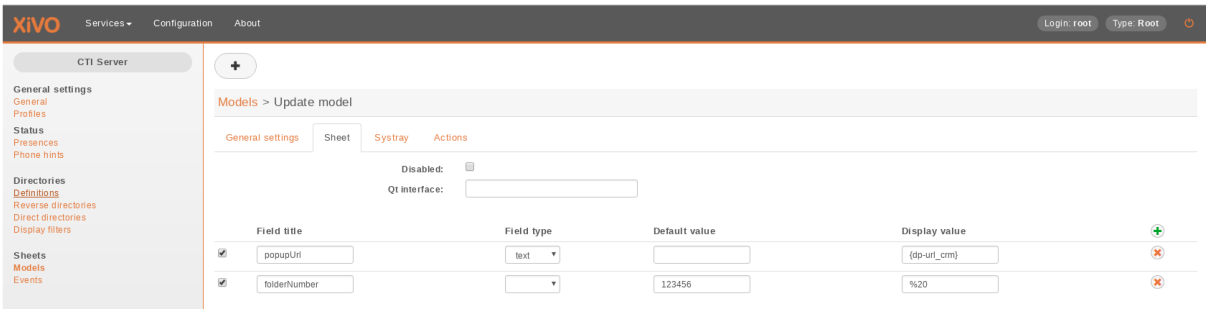
- *Services* > *CTI Server* > *Sheets* > *Models*:
  - Tab *General Settings*: Give a name
  - Tab *Sheet*: You must define a sheet with at least `folderNumber` and `popupUrl` fields set:
    - `folderNumber` (MANDATORY)
      - field type = `text`
      - It has to be defined. Can be calculated or use a default value not equal to "-"
      - Note: You could leave "empty" using a whitespace (in hexadecimal: %20)
    - `popupUrl` (MANDATORY)
      - field type = `text`
      - The url to open when call arrives : i.e. http://mycrm.com/customerInfo?folder= the folder number will be automatically appended at the end of the URL
      - Additionally to the existing xivo variables, you can also use here the following variables(only available in Web Agent and Desktop Agent):
      - `{xuc-token}`: will be replaced by a token used for xuc websocket and rest api, for example `http://mycrm.com/customerInfo?token={xuc-token}&folder=`
      - `{xuc-username}`: will be replaced by the username of the logged on user, for example `http://mycrm.com/customerInfo?username={xuc-username}&folder=`
    - `multiTab` (OPTIONAL)
      - field type = `text`
      - set to the text `true` to open each popup in a new window.
- *Services* > *CTI Server* > *Sheets* > *Events*: Choose the right events for opening the URL (if you choose two events, url will opened twice etc.)

Example : Using the caller number to open a customer info web page

- Define `folderNumber` with any default value i.e. 123456
- Define `popupUrl` with a display value of http://mycrm.com/customerInfo?nb={xivo-calleridnum}&fn= when call arrives web page http://mycrm.com/customerInfo?nb=1050&fn=123456 will be displayed

Login and Pause management using function keys

You can configure Login or Pause keys on an agent phone. Their state will be synchronized with the state in *XiVO CC* applications.



Behavior

- **Login Key**:
  - change status of agent: login if it was logged out, and logout if it was logged in
  - LED of phone key will be updated accordingly as well as the status in *XiVO CC* applications (CCAgent...)
  - if you login/logout via *XiVO CC* applications (CCAgent...), status will be updated and phone key LED will be updated.
- **Pause key**:
  - change status of agent: pause if it was ready, ready if it was paused or in wrapup
  - LED of phone key will be updated accordingly as well as the status in *XiVO CC* applications (CCAgent...)
  - if you pause/unpause via *XiVO CC* applications (CCAgent...), status will be updated and phone key LED will be updated
  - **Wrapup**: if agent is on wrapup, the phone key will *blink*. If you press key while on wrapup, agent status will be changed to ready

---

Note:

* The key blinks on Snom and Yealink phone sets. It doesn't blink on Polycom phone sets.
* To be able to terminate Wrapup via the key on Snom phones you must use correct version of plugin (see devices releasenotes).

Configuration

There are two types of customizable *function keys* that can be used

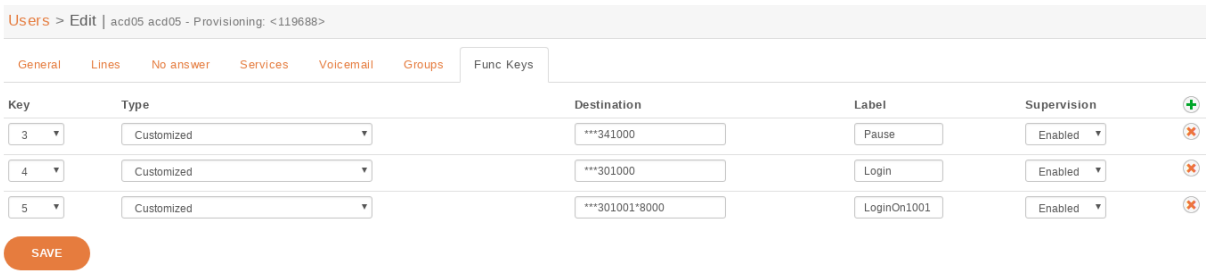
- Login: it will toggle login/logout of agent. There are two configuration patterns (see also below):
  - either `***30<PHONE NUMBER>`: in this case it will ask for agent number and will then login the given agent on phone `<PHONE NUMBER>`
  - or `***30<PHONE NUMBER>*<AGENT NUMBER>`: in this case it will log agent `<AGENT NUMBER>` on phone `<PHONE NUMBER>`
- Pause: it will toggle pause/unpause of agent (and will blink if agent is on Wrapup). Configuration pattern (see also below):
  - `***34<PHONE NUMBER>` : it will toggle pause/unpause of agent logged on phone `<PHONE NUMBER>`

To use it you must:

1. On on XiVO PBX edit `/etc/xivo-xuc.conf` and change variables:

- `XUC_SERVER_IP` to IP address of XivoCC
- `XUC_SERVER_PORT` to port of XUC Server (default is 8090)

2. Configure function key on user (example with user 1000, 1001 and agent 8000 associated to user 1000):
   - Open *Services* > *IPBX* > *IPBX settings* > *Users*
   - Edit the user, open *Func Keys* tab and add keys like:
   - For Pause/Unpause agent logged on phone 1000 set:
     - *Type*: `Customized`,
     - *Destination*: `***341000`,
     - *Label*: `Pause`,
     - *Supervision*: `Enabled`
   - For Login/Logout agent on phone 1000 set:
     - *Type*: `Customized`,
     - *Destination*: `***301000`,
     - *Label*: `Login`,
     - *Supervision*: `Enabled`
   - For Login/Logout agent 8000 on phone 1001 set:
     - *Type*: `Customized`,
     - *Destination*: `***301001*8000`,
     - *Label*: `LoginOn1001`,
     - *Supervision*: `Enabled`



CC Manager

See *CC Manager features*.

Access authorizations in CCManager

Note: Behavior was changed in 2017.LTS1 (see 2017.LTS1 release notes in *Release Notes*)

By default, CCManager access is authorized only for users with *Administrateur* or *Superviseur* rigths (as defined in the Configuration Management server). If required, you can authorize all users to connect to the CCManager interface by setting the `ENFORCE_MANAGER_SECURITY` environment variable to `false` in the `/etc/docker/compose/custom.env` file:

```
...
ENFORCE_MANAGER_SECURITY=false
```

Then you need to recreate the xucmgt container with `xivocc-dcomp up -d`. Then each user will be able to log in the CCManager. Otherwise, each user that wants to connect to the CCManager will need to have a *Administrateur* or *Superviseur* profile in the Configuration Management server.

Recording configuration

This page describes how to configure the recording feature.

Configure recording

To configure recording there are two steps to follow on *XiVO PBX*:

1. Add link towards Recording Server,
2. and then enable recording, which can be done either:
   - in the Queue configuration
   - or via subroutines

1. Add link towards Recording Server

Note: Steps to be done on **XiVO PBX**

The first step is to configure the link towards the Recording Server by running the configuration script:

```
xivocc-recording-config
```

During the configuration, you will be asked for :

- the Recording Server IP (i.e. 192.168.0.2)
- the *XiVO PBX* name (it must not contain any space or "-" character). If you configure more more than one *XiVO PBX* on the same Recording Server, you must give a different name to each of them.

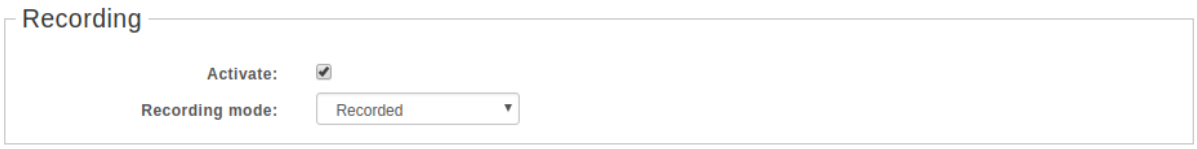
After having configured the recording, you have to enable it via sub-routines. See below.

2. Enable recording

Enable recording in the Queue configuration

Note: Steps to be done on **XiVO PBX**

To enable recording on a queue, go to *Services -> Contact Center -> Queues* and edit the queue.



Then, in the recording section:

- *Recording mode* set it to *Recorded* or *Recorded on demand*
  - *Recorded*: call will be recorded

  - *Recorded on demand*: recording starts in paused state and can be activated by the agent (see *agent recording configuration*)

- *Activate* check it for the recording mode to be active
  - You need to check the *Activate* parameter for the recording to be enabled. When *Activate* is checked, the recording will be enabled according to the mode selected.

Enable recording via subroutines

Note: Steps to be done on **XiVO PBX**

To enable the recording you have to configure one of the shipped subroutines.

The package `xivocc-recording` (see *recording installation section*) ships the following dialplan subroutines :

Subroutine	Description
`xivocc-incall-recording`	Records incoming calls
`xivocc-incall-recording-paused`	Records incoming calls, but record starts in paused state and can be activated by the agent (see *agent recording configuration*)
`xivocc-outcall-recording`	Records outgoing calls
`xivocc-outcall-recording-paused`	Records outgoing calls, but record starts in paused state and can be activated by the agent (see *agent recording configuration*)

These subroutines are to be configured on the following *XiVO PBX* objects (either globally or per-object):

Warning: They **MUST** be configured **only** on the following objects. Other configuration **are not supported**.

- Incalls,
- and/or Users,
- and/or Outcall

Note: Here is an **example** if you want to enable recording for:

- *All* outbound calls but started in pause state,
- And *only on* incoming call 0123456789

Then you would have to:

1. Create a `custom_global_subr.conf` file in the `/etc/asterisk/extensions_extra.d` directory
2. If not already defined elsewhere define the global subroutine:

```
[xivo-subrgbl-outcall]
exten = s,1,NoOp(=== Recording outbound calls in pause ===)
same = n,Gosub(xivocc-outcall-recording-paused,s,1)
same = n,Return()
```

3. Enable the call recording for incall 0123456789 by editing it via the *XiVO PBX* web interface and set the field *Pre-process subroutine* to `xivocc-incall-recording`

Stop recording upon external transfer

By default recording is stopped when both parties of the call are external.

This can be deactivated by adding `STOP_RECORDING_UPON_EXTERNAL_XFER` environment variable to the xuc section of your `docker-xivocc.yml` file:

```
xuc:
  image: ...

  environment:
  - ...
  - SECURED_KRB5_PRINCIPAL
  - STOP_RECORDING_UPON_EXTERNAL_XFER
```

and `STOP_RECORDING_UPON_EXTERNAL_XFER=false` value to your `custom.env`.

```
XIVO_HOST=192.168.1.1
XUC_HOST=192.168.1.2
XUC_PORT=8090
...
STOP_RECORDING_UPON_EXTERNAL_XFER=false
```

and then relaunch the xivocc services with `xivocc-dcomp up -d` command.

Recording filtering configuration

Note: Steps to be done on **XiVO CC**

After having followed above paragraphs, you can also configure the recording filtering.

1. Add a user with Administrateur rights for Recording Server:
    1. Connect to the Config Management interface : http://<XIVO_CC_IP>:9100 (login avencall/superpass),
    2. Add one of the *XiVO PBX* user giving him *Administrateur* rights,
2. Configure excluded numbers on Recording Server
    1. Then, connect with this user to the Recording Server interface : http://<XIVO_CC_IP>:9400
    2. Navigate to the page `Contrôle d'enregistrement` and add the numbers to be excluded from the recording.

In list *Destinataire de l'appel (Numéro entrant, File d'attente, Utilisateur)* declare the:

- XiVO Incalls numbers,
- XiVO Queues numbers,
- or XiVO Users numbers

to be excluded from the recording on incoming or internal call. These numbers will be checked by the `xivocc-incall-recording` subroutines.

Note: numbers must be entered as they first appear in dialplan (check is made against XIVO_DSTNUM dialplan variable).

In list *Emetteur ou destinataire d'un appel sortant (Utilisateur ou numéro appelé externe)* declare the:

- XiVO Users internal numbers

to be excluded from recording on outgoing calls. These numbers will be checked by the `xivocc-outcall-recording` subroutines.

Note: check is made against XIVO_SRCNUM dialplan variable.

Reporting

Totem Panels

Data replication can take some time if there are a lot of data in xivo cel and queue log tables. You may check xivo-db-replication log files (/var/log/xivocc/xivo-db-replication.log).

Preconfigured panels are available on http://@IP/kibana/#/dashboard/file/queues.json et http://@IP/kibana/#/dashboard/file/agents.json to be able to save this panels in elasticsearch database you have to sign on on request user admin/Kibana

Defaut data persistance in totem panels is 7 days, you may change this default by defining ELASTICTTL environment variables in xivo-replic section, see below for an example of how to keep 15 days.

```
xivo_replic:
  image: xivoxc/xivo-db-replication:latest

  links:
  - pgxivocc:db
  - elasticsearch:elasticsearch

  environment:
  - JAVA_OPTS=-Xms256m -Xmx256m
  - XIVO_HOST=192.168.250.11
  - ELASTICTTL=15
...
```

Disable Totem Panels

If you do not use pannels and elasticsearch, it is possible to disable then.

You need to edit the docker-xivocc.yml file.

First, in *xivo_replic* section:

- remove or comment the elasticsearch link,
- add elasticsearch as en *extra_host* on 127.0.0.1
- and add the *DISABLEELASTIC=true* environment variable

```
xivo_replic:
  image: xivoxc/xivo-db-replication:latest

  links:
  - pgxivocc:db
  #- elasticsearch:elasticsearch

  extra_hosts:
  - "elasticsearch:127.0.0.1"

  environment:
  - JAVA_OPTS=-Xms256m -Xmx256m
  - XIVO_HOST=192.168.250.11
```

(continues on next page)

(continued from previous page)

```
  - DISABLEELASTIC=true
...
```

Secondly, remove the *elasticsearch* and *kibana_volumes* sections,

Thirdly, in *nginx* section:

- remove or comment the elasticsearch link,
- add elasticsearch as en *extra_host* on 127.0.0.1
- remove or comment the kibana_volumes

```
nginx:
  image: xivoxc/xivoxc_nginx:latest

  ports:
  - 80:80

  links:
  #- elasticsearch
  - xuc
  - xucmgt

  extra_hosts:
  - "xivo_host:192.168.250.11"
  - "elasticsearch:127.0.0.1"

  volumes_from:
  #- kibana_volumes
  - fingerboard
...
```

Web / Desktop Application

Disabling WebRTC

WebRTC can be disabled globally by setting the `DISABLE_WEBRTC` environment varibale to `true` in `/etc/docker/compose/custom.env` file.

WebRTC configuration

See *Web RTC feature description*.

Signed SSL/TLS certificate for WebRTC

XivoCC installation generates self-signed SSL/TLS into nginx server running as part of XivoCC. This limits WebRtc usage:

- *UC Assistant* shows warning about unsecure page and exception must be confirmed by user.
- *Desktop Applications* must be started with `--ignore-certificate-errors` parameter, which degrades security.

To avoid this, SSL/TLS certificate signed by authority recognized by Chrome in PEM format is required, see *Install trusted certificate for nginx*.

Note: Do **NOT** forget to check that XUC_HOST in /etc/docker/compose/custom.env is also configured with the same FQDN as in the certificate, not the IP address.

4.2.3 XiVOcc Administration

Start, stop or restart containers

Using the *xivocc-dcomp* script, you can control the run of the XiVO CC components:

```
xivocc-dcomp [command] [container]
```

List of commands:

- `up -d` - run containers
- `stop` - stop containers
- `restart` - restart containers

If you don't enter container name, the command applies on all containers. Use container names from this *list*.

Warning: Restarting xuc server with active calls may result in some agent's having incorrect state after the restart. Hang-up of such call will return agent into correct state.

Show status

```
xivocc-dcomp ps
```

Show containers and images versions

Note: Introduced in 2017.03.03 release.

Docker images are labelled with the exact version of the embedded application.

You can display the:

- Version of the running docker containers by typing:

```
xivocc-dcomp version
```

- Version of all docker containers (including stopped ones) by typing:

```
xivocc-dcomp version -a
```

- Version of docker images:

```
xivocc-dcomp version -i
```

For example :

```
# xivocc-dcomp version
NAMES                       VERSION
xivocc_nginx_1
xivocc_xuc_1                2017.03.03
```

(continues on next page)

(continued from previous page)

```
xivocc_recording_server_1    2017.03.03
xivocc_xucmgt_1              2017.03.03
xivocc_config_mgt_1          2017.03.02
xivocc_pack_reporting_1      2017.03.02
xivocc_spagobi_1             2017.03.02
xivocc_xivo_stats_1          2017.03.02
xivocc_xivo_replic_1         2017.03.02
xivocc_pgxivocc_1
xivocc_elasticsearch_1
xivocc_fingerboard_1
xivocc_recording_rsync_1
xivocc_kibana_volumes_1
```

This only applies to the following images:

- config_mgt
- pack_reporting
- recording
- spagobi
- xivo_replic
- xivo_stats
- xuc
- xucmgt

Note: Labels were in fact introduced in 2017.03.02 but in this version you had to use docker commands to display them:

- List all running containers with the exact version of application

```
$ docker ps --format 'table {{.Names}}\t{{.Image}}\t{{.Label "version"}}\t{{.
↪Status}}'
NAMES                        IMAGE                                                 
↪VERSION              STATUS
xivocc_spagobi_1             xivoxc/spagobi:2017.03.latest                  2017.
↪03.02           Up 14 hours
xivocc_nginx_1               xivoxc/xivoxc_nginx:latest                            
↪                Up 32 hours
xivocc_xuc_1                 xivoxc/xuc:2017.03.latest                      2017.
↪03.02           Up 13 hours
xivocc_recording_server_1    xivoxc/recording-server:2017.03.latest         2017.
↪03.02           Up 32 hours
xivocc_xivo_replic_1         xivoxc/xivo-db-replication:2017.03.latest      2017.
↪03.02           Up 32 hours
xivocc_config_mgt_1          xivoxc/config-mgt:2017.03.latest               2017.
↪03.02           Up 32 hours
xivocc_pack_reporting_1      xivoxc/pack-reporting:2017.03.latest           2017.
↪03.02           Up 32 hours
xivocc_xivo_stats_1          xivoxc/xivo-full-stats:2017.03.latest          2017.
↪03.02           Up 32 hours
xivocc_pgxivocc_1            xivoxc/pgxivocc:latest                                
↪                Up 32 hours
xivocc_xucmgt_1              xivoxc/xucmgt:2017.03.latest                   2017.
↪03.02           Up 32 hours
xivocc_elasticsearch_1       elasticsearch:1.7.2                                   
↪                Up 32 hours
xivocc_fingerboard_1         xivoxc/fingerboard:latest                             
↪                Up 32 hours
```

(continues on next page)

(continued from previous page)

```
xivocc_recording_rsync_1    xivoxc/recording-rsync:latest
↪               Up 32 hours
xivocc_kibana_volumes_1     xivoxc/kibana_volume:latest
↪               Up 32 hours
```

- You can also inspect an image or container to get it's exact version:

```
# Inspect an image
$ docker inspect --format '{{(index .Config.Labels "version")}}' xivoxc/
↪xuc:2017.03.latest
2017.03.02

# Inspect a running container
$ docker inspect --format '{{(index .Config.Labels "version")}}' xivocc_xuc_1
2017.03.02
```

Log

The log of each container can be found in the */var/log/xivocc* directory. Currently (it may change) the structure looks like this :

```
/var/log/xivocc:
├── config-mgt
│   └── config-mgt.log
├── purge-reporting-database.log
├── recording-server
│   ├── downloads.log
│   └── recording-server.log
├── spagobi
│   ├── Quartz.log
│   ├── SpagoBIBirtReportEngine.log
│   ├── SpagoBIChartEngine.log
│   ├── SpagoBIJasperReports.log
│   ├── SpagoBI.log
│   ├── SpagoBIQbeEngineAudit.log
│   ├── SpagoBIQbeEngine.log
│   └── SpagoBITalendEngine.log
├── specific-stats.log
├── xivo-db-replication
│   └── xivo-db-replication.log
├── xivo-full-stats
│   └── xivo-full-stats.log
├── xuc
│   ├── xuc_ami.log
│   └── xuc.log
└── xucmgt
    └── xucmgt.log
```

Backup

You may backup your databases by using a similar command as below, make sure you have enough space on disk.

```
cd /var/backups
mkdir xivocc
cd xivocc

docker run --rm --link xivocc_pgxivocc_1:db -v $(pwd):/backup -e PGPASSWORD=***
↪xivoxc/pgxivocc pg_dump -h db -U postgres --format=c -f /backup/xuc_rights dump
↪xuc_rights
```

(continues on next page)

(continued from previous page)

```
docker run --rm --link xivocc_pgxivocc_1:db -v $(pwd):/backup -e PGPASSWORD=*** ␣
↪xivoxc/pgxivocc pg_dump -h db -U postgres --format=c -f /backup/spagobi_dump ␣
↪spagobi
docker run --rm --link xivocc_pgxivocc_1:db -v $(pwd):/backup -e PGPASSWORD=*** ␣
↪xivoxc/pgxivocc pg_dump -h db -U postgres --format=c -f /backup/recording_dump ␣
↪recording
docker run --rm --link xivocc_pgxivocc_1:db -v $(pwd):/backup -e PGPASSWORD=*** ␣
↪xivoxc/pgxivocc pg_dump -h db -U postgres --format=c -f /backup/xivo_stats_dump ␣
↪xivo_stats
```

Restore

You may restore a backup using a similar command (to be adapted)

```
#Restore

cd /var/backups/xivocc

docker run --rm -it --link xivocc_pgxivocc_1:db -v $(pwd):/backup xivoxc/pgxivocc ␣
↪pg_restore -j 10 -h db -c -U postgres -d xivo_stats /backup/xivo_stats_dump
docker run --rm -it --link xivocc_pgxivocc_1:db -v $(pwd):/backup xivoxc/pgxivocc ␣
↪pg_restore -j 10 -h db -c -U postgres -d spagobi /backup/spagobi_dump
docker run --rm -it --link xivocc_pgxivocc_1:db -v $(pwd):/backup xivoxc/pgxivocc ␣
↪pg_restore -j 10 -h db -c -U postgres -d xuc_rights /backup/xuc_rights_dump
docker run --rm -it --link xivocc_pgxivocc_1:db -v $(pwd):/backup xivoxc/pgxivocc ␣
↪pg_restore -j 10 -h db -c -U postgres -d recording /backup/recording_dump
```

4.3 Troubleshooting

Important:

- When reading this section, keep in mind the *Architecture & Flows* diagram.
- If you want to troubleshoot your installation see *XiVOcc Installation Troubleshooting*
- For desktop application see *Troubleshoot Application*

4.3.1 Troubleshooting

Transfers using DTMF

When transfering a call using DTMF (*1) you get an *invalid extension* error when dialing the extension.

The workaround to this problem is to create a preprocess subroutine and assign it to the destinations where you have the problem.

Under *Services* → *IPBX* → *IPBX configuration* → *Configuration files* add a new file containing the following dialplan:

```
[allow-transfer]
exten = s,1,NoOp(## Setting transfer context ##)
same = n,Set(__TRANSFER_CONTEXT=<internal-context>)
same = n,Return()
```

Do not forget to substitute <internal-context> with your internal context.

Some places where you might want to add this preprocess subroutine is on queues and outgoing calls to be able to transfer the called person to another extension.

Fax detection

XiVO **does not currently support Fax detection**. The following describe a workaround to use this feature. The behavior is to answer all incoming (external) call, wait for a number of seconds (4 in this example) : if a fax is detected, receive it otherwise route the call normally.

Note: This workaround works only :

- on incoming calls towards an User (and an User only),
- if the incoming trunk is a DAHDI or a SIP trunk,
- if the user has a voicemail which is activated and with the email field filled
- XiVO >= 13.08 (needs asterisk 11)

Be aware that this workaround will probably not survive any upgrade.

---

1. In the Web Interface and under *Services* → *IPBX* → *IPBX configuration* → *Configuration files* add a new file named *fax-detection.conf* containing the following dialplan:

```
;; Fax Detection
[pre-user-global-faxdetection]
exten = s,1,NoOp(Answer call to be able to detect fax if call is external AND␣
↪user has an email configured)
same  =   n,GotoIf($["${XIVO_CALLORIGIN}" = "extern"]?:return)
same  =   n,GotoIf(${XIVO_USEREMAIL}?:return)
same  =   n,Set(FAXOPT(faxdetect)=yes) ; Activate dynamically fax detection
same  =   n,Answer()
same  =   n,Wait(4) ; You can change the number of seconds it will wait for␣
↪fax (4 to 6 is good)
same  =   n,Set(FAXOPT(faxdetect)=no) ; If no fax was detected deactivate␣
↪dyamically fax detection (needed if you want directmedia to work)
same  =   n(return),Return()

exten = fax,1,NoOp(Fax detected from ${CALLERID(num)} towards ${XIVO_DSTNUM} -␣
↪will be sent upon reception to ${XIVO_USEREMAIL})
same  =     n,GotoIf($["${CHANNEL(channeltype)}" = "DAHDI"]?
↪changeechocan:continue)
same  =     n(changeechocan),Set(CHANNEL(echocan_mode)=fax) ; if chan type is␣
↪dahdi set echo canceller in fax mode
same  =     n(continue),Gosub(faxtomail,s,1(${XIVO_USEREMAIL}))
```

2. In the file `/etc/xivo/asterisk/xivo_globals.conf` set the global user subroutine to `pre-user-global-faxdetection` : this subroutine will be executed each time a user is called:

```
XIVO_PRESUBR_GLOBAL_USER = pre-user-global-faxdetection
```

3. Reload asterisk configuration (both for dialplan and dahdi):

```
asterisk -rx 'core reload'
```

Berofos Integration with PBX

You can use a Berofos failover switch to secure the ISDN provider lines when installing a XiVO in front of an existing PBX. The goal of this configuration is to mitigate the consequences of an outage of the XiVO : with this

equipment the ISDN provider links could be switched to the PBX directly if the XiVO goes down.

XiVO **does not offer natively** the possibility to configure Berofos in this failover mode. This section describes a workaround.

Logical view:

```
                +------+                                +-----+
-- Provider ----| XiVO | -- ISDN Interconnection  --| PBX | -- Phones
                +------+                                +-----+
```

Connection:

```
  +-------------Bero*fos----------------+
  | A        B        C        D       |
  | o o o o  o o o o  o o o o  o o o o |
  +-+-+------+-+------+-+------+-+-----+
    | |      | |      | |      | |
   / /       | |      | |      | |
  / /    +---------+   / /   +----------+
2 T2     |  XiVO   |  / /    |   PBX    |
         +---------+ / /     +----------+
             | |   / /
             \ \__/ /
              \____/
```

The following describes how to configure your XiVO and your Berofos.

1. Follow the Berofos general configuration (firmware, IP, login/password) described in the the *Berofos Installation and Configuration* page.
2. When done, apply these specific parameters to the berofos:

```
bnfos --set scenario=1   -h 10.105.2.26 -u admin:berofos
bnfos --set mode=1       -h 10.105.2.26 -u admin:berofos
bnfos --set modedef=1    -h 10.105.2.26 -u admin:berofos
bnfos --set wdog=1       -h 10.105.2.26 -u admin:berofos
bnfos --set wdogdef=1    -h 10.105.2.26 -u admin:berofos
bnfos --set wdogitime=60 -h 10.105.2.26 -u admin:berofos
```

3. Add the following script `/usr/local/sbin/berofos-workaround`:

```
#!/bin/bash
# Script workaround for berofos integration with a XiVO in front of PABX

res=$(/usr/sbin/service asterisk status)
does_ast_run=$?
if [ $does_ast_run -eq 0 ]; then
    /usr/bin/logger "$0 - Asterisk is running"
    # If asterisk is running, we (re)enable wdog and (re)set the mode
    /usr/bin/bnfos --set mode=1 -f fos1 -s
    /usr/bin/bnfos --set modedef=1 -f fos1 -s
    /usr/bin/bnfos --set wdog=1 -f fos1 -s

    # Now 'kick' berofos ten times each 5 seconds
    for ((i == 1; i <= 10; i += 1)); do
        /usr/bin/bnfos --kick -f fos1 -s
        /bin/sleep 5
    done
else
    /usr/bin/logger "$0 - Asterisk is not running"
fi
```

4. Add execution rights to script:

```
chmod +x /usr/local/sbin/berofos-workaround
```

5. Create a cron to launch the script every minutes `/etc/cron.d/berofos-cron-workaround`:

```
# Workaround to berofos integration
MAILTO=""

*/1 * * * * root /usr/local/sbin/berofos-workaround
```

Upgrading from XiVO 1.2.3

1. There is an issue with `xivo-libsccp` and `pf-xivo-base-config` during an upgrade from 1.2.3:

```
dpkg: error processing /var/cache/apt/archives/pf-xivo-base-config_13%3a1.2.4-
↪1_all.deb (--unpack):
trying to overwrite '/etc/asterisk/sccp.conf', which is also in package xivo-
↪libsccp 1.2.3.1-1
...
Errors were encountered while processing:
/var/cache/apt/archives/pf-xivo-base-config_13%3a1.2.4-1_all.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

2. You have to remove `/var/lib/dpkg/info/xivo-libsccp.conffiles`:

```
rm /var/lib/dpkg/info/xivo-libsccp.conffiles
```

3. You have to edit `/var/lib/dpkg/info/xivo-libsccp.list` and remove the following line:

```
/etc/asterisk/sccp.conf
```

4. and remove `/etc/asterisk/sccp.conf`:

```
rm /etc/asterisk/sccp.conf
```

5. Now, you can launch `xivo-upgrade` to finish the upgrade process

CTI server is unexpectedly terminating

If you observes that your CTI server is sometimes unexpectedly terminating with the following message in `/var/log/xivo-ctid.log`:

```
(WARNING) (main): AMI: CLOSING
```

Then you might be in the case where asterisk generates lots of data in a short period of time on the AMI while the CTI server is busy processing other thing and is not actively reading from its AMI connection. If the CTI server takes too much time before consuming some data from the AMI connection, asterisk will close the AMI connection. The CTI server will terminate itself once it detects the connection to the AMI has been lost.

There's a workaround to this problem called the ami-proxy, which is a process which buffers the AMI connection between the CTI server and asterisk. This should only be used as a last resort solution, since this increases the latency between the processes and does not fix the root issue.

To enable the ami-proxy, you must:

1. Add a file `/etc/systemd/system/xivo-ctid.service.d/ami-proxy.conf`:

```
mkdir -p /etc/systemd/system/xivo-ctid.service.d
cat >/etc/systemd/system/xivo-ctid.service.d/ami-proxy.conf <<EOF
[Service]
```

(continues on next page)

(continued from previous page)

```
Environment=XIVO_CTID_AMI_PROXY=1
EOF
systemctl daemon-reload
```

2. Restart the CTI server:

```
systemctl restart xivo-ctid.service
```

If you are on a XiVO cluster, you must do the same procedure on the slave if you want the ami-proxy to also be enabled on the slave.

To disable the ami-proxy:

```
rm /etc/systemd/system/xivo-ctid.service.d/ami-proxy.conf
systemctl daemon-reload
systemctl restart xivo-ctid.service
```

Agents receiving two ACD calls

Warning: Procedure was removed since bug was fixed in asterisk version shipped in 2017.LTS1 (2017.03)

PostgreSQL localization errors

The database and the underlying database cluster used by XiVO is sensitive to the system locale configuration. The locale used by the database and the database cluster is set when XiVO is installed. If you change your system locale without particular attention to PostgreSQL, you might make the database and database cluster temporarily unusable.

When working with locale and PostgreSQL, there's a few useful commands and things to know:

- `locale -a` to see the list of currently available locales on your system
- `locale` to display information about the current locale of your shell
- `grep ^lc_ /etc/postgresql/9.4/main/postgresql.conf` to see the locale configuration of your database cluster
- `sudo -u postgres psql -l` to see the locale of your databases
- the `/etc/locale.gen` file and the associated `locale-gen` command to configure the available system locales
- `systemctl restart postgresql.service` to restart your database cluster
- the PostgreSQL log file located at `/var/log/postgresql/postgresql-9.4-main.log`

Note: You can use any locale with XiVO as long as it uses an UTF-8 encoding.

Database cluster is not starting

If the database cluster doesn't start and you have the following errors in your log file:

```
LOG:  invalid value for parameter "lc_messages": "en_US.UTF-8"
LOG:  invalid value for parameter "lc_monetary": "en_US.UTF-8"
LOG:  invalid value for parameter "lc_numeric": "en_US.UTF-8"
LOG:  invalid value for parameter "lc_time": "en_US.UTF-8"
FATAL:  configuration file "/etc/postgresql/9.4/main/postgresql.conf" contains␣
↪errors
```

(continues on next page)

(continued from previous page)

Then this usually means that the locale that is configured in `postgresql.conf` (here `en_US.UTF-8`) is not currently available on your system, i.e. does not show up the output of `locale -a`. You have two choices to fix this issue:

- either make the locale available by uncommenting it in the `/etc/locale.gen` file and running `locale-gen`
- or modify the `/etc/postgresql/9.4/main/postgresql.conf` file to set the various `lc_*` options to a locale that is available on your system

Once this is done, restart your database cluster.

Can't connect to the database

If the database cluster is up but you get the following error when trying to connect to the `asterisk` database:

```
FATAL:  database locale is incompatible with operating system
DETAIL:  The database was initialized with LC_COLLATE "en_US.UTF-8",  which is not␣
↪recognized by setlocale().
HINT:  Recreate the database with another locale or install the missing locale.
```

Then this usually means that the database locale is not currently available on your system. You have two choices to fix this issue:

- either make the locale available by uncommenting it in the `/etc/locale.gen` file, running `locale-gen` and restarting your database cluster
- or *recreate the database using a different locale*

Error during the upgrade

Then you are mostly in one of the cases described above. Check your log file.

Error while restoring a database backup

If during a database restore, you get the following error:

```
pg_restore: [archiver (db)] Error while PROCESSING TOC:
pg_restore: [archiver (db)] Error from TOC entry 4203; 1262 24745 DATABASE␣
↪asterisk asterisk
pg_restore: [archiver (db)] could not execute query: ERROR:  invalid locale name:
↪"en_US.UTF-8"
    Command was: CREATE DATABASE asterisk WITH TEMPLATE = template0 ENCODING =
↪'UTF8' LC_COLLATE = 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8';
```

Then this usually means that your database backup has a locale that is not currently available on your system. You have two choices to fix this issue:

- either make the locale available by uncommenting it in the `/etc/locale.gen` file, running `locale-gen` and restarting your database cluster
- or if you want to restore your backup using a different locale (for example `fr_FR.UTF-8`), then restore your backup using the following commands instead:

```
sudo -u postgres dropdb asterisk
sudo -u postgres createdb -l fr_FR.UTF-8 -O asterisk -T template0 asterisk
sudo -u postgres pg_restore -d asterisk asterisk-*.dump
```

Error during master-slave replication

Then the slave database is most likely not using an UTF-8 encoding. You'll need to *recreate the database using a different locale*

Changing the locale (LC_COLLATE and LC_CTYPE) of the database

If you have decided to change the locale of your database, you must:

- make sure that you have enough space on your hard drive, more precisely in the file system holding the `/var/lib/postgresql` directory. You'll have, for a moment, two copies of the `asterisk` database.
- prepare for a service interruption. The procedure requires the services to be restarted twice, and the system performance will be degraded while the database with the new locale is being created, which can take a few hours if you have a really large database.
- make sure the new locale is available on your system, i.e. shows up in the output of `locale -a`

Then use the following commands (replacing `fr_FR.UTF-8` by your locale):

```
xivo-service restart all
sudo -u postgres createdb -l fr_FR.UTF-8 -O asterisk -T template0 asterisk_
↪newlocale
sudo -u postgres pg_dump asterisk | sudo -u postgres psql -d asterisk_newlocale
xivo-service stop
sudo -u postgres psql <<'EOF'
DROP DATABASE asterisk;
ALTER DATABASE asterisk_newlocale RENAME TO asterisk;
EOF
xivo-service start
```

You should also modify the `/etc/postgresql/9.4/main/postgresql.conf` file to set the various `lc_*` options to the new locale value.

For more information, consult the official documentation on PostgreSQL localization support.

Originate a call from the Asterisk console

It is sometimes useful to ring a phone from the asterisk console. For example, if you want to call the `1234` extension in context `default`:

```
channel originate Local/1234@default extension 42@xivo-callme
```

WebRTC

- *http.conf* - asterisk's webserver must accept connection from outside, the listen address must be updated, for the sake of simplicity let's use 0.0.0.0, you can also pick an address of one of the network interfaces:

```
[general]
enabled=yes
bindaddr=0.0.0.0
bindport=5039
prefix=
tlsenable=yes
tlsbindaddr=127.0.0.1:5040
tlscertfile=/usr/share/xivo-certs/server.crt
tlsprivatekey=/usr/share/xivo-certs/server.key
servername=XiVO PBX
```

Do not forget to reload the configuration by the *module reload http* command on the Asterisk CLI.

- *rtp.conf* - the ICE support must be activated:

```
;
; RTP Configuration
;
[general]
;
; RTP start and RTP end configure start and end addresses
;
; Defaults are rtpstart=5000 and rtpend=31000
;
rtpstart=10000
rtpend=20000
;
; Whether to enable or disable UDP checksums on RTP traffic
;
;rtpchecksums=no
;
; The amount of time a DTMF digit with no 'end' marker should be
; allowed to continue (in 'samples', 1/8000 of a second)
;
;dtmftimeout=3000
icesupport=yes
stunaddr=stun.l.google.com:19302
```

The configuration is reloaded by *module reload res_rtp_asterisk.so*.

- WebRTC requires DTLS keys to be generated in */etc/asterisk/keys*. If you need to manually generate the DTLS certificates following instructions on the Asterisk Wiki: https://wiki.asterisk.org/wiki/display/AST/Secure+Calling+Tutorial. You just need to generate the TLS certificates (first call of *ast_tls_cert*), other steps are not necessary. Make sure asterisk can read files by executing: `chown -R asterisk.asterisk /etc/asterisk/keys`

4.3.2 Xuc & Xucmgt (CC & UC applications)

XUC overview page

XUC overview page available at @XUC_IP:PORT, usually @SERVER_IP:8090. You have to check if the "Internal configuration cache database" contains agents, queues etc.

XUC sample page

XUC sample page available at @XUC_IP:PORT/sample, usually @SERVER_IP:8090/sample. You can use this page to check user login and other API functions. CCManager, agent and assistant web use functions available on the sample page.

XUC Internal Metrics

Internal metrics are also available - see *XUC Internal Metrics* page.

XUC Internal Metrics

Contents

- *XUC Internal Metrics*

- *Introduction*
- *Configuration*
  - *Enable JMX*
  - *Explore JMX*
  - *Expose JMX through REST*
- *Metrics description*
  - *Historical metrics*
  - *New metrics*
  - *Other JVM metrics*

Introduction

The XUC process exposes some metrics to troubleshoot or monitor the health of the process. Some of these metrics were previously exposed in a sub-page of the XUC overview page. The metrics are not exposed using the JMX technology available in java.

Configuration

Enable JMX

JMX is enabled by default in java but only available on the local machine running the process. Moreover as we are using docker, it's only available inside the docker container itself. To make it available from the outside of the container and host running the XUC process, you need to explicitely configure it to do so.

In the following configuration, replace

- JMX_PORT with the port number where you want to expose the JMX
- JMX_HOST by the docker host IP address (not the container one but the IP of the server running docker)

Edit your docker compose file (/etc/docker/compose/docker-xivocc.yml) and change the configuration of the xuc container:

```
xuc:
  [...]
  ports:
  - JMX_PORT:JMX_PORT
  [...]

  environment:
  - JAVA_OPTS=-Xms512m -Xmx1024m -DtechMetrics.jmxReporter=true -Dcom.sun.
↪management.jmxremote -Dcom.sun.management.jmxremote.port=JMX_PORT -Dcom.sun.
↪management.jmxremote.local.only=false -Dcom.sun.management.jmxremote.
↪authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.rmi.server.
↪hostname=JMX_HOST -Dcom.sun.management.jmxremote.rmi.port=JMX_PORT
  [...]
```

For example, if we have JMX_PORT=15701 and JMX_HOST=192.168.228.100, you should set

```
xuc:
  [...]
  ports:
  - 15701:15701
```

(continues on next page)

(continued from previous page)

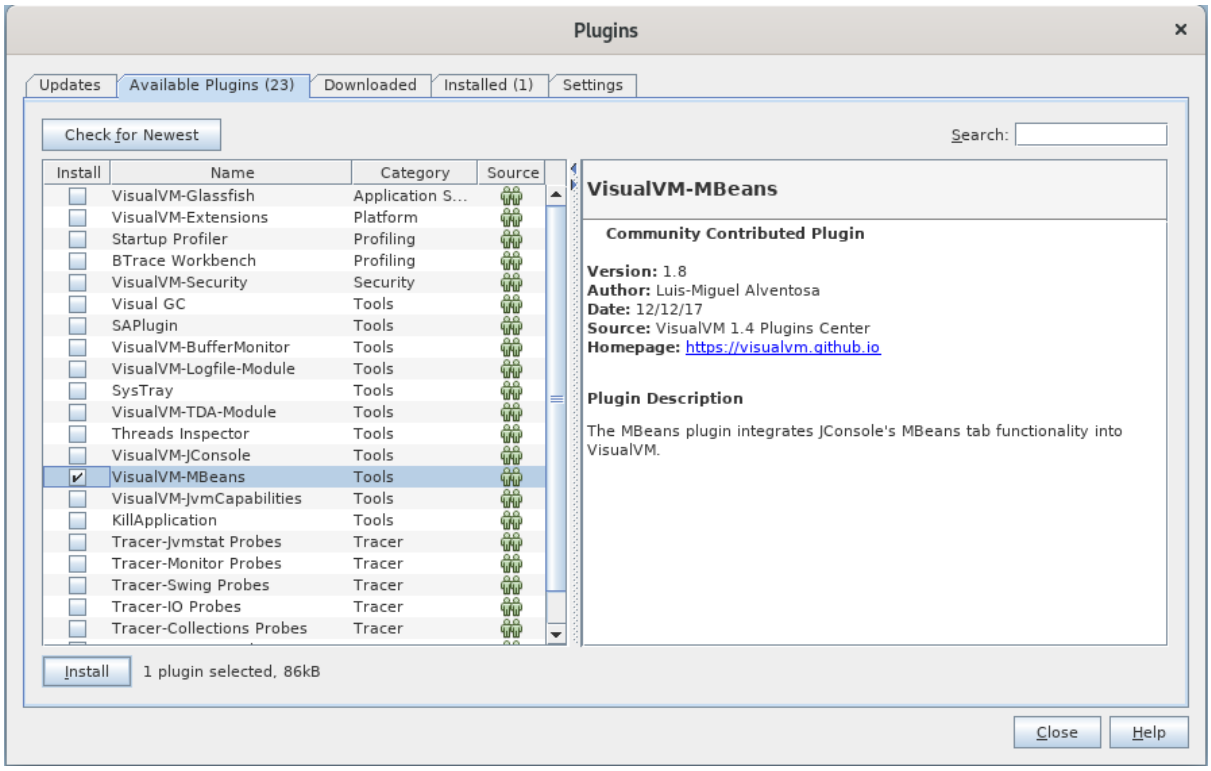
```
  [...]

  environment:
  - JAVA_OPTS=-Xms512m -Xmx1024m -DtechMetrics.jmxReporter=true -Dcom.sun.
↪management.jmxremote -Dcom.sun.management.jmxremote.port=15701 -Dcom.sun.
↪management.jmxremote.local.only=false -Dcom.sun.management.jmxremote.
↪authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.rmi.server.
↪hostname=192.168.228.100 -Dcom.sun.management.jmxremote.rmi.port=15701
  [...]
```

Then restart the XUC process with the new configuration by running the command `xivocc-dcomp up -d xuc`

Explore JMX

Once restarted you can then use tools to explore the metrics: jconsole, visualvm with MBeans plugin, eclipse,... For example, here are the steps to configure visualvm and explore the JMX metrics:

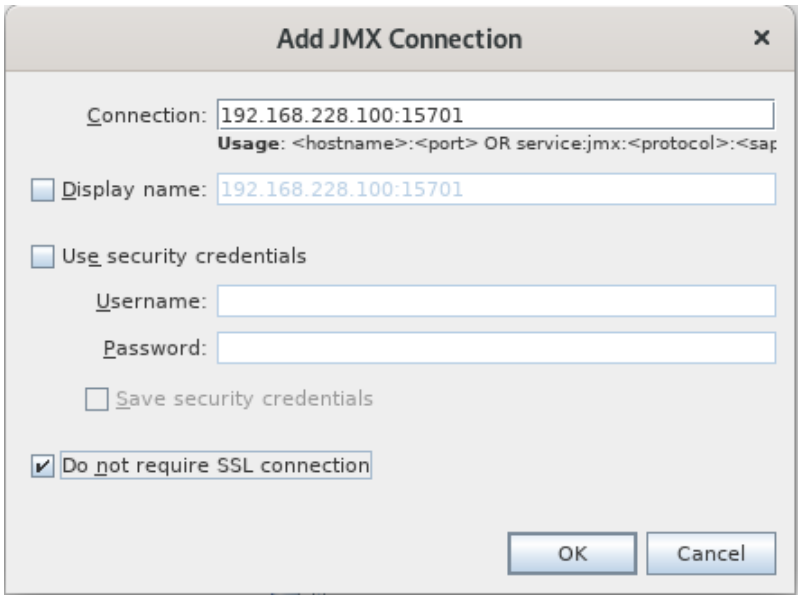
- Download and install visualvm https://visualvm.github.io/
- Enable MBeans plugin

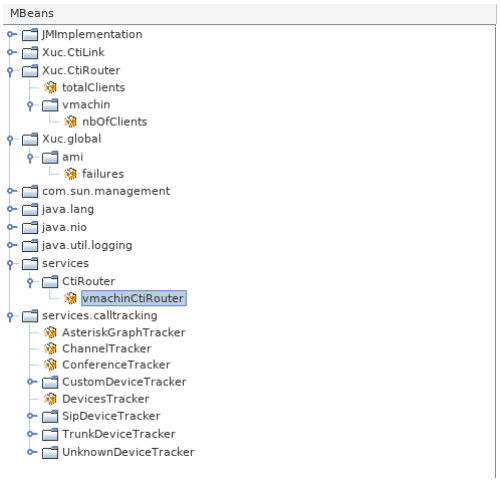


- Add remote host
- Double click on process under the newly added host
- Click on the MBeans tab
- Explore tree

Expose JMX through REST

Alternatively you could integrate a JMX plugin to your running process which allows to gather JMX metrics over HTTP. You need to download jolokia JVM agent from their website: https://jolokia.org/ and transfer the jar file


Add JMX Connection
Connection:
192.168.228.100:15701
Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap
Display name:
192.168.228.100:15701
Use security credentials
Username:
Password:
Save security credentials
Do not require SSL connection
OK
Cancel


MBeans
JMImplementation
Xuc.CtiLink
Xuc.CtiRouter
totalClients
vmachin
nbOfClients
Xuc.global
ami
failures
com.sun.management
java.lang
java.nio
java.util.logging
services
CtiRouter
vmachinCtiRouter
services.calltracking
AsteriskGraphTracker
ChannelTracker
ConferenceTracker
CustomDeviceTracker
DevicesTracker
SipDeviceTracker
TrunkDeviceTracker
UnknownDeviceTracker

on the server hosting the XUC container (for example in `/etc/docker/jolokia/jolokia-jvm-1.6.2-agent.jar`).

Then you should change your docker compose configuration for the xuc process in `docker-xivocc.yml`:

```
xuc:
  [...]
  ports:
  - JMX_HTTP_PORT:JMX_HTTP_PORT
  [...]

  environment:
  - JAVA_OPTS=-Xms512m -Xmx1024m -DtechMetrics.jmxReporter=true -javaagent:/opt/
↪jolokia/jolokia-jvm-1.6.2-agent.jar=port=JMX_HTTP_PORT,host=JMX_HTTP_HOST
  [...]

  volumes:
  - /etc/docker/jolokia:/opt/jolokia
```

Then restart the XUC process with the new configuration by running the command `xivocc-dcomp up -d xuc`. The JMX metrics are now available over HTTP, see jolokia website for help on available endpoints: https://jolokia.org/documentation.html

Here are some example url to test: * List all jmx metrics available: `curl http://JMX_HTTP_HOST:JMX_HTTP_PORT/jolokia/list` * Get metrics of a specific service: `curl http://JMX_HTTP_HOST:JMX_HTTP_PORT/jolokia/read/services.calltracking:type=AsteriskGraphTracker`

Metrics description

Historical metrics

These metrics were previously exposed in a sub-page of the XUC overview page.

- Xuc.CtiLink.*: Information on the link per user between XUC and ctid on the XiVO PBX
- Xuc.CtiRouter.totalClients: Total number of client connected to the XUC
- Xuc.CtiRouter.<username>.nbOfClients: Number of client connected to the XUC with the given <username>
- Xuc.global.ami.failures: Number of failure/lost connection to the asterisk AMI

New metrics

- services.CtiRouter.<username>CtiRouter: Information on the currently connected <username>
- services.calltracking.AsteriskGraphTracker * GraphSize: Size of the call graph * LoopDetected: Number of call loop detected * Notifications: Number of Notifications published internally * Watchers: Number of object monitoring the graph
- services.calltracking.ChannelTracker * HangupEvents: Number of Hangup received since the process started * NewChannelEvents: Number of channel created since the process started * Notifications: Number of Notifications published internally * Watchers: Number of object monitoring the channels
- services.calltracking.ConferenceTracker * Conferences: Number of conferences * Participants: Number of participants
- services.calltracking.DevicesTracker.Devices: Number of monitored device

- services.calltracking.SipDeviceTracker.<SIP_PEER_NAME>: Information about the SIP peer (Phone device) * Calls: Number of active calls * ChannelEvent: Number of channel event received * PartyInformation: Number of event received from remote party * PathsFromChannel: Number of event received from the AsteriskGraphTracker
- services.calltracking.TrunkDeviceTracker.<TRUNK_NAME>: Information about the trunk, same information as in SipDeviceTracker
- services.calltracking.CustomDeviceTracker.<CUSTOM_NAME>: Information about the custom device, same information as in SipDeviceTracker
- services.calltracking.UnknownDeviceTracker.<CUSTOM_NAME>: Information about other asterisk device, same information as in SipDeviceTracker

Other JVM metrics

You may also find these metrics interesting when troubleshooting the process: * java.lang.Memory.HeapMemoryUsage: Information about the java heap memory * java.lang.GarbageCollector: Information about the java garbage collector process

Agent states after XUC restart

Please see the note in *restarting* XUC server with active calls.

4.3.3 NGINX - proxy web

Basic check

On the standard HTTP port of the machine (80) you have the fingerboard page.

Docker says nginx is restarting

- Check logs for missing files or links, nginx refuses to start if one of servers is not accessible, e.g. xuc is down.

4.3.4 PostgreSQL

Container keeps on restarting after upgrade

After upgrading Docker it can sometimes happen that the container *xivocc_pgxivocc_1* gets stuck in restarting mode. Logs show the following error

```
FATAL:  data directory "/var/lib/postgresql/data" has group or world access
DETAIL:  Permissions should be u=rwx (0700).
```

Error is caused by wrong permissions on the PostgreSQL data directory. Follow the below steps in order to fix the issue:

1. Find the location of the directory inside the container information:

   ```
   docker inspect xivocc_pgxivocc_1 | grep volumes
   ```

2. Command output should give something like this:

   ```
   "Source": "/var/lib/docker/volumes/
   ↪82898d65ae41174865211ff709c12b1f5e7b3c1b7d26e73500b6e6c7cffb3f10/_data",
   ```

3. Use the result above to change the permissions:

```
chmod -R 700 /var/lib/docker/volumes/
↪82898d65ae41174865211ff709c12b1f5e7b3c1b7d26e73500b6e6c7cffb3f10/_data
```

4. Restart the container:

```
xivocc-dcomp restart pgxivocc
```

4.3.5 Reporting

xivo_replic does not replicate call data

After experiencing a 'no space left on device' and restarting containers, it can sometimes happen that the data from XiVO is not replicated anymore. Container *xivocc_replic_1* logs show the following error:

```
xivo_replic_1       | liquibase.exception.LockException: Could not acquire change␣
↪log lock.  Currently locked by fe80:0:0:0:42:acff:fe11:8%eth0␣
↪(fe80:0:0:0:42:acff:fe11:8%eth0) since 4/10/17 3:28 PM
```

Error is caused by a lock in liquibase db. Follow the below steps in order to fix the issue:

> **Warning:** This problem should not happen, so you should know what you are doing and not follow this procedure blindly.

1. With xivo_stats, xivo_replic and pack_reporting containers stopped find if there still is an active lock:

```
xivo_stats=# select * from databasechangeloglock;
id | locked |       lockgranted       |                              lockedby
----+--------+-------------------------+-------------------------------------
↪--------------------------
1 | t      | 2017-04-10 15:28:10.684 | fe80:0:0:0:42:acff:fe11:8%eth0␣
↪(fe80:0:0:0:42:acff:fe11:8%eth0)
```

2. If so, delete the lock:

```
xivo_stats=# truncate databasechangeloglock;
xivo_stats=# select * from databasechangeloglock;
id | locked | lockgranted | lockedby
----+--------+-------------+----------
```

3. Restart the containers:

```
xivocc-dcomp start xivo_replic
xivocc-dcomp start pack_reporting
xivocc-dcomp start xivo_stats
```

Now, if xivo_stats is stuck in exit (126), try a *docker rm -v {xivo_stats contener id}* followed by a *xivocc-dcomp up -d*.

CHAPTER 5

IPBX Configuration Guide

5.1 Advanced Configuration

This section describes the advanced system configuration.

5.1.1 XiVO General Settings

XiVO offers the possibility to configure the general settings via the *Configuration* → *Management* → *General* page.

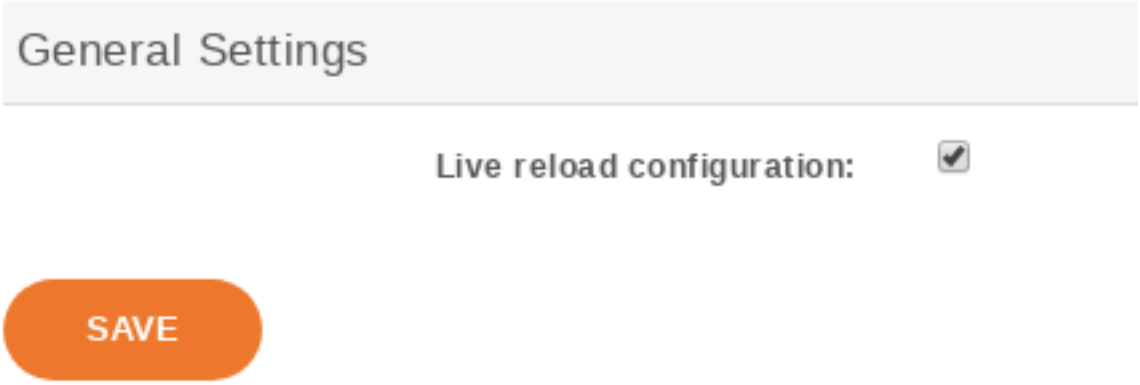


Fig. 1: Configure XiVO General Settings

Live reload configuration permit to reload its configuration on command received from WEBI (this option is enabled by default).

If you deactivate Live reload, apart from commands to xivo-ctid service, no config update command will be sent to udpate the services configuration. You must then reload the configuration accordingly.

The table below lists the parameters that will need a manual reload:

Services –> IPBX –> IPBX Settings			
Menu	**Section**	**Parameter**	**Reload command**
Users	General	*On-Hold Music*, *Caller ID*	`sip reload`
	Lines	*all*	`sip reload` `dialplan reload`
	Services	*Enable supervision*	`dialplan reload`
	Voicemail	*all*	`voicemail reload` `sip reload`
	Func Keys	*Type Customized* with *Supervision* activated	`dialplan reload` + phone reboot
Groups	General	• *Name*, *Ring strategy*, • *User reachability timeout*, • *Time before retrying a call to a user*, • *Call a member already in line*, • *On-Hold Music*	`queue reload all`
		Number	`dialplan reload`
	Users	*all*	`queue reload` `members`
	Application	N.A.	N.A.
	Call permissions	N.A.	N.A.
	No answer	N.A.	N.A.
	Schedules	N.A.	N.A.
Voicemails	*all*	*all*	`voicemail reload`
Conference rooms	General	*Number*, *PIN code*, *Organizer PIN code*	`dialplan reload`
Services –> IPBX –> Call management			
Menu	**Section**	**Parameter**	**Reload command**
Incoming calls	*all*	N.A.	N.A.
Outgoing calls	Exten	*all*	`dialplan reload`
Call permissions	*all*	N.A.	N.A.
Call filters	*all*	N.A.	N.A.
Call pickups	Interceptors	*all*	`sip reload`
	Intercepted	*all*	`sip reload`
Schedule	*all*	N.A.	N.A.
Services –> Call center –> Call management			
Menu	**Section**	**Parameter**	**Reload command**
Agents	*all*	*all*	`queue reload all`
Queues	General	*Name*, *Ring strategy*, *On-Hold Music*	`queue reload all`
		Number	`dialplan reload`
	Announces	*all*	`queue reload all`
	Members	*all*	`queue reload` `members`
	Application	N.A.	N.A.
	No answer	N.A.	N.A.
	Advanced	*all*	`queue reload all`
	Schedules	N.A.	N.A.

Continued on next page

Table 1 – continued from previous page

	Diversions	N.A.	N.A.
	Qualifications	N.A.	N.A.
Agents	*all*	*all*	`queue reload all`
Queues penalties	*all*	*all*	`queue reload all`
Skills	*all*	*all*	`queue reload all`
Skill rules	*all*	*all*	`queue reload all`
Qualifications	*all*	N.A.	N.A.

5.1.2 Activate Directmedia with SIP Provider & DTMF RFC 2833

Important: This page describes how to enable Directmedia on your XiVO PBX in **the specific following context**:

- with a SIP Provider authorizing directmedia (or requiring it),
- and with this SIP Provider using DTMF according to RFC 2833

Other use cases are not covered here.

Introduction

Directmedia is an Asterisk feature to optimize network streams. By default, when asterisk establishes a call between two phones, it establishes the media stream (voice or video RTP stream) via itself. Then, if A calls B, the media stream goes from A to asterisk and then from asterisk to B:

```
Without directmedia:

                            +-----------+
   A <---- SIP & RTP ----> |   XiVO    | <---- SIP & RTP ----> B
                            +-----------+
```

Enabling Directmedia make media streams to flow directly between A and B:

```
With directmedia activated:

                            +-----------+
   A <------- SIP -------> |   XiVO    | <------- SIP -------> B
                            +-----------+
   ^                                                          ^
   ^_________________________  RTP  _________________________^
```

This is particularly useful when your XiVO PBX and phones are not located on the same site. In this case, activating Directmedia could dramatically save network bandwidth.

Prerequisites

- As written in the note above we only cover the case where your SIP Provider will accept media stream directly between your endpoints and its media gateways.
- Also it describes how to configure your system if your SIP Provider accepts DTMF in RFC 2833.
- And you **must** ensure that network routing is working between your different endpoints - as enabling directmedia means that media stream will flow directly between endpoints (phones, gateways, operators etc.).
- Note also that in this context network QoS should be taken into account with great care, but is outside the scope of this page.

Configuration

Directmedia activation

First of all, enable directmedia on your XiVO PBX:

1. Go to *Services -> IPBX -> General settings -> SIP Protocol -> Default*
2. Set *Redirect media stream* to **Not behind NAT**
3. Set *DTMF* to **RFC2833**

Device configuration

Device configuration must be changed to activate the RFC 2833 DTMF mode:

1. Go to *Configuration -> Provisioning -> Template device*
2. Edit the *Default template device*
3. and set *DTMF* to **RTP-out-of-band**
4. **This requires a synchronization of all Devices** (see *Synchronize a device*)

SIP Provider Trunk Configuration

Verify that your SIP Provider trunk configuration has also directmedia activated:

1. Edit your SIP Provider trunk,
2. In tab *Advanced* set *Redirect media streams* to **Not behind NAT**
3. In tab *Signalling* set *DTMF* to **RFC2833**

Other SIP Trunks

If you have other SIP trunks on your installation, you should verify that the DTMF mode is according to what the endpoint support. And, as it is not covered in this guide, we recommend that you deactivate directmedia:

1. Edit the SIP trunk,
2. In tab *Advanced* set *Redirect media streams* to **No**
3. In tab *Signalling* set *DTMF* to the one supported by the endpoint.

User configuration

For directmedia to work with DTMF set in RFC 2833 mode you **must** deactivate the following options for users:

1. Go to *Services -> IPBX -> IPBX Settings -> Users*,
2. edit each user - **except Switchboard user**,
3. go to tab *Services*
4. and un-check all following options:

- *Enable call transfer*
- *Enable online call recording*

Groups configuration

For directmedia to work with DTMF set in RFC 2833 mode you **must** deactivate the following options for groups:

1. Go to *Services -> IPBX -> IPBX Settings -> Groups*,
2. edit each group,
3. go to tab *Application*
4. and un-check all following options:

- *Allow called one to transfer the caller*
- *Allow caller to transfer the call*
- *Allow called on to record the call*
- *Allow caller to record the call*

Queues configuration

For directmedia to work with DTMF set in RFC 2833 mode you **must** deactivate the following options for queues:

1. Go to *Services -> Call Center -> Configuration -> Queues*,
2. edit each queue - **except the Switchboard queue**,
3. go to menu *Application*
4. and un-check all following options:

- *Allow callee to hang up the call*
- *Allow caller to hang up the call*
- *Allow callee to transfer the call*
- *Allow caller to transfer the call*
- *Allow callee to record the call*
- *Allow caller to record the call*

Conclusion

After having followed the above configuration steps, your XiVO will be configured for directmedia with a SIP Provider using DTMF in RFC 2833 mode.

Limitations

This configuration is not compatible with:

- XiVO Client transfer method:
  - after having un-checked the option *Enable call transfer* transfer won't be possible from legacy XiVO Client.
- Switchboard: calls towards Switchboard will be without directmedia. Directmedia will be activated on the call after having been transfered (as the legacy transfer method must be used for Switchboard CTI application)
- Jitterbuffer: jitterbuffer must be disabled on your XiVO PBX.
- Recording: if the call recording is activated for calls, it will disable Directmedia on these calls.

- ChanSpy: listening to an agent (see CCManager application documentation) will disable temporarily the directmedia. When listening is stopped, media flow will be re-established directly between agent and caller.

5.1.3 Telephony certificates

XiVO offers the possibility to create and manage X.509 certificates via the the *Configuration* → *Management* → *Certificates* page.

These certificates can be used for:

- enabling SIP TLS
- enabling encryption between the CTI server and the XiVO clients

For the certificate used for HTTPS, see *HTTPS certificate*.

Creating certificates

You can add a certificate by clicking on the add button at the top right of the page. You'll then be shown this page:

You should look at the *examples* if you don't know which attributes to set when creating your certificates.

Removing certificates

When removing a certificate, you should remove all the files related to that certificates.

> **Warning:** If you remove a certificate that is used somewhere in XiVO, then you need to manually reconfigure that portion of XiVO.
>
> For example, if you remove the certificate files used for SIP TLS, then you need to manually disable SIP TLS or asterisk will look for certificate file but it won't be able to find them.

Examples

In the following examples, if a field is not specified than you should leave it at its default value.

Creating certificates for SIP TLS

You need to create both a CA certificate and a server certificate.

CA certificate:

- *Name* : phones-CA
- *Certification authority (checkbox)* : checked
- *Autosigned* : checked
- *Valid end date* : at least one month in the future
- *Common name* : the FQDN (Fully Qualified Domain Name) of your XiVO
- *Organization* : your organization's name, or blank
- *Email* : your email or organization's email

Server certificate:

- *Name* : phones
- *Certification authority (select)* : phones-CA

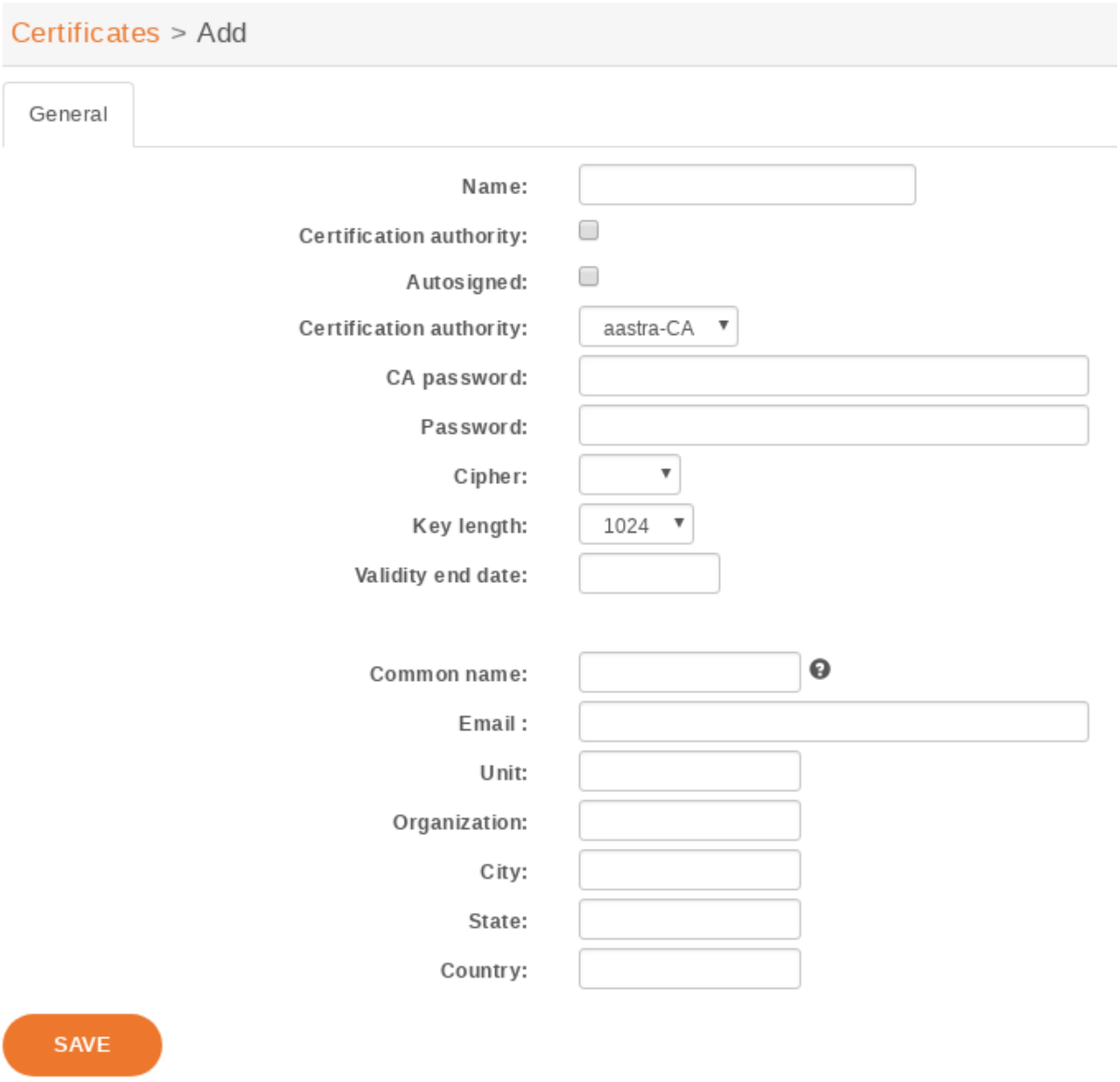


Fig. 2: Adding a certificate

- *Valid end date* : at least one month in the future
- *Common name* : the FQDN of your XiVO
- *Organization* : your organization's name, or blank
- *Email* : your email or organization's email

Creating certificate for CTI server

- *Name* : xivo-ctid
- *Autosigned* : checked
- *Valid end date* : at least one month in the future
- *Common name* : the FQDN of your XiVO
- *Organization* : your organization's name, or blank
- *Email* : your email or organization's email

Warning: You must *not* set a password for the certificate. If the certificate is password protected, the CTI server will not be able to use it.

5.2 Boss Secretary Filter

The boss secretary filter allow to set a secretary or a boss role to a user. Filters can then be created to filter calls directed to a boss using different strategies.

5.2.1 Quick Summary

In order to be able to use the boss secretary filter you have to :

- Select a boss role for one the users
- Select a secretary role for one ot the users
- Create a filter to set a strategy for this boss secretary filter
- Add a function key for the user boss and the user secretary

5.2.2 Defining a Role

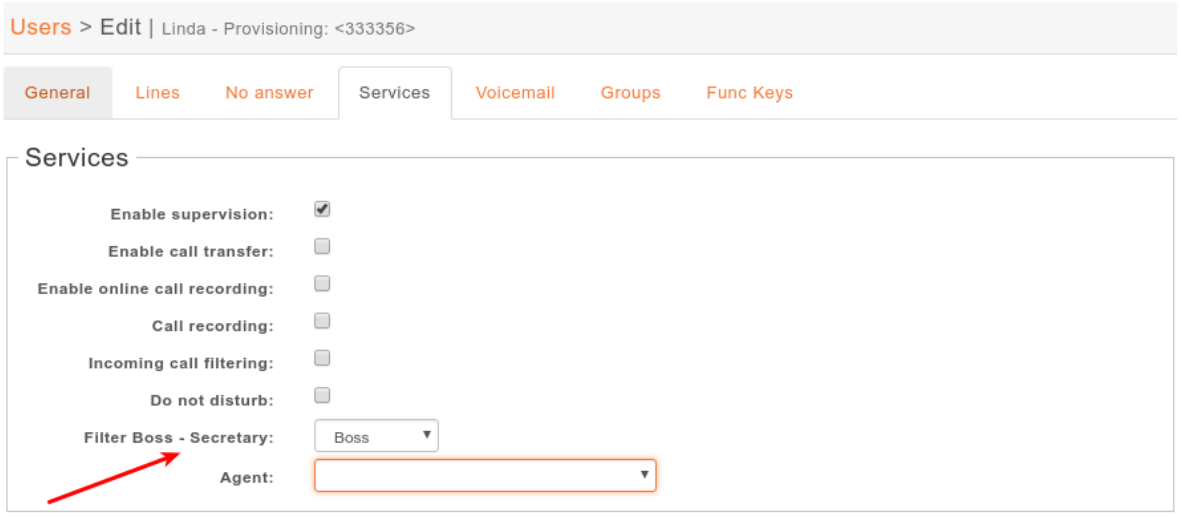
The secretary or boss role can be set in the user's configuration page under the service tab. To use this feature, at least one boss and one secretary must be defined.

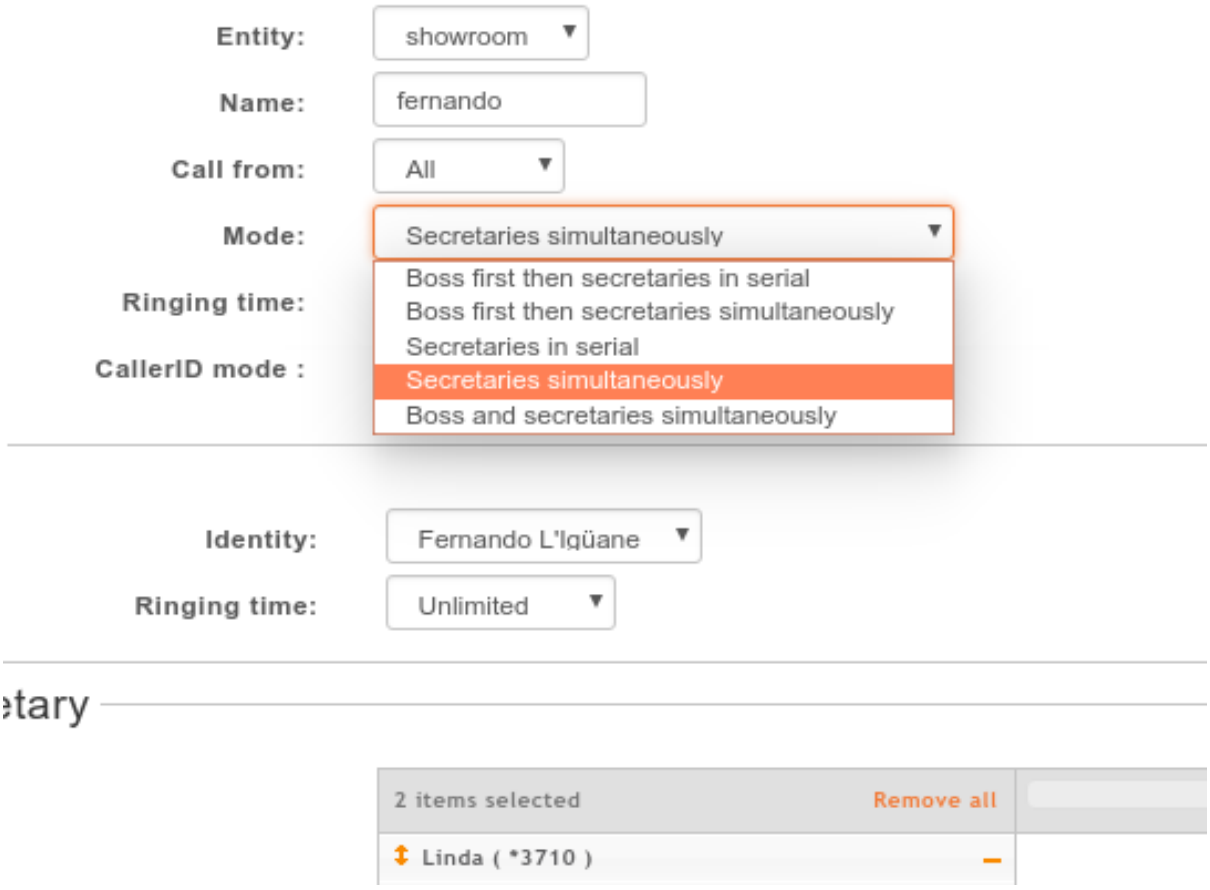
5.2.3 Creating a Filter

The filter is used to associate a boss to one or many secretaries and to set a ring strategy. The call filter is added in the *Services* → *IPBX* → *Call management* → *Call filters* page.

Different ringing strategies can be applied :

- Boss rings first then all secretaries one by one
- Boss rings first then secretaries are all ringing simultaneously
- Secretaries ring one by one


Users > Edit | Linda - Provisioning: <333356>
General
Lines
No answer
Services
Voicemail
Groups
Func Keys
Services
Enable supervision:
Enable call transfer:
Enable online call recording:
Call recording:
Incoming call filtering:
Do not disturb:
Filter Boss - Secretary:
Boss
Agent:


Entity:
showroom
Name:
fernando
Call from:
All
Mode:
Secretaries simultaneously
Boss first then secretaries in serial
Boss first then secretaries simultaneously
Secretaries in serial
Secretaries simultaneously
Boss and secretaries simultaneously
Ringing time:
CallerID mode :
Identity:
Fernando L'Iglüane
Ringing time:
Unlimited
etary
2 items selected
Remove all
Linda (*3710)

- Secretaries are all ringing simultaneously
- Boss and secretaries are ringing simultaneously
- Change the caller id if the secretary wants to know which boss was initialy called

When one of serial strategies is used, the first secretary called is the last in the list. The order can be modified by drag and drop in the list.

5.2.4 Usage

The call filter function can be activated and deactivated by the boss or the secretary using the *37 extension. The extension is defined in *IPBX services* > *Extensions*.

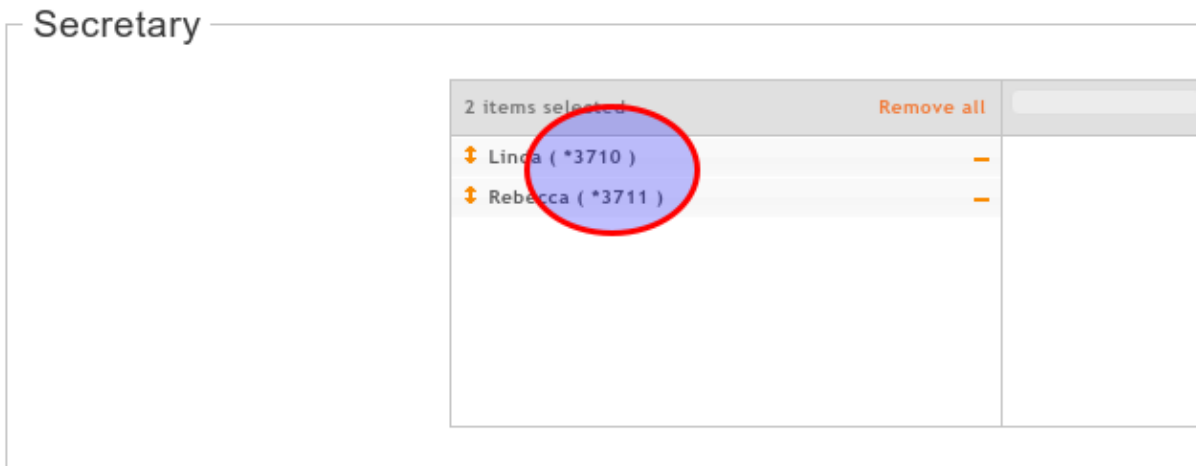
The call filter has to be activated for each secretary if more than one is defined for a given boss.

The extension to use is `*37<callfilter member id>`.

In this example, you would set 2 `Func Keys` `*373` and `*374` on the Boss.

On the secretary `Jina LaPlante` you would set `*373`.

On the secretary `Ptit Nouveau` you would set `*374`.



5.2.5 Function Keys

A more convenient way to active the boss secretary filter is to assign a function key on the boss' phone or the secretary's phone. In the user's configuration under `Func Keys`. A function key can be added for each secretaries of a boss.

If supervision is activated, the key will light up when filter is activated for this secretary. If a secretary also has a function key on the same boss/secretary combination the function key's BLF will be in sync between each phones.

Warning: With SCCP phones, you must configure a custom `Func Keys`.

5.3 Call Completion

The call completion feature (or CCSS, for Call Completion Supplementary Services) in XiVO allows for a caller to be automatically called back when a called party has become available.

1. To illustrate, let's say Alice attempts to call Bob.

2. Bob is currently on a phone call with Carol, though, so Bob rejects the call from Alice
3. Alice then dials *40 to request call completion.
4. Once Bob has finished his phone call, Alice will be automatically called back by the system.
5. When she answers, Bob will be called on her behalf.

This feature has been introduced in XiVO in version 14.17.

5.3.1 Description

Call completion can be used in two scenarios:

- when the called party is busy (Call Completion on Busy Subscriber)
- when the called party doesn't answer (Call Completion on No Response)

We have already discussed the busy scenario in the introduction section.

Let's now illustrate the no answer scenario:

1. Alice attempts to call Bob.
2. Bob doesn't answer the phone. Alternatively, Alice hangs up before Bob has the time to answer the call.
3. Alice then dial *40 to request call completion.
4. When Bob's phone becomes busy and then is no longer busy, Alice is automatically called back.
5. When she answers, Bob will be called on her behalf.

The important thing to note here is step #4. Bob's phone needs to become busy and then no longer busy for Alice to be called back. This means that if Bob was away when Alice called him, but when he came back he did not received nor placed any call, then Alice will not be called back.

In fact, in all scenarios, after call completion has been requested by the caller, the called phone needs to transition from busy to no longer busy for the caller to be called back. This means that in the following scenario:

1. Alice attempts to call Bob.
2. Bob is currently on a phone call, so he doesn't answer the call from Alice.
3. Bob finish his call a few seconds later.
4. Alice then dials *40 to request call completion (Bob is not busy anymore).

Then, for Alice to be called back, Bob needs to become busy and then not busy.

If Alice is busy when Bob becomes not busy, then the call completion callback will only happen after both Alice and Bob are not busy.

When call completion is active, it can be cancelled by dialing the *40 extension.

Some timers governs the use of call completion. These are:

- offer timer: the time the caller has to request call completion. Defaults to 30 (seconds).
- busy available timer: when call completion on busy subscriber is requested, if this timer expires before the called party becomes available, then the call completion attempt will be cancelled. Defaults to 900 (seconds).
- no response available timer: similar to the "busy available timer", but when call completion on no response is requested. Defaults to 900 (seconds).
- recall timer: when the caller who requested call completion is called back, how long the original caller's phone rings before giving up. Defaults to 30 (seconds).

It's currently impossible to modify the value of these timers in XiVO.

Special Scenarios

There are four special scenarios:

- the call completion will not activate
- the call completion will activate and call back for the original called party
- the call completion will activate and call back for the rerouted called party
- the call completion will activate and call back for the original called party but fail to join him

Call completion will not activate

It is not possible to activate call completion in the following two scenarios.

First scenario: Alice tries to call Bob, but Bob has currently reached its "simultaneous calls" limit. When activating call completion, Alice hears that the call completion can not be activated.

Note: The "simultaneous calls" option is configured per user via the XiVO web interface.

Second scenario: Alice tries to call Bob, but the call is redirected to Charlie.

This occurs when Bob redirects/rejects the call with any of the following:

- Unconditional call forwarding towards Charlie
- Closed schedule towards Charlie
- Call permission forbidding Alice to call Bob
- Preprocess subroutine forwarding the call towards Charlie

Call completion will activate and call back for the original called party

Scenario: Alice tries to call Bob, but the call is redirected to Charlie. When activating call completion, Alice hears that the call completion is activated and eventually Alice is called back to speak with Bob.

This occurs when Bob redirects/rejects the call with any of the following:

- No-answer call forwarding towards Charlie
- Busy call forwarding towards Charlie

Call completion will activate and call back for the rerouted called party

Scenario: Alice tries to call Bob, but the call is redirected to Charlie. When activating call completion, Alice hears that the call completion is activated and eventually Alice is called back to speak with Charlie.

This occurs when Bob redirects the call with any of the following:

- Boss-Secretary filter to the secretary Charlie

Call completion will activate and call back for the original called party but fail to join him

Scenario: Alice tries to call Bob, but the call is redirected to Charlie. When activating call completion, Alice hears that the call completion is activated and eventually Alice is called back to speak with Bob. But when Alice answers, Bob is not called. If Alice activates call completion again, she will hear that the call completion was cancelled.

This occurs when Bob redirects/rejects the call with any of the following:

- Do Not Disturb mode
- a new call forwarding rule that was applied after Alice activated call completion:
  - Unconditional call forwarding towards Charlie
  - Closed schedule towards Charlie
  - Call permission forbidding Alice to call Bob
  - Preprocess subroutine forwarding the call towards Charlie

Limitations

- Call completion can only be used with SIP lines. It can't be used with SCCP lines.
- It can't be used with outgoing calls and incoming calls, except if these calls are passing through a customized trunk of type Local.
- It can't be used with groups or queues.
- The call completion feature can't be enabled only for a few users; either all users have access to it, or none.

5.3.2 Configuration

The call completion extension is enabled via the *Services* → *IPBX* → *IPBX services* → *Extensions* page, in the *General* tab.

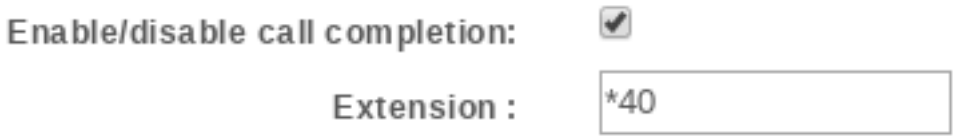


Fig. 3: Call Completion Extension

If your XiVO has been installed in version 14.16 or earlier, then this extension is by default disabled. Otherwise, this extension is by default enabled.

5.4 Call Permissions

You can manage call permissions via the *Services* → *IPBX* → *Call management* → *Call permissions* page.

Call permissions can be used for:

- denying a user from calling a specific extension
- denying a user of a group from calling a specific extension
- denying a specific extension on a specific outgoing call from being called
- denying an incoming call coming from a specific extension from calling you

More than one extension can match a given call permission, either by specifying more than one extension for that permission or by using extension patterns.

You can also create permissions that allow a specific extension to be called instead of being denied. This make it possible to create a general "deny all" permission and then an "allow for some" one.

Finally, instead of unconditionally denying calling a specific extension, call permissions can instead challenge the user for a password to be able to call that extension.

As you can see, you can do a lot of things with XiVO's call permissions. They can be used to create fairly complex rules. That said, it is probably *not* a good idea to so because it's pretty sure you'll get it somehow wrong.

5.4.1 Notes

Internal Calls

Note: You can only deny **internal calls** towards **users'** extensions.

If you apply a *Deny* rule towards an extension to a User, it will only work if it is the extension of a user (or an extension that go through an outgoing call). It won't work if the extension is the number of a Queue or a Conference room.

Forwarded Calls

Note: Forwarded calls will be checked against the **forwarder** call permissions (if forwarder is a user).

When a call is forwarded, the call permissions will be checked against the **forwarder** permissions (not the caller permissions).

This applies only to a call:

- forwarded by
  - **a user**
- thanks to
  - XiVO **forward extensions** (like unconditional forward (**21*), forward on no answer (**22*) ...),
  - or his phone **SIP forward** feature
- towards either:
  - another **user**
  - or a number which will go through an outgoing call

5.4.2 Examples

Note that when creating or editing a call permission, you must at least:

- fill the *Name* field
- have one extension / extension pattern in the *Extensions* field

Denying a user from calling a specific extension

- Add the extension in the extensions list
- In the *Users* tab, select the user

Note: User's *Rightcall Code* (*Services -> IPBX -> IPBX Settings -> Users* under *Services* tab) overwrite all password call permissions for the user.

Warning: The extension can be anything but it will only work if it's the extension of a user or an extension that pass through an outgoing call. It does *not* work, for example, if the extension is the number of a conference room.

Denying a user of a group from calling a specific extension

First, you must create a group and add the user to this group. Note that groups aren't required to have a number.

Then,

- Add the extension in the extensions list
- In the *Groups* tab, select the group

Denying users from calling a specific extension on a specific outgoing call

- Add the extension in the extensions list
- In the *Outgoing calls* tab, select the outgoing call

Note that selecting both a user and an outgoing call for the same call permission doesn't mean the call permission applies only to that user. In fact, it means that the user can't call that extension and that the extension can't be called on the specific outgoing call. This in redundant and you will get the same result by not selecting the user.

Denying an incoming call coming from a specific extension from calling you

Call permissions on incoming calls are semantically different from the other scenarios since the extension that you add to the permission will match the extension of the caller (i.e. the caller number) and *not* the extension that the caller dialed (i.e. the callee number).

- Add the extension in the extensions list.
- In the *Incoming calls* tab, select the incoming call

5.5 Call Logs

Call logs are pre-generated from CEL entries. The generation is done automatically by xivo-call-logd. xivo-call-logs is also run nightly to generate call logs from CEL that were missed by xivo-call-logd.

Note: The oldest call logs are periodically removed. See *Purge Logs* for more details.

5.5.1 Search Dashboard

Call logs can be accessed using the menu *Services* → *IPBX* → *Call management* → *Call Logs* page.

Specifying no start date returns all available call logs. Specifying a start date and no end date returns all call logs from start date until now.

Call logs are presented in a CSV format. Here's an example:

```
Call Date,Caller,Called,Period,user Field
2015-01-02T00:00:00,Alice (1001),1002,2,userfield
```

The CSV format has the following specifications:

- field names are listed on the first line

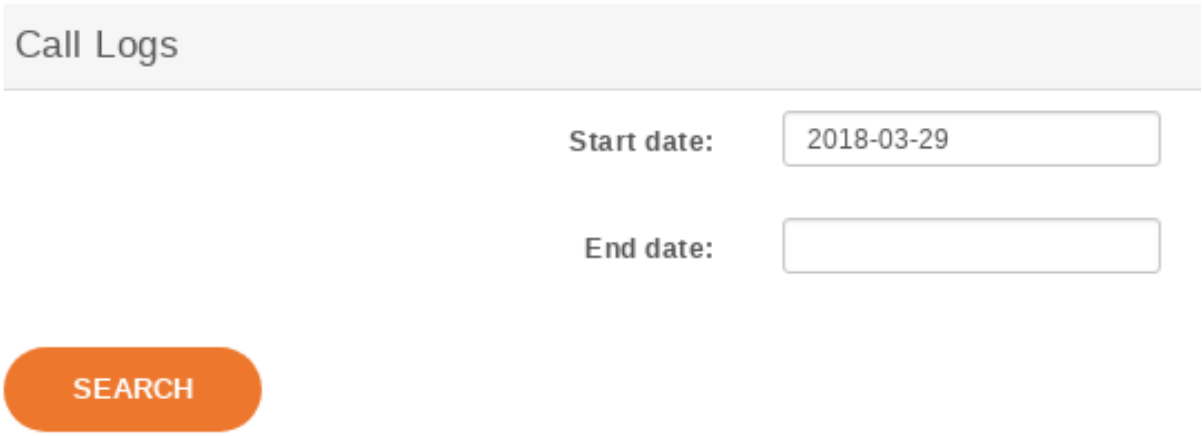


Fig. 4: Calls Records Dashboard

- fields are separated by commas: `,`
- if there is a comma in a field value, the value is surrounded by double quotes: `"`
- the UTF-8 character encoding is used

5.5.2 REST API

Call logs are also available from *xivo-confd REST API*.

5.5.3 Manual generation

Call logs can also be generated manually. To do so, log on to the target XiVO server and run:

```
xivo-call-logs
```

To avoid running for too long in one time, the call logs generation is limited to the N last unprocessed CEL entries (default 20,000). This means that successive calls to `xivo-call-logs` will process N more CELs, making about N/10 more calls available in call logs, going further back in history, while processing new calls as well.

You can specify the number of CEL entries to consider. For example, to generate calls using the 100,000 last unprocessed CEL entries:

```
xivo-call-logs -c 100000
```

5.6 Conference Room

5.6.1 Adding a conference room

In this example, we'll add a conference room with number 4010.

First, you need to define a conference room number range for the ''default" context via the *Services -> IPBX -> IPBX Configuration -> Contexts* page.

You can then create a conference room via the *Services -> IPBX -> IPBX Settings -> Conference rooms* page.

In this example, we have only filled the ''Name" and ''Number" fields, the others have been left to their default value.

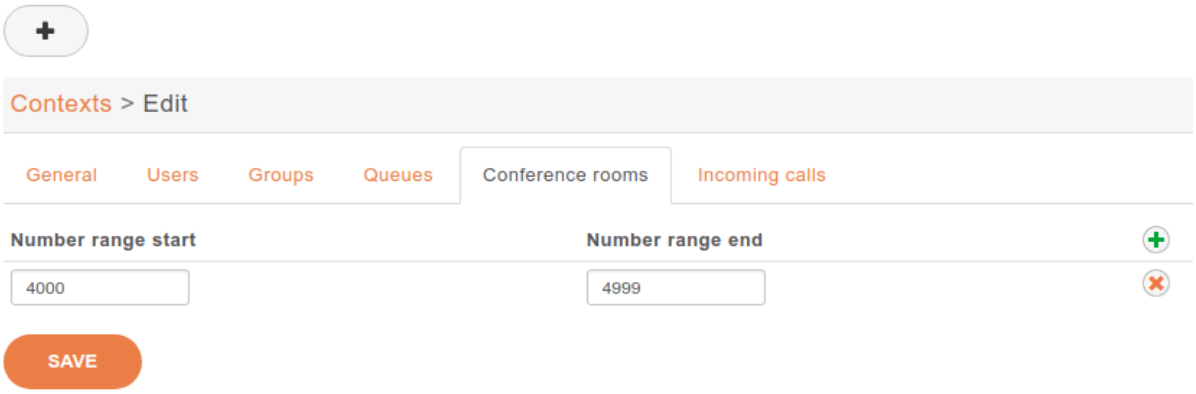


Fig. 5: Adding a conference room number range to the default context

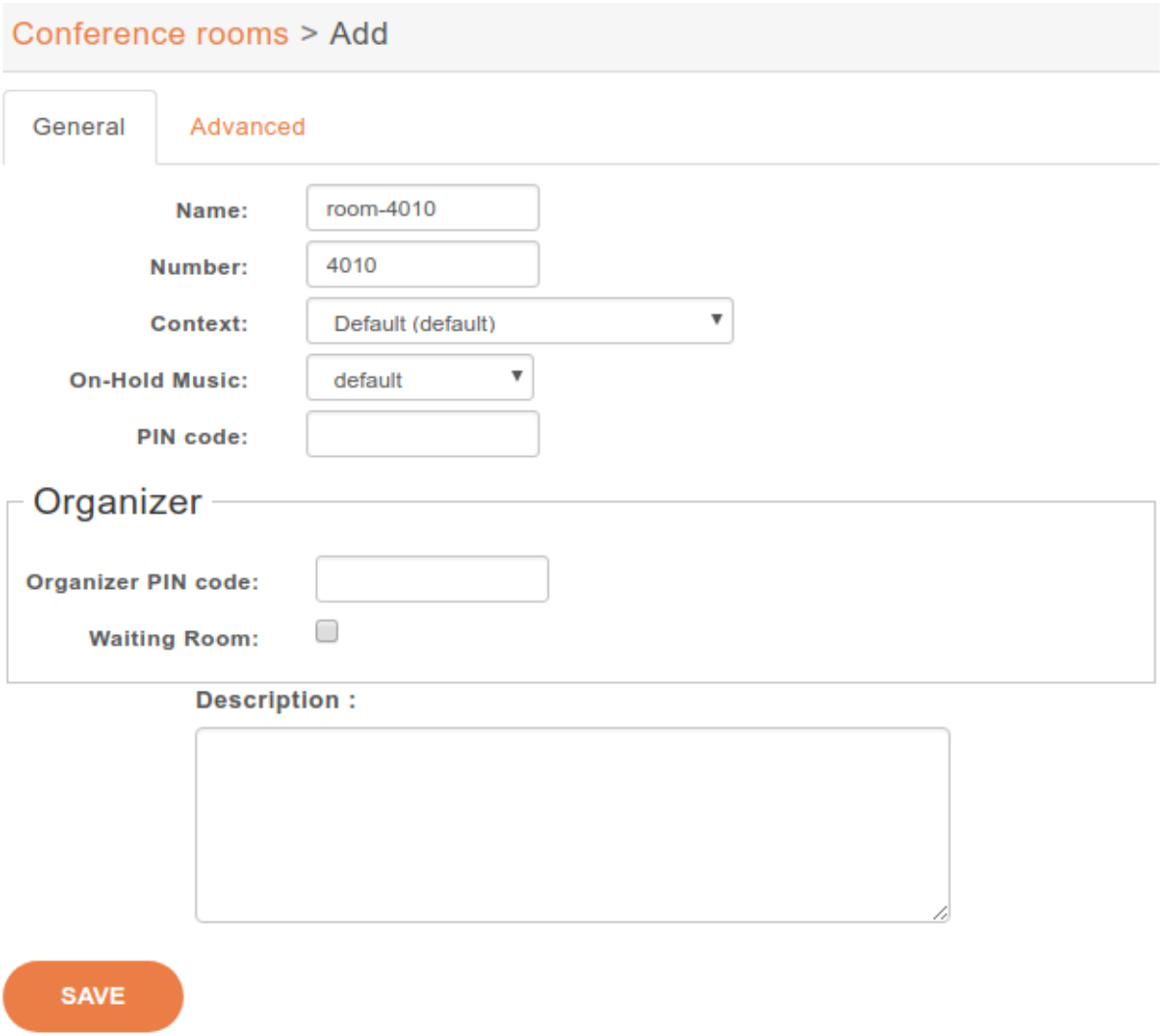


Fig. 6: Creating conference room 4010

As you can see, there's quite a few options when adding / editing a conference room. Here's a description of the most common one:

General Tab

- *Pin code*: Protects your conference room with a PIN number. People trying to join the room will be asked for the PIN code.
- *Organizer Pin Code*: This pin code is used to define an organizer for the conference, organizer will have the right to mute or kick participants with the *Conferences* monitoring in uc assistant application.
- *Waiting room*: Once this option is checked the users are not able to speak in the room and have to wait the organizer to join the conference.

Note: You must define a general pin code if you want to be able to organize conferences and use the organizers features.

Advanced Tab

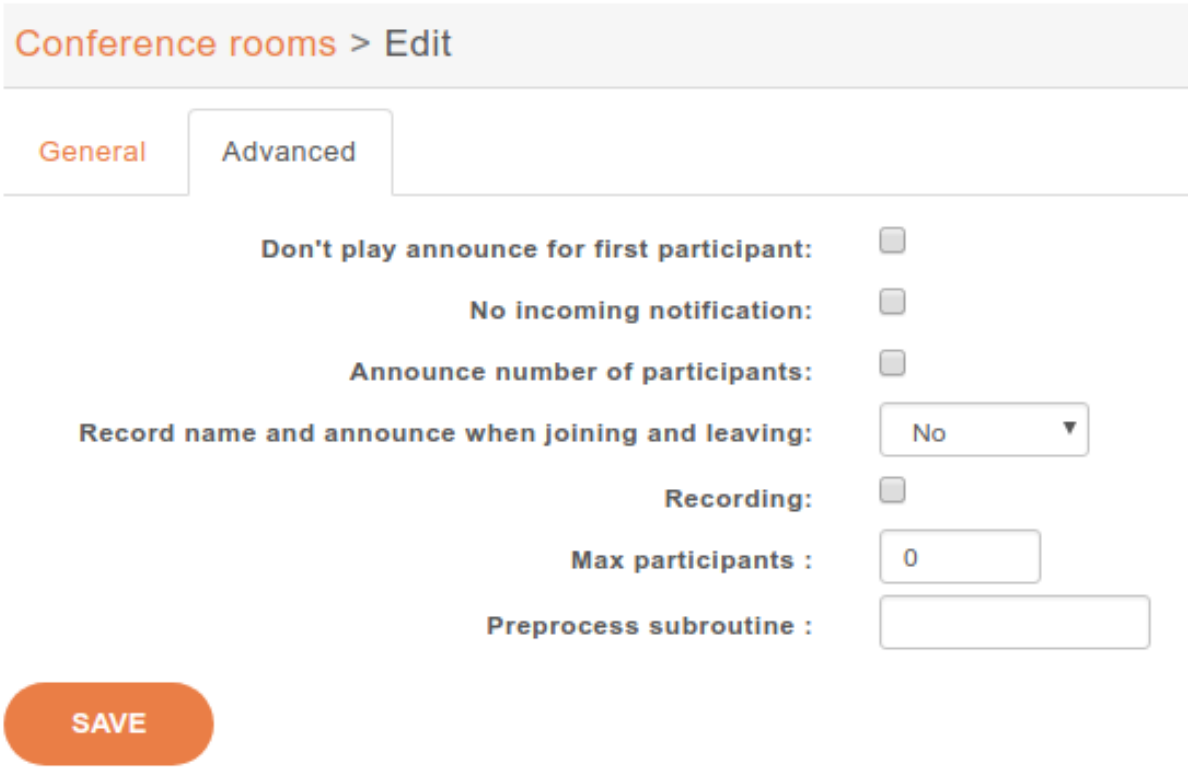


Fig. 7: Conference room configuration advanced tab

- *Don't play announce for first participant*: First user joining will not get any announce.
- *No incoming notification*: Users will not hear incoming announces. Deactivates the sound when user enters / leaves conference. It also deactivates the name recording feature, but not the announce of number of participants.
- *Announce number of participants*: Announce user(s) count on joining a conference.
- *Record name and announce when joining and leaving*: The user is allowed to record his name that will be heard by the other conference attendees. Can be reviewed afert record (yes) or not (No Review).

- *Recording*: This conference room is recorded.
- *Max participants*: Define the maximum number of participants, default to 0.
- *Preprocess subroutine*: See *Subroutine*.

5.7 CTI Server

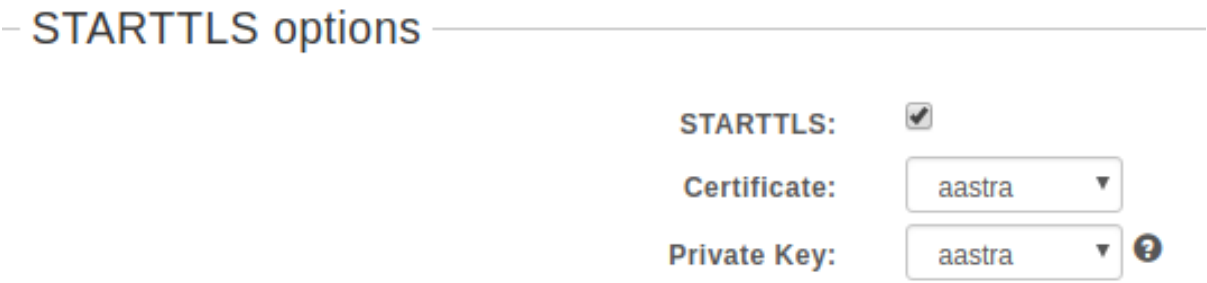
The CTI server configuration options can be found in the web-interface under the services tab.

5.7.1 General Options

The general options allow the administrator to manage network connections between the CTI server and the clients.

The section named `STARTTLS options` allows the administrator to enable encrypted communications between the clients and xivo-ctid and specify the certificate and private keys to use.

If no certificate and private key is configured, xivo-ctid will use the ones located in `/usr/share/xivo-certs`.



Parting options are used to isolate XiVO users from each other. These options should be used when using the same XiVO for different enterprises.

Context separation is based on the user's line context. A user with no line is not the member of any context and will not be able to do anything with the CTI client.

Note: xivo-dird must be restarted to take into account this parameter.

5.7.2 Authentication

xivo-ctid uses xivo-auth to authenticate users. The default authentication backend is *xivo_user*. To change the authentication backend, add a configuration file in */etc/xivo-ctid/conf.d* with the following content:

```
auth:
    backend: backend_name
```

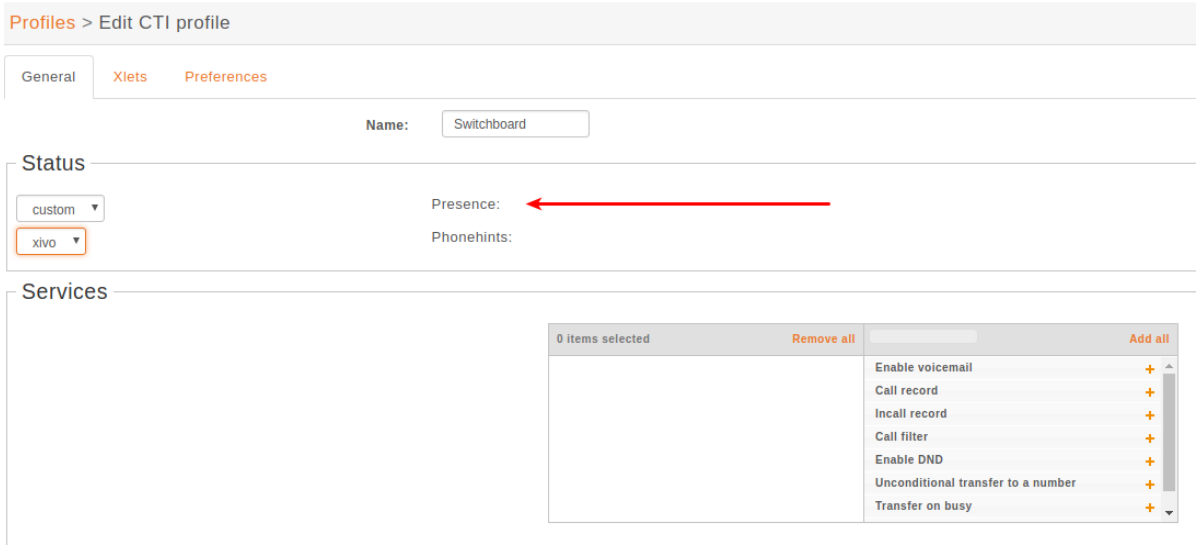
where *backend* name is the name of an enabled *xivo-auth* *Backends Plugins*.

	Presence Name	Description	Action
☐	berightback	Bientôt de retour	
☐	erreursaisie	Erreur Saisie	
☐	postappel	Autre Travail	
☐	away	Sorti	
☐	outtolunch	Parti Manger	
☐	donotdisturb	Ne pas déranger	
☐	disconnected	Déconnecté	
☐	backoffice	Back Off.	
☐	chat	Chat	
☐	available	Disponible	

5.7.3 Presence Option

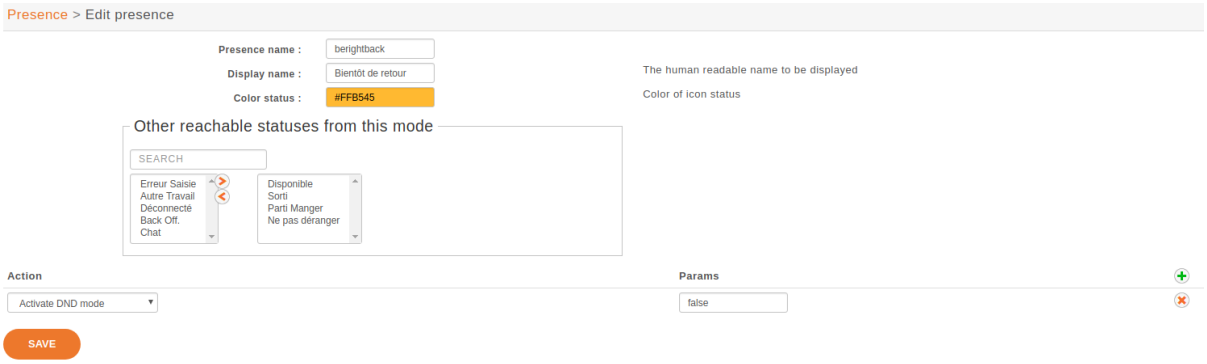
In the *Status* menu, under *Presences*, you can edit presences group. The default presence group is *francais*. When editing a group, you will see a list of presences and their descriptions.

To use another presence group, you can edit the CTI profile you are using and select the appropriate presence group for that profile.



Available configuration

- *Presence name* is the name of the presence
- *Display name* is the human readable representation of this presence
- *Color status* is the color associated to this presence
- *Other reachable statuses* is the list of presence that can be switched from this presence state
- *Actions* are post selection actions that are triggered by selecting this presence



Actions

action	param
Enable DND	*{'true','false'}*
Pause agent in all queues	
Unpause agent in all queues	
Agent logoff	

5.7.4 Enable encryption

To enable encryption of CTI communications between server and clients, you have to enable STARTTLS in *CTI Server → General settings → General*

Custom certificates can be added in *Configuration → Certificates* and used in *CTI Server → General settings → General*

In your XiVO Client, in the menu *XiVO Client → Configure → Connection*, click on the lock icon.

Note: A client which chooses to use encryption will not be able to connect to a server that does not have STARTTLS enabled.

Warning: For now, there is no mechanism for strong authentication of the server. The connection is encrypted, but the identity of the server is not verified.

5.7.5 CTI profiles

The CTI profiles define which features are made available to a user. You can configure which profile will be used by a user in the menu *IPBX → PBX Settings → Users*:

You can also customize the default profiles or add new profiles in the menu *CTI Server → Profiles*:

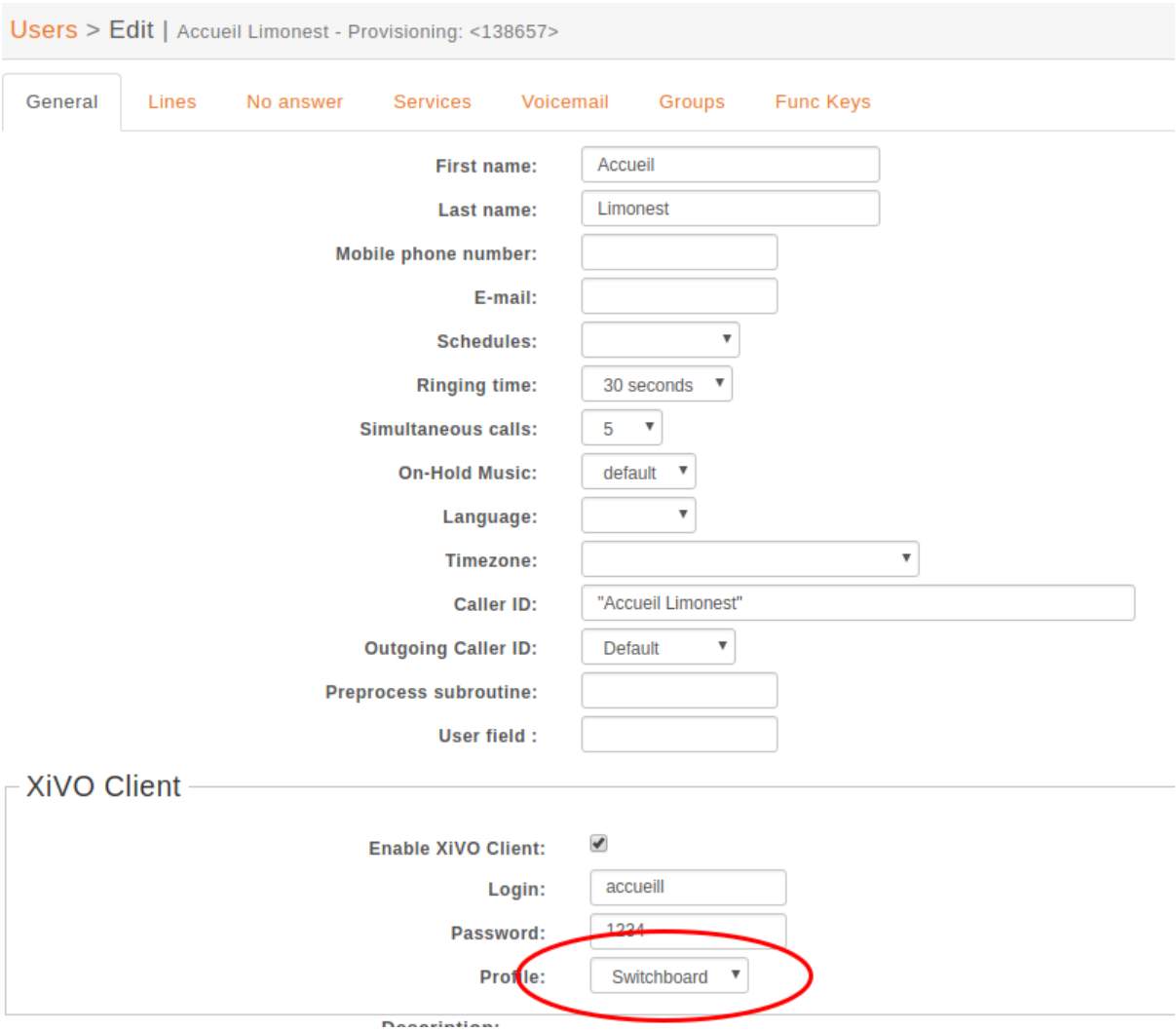
Xlets

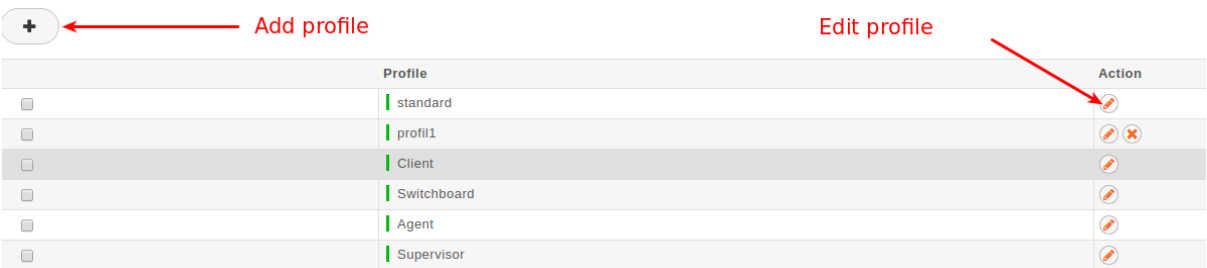
To choose which features are available to users using a profile, you have to select which *Xlets* will be available.

The *Position* attribute determines how the Xlets will be laid out:

- *dock* will display a Xlet in its own frame. This frame can have some options:
  - *Floating* means that the frame can be detached from the main window of the CTI Client.
  - *Closable* means that the Xlet can be hidden
  - *Movable* means that the Xlet can be moved (either inside the main window or outside)
  - *Scroll* means that the Xlet will display a scroll bar if the Xlet is too large.
- *grid* will display a Xlet inside the main window, and it will not be movable. Multiple *grid* Xlets will be laid out vertically (the second below the first).
- *tab* will display a Xlet inside a tab of the Xlet *Tabber*. Thus the Xlet *Tabber* is required and can't be in a *tab* position.

The *Number* attribute gives the order of the Xlets, beginning with 0. The order applies only to Xlets having the same *Position* attribute.


Users > Edit | Accueil Limonest - Provisioning: <138657>
General
Lines
No answer
Services
Voicemail
Groups
Func Keys
First name:
Accueil
Last name:
Limonest
Mobile phone number:
E-mail:
Schedules:
Ringing time:
30 seconds
Simultaneous calls:
5
On-Hold Music:
default
Language:
Timezone:
Caller ID:
"Accueil Limonest"
Outgoing Caller ID:
Default
Preprocess subroutine:
User field :
XiVO Client
Enable XiVO Client:
Login:
accueill
Password:
1234
Profile:
Switchboard


Add profile
Edit profile
Profile
Action
standard
profil1
Client
Switchboard
Agent
Supervisor

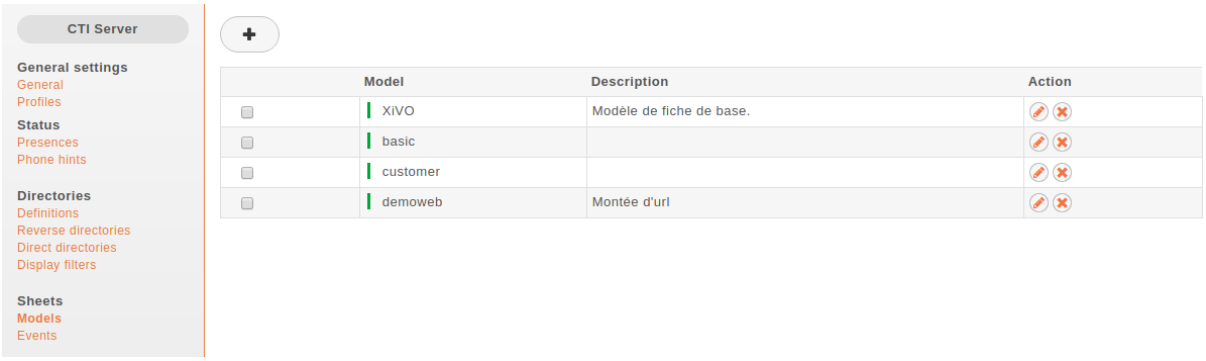
5.8 Display customer informations

5.8.1 Sheet Configuration

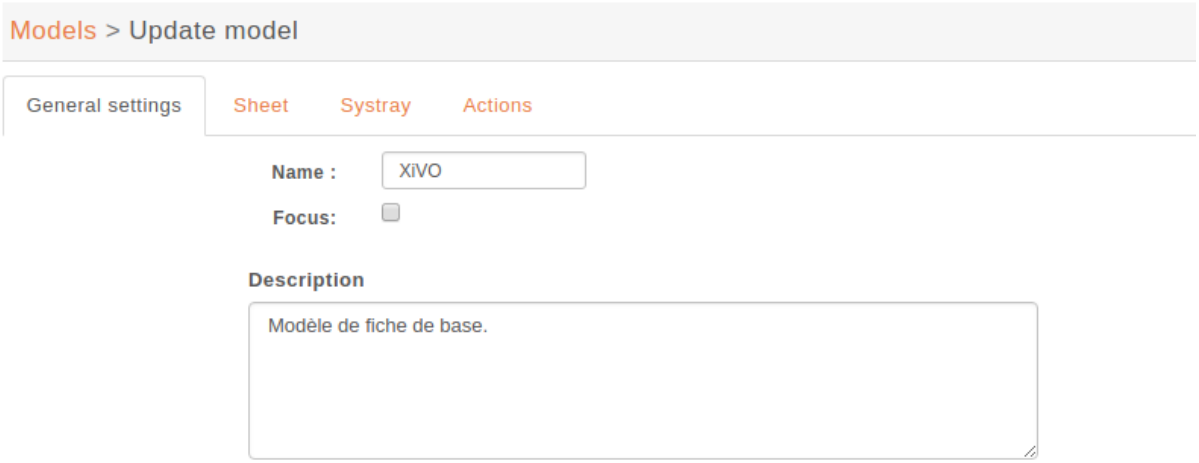
Sheets can be defined under *Services* → *CTI Server* → *Models* in the web interface. Once a sheet is defined, it has to be assigned to an event in the *Services* → *CTI Server* → *Events* menu.

Model The model contains the content of the displayed sheet.

Event Events are actions that trigger the defined sheet. A sheet can be assigned to many events. In that case, the sheet will be raised for each event.



General settings



You must give a name to your sheet to be able to select it later.

The `Focus` checkbox is deprecated and will be remove in next version.

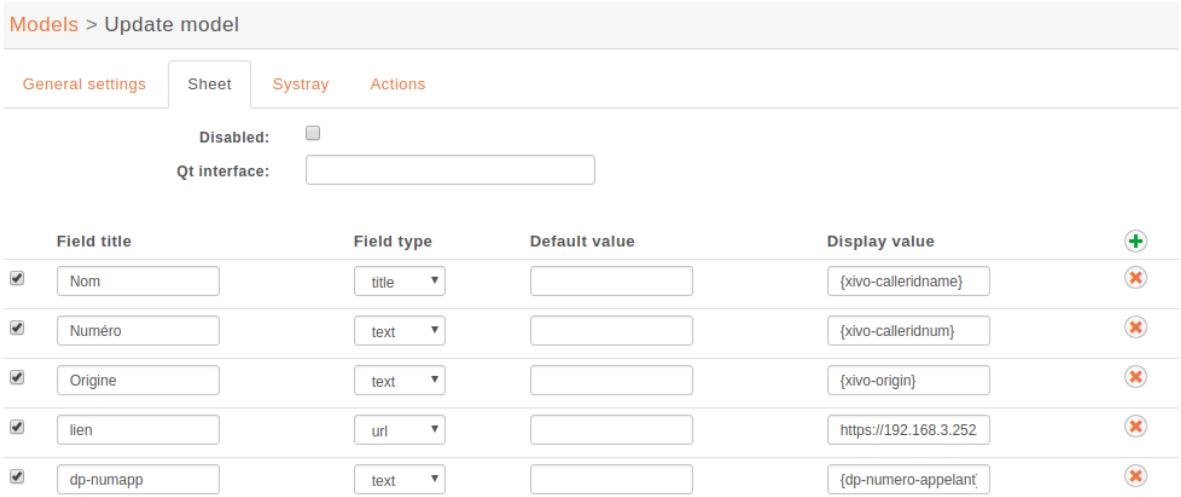
Sheets

Sheets allows to list different fields and associated content to be displayed in XivoCC application such as *CC Agent*. The information will be automatically laid out in a linear fashion and will be read-only.

List of fields

Default XiVO sheet example :

Each field is represented by the following parameters :



- Field title : name of your line used as label on the sheet.
- Field type : define the type of field displayed on the sheet. Supported field types :
  - `title` : to create a title on your sheet
  - `text` : show a text
  - `url` : a simple url link, open your default browser.
  - `urlx` : an url button **not implemented**
  - `phone` : create a tel: link, you can click to call on your sheet. **not implemented**
  - `form` : show the form from an ui predefined. **not implemented**
- **Default value** : if given, this value will be used when all substitutions in the display value field fail.
- **Display value** : you can define text, variables or both. See the *variable list* for more information.

Variables

Three kinds of variables are available :

- *xivo-* prefix is reserved and set inside the CTI server:
  - *xivo-where* for sheet events, event triggering the sheet
  - *xivo-origin* place from where the lookup is requested (did, internal, forcelookup)
  - *xivo-direction* incoming or internal
  - *xivo-did* DID number
  - *xivo-calleridnum*
  - *xivo-calleridname*
  - *xivo-calleridrdnis* contains information whether there was a transfer
  - *xivo-calleridton* Type Of Network (national, international)
  - *xivo-calledidnum*
  - *xivo-calledidname*
  - *xivo-ipbxid* (*xivo-astid* in 1.1)
  - *xivo-directory* : for directory requests, it is the directory database the item has been found

  - *xivo-queuename* queue called
  - *xivo-agentnumber* agent number called
  - *xivo-date* formatted date string
  - *xivo-time* formatted time string, when the sheet was triggered
  - *xivo-channel* asterisk channel value (for advanced users)
  - *xivo-uniqueid* asterisk uniqueid value (for advanced users)
- *db-* prefixed variables are defined when the reverse lookup returns a result.

For example if you want to access to the reverse lookup full name, you need to define a field `fullname` in the directory definition, mapping to the full name field in your directory. The `{db-fullname}` will be replaced by the caller full name. Every field of the directory is accessible this way.

- *dp-* prefixed ones are the variables set through the dialplan (through UserEvent application)

For example if you want to access from the dialplan to a variable dp-test you need to add in your dialplan this line (in a subroutine):

```
UserEvent(dialplan2cti,UNIQUEID: ${UNIQUEID},CHANNEL: ${CHANNEL},
↪VARIABLE: test,VALUE: "Salut")
```

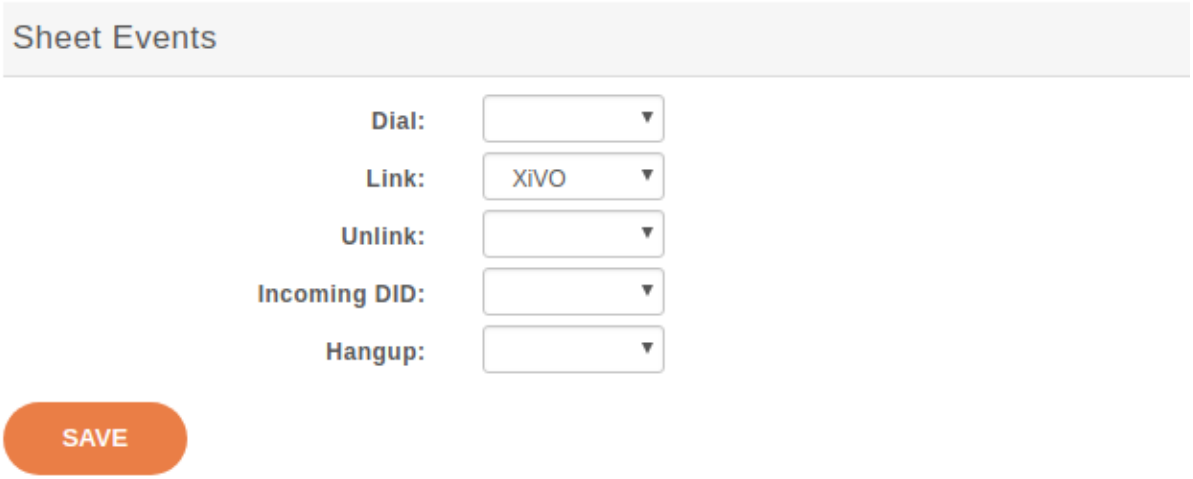
The `{dp-test}` displays Salut.

Event configuration

You can configure a sheet when a specific event is called. For example if you want to receive a sheet when an agent answers to a call, you can choose a sheet model for the Link event.

The following events are available :

- Dial: When the member's phone starts ringing for calls on a group or queue or when the user receives a call
- Link: When a user or agent answers a call
- Unlink: When a user or agent hangup a call received from a queue
- Incoming DID: Received a call in a DID
- Hangup: Hangup the call



The informations about a call are displayed via the XiVO Client on forms called sheets.

5.8.2 Example: Display a Web page when an agent answers a call

The first step is to assign the URL to a dialplan variable. Go in the *Services* → *IPBX* → *Configuration files* and create a new file called `setsheeturl.conf`. In this file, put the following:

```
[setsheeturl]
exten = s,1,NoOp(Starting Set Sheet URL)
same  = n,Set(SHEET_URL_CTI=http://documentation.xivo.solutions)
same  = n,UserEvent(dialplan2cti,UNIQUEID: ${UNIQUEID},CHANNEL: ${CHANNEL},
↪VARIABLE: mysheeturl,VALUE: ${SHEET_URL_CTI})
same  = n,Return()
```

You can replace `documentation.xivo.solutions` by the URL you want.

The second step is to set the URL when the call is queued. To do that, we will use a preprocessing subroutine. This is configured in the queue configuration : go to *Services* → *Call center* → *Queues* and edit the queue. Set the field `Preprocessing subroutine` to `setsheeturl` (the same as above).

The third step is to configure the sheet to open the wanted URL. Go to *Services* → *CTI Server* → *Sheets* → *Models* and create a new sheet. Keep the default for everything except the Action tab, add a field and set it to `{dp-mysheeturl}` (the same as above).

The fourth and final step is to trigger the sheet when the agent answers the queued call. Go to *Services* → *CTI Server* → *Sheets* → *Events* and link the event `Agent linked` to the sheet you just created.

That's it, you can assign agents to your queue, log the agents and make them answer calls with the XiVO Client opened, and your browser should open the specified URL.

5.9 Devices

5.9.1 Synchronize a device

First you have to display the list of devices.

Fig. 8: Click on the synchronize button for a device.

	MAC	IP	Phone number	Entity	Vendor	Model	Plugin	Action
☐	00:04:13:70:31:46	10.70.0.100	1010	xivo	Snom	720	xivo-snom-8.7.5.35	
☐	00:08:5d:34:6a:17	10.70.0.101	-	-	Aastra	6757i	xivo-aastra-3.3.1-SP4	
☐	00:15:65:b2:e2:4e	10.70.0.104	1020	xivo	Yealink	T23G	xivo-yealink-v81	
☐	00:15:65:c4:c2:73	10.70.0.105	1030	xivo	Yealink	T46S	xivo-yealink-v81	

Fig. 9: List devices

You will see a pop-up to confirm synchronization Click on the <ok> button.

You must wait until the full synchronization process has completed to determine the state returned back from the device. This can take several seconds. It is important to wait and do nothing during this time.

If synchronization is successful, a blue information balloon notifies you of success.

If synchronization fails, a red information balloon warns you of failure.

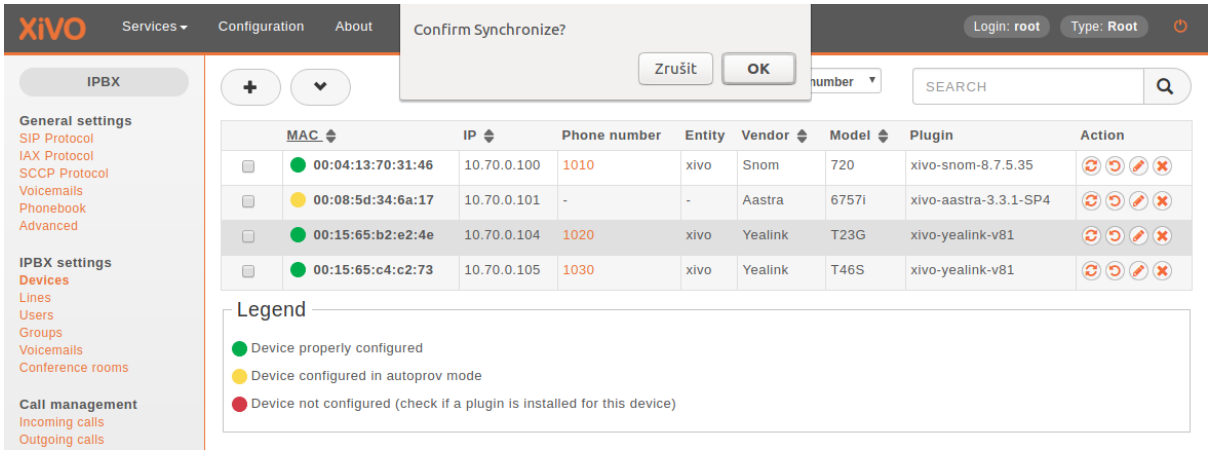


Fig. 10: Alert confirm synchronize

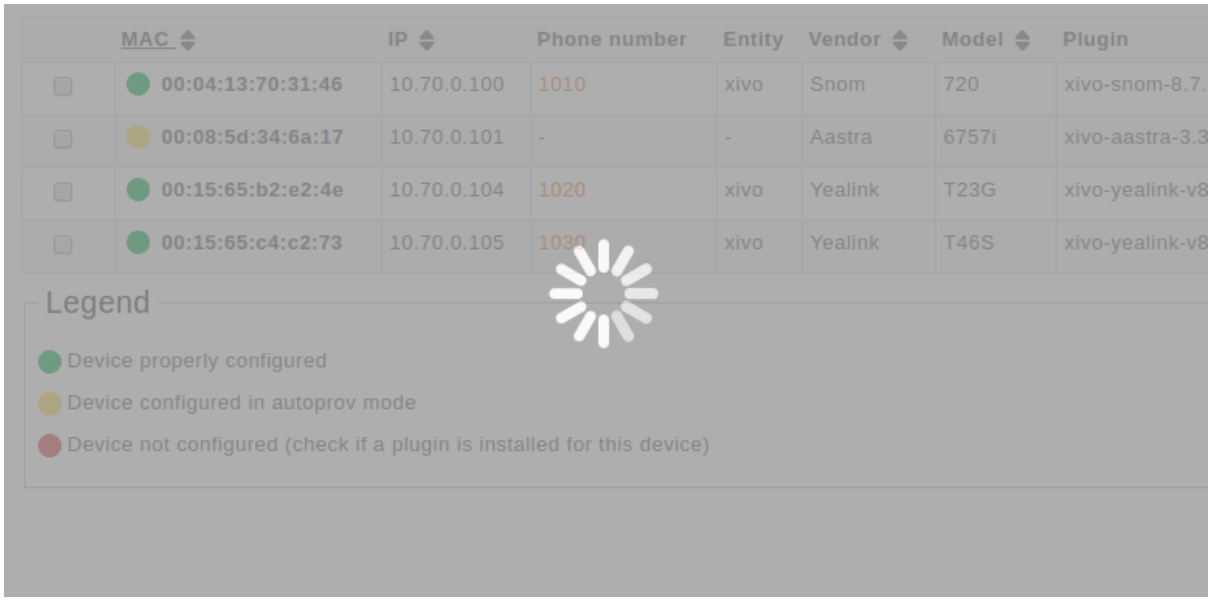


Fig. 11: Request synchronization processing

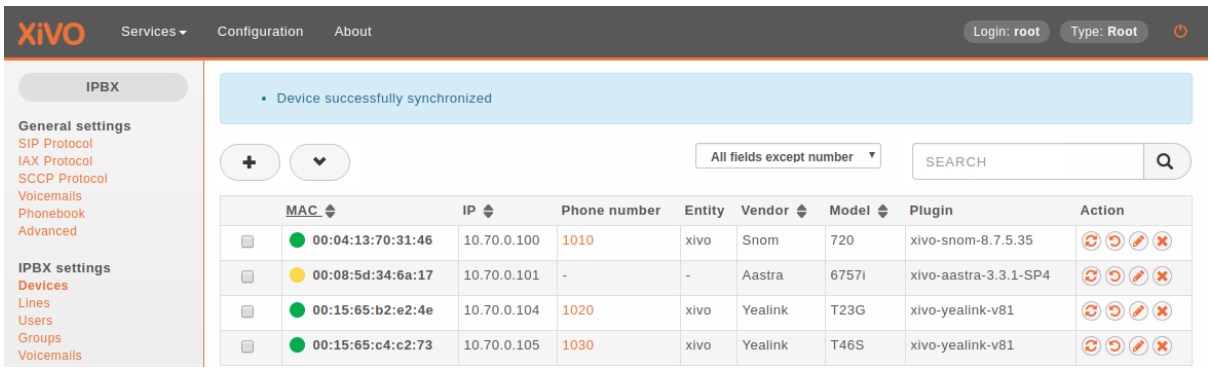


Fig. 12: Device successfully synchronized

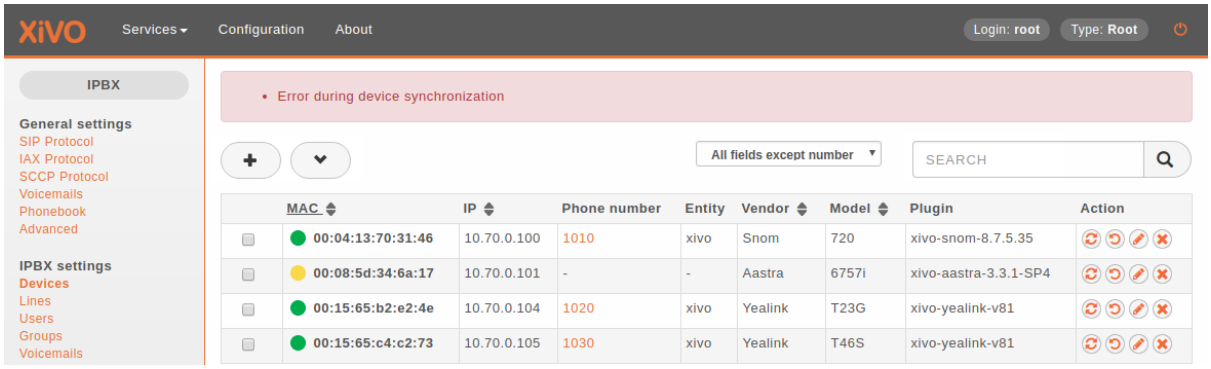


Fig. 13: Error during device synchronization

5.9.2 Synchronize multiple devices

Warning: When using multiple synchronization, the individual return states will not be displayed.

Select the devices you want to synchronize by checking the boxes.

Fig. 14: Synchronize selected devices

A pop-up will appear requesting confirmation.

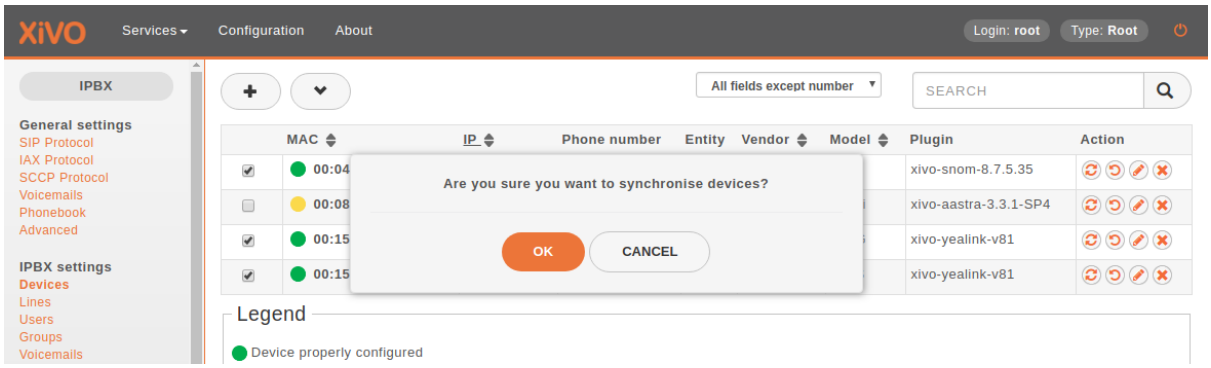


Fig. 15: Synchronize selected devices confirmation

If mass synchronization was successfully sent to the devices, a blue information balloon notifies you of success.

5.10 Directories

This page documents how to add and configure directories from custom sources. Directories added from custom sources can be used for lookup via the directory feature of phones or for *reverse lookup* on incoming calls.

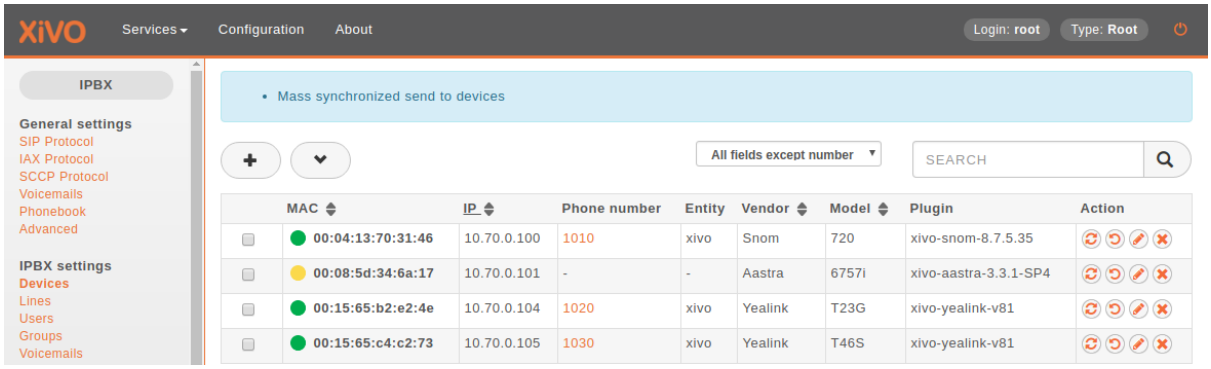


Fig. 16: Mass synchronization request sent successfully

An example of *adding a source* and *configuring source access* is made for each type of source:

5.10.1 XiVO directories

This type of directory is used to query the users of a XiVO. On a fresh install, the local XiVO is already configured. The URI field for this type of directory should be the base URL of a *xivo-confd* server.

This directory type matches the *xivo* backend in *xivo-dird*.

Available fields

- id
- agent_id
- line_id
- firstname
- lastname
- email
- exten
- context
- mobile_phone_number
- userfield
- description
- voicemail_number

Example

Adding a source

Configuring source access

Here is an example of a configuration where the userfield was used as a free field to store the DID number of the user and the description to store it's location.

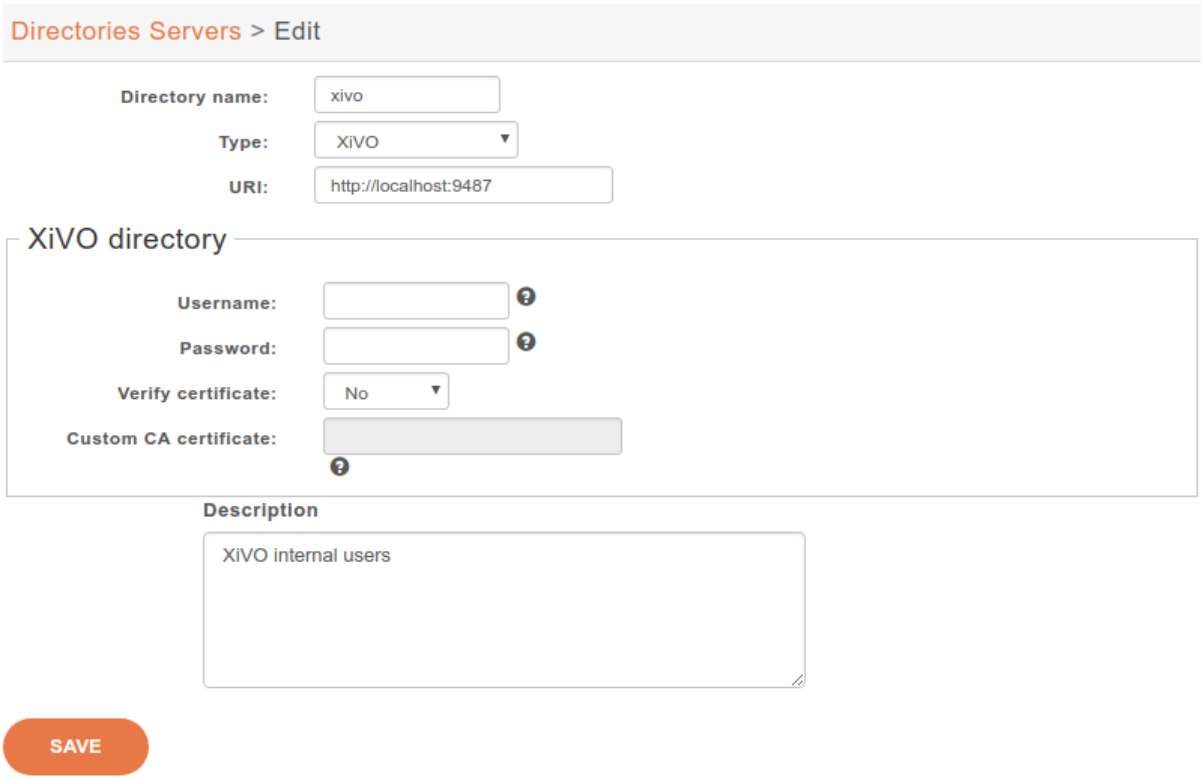


Fig. 17: *Configuration → Management → Directories*

5.10.2 LDAP

XiVO offers the possibility to integrate LDAP servers. Once configured properly, you'll be able to search your LDAP servers from your XiVO client and from your phones (if they support this feature).

Note: This page describes how to add LDAP servers as sources of contacts. For other sources of contacts, see *Directories*.

Add a LDAP Server

You can add a LDAP server by clicking on the add button at the top right corner of the *Configuration → Management → LDAP Servers* page. You'll then be shown this page:

Enter the following information:

- Name: the server's display name
- Host: the hostname or IP address
- Port: the port number (default: 389)
- Security layer: select SSL if it is activated on your server and you want to use it (default: disabled)
  - SSL means TLS/SSL (doesn't mean StartTLS) and port 636 should then be used
- Protocol version: the LDAP protocol version (default: 3)

Warning: When editing an LDAP server, you'll have to restart the CTI server for the changes to be taken into account.

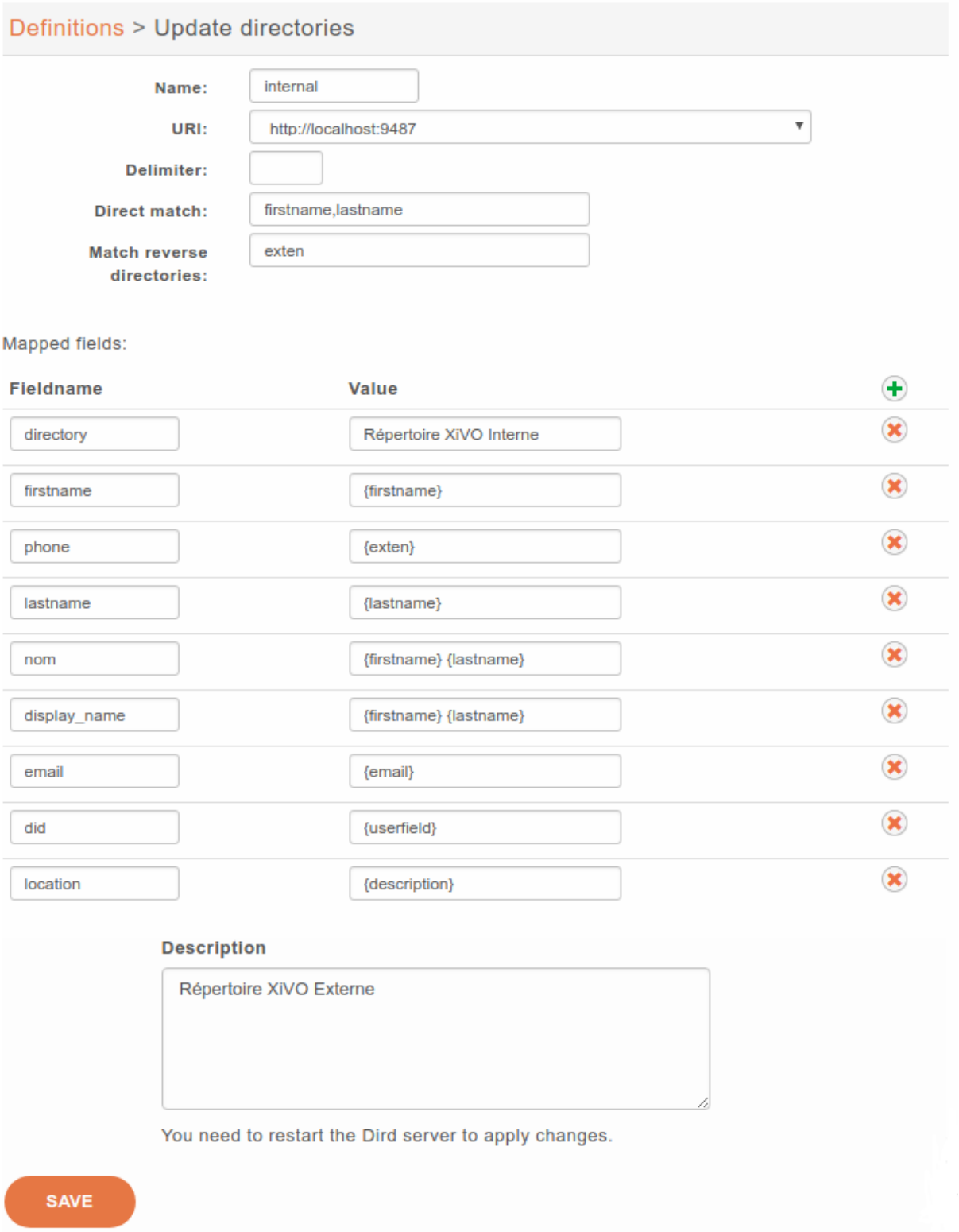


Fig. 18: *Services → CTI Server → Directories → Definitions*

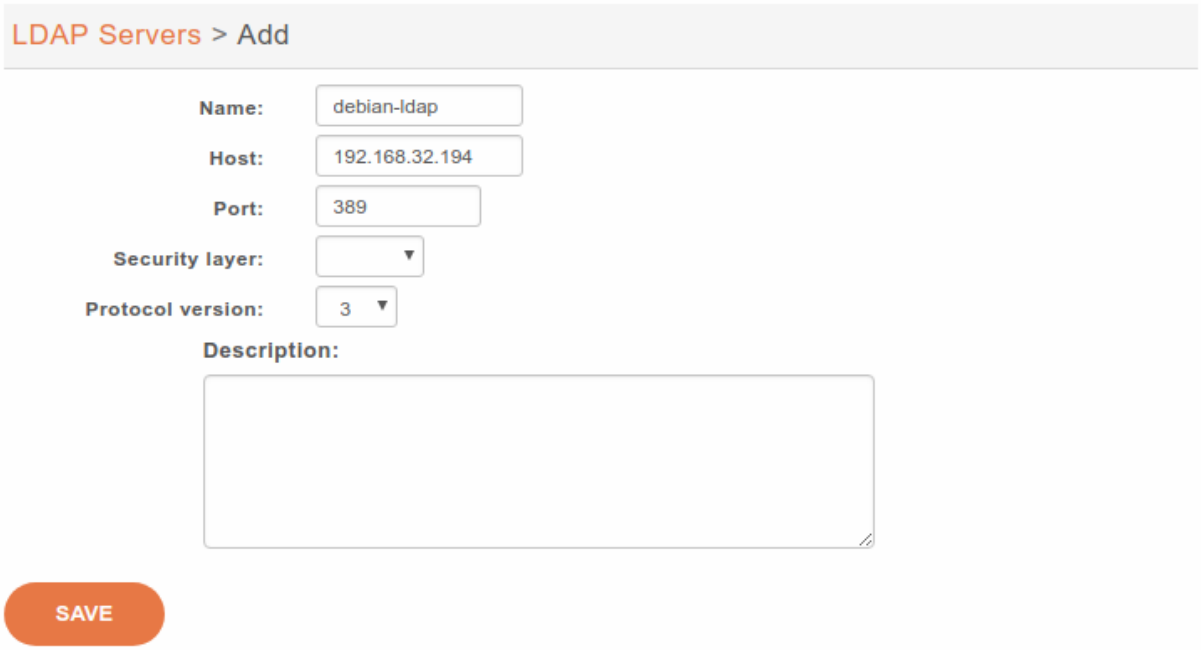


Fig. 19: Adding a LDAP server

Notes on SSL/TLS usage

If you are using SSL with an LDAP server that is using a CA certificate from an unknown certificate authority, you'll have to put the certificate file as a single file ending with `.crt` into `/usr/local/share/ca-certificates` and run `update-ca-certificates`.

You also need to make sure that the `/etc/ldap/ldap.conf` file contains a line `TLS_CACERT /etc/ssl/certs/ca-certificates.crt`.

After that, restart spawn-fcgi with `service spawn-fcgi restart`.

Also, make sure to use the FQDN of the server in the host field when using SSL. The host field must match exactly what's in the CN attribute of the server certificate.

Add a LDAP Filter

Next thing to do after adding a LDAP server is to create a LDAP filter via the *Services* → *IPBX configuration* → *LDAP Filters* page.

You can add a LDAP filter by clicking on the add button at the top right of the page. You'll then be shown this page:

Enter the following information:

- Name: the filter's display name
- LDAP server: the LDAP server this filter applies to
- User: the `dn` of the user used to do search requests
- Password: the password of the given user
- Base DN: the base `dn` of search requests
- Filter: if specified, *it replace the default filter*

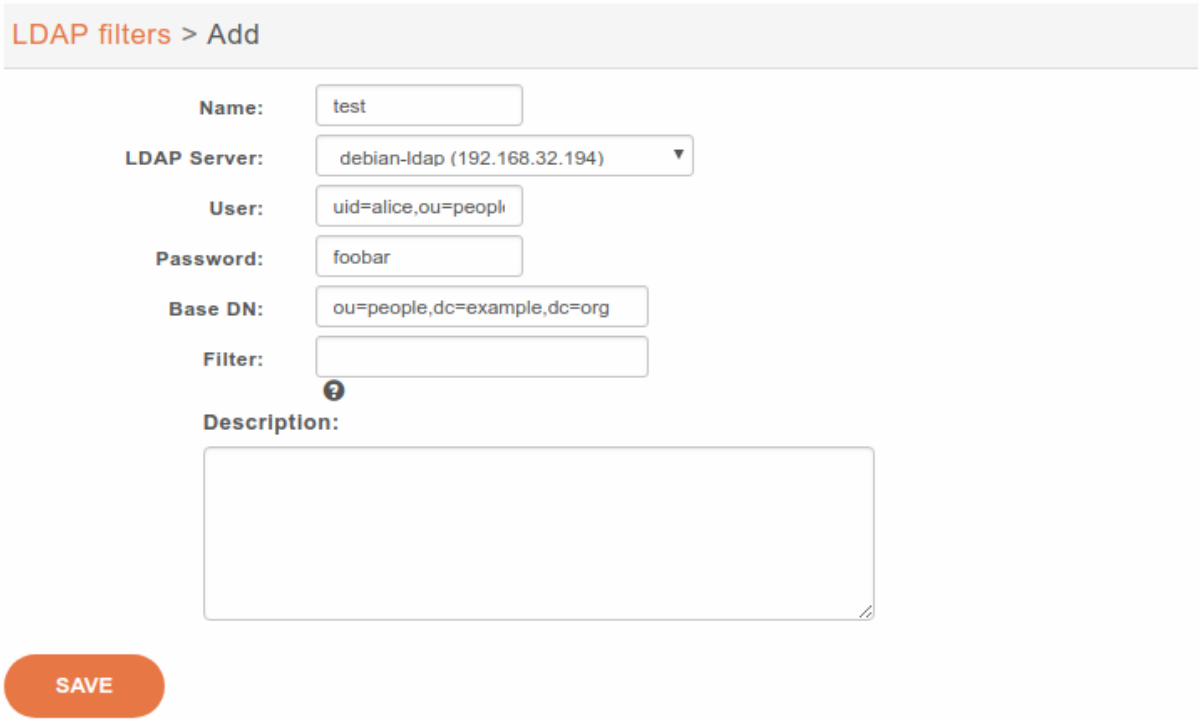


Fig. 20: Adding a LDAP Filter

Use a Custom Filter

In some cases, you might have to use a custom filter for your search requests instead of the default filter.

In custom filters, occurrence of the pattern `%Q` is replaced by what the user entered on its phone.

Here's some examples of custom filters:

- `cn=*%Q*`
- `&(cn=*%Q*)(mail=*@example.org)`
- `|(cn=*%Q*)(displayName=*%Q*)`

Add a Directory Definition

The next step is to add a directory defintion for the LDAP filter you just created. See the *directories* section for more information.

Here's an example of an LDAP directory definition:

If a custom filter is defined in the LDAP filter configuration, the fields in *direct match* will be added to that filter using an *&*. To only use the *filter* field of your LDAP filter configuration, do not add any *direct match* fields in your directory definition.

Example:

- Given an LDAP filter with *filter* `st=Canada`
- Given a directory definition with a *direct match* `cn,o`
- Then the resulting filter when doing a search will be `&(st=Canada)(|(cn=*%Q*)(o=*%Q*))`

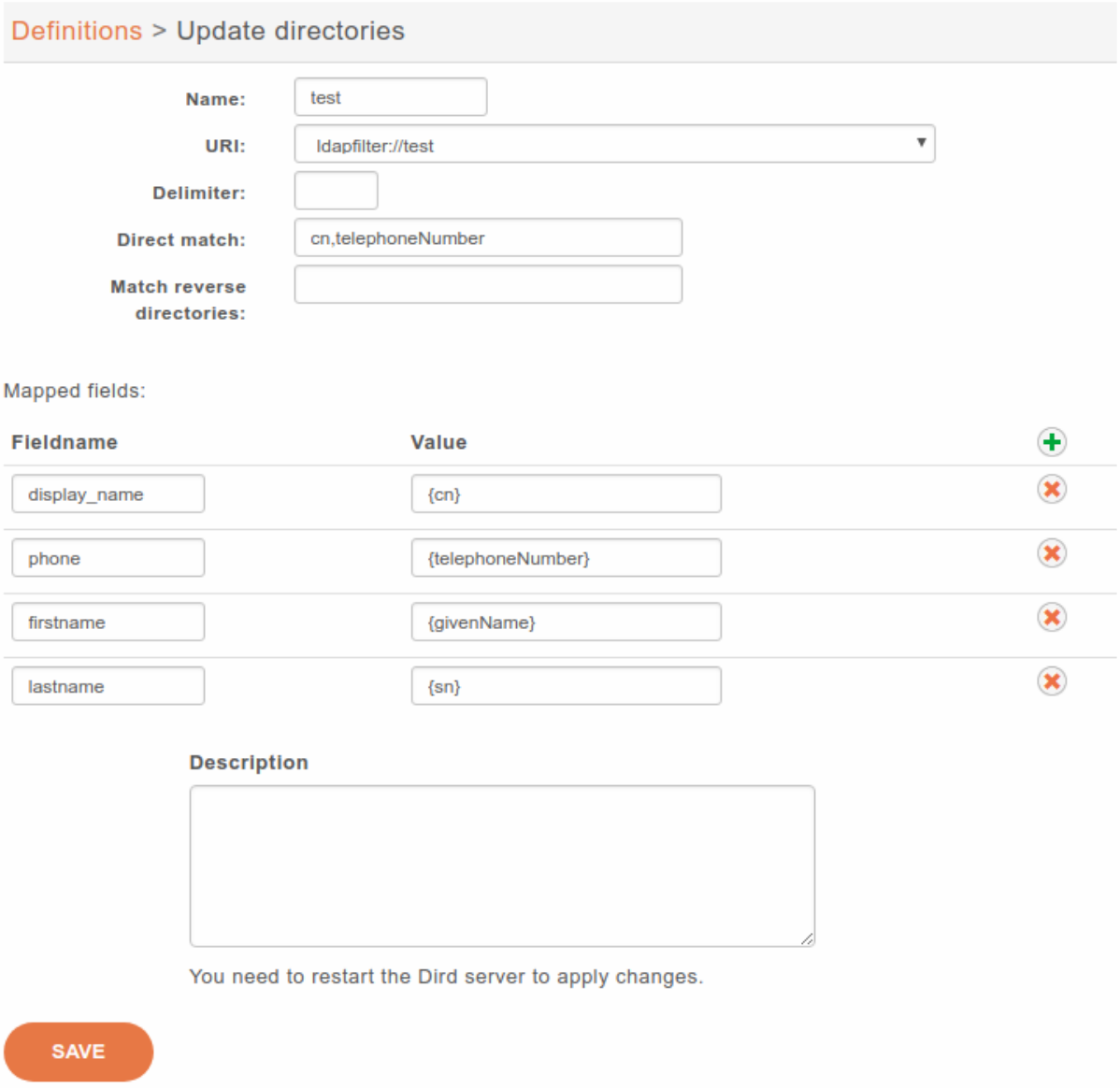


Fig. 21: *Services → IPBX → IPBX configuration → LDAP filters*

5.10.3 CSV File directories

The source file of the directory must be in CSV format. You will be able to choose the headers and the separator in the next steps. For example, the file will look like:

```
title|firstname|lastname|displayname|society|mobilenumber|email
mr|Emmett|Brown|Brown Emmett|DMC|5555551234|emmet.brown@dmc.example.com
```

This directory type matches the *csv* backend in *xivo-dird*.

For file directories, the *Direct match* and the *Match reverse directories* must be filled with the name of the column used to match entries.

Available fields

Available fields are the one's contained in the CSV file.

Example

csv-phonebook.csv:

```
title|firstname|lastname|displayname|society|phone|email
mr|Emmett|Brown|Brown Emmett|DMC|5555551234|emmet.brown@dmc.example.com
ms|Alice|Wonderland|Wonderland Alice|DMC|5555551235|alice.wonderland@dmc.example.
↪com
```

Adding a source

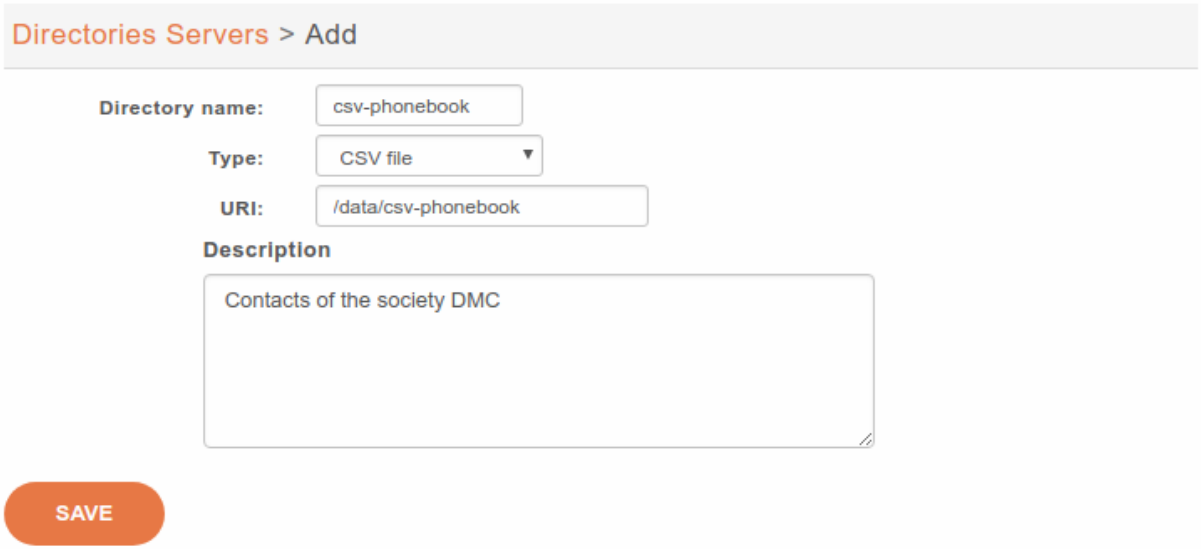


Fig. 22: *Configuration → Management → Directories*

Configuring source access

5.10.4 CSV Web service directories

The data returned by the Web service must have the same format than the file directory. In the same way, you will be able to choose the headers and the separator in the next step.

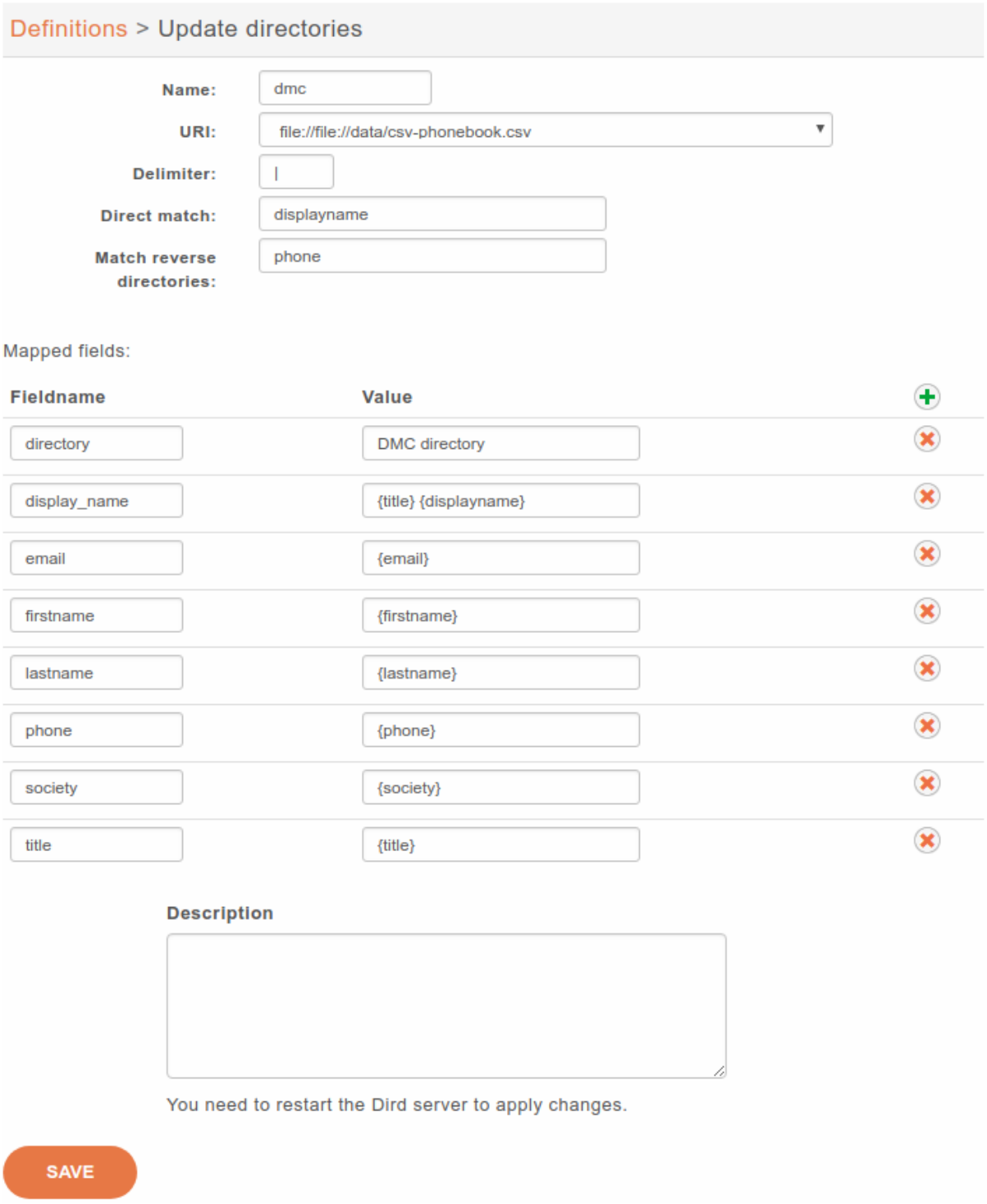


Fig. 23: *Services → CTI Server → Directories → Definitions*

This directory type matches the *CSV web service* backend in *xivo-dird*.

For web service directories, the *Direct match* and the *Match reverse directories* must be filled with the name of the HTTP query parameter that will be used when doing the HTTP requests.

Note that the CSV returned by the Web service is not further processed.

Manual configuration needs to be done to use a secure (SSL) connection. See *CSV web service* for more details.

Available fields

Available fields are the ones contained in the CSV result.

Example

http://example.org:8000/ws-phonebook return csv:

```
title|firstname|lastname|displayname|society|phone|email
mr|Emmett|Brown|Brown Emmett|DMC|5555551234|emmet.brown@dmc.example.com
ms|Alice|Wonderland|Wonderland Alice|DMC|5555551235|alice.wonderland@dmc.example.
↪com
```

Adding a source

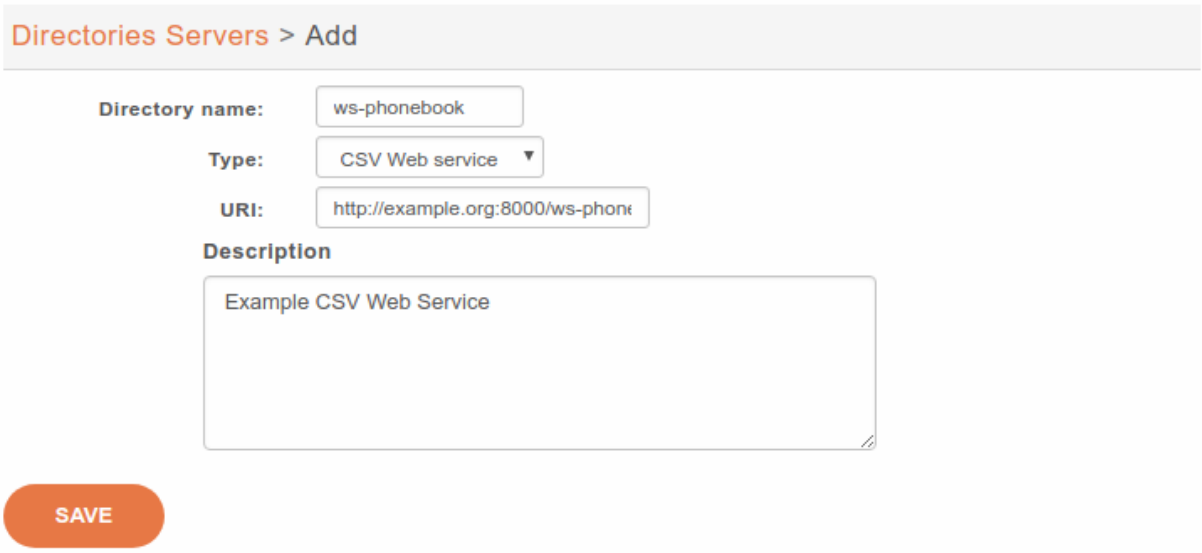


Fig. 24: *Configuration → Management → Directories*

Configuring source access

Given you have the following directory definition:

- *Direct match* : `search`
- *Match reverse directories* : `phone`

When a direct lookup for “Alice” is performed, then the following HTTP request:

```
GET /ws-phonebook?search=Alice HTTP/1.1
```

is emitted. When a reverse lookup for “5555551234” is performed, then the following HTTP request:

```
GET /ws-phonebook?phone=5555551234 HTTP/1.1
```

is emitted. On the reverse lookup, a filtering is performed on the result. In this example, it should have `phone` as column.

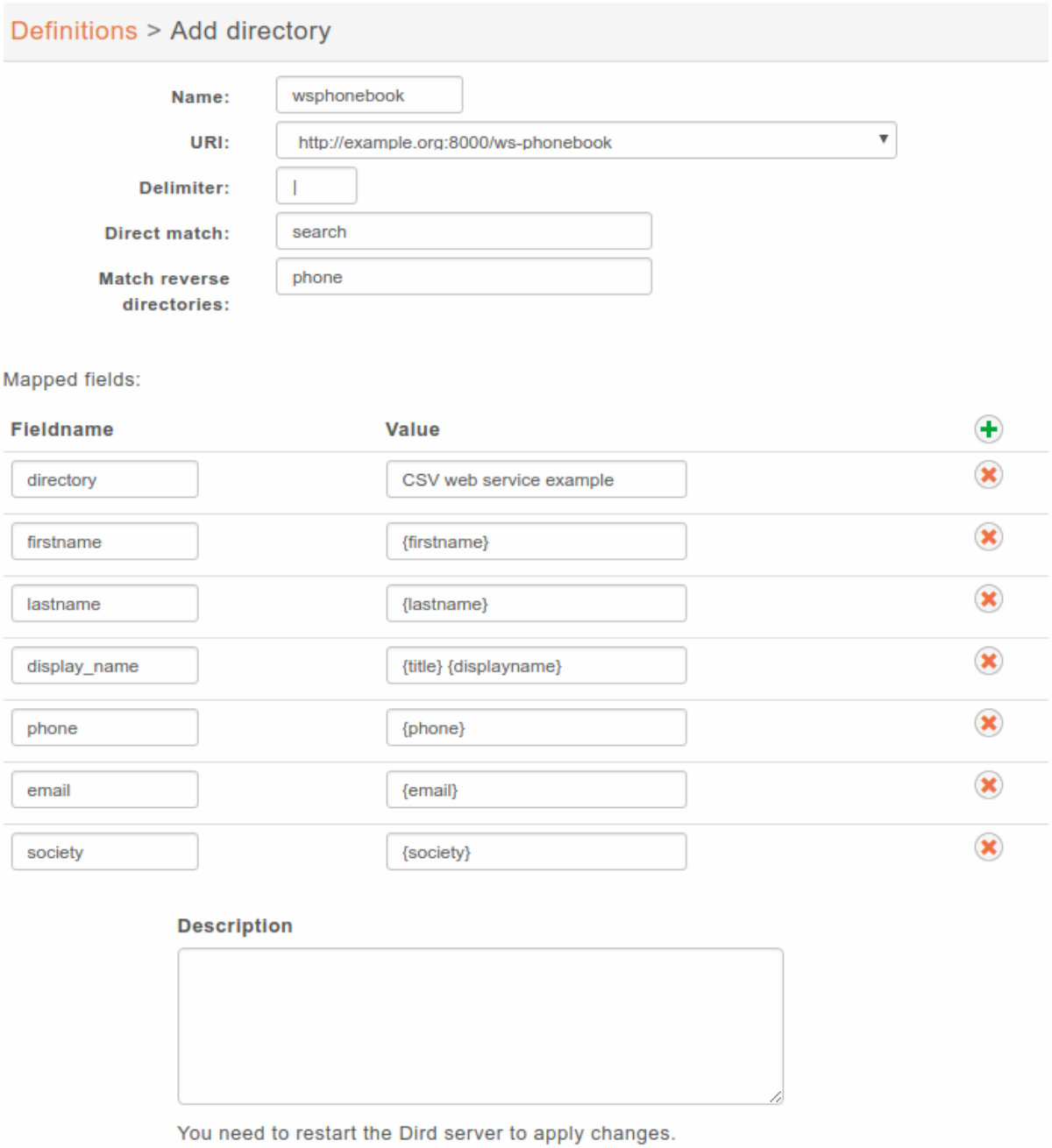


Fig. 25: *Services → CTI Server → Directories → Definitions*

5.10.5 Phonebook directories

This type of directory source is the internal phonebook of a XiVO. The *URI* field is the one used to query the phonebook.

This directory type matches the *phonebook* backend in *xivo-dird*.

Available fields

General phone book section

These fields are set in the General tab of the phone book.

- phonebook.description
- phonebook.displayname
- phonebook.email
- phonebook.firstname
- phonebook.fullname (this value is automatically generated as *"<firstname> <lastname>"*, e.g. *"John Doe"*)
- phonebook.lastname
- phonebook.society
- phonebook.title
- phonebook.url

Phone numbers

These are the different phone numbers that are available

- phonebooknumber.fax.number
- phonebooknumber.home.number
- phonebooknumber.mobile.number
- phonebooknumber.office.number
- phonebooknumber.other.number

Addresses

Each configured address can be accessed

Address uses the following syntax *phonebookaddress.[location].[field]*, e.g. *phonebookaddress.office.zipcode*.

Locations

- home
- office
- other

Fields

- address1
- address2
- city
- country
- state

- zipcode

Example

Adding a source

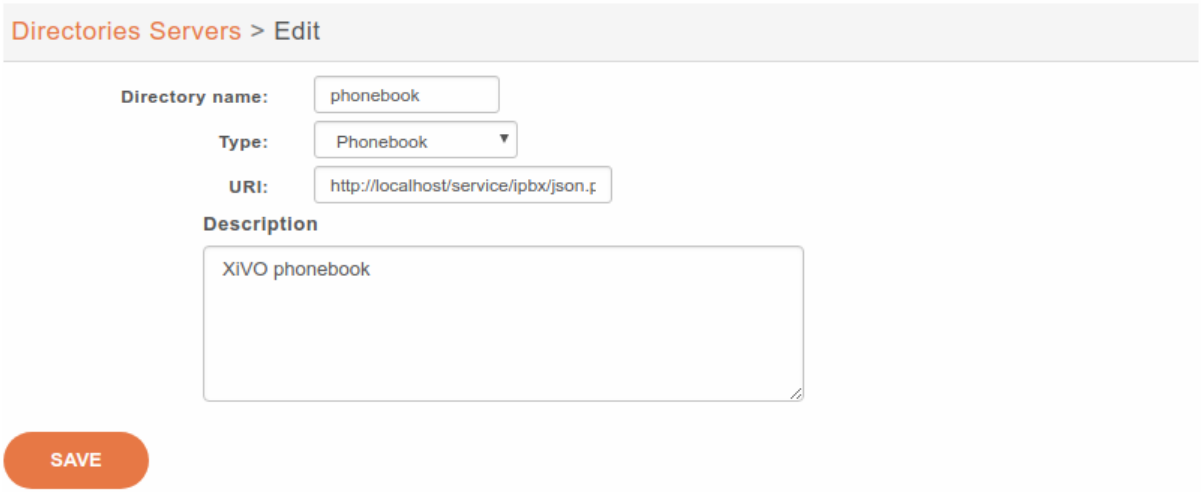


Fig. 26: *Configuration → Management → Directories*
URI : `http://localhost/service/ipbx/json.php/private/pbx_services/phonebook`

Configuring source access

Default phonebook is set in *Directories -> Definitions -> xivodir*.

Note: Phone IP should be in the authorized subnet to access the directories. See *Remote directory*.

5.10.6 Adding a source

Note: See *LDAP* for adding this source.

You can add new data sources via the *Configuration → Management → Directories* page.

- *Directory name*: the name of the directory
- *Type*: there are 4 types of directory:
  - *XiVO*
  - *CSV File*
  - *CSV Web service*
  - *Phonebook*
- *URI*: the data source
- *Description*: (optional) a description of the directory

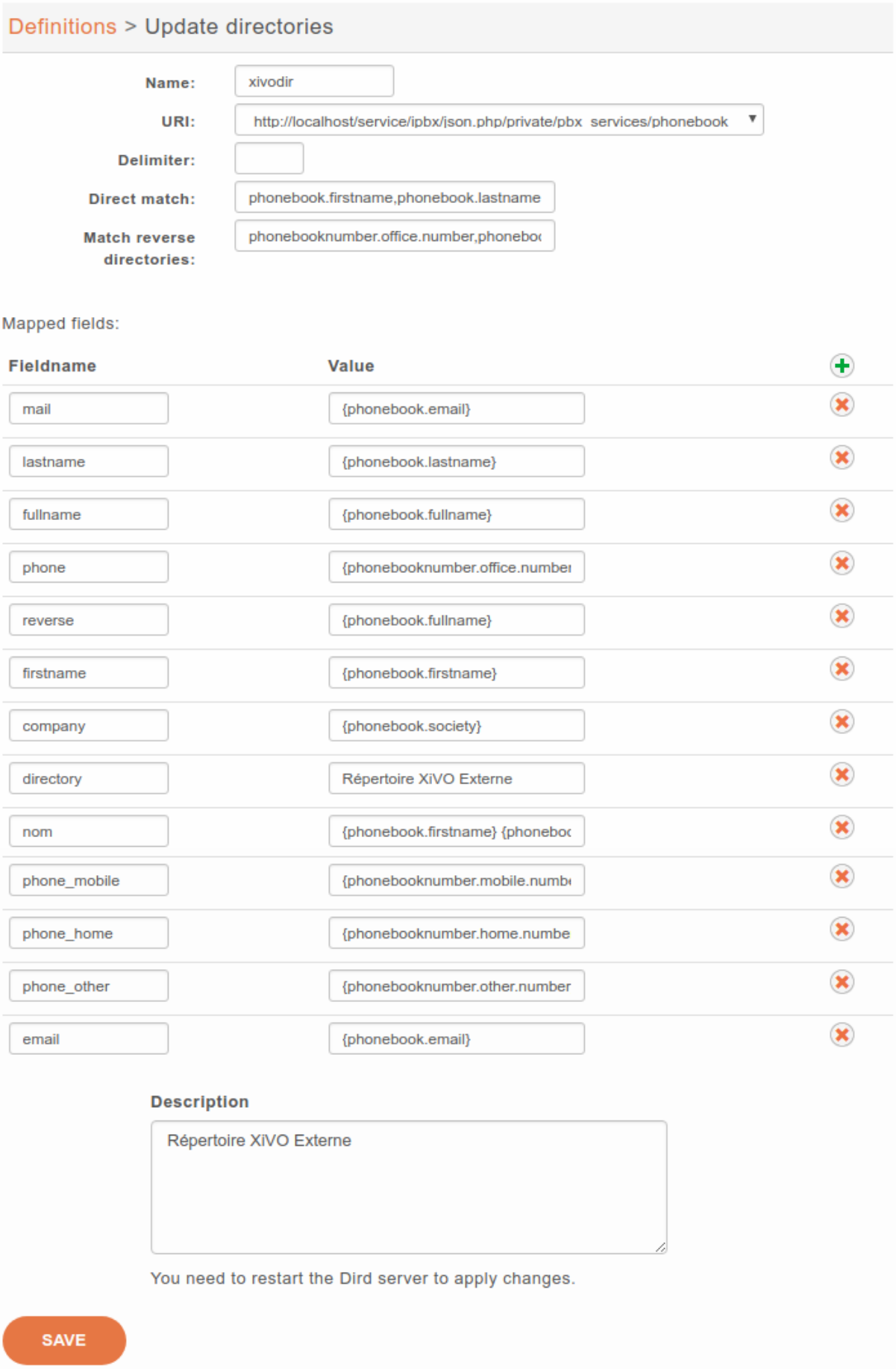


Fig. 27: *Services → CTI Server → Directories → Definitions*

5.10.7 Configuring source access

Go in *Services* → *CTI Server* → *Directories* → *Definitions* and add a new directory definition.

- *Name*: the name of the directory definition
- *URI*: the data source
- *Delimiter*: (optional) the field delimiter in the data source
- *Direct match*: the list used to match entries for direct lookup (comma separated)
- *Match reverse directories*: (optional) the list used to match entries for reverse lookup (comma separated)
- *Mapped fields*: used to add or modify columns in this directory source
  - *Fieldname*: the identifier for this new field
  - *Value*: a python format string that can be used to modify the data returned from a data source

Reverse lookup

It's possible to do reverse lookups on incoming calls to show a better caller ID name when the caller is in one of our directories.

Reverse lookup will only be tried if at least one of the following conditions is true:

- The caller ID name is the same as the caller ID number
- The caller ID name is "unknown"

Also, reverse lookup is performed after *caller ID number normalization* (since XiVO 13.11).

To enable reverse lookup, you need to add an entry in *Mapped fields*:

- *Fieldname*: `reverse`
- *Value*: the header of your data source that you want to see as the caller ID on your phone on incoming calls

Example

- *Match reverse directories*: `phonebooknumber.office.number,phonebooknumber.mobile.number,phonebooknumber.home.number`
- *Fieldname*: `reverse`
- *Value*: `phonebook.society`

This configuration will show the contact's company name on the caller ID name, when the incoming call will match office, mobile or home number.

Phone directory

Phone directory takes 2 *Fieldname* by default:

- `display_name`: the displayed name on the phone
- `phone`: the number to call

Examples:

You will find below some useful configurations of *Mapped fields*.

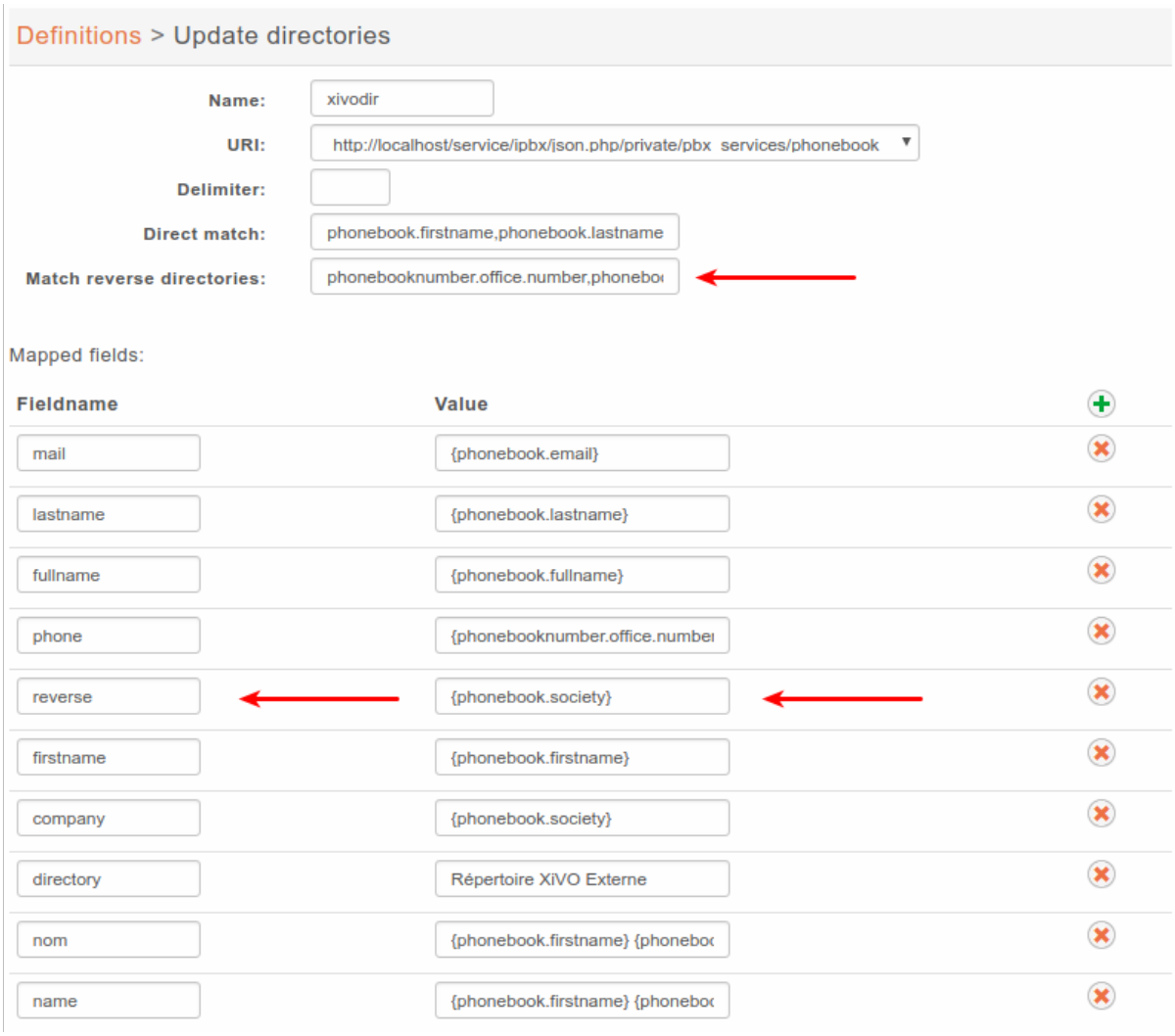


Fig. 28: *Services → CTI Server → Directories → Definitions*

Adding a name field from firstname and lastname

Given a configuration where the directory source returns results with fields firstname and lastname . To add a *name* column to a directory, the administrator would add the following *Mapped fields*:

- *Fieldname*: `name`
- *Value*: `{firstname} {lastname}`

Prefixing a field

Given a directory source that need a prefix to be called, a new field can be created from an exising one. To add a prefix *9* to the numbers returned from a source, the administrator would add the following *Mapped fields*:

- *Fieldname*: `number`
- *Value*: `9{number}`

Adding a static field

Sometimes, it can be useful to add a field to the search results. A string can be added without any formatting. To add a *directory* field to the *xivodir* directory, the administrator would add the following *Mapped fields*:

- *Fieldname*: `directory`
- *Value*: `XiVO internal directory`

5.10.8 Configuring source display

XiVO Client

Edit the default display filter or create your own in *Services* → *CTI Server* → *Directories* → *Display filters*.

Each line in the display filter will result in a header in your XiVO Client.

- *Field title*: text displayed in the header.
- *Field type*: type of the column, this information is used by the XiVO Client. (see *type description*)
- *Default value*: value that will be used if this field is empty for one of the configured sources.
- *Field name*: name of the field in the directory definitions. The specified names should be available in the configured sources. To add new column name to a directory definition see above.

Phone

The only way to configure display phone directory is through *XiVO dird configuration*.

5.10.9 Adding a directory

To include a directory in direct directory definition:

1. Go to *Services* → *CTI Server* → *Directories* → *Direct directories*.
2. Edit your context.
3. Select your display filter.
4. Add the directories in the *Directories* section.

To include a directory in reverse directory definition:

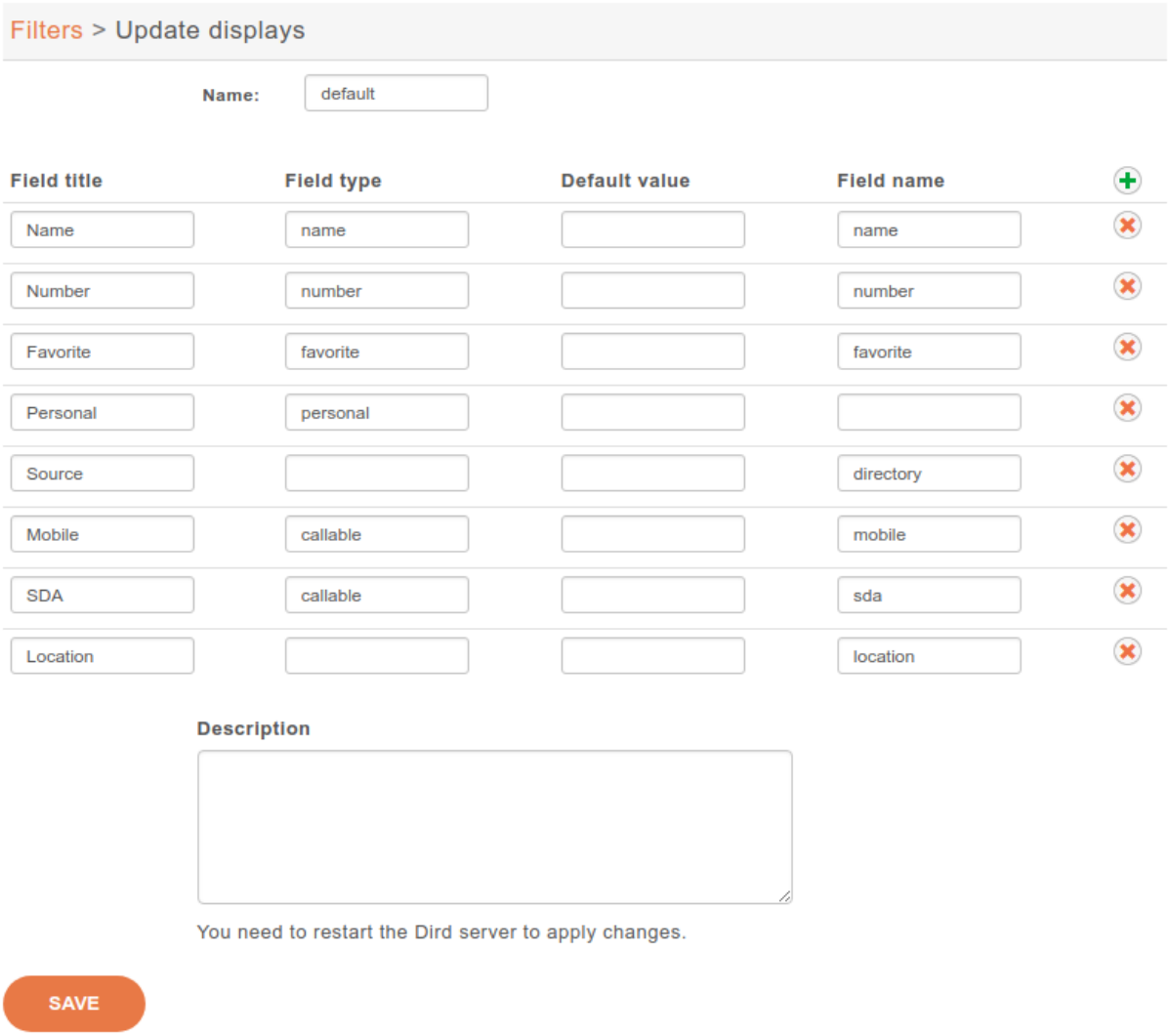


Fig. 29: *Services → CTI Server → Directories → Display filters*

1. Go to *Services* → *CTI Server* → *Directories* → *Reverse directories*.
2. Add the directories to include to reverse lookups in the *Related directories* section.

5.10.10 Applying changes

To reload the directory configuration for XiVO Client, phone lookups and reverse lookups, use *one* of these methods:

- *Services* → *IPBX* → *Control* → *Restart Dird server*
- console `service xivo-dird restart`

5.11 Directed Pickup

Directed pickup allows a user to intercept calls made to another user.

For example, if a user with number 1001 is ringing, you can dial *81001 from your phone and it will intercept (i.e. pickup) the call to this user.

The extension prefix used to pickup calls can be changed via the *Services* → *IPBX* → *IPBX services* → *Extensions* page.

5.11.1 Custom Line Limitation

There is a case where directed pickup does not work, which is the following:

```
Given you have a user U with a line of type "customized"
Given this custom line is using DAHDI technology
Given this user is a member of group G
When a call is made to group G
Then you won't be able to intercept the call made to U by pressing *8<line number␣
↪of U>
```

If you find yourself in this situation, you'll need to write a bit of dialplan.

For example, if you have the following:

- a user with a custom line with number 1001 in context default
- a custom line with interface `DAHDI/g1/5551234`

Then add the following, or similar:

```
[custom_lines]
exten = line1001,1,NoOp()
same  = n,Set(__PICKUPMARK=1001%default)
same  = n,Dial(DAHDI/g1/5551234)
same  = n,Hangup()
```

And do a `dialplan reload` in the asterisk CLI.

Then, edit the line of the user and change the interface value to `Local/line1001@custom_lines`

Note that you'll need to update your dialplan if you update the number of the line or the context.

5.12 Entities

5.12.1 Purpose

In some cases, as the telephony provider, you want different independent organisations to have their telephony served by your XiVO, e.g. different departments using the same telephony infrastructure, but you do not want each organisation to see or edit the configuration of other organisations.

5.12.2 Configuration

In *Configuration* → *Entities*, you can create entities, one for each independant organisation.

In *Configuration* → *Users*, you can select an entity for each administrator.

Note: Once an entity is linked with an administrator, it can not be deleted. You have to unlink the entity from all administrator to be able to delete it.

For the new entity to be useful, you need to create contexts in this entity. You may need:

- an Internal context for users, groups, queues, etc.
- an Incall context for incoming calls
- an Outcall context for outgoing calls, which should be included in the Internal context for the users to be able to call external numbers

5.12.3 Limitations

Global Fields

Some fields are globally unique and will collide when the same value is used in different entities:

- User CTI login
- Agent number
- Queue name
- Context name

An error message will appear when creating resources with colliding parameters, saying the resource already exists, even if the entity-linked administrator can not see them.

Affected Lists

Only the following lists may be filtered by entity:

- Lines
- Users
- Devices
- Groups
- Voicemails
- Conference Rooms
- Incoming calls
- Call filters

- Call pickups
- Schedules
- Agents
- Queues

For the devices:

- The filtering only applies to the devices associated with a line.
- The devices in autoprov mode or not configured mode are visible by every administrator.

REST API

The REST API does not have the notion of entity. When creating a resource without context via REST API, the resource will be associated to an arbitrary entity. Affected resources are:

- Contexts
- Call filters
- Group pickups
- Schedules
- Users

5.13 Fax

5.13.1 Fax transmission

It's possible to send faxes from XiVO using the fax Xlet in the XiVO client.

The file to send must be in PDF format.

5.13.2 Fax reception

Note: Only works for Fax in A4 paper format.

Adding a fax reception DID

If you want to receive faxes from XiVO, you need to add incoming calls definition with the *Application* destination and the *FaxToMail* application for every DID you want to receive faxes from.

This applies even if you want the action to be different from sending an email, like putting it on a FTP server. You'll still need to enter an email address in these cases even though it won't be used.

Note that, as usual when adding incoming call definitions, you must first define the incoming call range in the used context.

Changing the email body

You can change the body of the email sent upon fax reception by editing `/etc/xivo/mail.txt`.

The following variable can be included in the mail body:

- `%(dstnum)s`: the DID that received the fax

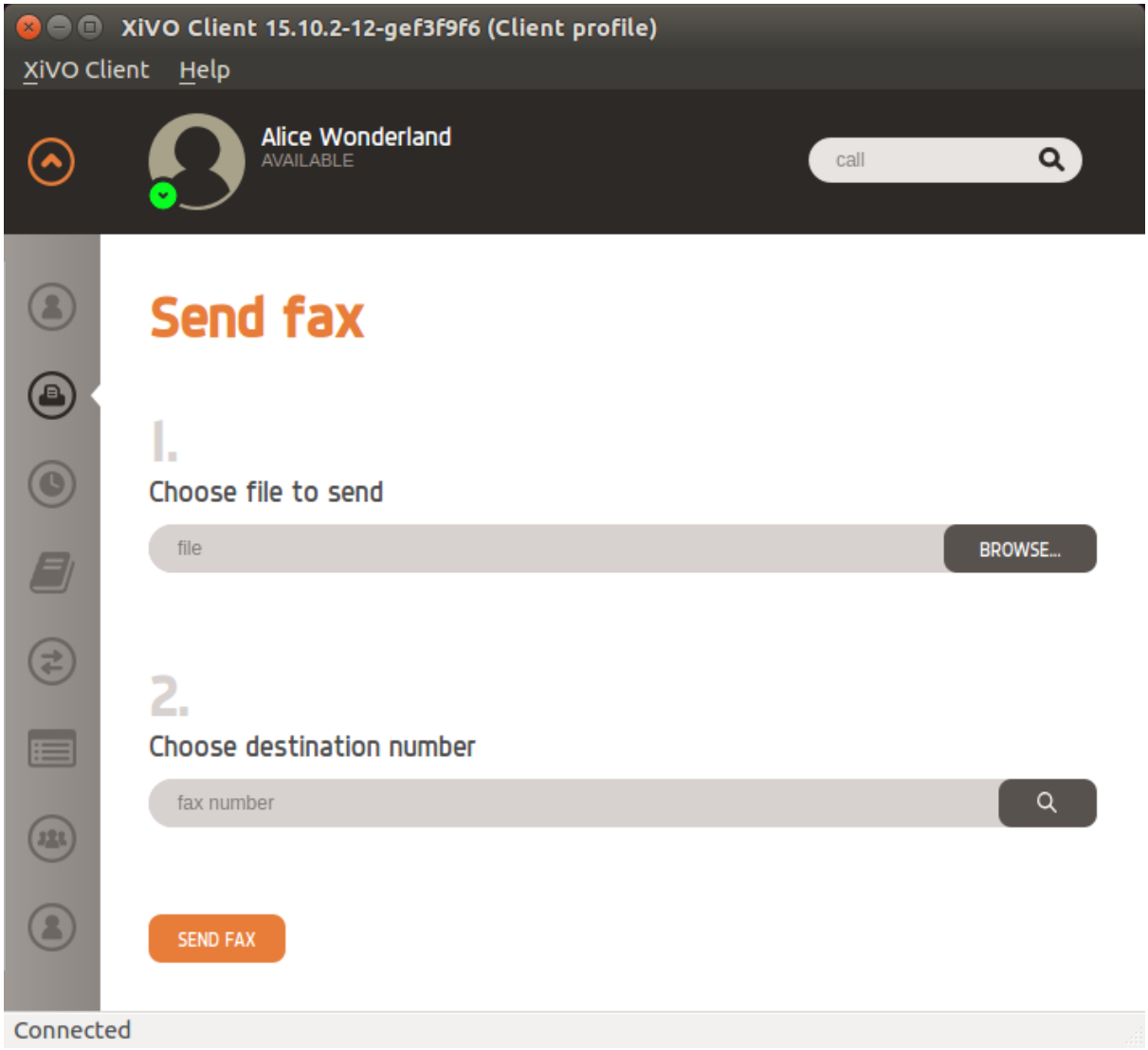
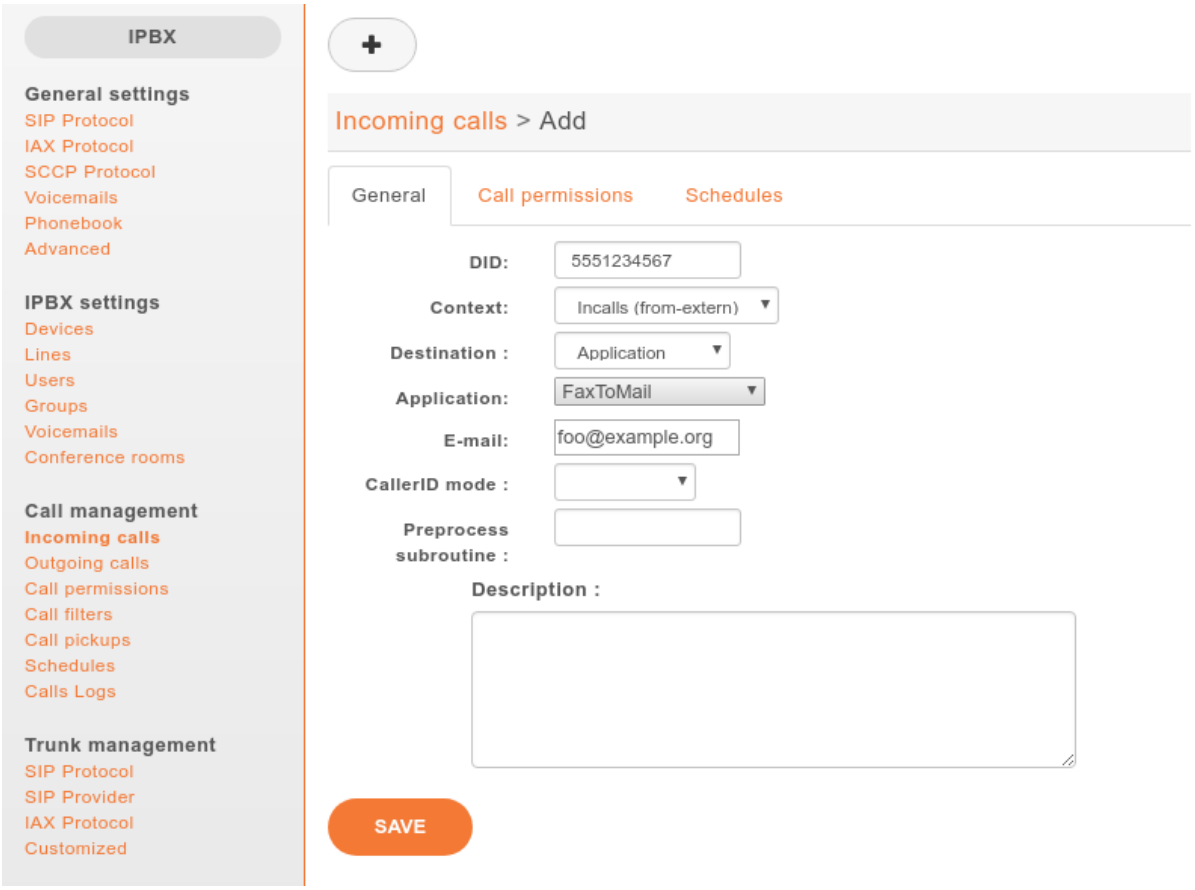


Fig. 30: The fax Xlet in the XiVO Client



- `%(srcnum)s`: the sender number (as we received it) - since Borealis.10

If you want to include a regular percent character, i.e. `%`, you must write it as `%%` in `mail.txt` or an error will occur when trying to do the variables substitution.

The `agid` service must be restarted to apply changes:

```
service xivo-agid restart
```

Changing the email subject

You can change the subject of the email sent upon fax reception by editing `/etc/xivo/asterisk/xivo_fax.conf`.

Look for the `[mail]` section, and in this section, modify the value of the `subject` option.

The available variable substitution are the same as for the email body.

The `agid` service must be restarted to apply changes:

```
service xivo-agid restart
```

Changing the email from

You can change the from of the email sent upon fax reception by editing `/etc/xivo/asterisk/xivo_fax.conf`.

Look for the `[mail]` section, and in this section, modify the value of the `email_from` option.

The `agid` service must be restarted to apply changes:

```
service xivo-agid restart
```

Changing the email realname

You can change the realname of the email sent upon fax reception by editing `/etc/xivo/asterisk/xivo_fax.conf`.

Look for the `[mail]` section, and in this section, modify the value of the `email_realname` option.

The `agid` service must be restarted to apply changes:

```
service xivo-agid restart
```

Using the advanced features

The following features are only available via the `/etc/xivo/asterisk/xivo_fax.conf` configuration file. They are not available from the web-interface.

The way it works is the following:

- you first declare some backends, i.e. actions to be taken when a fax is received. A backend name looks like `mail`, `ftp_example_org` or `printer_office`.
- once your backends are defined, you can use them in your destination numbers. For example, when someone calls the DID 100, you might want the `ftp_example_org` and `mail` backend to be run, but otherwise, you only want the `mail` backend to be run.

Here's an example of a valid `/etc/xivo/asterisk/xivo_fax.conf` configuration file:

```
[general]
tiff2pdf = /usr/bin/tiff2pdf
mutt = /usr/bin/mutt
lp = /usr/bin/lp

[mail]
subject = FAX reception to %(dstnum)s
content_file = /etc/xivo/mail.txt
email_from = no-reply+fax@xivo.solutions
email_realname = Service Fax

[ftp_example_org]
host = example.org
username = foo
password = bar
directory = /foobar

[dstnum_default]
dest = mail

[dstnum_100]
dest = mail, ftp_example_org
```

The section named `dstnum_default` will be used only if no DID-specific actions are defined.

After editing `/etc/xivo/asterisk/xivo_fax.conf`, you need to restart the agid server for the changes to be applied:

```
service xivo-agid restart
```

Using the FTP backend

The FTP backend is used to send a PDF version of the received fax to an FTP server.

An FTP backend is always defined in a section beginning with the `ftp` prefix. Here's an example for a backend named `ftp_example_org`:

```
[ftp_example_org]
host = example.org
port = 2121
username = foo
password = bar
directory = /foobar
convert_to_pdf = 0
```

The `port` option is optional and defaults to 21.

The `directory` option is optional and if not specified, the document will be put in the user's root directory.

The `convert_to_pdf` option is optional and defaults to 1. If it is set to 0, the TIFF file will not be converted to PDF before being sent to the FTP server.

The uploaded file are named like `${XIVO_SRCNUM}-${EPOCH}.pdf`.

Using the printer backend

To use the printer backend, you must have the `cups-client` package installed on your XiVO:

```
$ apt-get install cups-client
```

The printer backend uses the `lp` command to print faxes.

A printer backend is always defined in a section beginning with the `printer` prefix. Here's an example for a backend named `printer_office`:

```
[printer_office]
name = office
convert_to_pdf = 1
```

When a fax will be received, the system command `lp -d office <faxfile>` will be executed.

The `convert_to_pdf` option is optional and defaults to 1. If it is set to 0, the TIFF file will not be converted to PDF before being printed.

> **Warning:** You need a CUPS server set up somewhere on your network.

Using the mail backend

By default, a mail backend named `mail` is defined. You can define more mail backends if you want. Just look what the default mail backend looks like.

5.13.3 Fax detection

XiVO **does not currently support Fax Detection**. A workaround is described in the *Fax detection* section.

5.13.4 Using analog gateways

XiVO is able to provision Cisco SPA122 and Linksys SPA2102, SPA3102 and SPA8000 analog gateways which can be used to connect fax equipments. This section describes the creation of custom template *for SPA3102* which modifies several parameters.

Note: **With SPA ATA plugins >= v0.8**, you **should not need** to follow this section anymore since all of these parameters are now set in the base templates of all, except for Echo_Canc_Adapt_Enable, Echo_Supp_Enable, Echo_Canc_Enable.

Note: Be aware that most of the parameters are or could be country specific, i.e. :

- Preferred Codec,
- FAX Passthru Codec,
- RTP Packet Size,
- RTP-Start-Loopback Codec,
- Ring Waveform,
- Ring Frequency,
- Ring Voltage,
- FXS Port Impedance

---

1. Create a custom template for the SPA3102 base template:

```
cd /var/lib/xivo-provd/plugins/xivo-cisco-spa3102-5.1.10/var/templates/
cp ../../templates/base.tpl .
```

2. Add the following content before the `</flat-profile>` tag:

```
<!-- CUSTOM TPL - for faxes - START -->

{% for line_no, line in sip_lines.iteritems() %}
<!-- Dial Plan: L{{ line_no }} -->
<Dial_Plan_{{ line_no }}_ ua="na">([x*#].)</Dial_Plan_{{ line_no }}_>

<Call_Waiting_Serv_{{ line_no }}_ ua="na">No</Call_Waiting_Serv_{{ line_no }}_>
<Three_Way_Call_Serv_{{ line_no }}_ ua="na">No</Three_Way_Call_Serv_{{ line_no␣
↪}}_>

<Preferred_Codec_{{ line_no }}_ ua="na">G711a</Preferred_Codec_{{ line_no }}_>
<Silence_Supp_Enable_{{ line_no }}_ ua="na">No</Silence_Supp_Enable_{{ line_no␣
↪}}_>
<Echo_Canc_Adapt_Enable_{{ line_no }}_ ua="na">No</Echo_Canc_Adapt_Enable_{{␣
↪line_no }}_>
<Echo_Supp_Enable_{{ line_no }}_ ua="na">No</Echo_Supp_Enable_{{ line_no }}_>
<Echo_Canc_Enable_{{ line_no }}_ ua="na">No</Echo_Canc_Enable_{{ line_no }}_>
<Use_Pref_Codec_Only_{{ line_no }}_ ua="na">yes</Use_Pref_Codec_Only_{{ line_
↪no }}_>
<DTMF_Tx_Mode_{{ line_no }}_ ua="na">Normal</DTMF_Tx_Mode_{{ line_no }}_>

<FAX_Enable_T38_{{ line_no }}_ ua="na">Yes</FAX_Enable_T38_{{ line_no }}_>
<FAX_T38_Redundancy_{{ line_no }}_ ua="na">1</FAX_T38_Redundancy_{{ line_no }}_
↪>
<FAX_Passthru_Method_{{ line_no }}_ ua="na">ReINVITE</FAX_Passthru_Method_{{␣
↪line_no }}_>
<FAX_Passthru_Codec_{{ line_no }}_ ua="na">G711a</FAX_Passthru_Codec_{{ line_
↪no }}_>
```

(continues on next page)

(continued from previous page)

```
<FAX_Disable_ECAN_{{ line_no }}_ ua="na">yes</FAX_Disable_ECAN_{{ line_no }}_>
<FAX_Tone_Detect_Mode_{{ line_no }}_ ua="na">caller or callee</FAX_Tone_Detect_
↪Mode_{{ line_no }}_>

<Network_Jitter_Level_{{ line_no }}_ ua="na">very high</Network_Jitter_Level_{
↪{ line_no }}_>
<Jitter_Buffer_Adjustment_{{ line_no }}_ ua="na">disable</Jitter_Buffer_
↪Adjustment_{{ line_no }}_>
{% endfor %}

<!-- SIP Parameters -->
<RTP_Packet_Size ua="na">0.020</RTP_Packet_Size>
<RTP-Start-Loopback_Codec ua="na">G711a</RTP-Start-Loopback_Codec>

<!-- Regional parameters -->
<Ring_Waveform ua="rw">Sinusoid</Ring_Waveform> <!-- options: Sinusoid/
↪Trapezoid -->
<Ring_Frequency ua="rw">50</Ring_Frequency>
<Ring_Voltage ua="rw">85</Ring_Voltage>

<FXS_Port_Impedance ua="na">600+2.16uF</FXS_Port_Impedance>
<Caller_ID_Method ua="na">Bellcore(N.Amer,China)</Caller_ID_Method>
<Caller_ID_FSK_Standard ua="na">bell 202</Caller_ID_FSK_Standard>

<!-- CUSTOM TPL - for faxes - END -->
```

3. Reconfigure the devices with:

```
xivo-provd-cli -c 'devices.using_plugin("xivo-cisco-spa3102-5.1.10").
↪reconfigure()'
```

4. Then reboot the devices:

```
xivo-provd-cli -c 'devices.using_plugin("xivo-cisco-spa3102-5.1.10").
↪synchronize()'
```

Most of this template can be copy/pasted for a SPA2102 or SPA8000.

5.13.5 Using a SIP Trunk

Fax transmission, to be successful, *MUST* use G.711 codec. Fax streams cannot be encoded with lossy compression codecs (like G.729a).

That said, you may want to establish a SIP trunk using G.729a for all other communications to save bandwith. Here's a way to be able to receive a fax in this configuration.

Note: There are some prerequisites:

- your SIP Trunk must offer both G.729a and G.711 codecs
- your fax users must have a customized outgoing calleridnum (for the codec change is based on this variable)

---

1. We assume that outgoing call rules and fax users with their DID are created
2. Create the file `/etc/asterisk/extensions_extra.d/fax.conf` with the following content:

```
;; For faxes :
; The following subroutine forces inbound and outbound codec to alaw.
; For outbound codec selection we must set the variable with inheritance.
```

(continues on next page)

(continued from previous page)

```
; Must be set on each Fax DID
[pre-incall-fax]
exten = s,1,NoOp(### Force alaw codec on both inbound (operator side) and␣
↪outbound (analog gw side) when calling a Fax ###)
exten = s,n,Set(SIP_CODEC_INBOUND=alaw)
exten = s,n,Set(__SIP_CODEC_OUTBOUND=alaw)
exten = s,n,Return()

; The following subroutine forces outbound codec to alaw based on outgoing␣
↪callerid number
; For outbound codec selection we must set the variable with inheritance.
; Must be set on each outgoing call rule
[pre-outcall-fax]
exten = s,1,NoOp(### Force alaw codec if caller is a Fax ###)
exten = s,n,GotoIf($["${CALLERID(num)}" = "0112697845"]?alaw:)
exten = s,n,GotoIf($["${CALLERID(num)}" = "0112697846"]?alaw:end)
exten = s,n(alaw),Set(__SIP_CODEC_OUTBOUND=alaw)
exten = s,n(end),Return()
```

3. For each Fax users' DID add the following string in the `Preprocess subroutine` field:

```
pre-incall-fax
```

4. For each Outgoing call rule add the the following string in the `Preprocess subroutine` field:

```
pre-outcall-fax
```

5.14 Graphics

The Services/Graphics section gives a historical overview of a XiVO system's activity based on snapshots recorded every 5 minutes. Graphics are available for the following resources :

- CPU
- Entropy
- Interruptions
- IRQ Stats
- System Load
- Memory Usage
- Open Files
- Open Inodes
- Swap Usage

Each section is presented as a series of 4 graphics : daily, weekly, monthly and yearly history. Each graphic can be clicked on to zoom. All information presented is read only.

5.15 Groups

Groups are used to be able to call a set or users.

Group name cannot be `general` reserved in asterisk configuration.

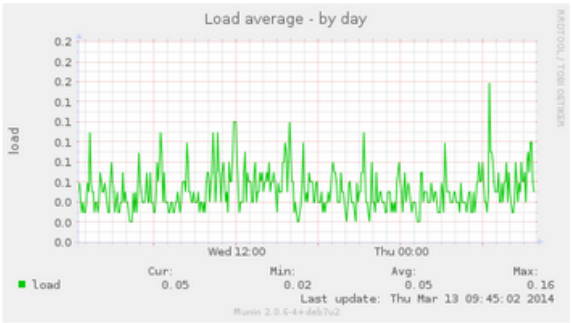
	Cur:	Min:	Avg:	Max:
Local timer interrupts	349.73	219.98	348.42	412.76
Spurious interrupts	0.00	0.00	0.00	0.00
Performance monitoring interrupts	0.00	0.00	0.00	0.00
IRQ work interrupts	0.00	0.00	0.00	0.00
Rescheduling interrupts	0.00	0.00	0.00	0.00
Function call interrupts	0.00	0.00	0.00	0.00
TLB shootdowns	0.00	0.00	0.00	0.00
Thermal event interrupts	0.00	0.00	0.00	0.00
Threshold APIC interrupts	0.00	0.00	0.00	0.00
Machine check exceptions	0.00	0.00	0.00	0.00
Machine check polls	3.33m	3.22m	3.33m	3.45m
ERR	0.00	0.00	0.00	0.00
MIS	0.00	0.00	0.00	0.00

Last update: Thu Mar 13 09:45:01 2014

	Cur:	Min:	Avg:	Max:
Local timer interrupts	350.26	93.43	363.09	7.12k
Spurious interrupts	0.00	0.00	0.00	0.00
Performance monitoring interrupts	0.00	0.00	0.00	0.00
IRQ work interrupts	0.00	0.00	0.00	0.00
Rescheduling interrupts	0.00	0.00	0.00	0.00
Function call interrupts	0.00	0.00	0.00	0.00
TLB shootdowns	0.00	0.00	0.00	0.00
Thermal event interrupts	0.00	0.00	0.00	0.00
Threshold APIC interrupts	0.00	0.00	0.00	0.00
Machine check exceptions	0.00	0.00	0.00	0.00
Machine check polls	3.33m	2.88m	3.33m	6.67m
ERR	0.00	0.00	0.00	0.00
MIS	0.00	0.00	0.00	0.00

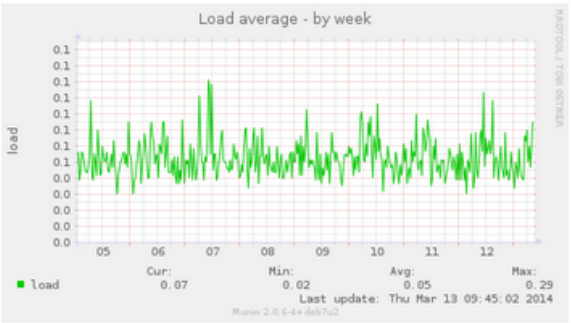
Last update: Thu Mar 13 09:45:01 2014

System load



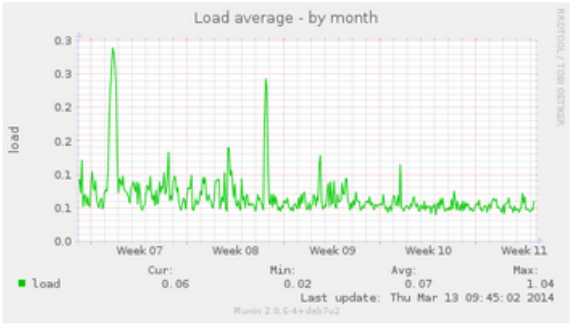
	Cur:	Min:	Avg:	Max:
load	0.05	0.02	0.05	0.16

Last update: Thu Mar 13 09:45:02 2014



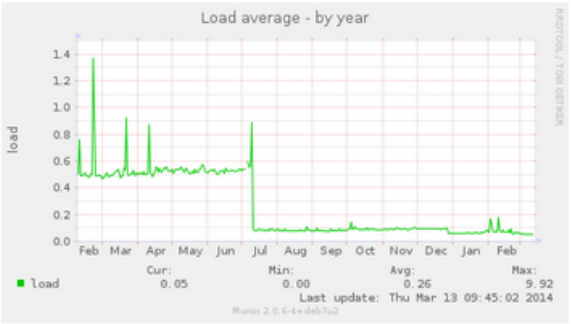
	Cur:	Min:	Avg:	Max:
load	0.07	0.02	0.05	0.29

Last update: Thu Mar 13 09:45:02 2014



	Cur:	Min:	Avg:	Max:
load	0.06	0.02	0.07	1.04

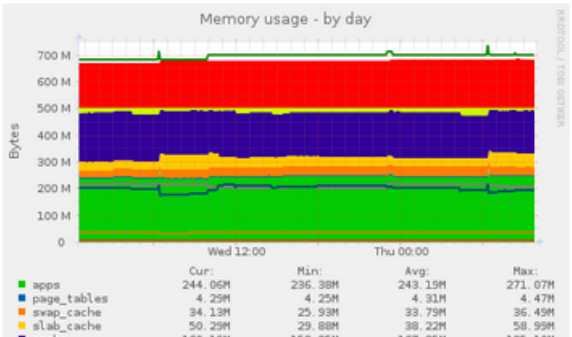
Last update: Thu Mar 13 09:45:02 2014



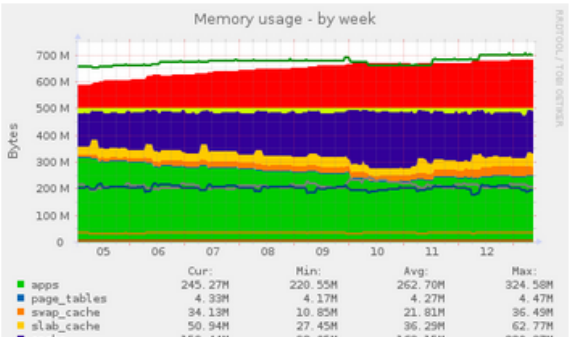
	Cur:	Min:	Avg:	Max:
load	0.05	0.00	0.26	9.92

Last update: Thu Mar 13 09:45:02 2014

Memory usage



	Cur:	Min:	Avg:	Max:
apps	244.06M	236.38M	243.19M	271.07M
page_tables	4.25M	4.25M	4.31M	4.47M
swap_cache	34.13M	25.93M	33.79M	36.49M
slab_cache	50.29M	29.88M	38.22M	58.99M
cache	160.19M	152.25M	167.85M	185.10M
buffers	56.00k	36.00k	55.81k	166.25k
unused	9.36M	6.18M	14.97M	27.96M
swap	182.28M	168.05M	175.99M	182.48M
inactive	209.53M	203.71M	212.85M	223.74M
committed	704.07M	684.67M	699.06M	735.48M
active	198.19M	176.96M	204.09M	220.10M
vmalloc_used	7.66M	7.66M	7.66M	7.66M
mapped	37.92M	36.35M	38.27M	39.79M



	Cur:	Min:	Avg:	Max:
apps	245.27M	220.55M	262.70M	324.58M
page_tables	4.33M	4.17M	4.27M	4.47M
swap_cache	34.13M	10.85M	21.81M	36.49M
slab_cache	50.94M	27.45M	36.29M	62.77M
cache	159.44M	98.05M	163.15M	220.37M
buffers	56.00k	36.00k	52.70k	166.72k
unused	8.23M	4.93M	14.11M	27.96M
swap	182.34M	84.75M	146.32M	182.48M
inactive	209.82M	195.29M	213.82M	253.62M
committed	704.67M	657.55M	678.42M	735.48M
active	197.56M	165.42M	205.51M	223.70M
vmalloc_used	7.66M	7.66M	7.66M	7.66M
mapped	38.02M	33.28M	37.65M	40.40M

5.16 Group Pickup

Pickup groups allow users to intercept calls directed towards other users of the group. This is done either by dialing a special extension or by pressing a function key.

5.16.1 Quick Summary

In order to be able to use group pickup you have to:

- Create a pickup group
- Enable an extension to intercept calls
- Add a function key to interceptors

5.16.2 Creating a Pickup Group

Pickup groups can be created in the *Services* → *IPBX* → *Call management* → *Call pickups* page.

In the *general* tab, you can define a name and a description for the pickup group. In the *Interceptors* tab, you can define a list of users, groups or queues that can intercept calls. In the *Intercepted* tab, you can define a list of users, groups or queues that can be intercepted.

5.16.3 Enabling an Interception Extension

The pickup extension can be defined in the *Services* → *IPBX* → *IPBX services* → *Extensions* page.

The extension used by group pickup is called *Group interception* it's default value is **8*.

> **Warning:** The extension must be enabled even if a function key is used.

5.16.4 Adding a Function Key to an Interceptor

To assign a function to an interceptor, go to *Services* → *IPBX* → *IPBX settings* → *Users*, edit an interceptor and go to the *Func Keys* tab.

Add a new function key of type *Group Interception* and save.

5.17 Incall

5.17.1 General Configuration

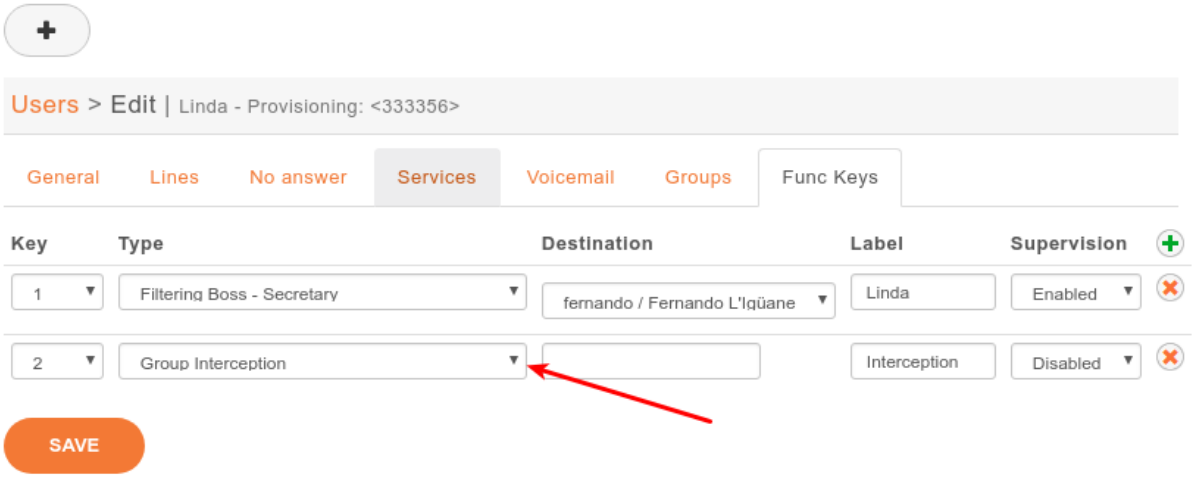
You can configure incoming calls settings in *Services* → *IPBX* → *Call Management* → *Incoming calls*.

DID (Direct Inward Dialing) Configuration

You can use special characters to match extensions. The most useful are:

```
. (period): will match one or more characters
X: will match only one character
```

Pickup groups > Edit | test
General
Interceptors
Intercepted
Groups
0 items selected
Remove all
Add all
groupecycle (10051@default)
groupegal (10050@default)
sda-tournante (10052@default)
Queues
0 items selected
Remove all
Add all
Account Dpt WR (3553@loadtest)
Accueil Limonest (3002@default)
Accueil Limonest hold (3003@default)
blue (3500@loadtest)
Car Rental RRM (3557@loadtest)
commerce (3200@default)
Hotline FC (3556@loadtest)
Users
3 items selected
Remove all
Add all
Père Noël (2702@default)
Linda (2701@default)
Fernando L'Igüane (2700@default)
Accueil Limonest (2300@default)
Accueil Rennes (2800@default)
Accueil Suresnes (2400@default)
acd01 acd01 (2551@loadtest)
acd02 acd02 (2552@loadtest)
acd03 acd03 (2553@loadtest)
acd04 acd04 (2554@loadtest)
SAVE



You can find more details about pattern matching in Asterisk (hence in XiVO) on the Asterisk wiki.

DID will not be validated against context ranges if pattern matching is used. Patterns are entered and displayed without the "_" prefix in the web interface.

5.18 Interconnections

5.18.1 Interconnect two XiVO directly

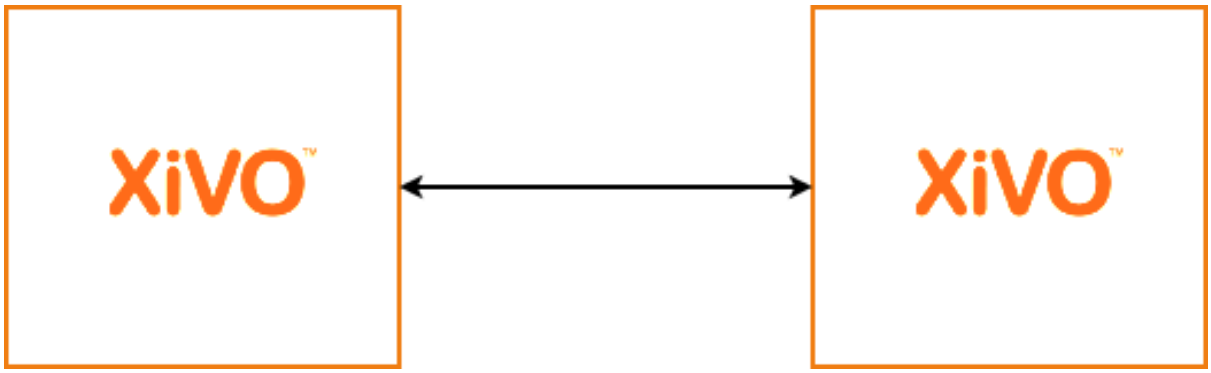


Fig. 31: Situation diagram

Interconnecting two XiVO will allow you to send and receive calls between the users configured on both sides.

The steps to configure the interconnections are:

- Establish the trunk between the two XiVO, that is the SIP connection between the two servers
- Configure outgoing calls on the server(s) used to emit calls
- Configure incoming calls on the server(s) used to receive calls

For now, only SIP interconnections have been tested.

Establish the trunk

The settings below allow a trunk to be used in both directions, so it doesn't matter which server is A and which is B.

Consider XiVO A wants to establish a trunk with XiVO B.

On XiVO B, go on page *Services → IPBX → Trunk management → SIP Protocol*, and create a SIP trunk:

```
Name : xivo-trunk
Username: xivo-trunk
Password: pass
Connection type: Friend
IP addressing type: Dynamic
Context: <see below>
```

Note: For the moment, Name and Username need to be the same string.

The `Context` field will determine which extensions will be reachable by the other side of the trunk:

- If `Context` is set to `default`, then every user, group, conf room, queue, etc. that have an extension if the `default` context will be reachable directly by the other end of the trunk. This setting can ease configuration if you manage both ends of the trunk.
- If you are establishing a trunk with a provider, you probably don't want everything to be available to everyone else, so you can set the `Context` field to `Incalls`. By default, there is no extension available in this context, so we will be able to configure which extension are reachable by the other end. This is the role of the incoming calls: making bridges from the `Incalls` context to other contexts.

On XiVO A, create the other end of the SIP trunk on the *Services → IPBX → Trunk management → SIP Protocol*:

```
Name: xivo-trunk
Username: xivo-trunk
Password: pass
Identified by: Friend
Connection type: Static
Address: <XiVO B IP address or hostname>
Context: Incalls
```

Register tab:

```
Register: checked
Transport: udp
Username: xivo-trunk
Password: pass
Remote server: <XiVO B IP address or hostname>
```

On both XiVO, activate some codecs, *Services → IPBX → General Settings → SIP protocol*, tab `Signaling`:

```
Enabled codecs: at least GSM (audio)
```

Warning: Without customizing the codecs, problems with sound quality or one-way sound may occur.

At that point, the Asterisk command `sip show registry` on XiVO B should print a line showing that XiVO A is registered, meaning your trunk is established.

Set the outgoing calls

The outgoing calls configuration will allow XiVO to know which extensions will be called through the trunk.

On the call emitting server(s), go on the page *Services → IPBX → Call management → Outgoing calls* and add an outgoing call.

Tab General:

```
Trunks: xivo-trunk
```

Tab Exten:

```
Exten: **99. (note the period at the end)
Stripnum: 4
```

This will tell XiVO: if any extension begins with `**99`, then try to dial it on the trunk `xivo-trunk`, after removing the 4 first characters (the `**99` prefix).

The most useful special characters to match extensions are:

```
. (period): will match one or more characters
X: will match only one character
```

You can find more details about pattern matching in Asterisk (hence in XiVO) on the Asterisk wiki.

Set the incoming calls

Now that we have calls going out from a XiVO, we need to route incoming calls on the XiVO destination.

Note: This step is only necessary if the trunk is linked to an Incoming calls context.

To route an incoming call to the right destination in the right context, we will create an incoming call in *Services → IPBX → Call management → Incoming calls*.

Tab General:

```
DID: 101
Context: Incalls
Destination: User
Redirect to: someone
```

This will tell XiVO: if you receive an incoming call to the extension `101` in the context `Incalls`, then route it to the user `someone`. The destination context will be found automatically, depending on the context of the line of the given user.

So, with the outgoing call set earlier on XiVO A, and with the incoming call above set on XiVO B, a user on XiVO A will dial `**99101`, and the user `someone` will ring on XiVO B.

5.18.2 Interconnect XiVO with a known SIP Provider

Connection to global telephony network can be configured automatically in this version of XiVO. For instructions how to configure it manually, see *Interconnect XiVO with any VoIP provider*.

Requirements

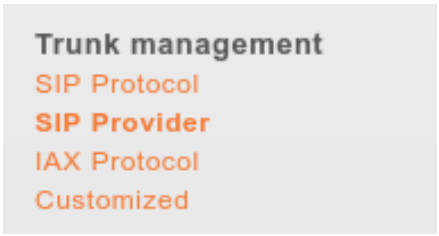
This is a premium feature that is not included in XiVO and is not freely available. To enable it, please contact XiVO.Solutions customer support.

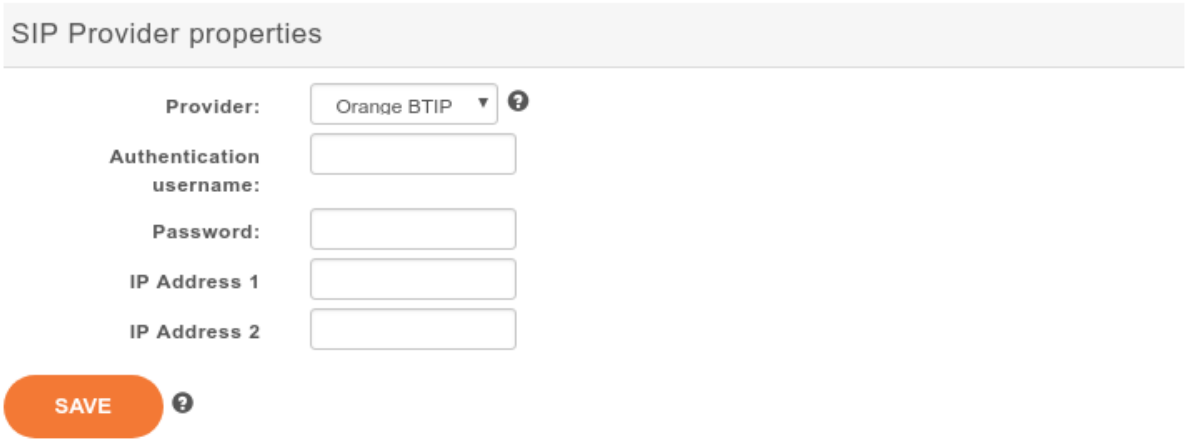
Overview

The SIP provider configuration can be applied from the SIP provider page.

On this page, you will not see all the settings that will be applied, but only those that must be personalised.

These settings will apply when you save the form:





- SIP trunk with the settings required by the provider will be created
- Data entered to the form will be included in the trunk
- If it is required by the provider, other XiVO settings will be changed. These settings can't be reverted by removing the trunk

Installation

- Open SIP Provider page from menu *Services* → *IPBX* → *Trunk management* → *SIP Provider*
- Choose provider
- Fill up the form
- Save
- Some *manual steps* may be required to complete the configuration

5.18.3 Interconnect XiVO with any VoIP provider

When you want to send and receive calls to the global telephony network, one option is to subscribe to a VoIP provider. To receive calls, your XiVO needs to tell your provider that it is ready and to which IP the calls must be sent. To send calls, your XiVO needs to authenticate itself, so that the provider knows that your XiVO is authorized to send calls and whose account must be credited with the call fare.

The steps to configure the interconnections are:

- Establish the trunk between the two XiVO, that is the SIP connection between the two servers
- Configure outgoing calls on the server(s) used to emit calls
- Configure incoming calls on the server(s) used to receive calls

Establish the trunk

You need the following information from your provider:

- a username
- a password
- the name of the provider VoIP server
- a public phone number

On your XiVO, go on page *Services → IPBX → Trunk management → SIP Protocol*, and create a SIP/IAX trunk:

```
Name : provider_username
Username: provider_username
Password: provider_password
Connection type: Peer
IP addressing type: voip.provider.example.com
Context: Incalls (or another incoming call context)
```

Register tab:

```
Register: checked
Transport: udp
Name: provider_username
Username: provider_username
Password: provider_password
Remote server: voip.provider.example.com
```

Note: For the moment, Name and Username need to be the same value.

If your XiVO is behind a NAT device or a firewall, you should set the following:

```
Monitoring: Yes
```

This option will make Asterisk send a signal to the VoIP provider server every 60 seconds (default settings), so that NATs and firewall know the connection is still alive. If you want to change the value of this cycle period, you have to select the appropriate value of the following parameter:

```
Qualify Frequency:
```

At that point, the Asterisk command `sip show registry` should print a line showing that you are registered, meaning your trunk is established.

Set the outgoing calls

The outgoing calls configuration will allow XiVO to know which extensions will be called through the trunk.

Go on the page *Services → IPBX → Call management → Outgoing calls* and add an outgoing call.

Tab General:

```
Trunks: provider_username
```

Tab Exten:

```
Exten: 418. (note the period at the end)
```

This will tell XiVO: if an internal user dials a number beginning with `418`, then try to dial it on the trunk `provider_username`.

The most useful special characters to match extensions are:

```
. (period): will match one or more characters
X: will match only one character
```

You can find more details about pattern matching in Asterisk (hence in XiVO) on the Asterisk wiki.

Set the incoming calls

Now that we have calls going out, we need to route incoming calls.

To route an incoming call to the right destination in the right context, we will create an incoming call in *Services → IPBX → Call management → Incoming calls*.

Tab General:

```
DID: your_public_phone_number
Context: Incalls (the same than configured in the trunk)
Destination: User
Redirect to: the_front_desk_guy
```

This will tell XiVO: if you receive an incoming call to the public phone number in the context `Incalls`, then route it to the user `the_front_desk_guy`. The destination context will be found automatically, depending on the context of the line of the given user.

5.18.4 Interconnect XiVO with a PBX via an ISDN link

The goal of this architecture can be one of:

- start a smooth migration between an old telephony system towards IP telephony with XiVO
- bring new features to the PBX like voicemail, conference, IVR etc.

First, XiVO is to be integrated transparently between the operator and the PBX. Then users or features are to be migrated from the PBX to the XiVO.

Warning: It requires a special call routing configuration on both the XiVO **and the PBX**.

Hardware

General uses

You must have an ISDN card able to support both the provider and PBX ISDN links.

Example : If you have two provider links towards the PBX, XiVO should have a 4 spans card : two towards the provider, and two towards the PBX.

If you use two cards

If you use two cards, you have to :

- Use a cable for clock synchronization between the cards
- Configure the *wheel* to define the cards order in the system.

Please refer to the section *Sync cable*

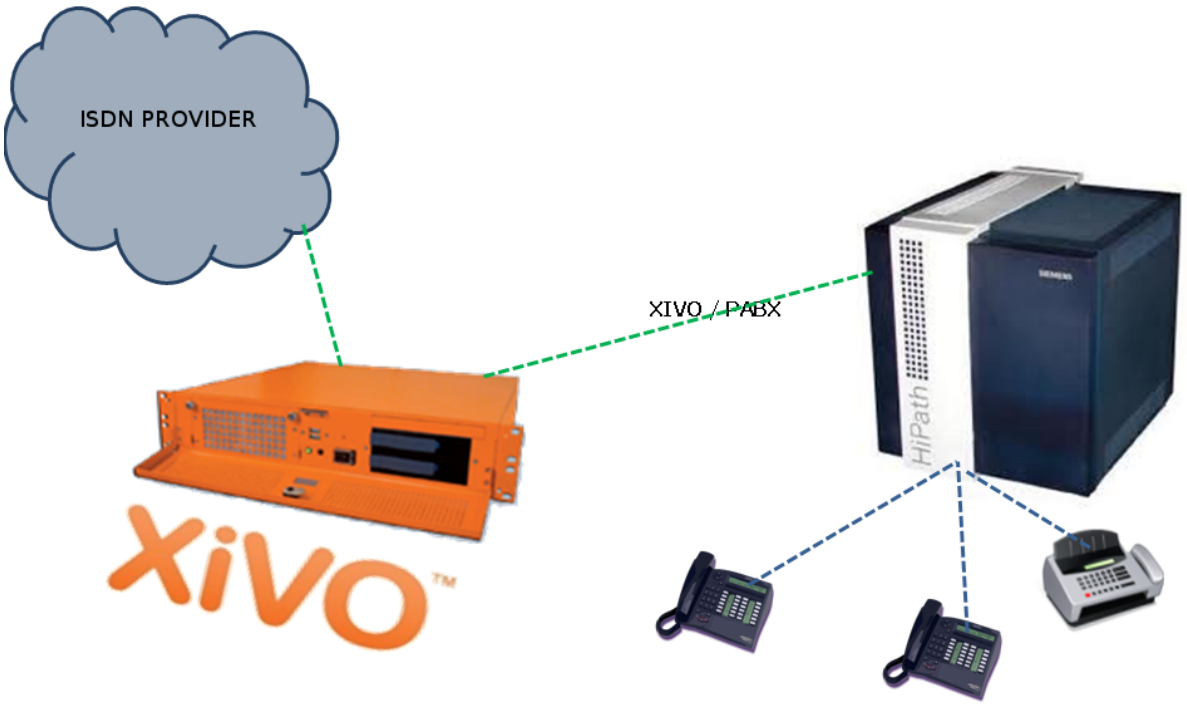


Fig. 32: Interconnect XiVO with a PBX

Configuration

You have now to configure two files :

1. `/etc/dahdi/system.conf`
2. `/etc/asterisk/dahdi-channels.conf`

system.conf

You mainly need to configure the `timing` parameter on each *span*. As a general rule :

- Provider *span* - XiVO will get the clock from the provider : the `timing` value is to be different from 0 (see */etc/dahdi/system.conf* section)
- PBX *span* - XiVO will provide the clock to the PBX : the `timing` value is to be set to 0 (see */etc/dahdi/system.conf* section)

Below is an example with two provider links and two PBX links:

```
# Span 1: TE4/0/1 "TE4XXP (PCI) Card 0 Span 1" (MASTER)
span=1,1,0,ccs,hdb3          # Span towards Provider
bchan=1-15,17-31
dchan=16
echocanceller=mg2,1-15,17-31

# Span 2: TE4/0/2 "TE4XXP (PCI) Card 0 Span 2"
span=2,2,0,ccs,hdb3          # Span towards Provider
bchan=32-46,48-62
dchan=47
echocanceller=mg2,32-46,48-62

# Span 3: TE4/0/3 "TE4XXP (PCI) Card 0 Span 3"
span=3,0,0,ccs,hdb3          # Span towards PBX
bchan=63-77,79-93
dchan=78
echocanceller=mg2,63-77,79-93

# Span 4: TE4/0/4 "TE4XXP (PCI) Card 0 Span 4"
span=4,0,0,ccs,hdb3          # Span towards PBX
bchan=94-108,110-124
dchan=109
echocanceller=mg2,94-108,110-124
```

dahdi-channels.conf

In the file `/etc/asterisk/dahdi-channels.conf` you need to adjust, for each span :

- `group` : the group number (e.g. `0` for provider links, `2` for PBX links),
- `context` : the context (e.g. `from-extern` for provider links, `from-pabx` for PBX links)
- `signalling` : `pri_cpe` for provider links, `pri_net` for PBX side

Warning: most of the PBX uses overlap dialing for some destination (digits are sent one by one instead of by block). In this case, the `overlapdial` parameter has to be activated on the PBX spans:

```
overlapdial = incoming
```

Below an example of `/etc/asterisk/dahdi-channels.conf`:

```
; Span 1: TE4/0/1 "TE4XXP (PCI) Card 0 Span 1" (MASTER)
group=0,11
context=from-extern
switchtype = euroisdn
signalling = pri_cpe
channel => 1-15,17-31

; Span 2: TE4/0/2 "TE4XXP (PCI) Card 0 Span 2"
group=0,12
context=from-extern
switchtype = euroisdn
signalling = pri_cpe
channel => 32-46,48-62

; PBX link #1
; Span 3: TE4/0/3 "TE2XXP (PCI) Card 0 Span 3"
group=2,13
context=from-pabx     ; special context for PBX incoming calls
overlapdial=incoming  ; overlapdial activation
switchtype = euroisdn
signalling = pri_net  ; behave as the NET termination
channel => 63-77,79-93

; PBX link #2
; Span 4: TE4/0/4 "T4XXP (PCI) Card 0 Span 4"
group=2,14
context=from-pabx     ; special context for PBX incoming calls
overlapdial=incoming  ; overlapdial activation
switchtype = euroisdn
signalling = pri_net  ; behave as the NET termination
channel => 94-108,110-124
```

Passthru function

Route PBX incoming calls

We first need to create a route for calls coming from the PBX

Create a file named `pbx.conf` in the directory `/etc/asterisk/extensions_extra.d/`, # Add the following lines in the file:

```
[from-pabx]
exten = _X.,1,NoOp(### Call from PBX ${CARLLERID(num)} towards ${EXTEN} ###)
exten = _X.,n,Goto(default,${EXTEN},1)
```

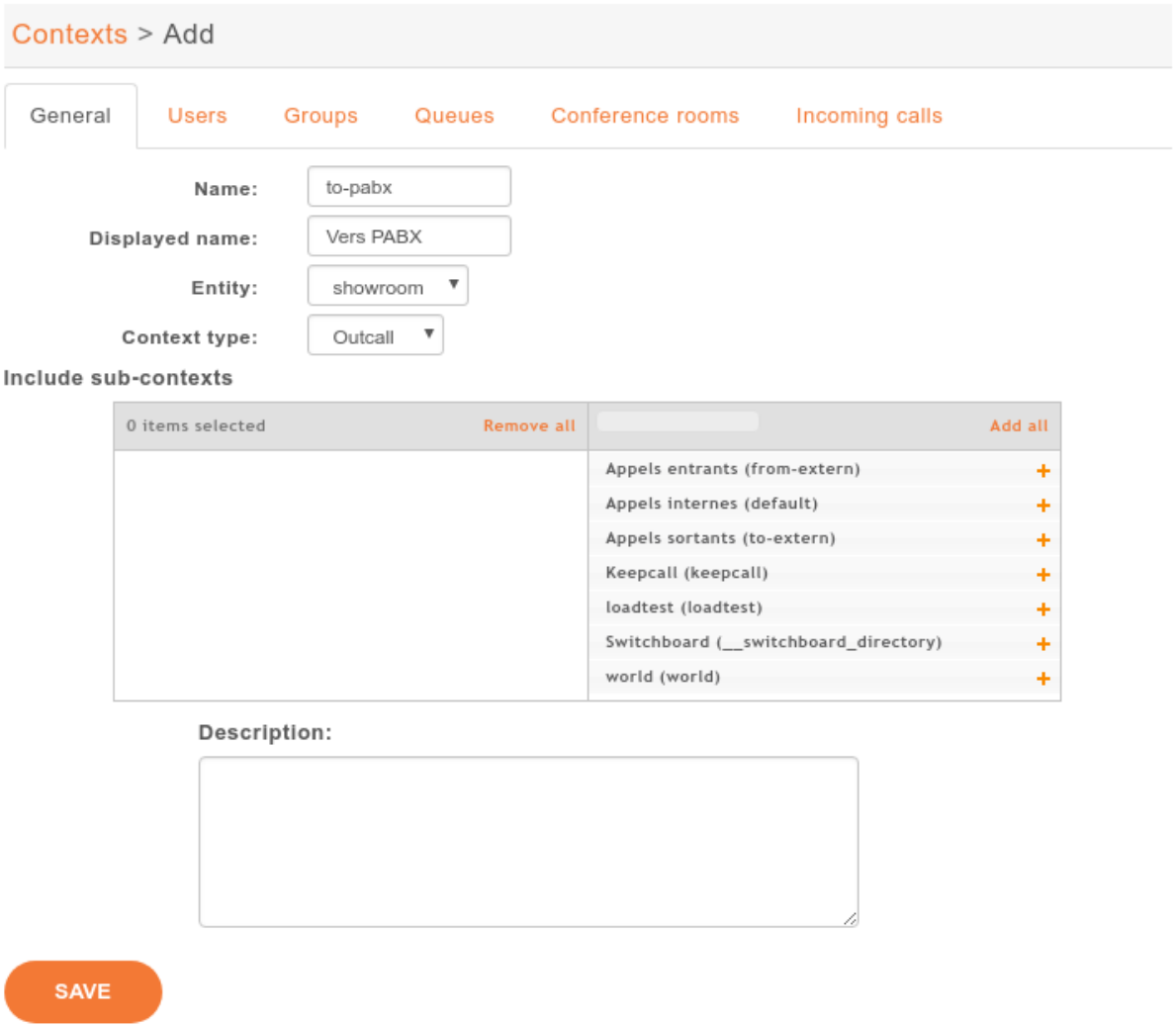
This dialplan routes incoming calls from the PBX in the `default` context of XiVO. It enables call from the PBX : * towards a SIP phone (in `default` context) * towards outgoing destniation (via the `to-extern` context included in `default` context)

Create the to-pabx context

In the webi, create a context named `to-pabx`:

- Name : to-pabx
- Display Name : TO PBX
- Context type : Outcall
- Include sub-contexts : No context inclusion

This context will permit to route incoming calls from the XiVO to the PBX.



Route incoming calls to PBX

In our example, incoming calls on spans 1 and 2 (spans pluged to the provider) are routed by from-extern context. We are going to create a default route to redirect incoming calls to the PBX.

Create an incoming call as below :

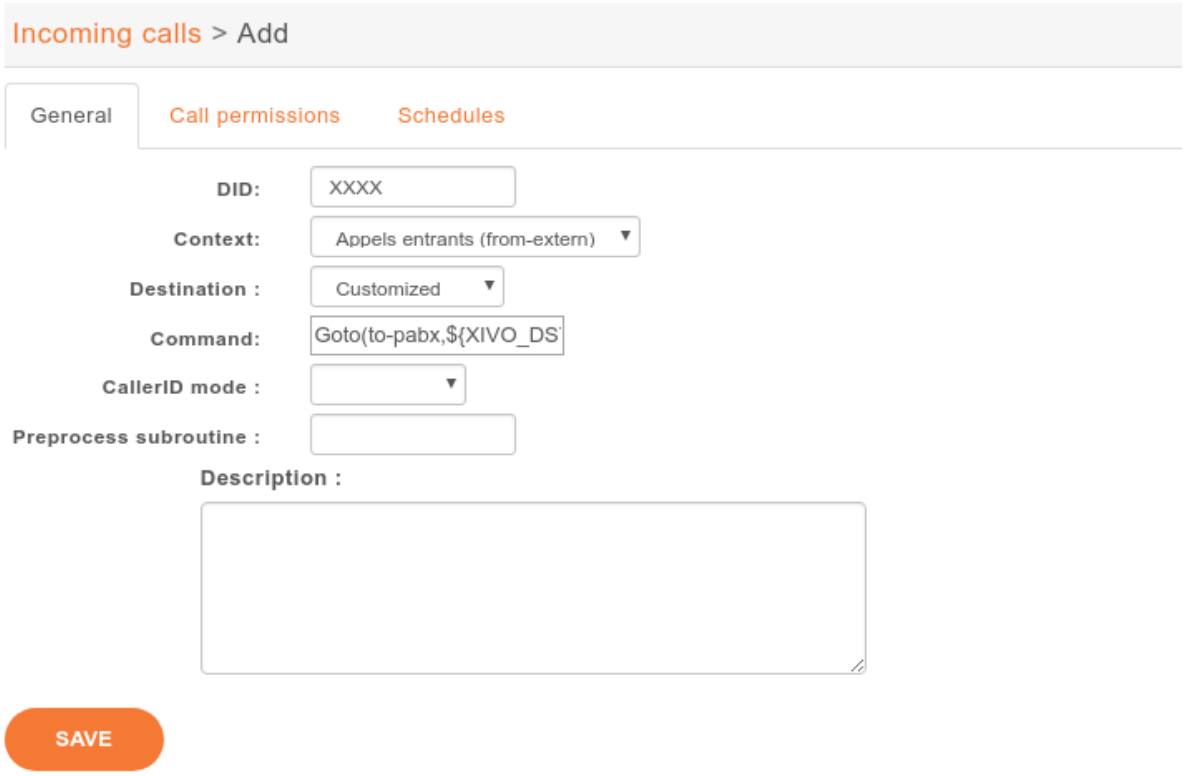
- DID : XXXX (according to the number of digits sent by the provider)
- Context : Incoming calls
- Destination : Customized
- Command : Goto(to-pabx,${XIVO_DSTNUM},1)

Create the interconnections

You have to create two interconnections :

- provider side : dahdi/g0
- PBX side : dahdi/g2

In the menu *Services* → *IPBX* → *Trunk management* → *Customized* page :



- Name : t2-operateur
- Interface : dahdi/g0
- Context : to-extern

The second interconnection :

- Name : t2-pabx
- Interface : dahdi/g2
- Context : to-pabx

Create outgoing calls

You must create two rules of outgoing calls in the menu *Services → IPBX → Call management → Outgoing calls* page :

1. Redirect calls to the PBX :

- Name : fsc-pabx
- Context : to-pabx
- Trunks : choose the *t2-pabx* interconnection

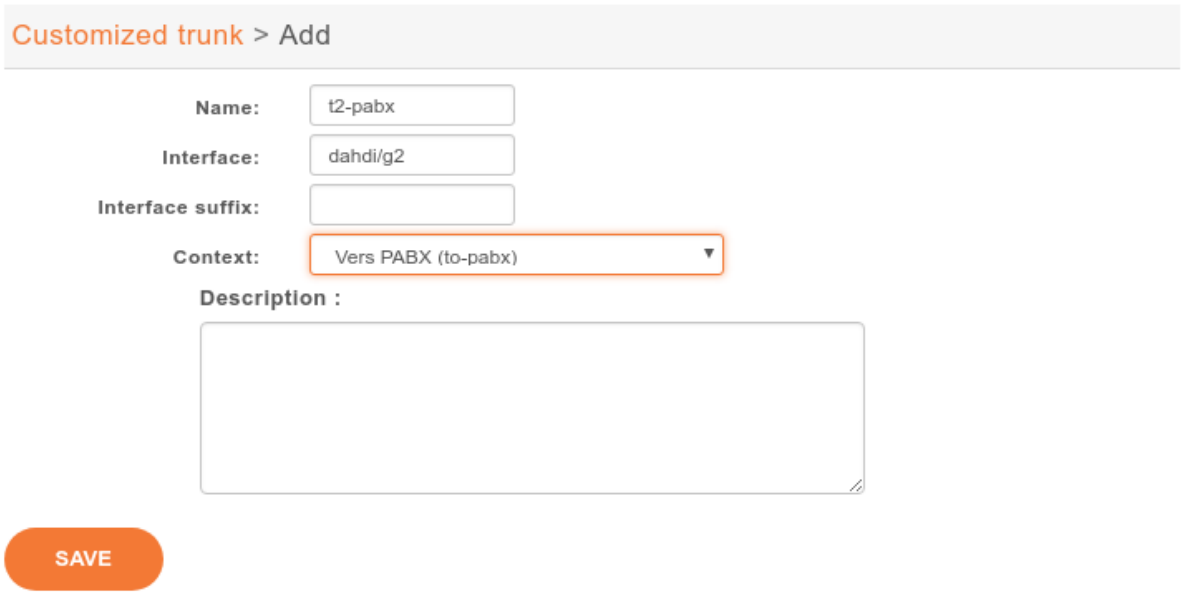
In the extensions tab :

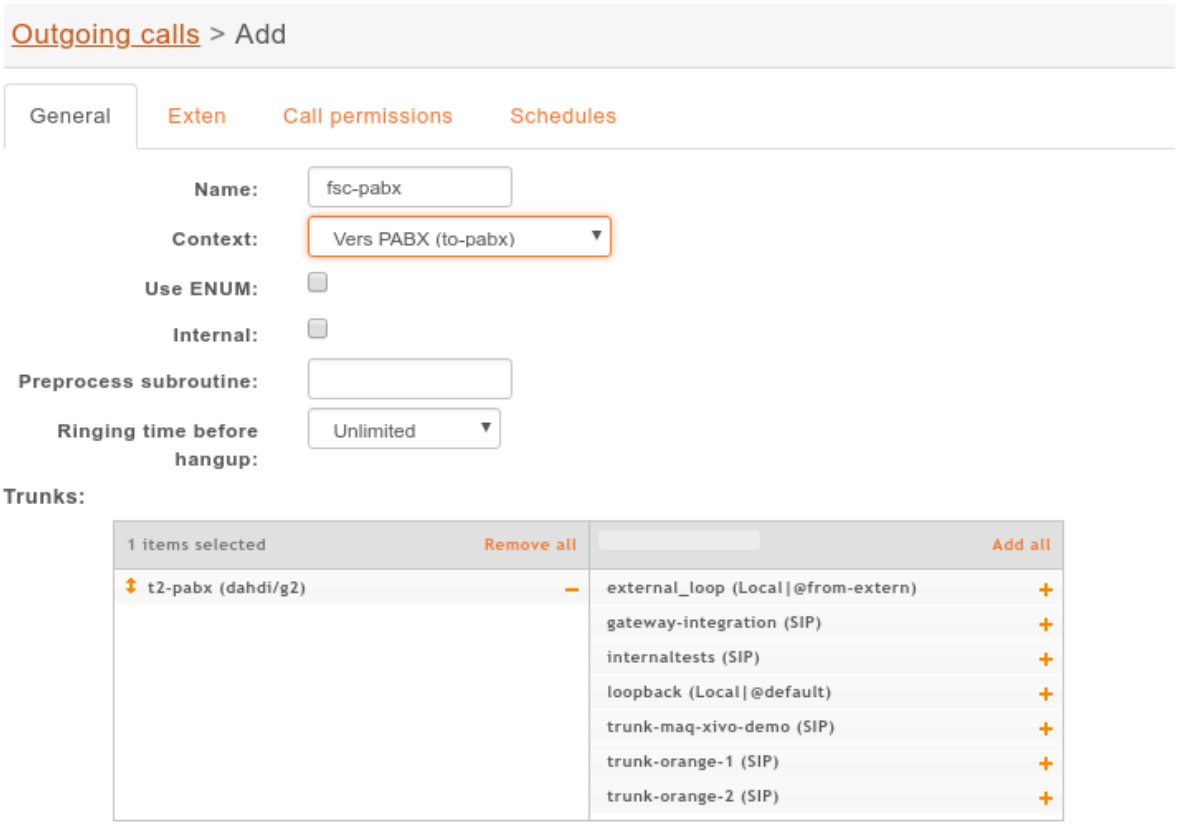
- Exten : XXXX

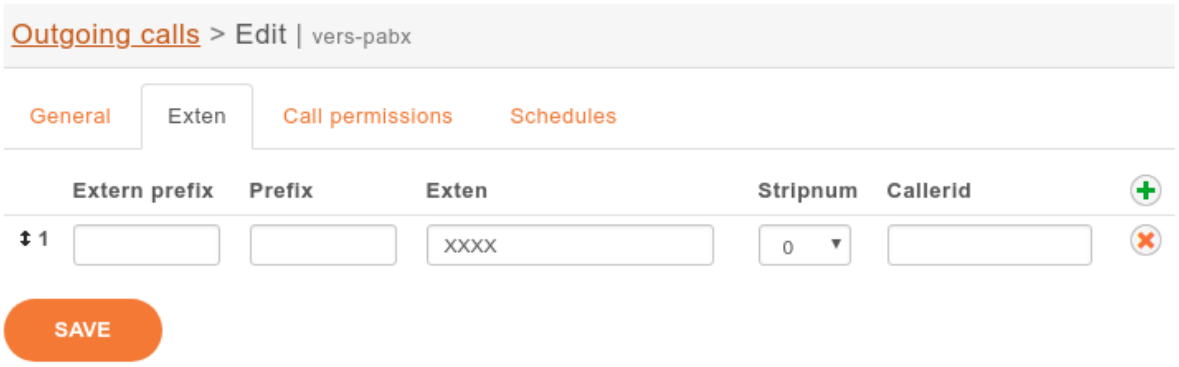
2. Create a rule “fsc-operateur”:

- Name : fsc-operateur
- Context : to-extern
- Trunks : choose the “t2-operateur” interconnection

Customized trunk > Add
Name:
t2-operateur
Interface:
dahdi/g0
Interface suffix:
Context:
Appels sortants (to-extern)
Description :
SAVE


Customized trunk > Add
Name:
t2-pabx
Interface:
dahdi/g2
Interface suffix:
Context:
Vers PABX (to-pabx)
Description :
SAVE


Outgoing calls > Add
General
Exten
Call permissions
Schedules
Name:
fsc-pabx
Context:
Vers PABX (to-pabx)
Use ENUM:
Internal:
Preprocess subroutine:
Ringing time before hangup:
Unlimited
Trunks:
1 items selected
Remove all
Add all
t2-pabx (dahdi/g2)
external_loop (Local|@from-extern)
gateway-integration (SIP)
internaltests (SIP)
loopback (Local|@default)
trunk-maq-xivo-demo (SIP)
trunk-orange-1 (SIP)
trunk-orange-2 (SIP)


Outgoing calls > Edit | vers-pabx
General
Exten
Call permissions
Schedules
Extern prefix
Prefix
Exten
Stripnum
Callerid
1
XXXX
0
SAVE

In the extensions tab:

```
exten = X.
```

5.18.5 Create an interconnection

There are three types of interconnections :

- Customized
- SIP
- IAX

Customized interconnections

Customized interconnections are mainly used for interconnections using DAHDI or Local channels:

- *Name* : it is the name which will appear in the outcall interconnections list,
- *Interface* : this is the channel name (for DAHDI see *DAHDI interconnections*)
- *Interface suffix* (optional) : a suffix added after the dialed number (in fact the Dial command will dial:

  ```
  <Interface>/<EXTEN><Interface suffix>
  ```

- *Context* : currently not relevant

SIP interconnections

- *General*, *Signaling* and *Advanced* tabs create the SIP peer information
- *Register* tab creates the registration chain

Note: in *XiVO PBX* Web interface slash "/" character is not supported in the password field.

DAHDI interconnections

To use your DAHDI links you must create a customized interconnection.

Name : the name of the interconnection like **e1_span1** or **bri_port1**

Interface : must be of the form `dahdi/[group order][group number]` where :

- `group order` is one of :
  - `g` : pick the first available channel in group, searching from lowest to highest,
  - `G` : pick the first available channel in group, searching from highest to lowest,
  - `r` : pick the first available channel in group, going in round-robin fashion (and remembering where it last left off), searching from lowest to highest,
  - `R` : pick the first available channel in group, going in round-robin fashion (and remembering where it last left off), searching from highest to lowest.
- `group number` is the group number to which belongs the span as defined in the */etc/asterisk/dahdi-channels.conf*.

Warning: if you use a BRI card you MUST use per-port dahdi groups. You should not use a group like g0 which spans over several spans.

For example, add an interconnection to the menu *Services* → *IPBX* → *Trunk management* → *Customized*

```
Name : interconnection name
Interface : dahdi/g0
```

5.18.6 Debug

Interesting Asterisk commands:

```
sip show peers
sip show registry
sip set debug on
```

5.18.7 Caller ID

When setting up an interconnection with the public network or another PBX, it is possible to set a caller ID in different places. Each way to configure a caller ID has it's own use case.

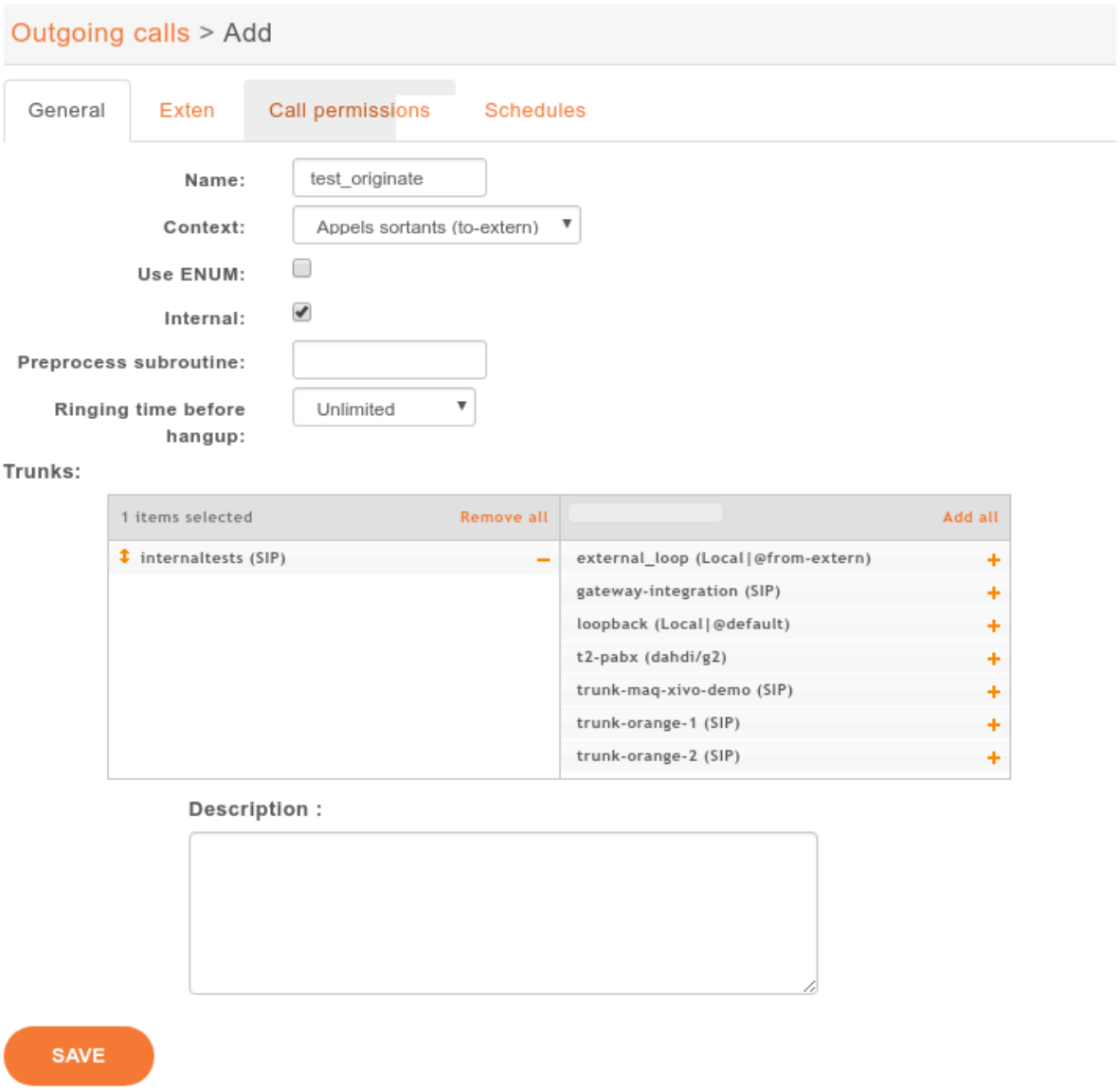
The format for a caller ID is the following `"My Name" <9999>` If you don't set the number part of the caller ID, the dialplan's number will be used instead. This might not be a good option in most cases. If you only need to set a *number* as an outgoing caller ID, you just have to put the number in the caller ID field like `0123456789`.

5.18.8 Outgoing call caller ID

There are several behavior for the outgoing caller ID.

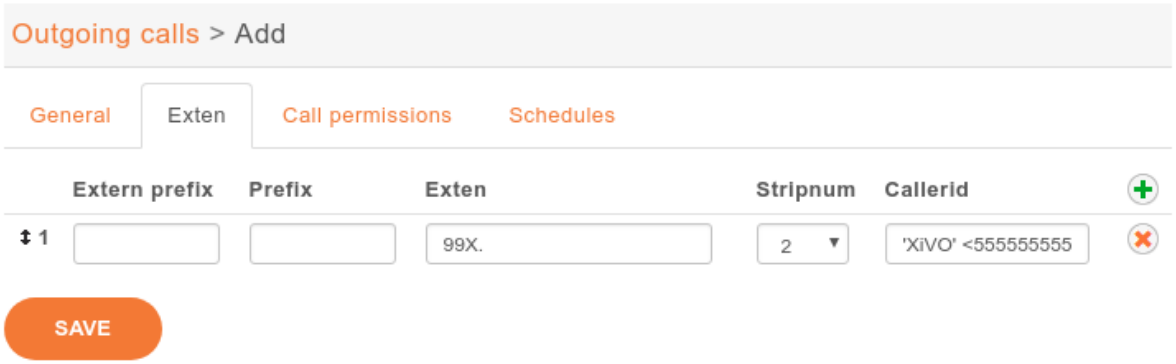
Use internal caller ID

When you create an outgoing call, it's possible to set it to internal, using the check box in the outgoing call configuration menu. When this option is activated, the internal caller's caller ID will be forwarded to the trunk. This option is useful when the other side of the trunk can reach the user with it's caller ID number.


Outgoing calls > Add
General
Exten
Call permissions
Schedules
Name:
test_originate
Context:
Appels sortants (to-extern)
Use ENUM:
Internal:
Preprocess subroutine:
Ringing time before hangup:
Unlimited
Trunks:
1 items selected
Remove all
Add all
internaltests (SIP)
external_loop (Local|@from-extern)
gateway-integration (SIP)
loopback (Local|@default)
t2-pabx (dahdi/g2)
trunk-maq-xivo-demo (SIP)
trunk-orange-1 (SIP)
trunk-orange-2 (SIP)
Description :
SAVE

Use outgoing caller ID

When the internal caller's caller ID is not usable to the called party, the outgoing call's caller id can be fixed to a given value that is more useful to the outside world. Giving the public number here might be a good idea.



A user can also have a forced caller ID for outgoing calls. This can be useful for a user who has his own public number (DID number). This option can be set in the user's configuration page. For this, the *Outgoing Caller ID* option must be set to Customize.

The user can also set his outgoing caller ID to Anonymous.

If you use a SIP provider trunk, and if your provider supports the RFC3325 for Anonymous calls, you have to set the Send the Remote-Party-ID option of your SIP trunk to **PAI**:

1. *Services* → *Trunk management* → *SIP Protocol* → *Edit* → *tab Advanced*
2. set parameter *Send the Remote-Party-ID* to **PAI**

With this option anonymous calls will be sent to your SIP provider with the RFC 3325 standard. Note that in this case, the *P-Asserted-Identity* SIP Header will contain the Outgoing caller ID number if set. Otherwise it will use the user's internal caller id, which not a good idea. So you should configure a default caller ID in the outgoing call.

Order of precedence

The order of precedence when setting the caller ID in multiple places is the following.

1. Internal
2. User's outgoing caller ID
3. Outgoing call caller ID
4. Default caller ID

5.19 Interactive Voice Response

5.19.1 Introduction

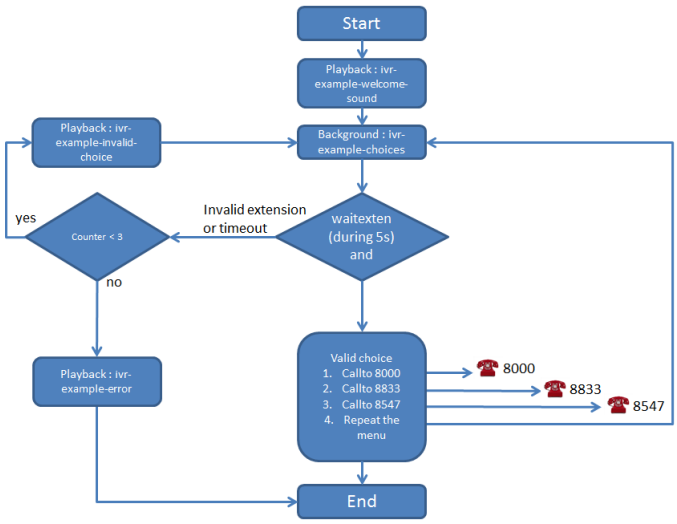
> *Interactive voice response (IVR) is a technology that allows a computer to interact with humans through the use of voice and DTMF tones input via keypad. In telecommunications, IVR allows customers to interact with a company's host system via a telephone keypad or by speech recognition, after which they can service their own inquiries by following the IVR dialogue.*
>
> —Wikipedia

The IVR can be easily added to XiVO using scripts. These scripts are written using the asterisk embeded language also named dialplan .

Users > Edit
General
Lines
No answer
Services
Voicemail
Groups
Func Keys
First name:
User1
Last name:
Mobile phone number:
E-mail:
Schedules:
Ringing time:
30 seconds
Simultaneous calls:
2
On-Hold Music:
default
Language:
fr FR
Timezone:
Caller ID:
User1
Outgoing Caller ID:
Customize
"Bob" <555555123
Preprocess subroutine:
User field :
XiVO Client
Enable XiVO Client:
Login:
bob
Password:
passss
Profile:
Client
Description:
SAVE

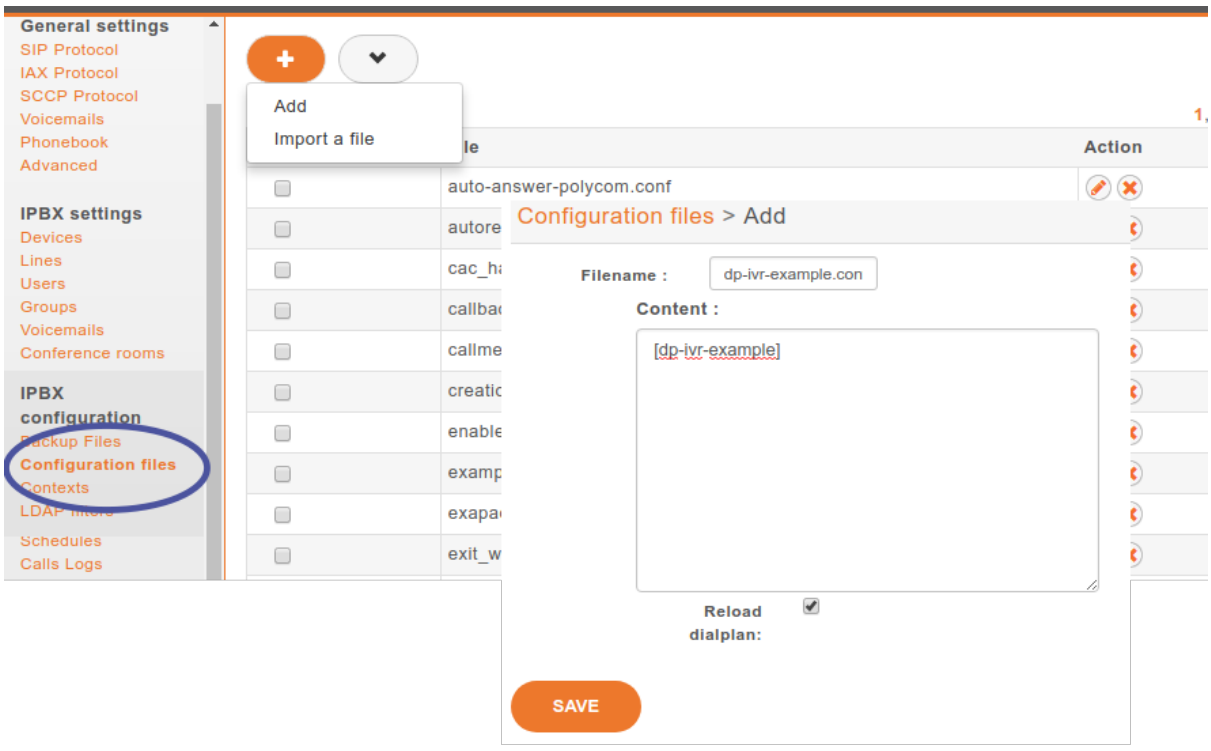
5.19.2 Use Case: Minimal IVR

Flowchart



Configuration File and Dialplan

First step, you need to create a configuration file, that contain an asterisk context and your IVR dialpan. In our example, both (file and context) are named dp-ivr-example.



Copy all these lines in the newly created configuration file (in our case, dp-ivr-example) :

```
[dp-ivr-example]

exten = s,1,NoOp(### dp-ivr-example.conf ###)
```

(continues on next page)

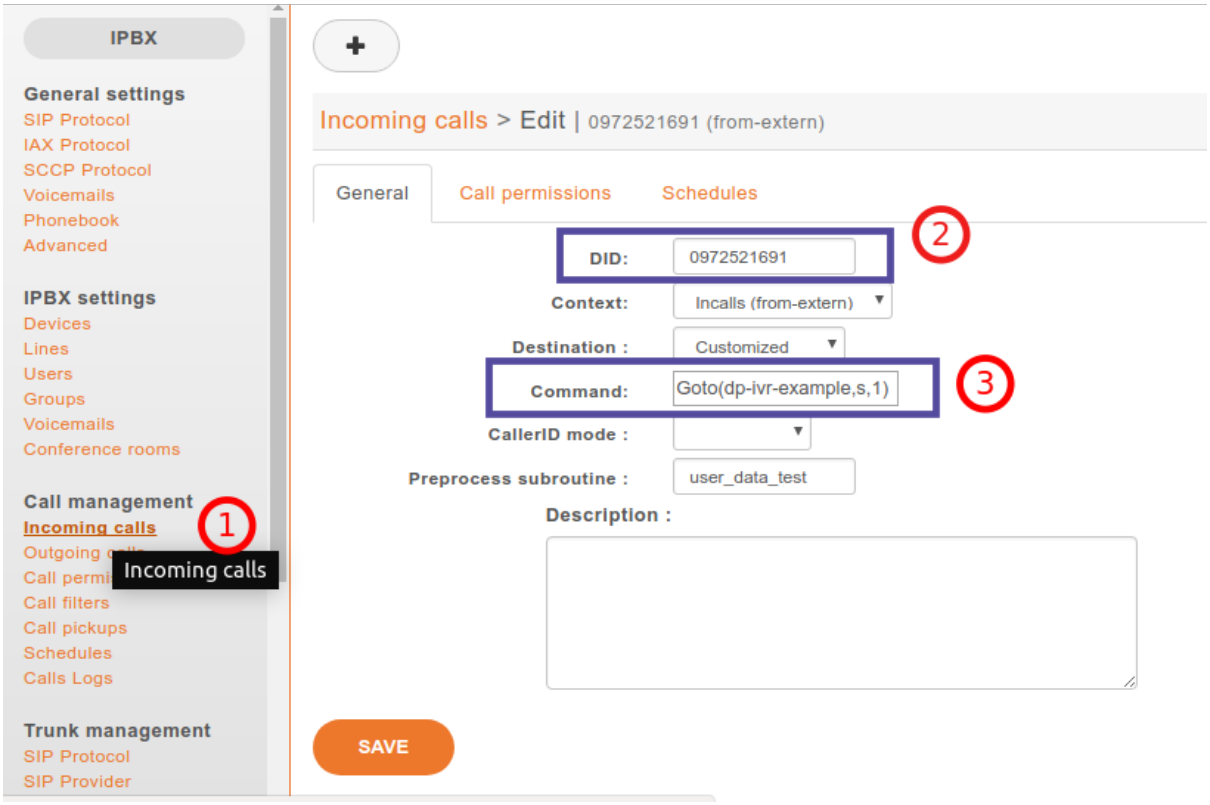
(continued from previous page)

```
same = n,NoOp(Set the context containing your ivr destinations.)
same = n,Set(IVR_DESTINATION_CONTEXT=my-ivr-destination-context)
same = n,NoOp(Set the directory containing your ivr sounds.)
same = n,Set(GV_DIRECTORY_SOUNDS=/var/lib/xivo/sounds/ivr-sounds)
same = n,NoOp(the system answers the call and waits for 1 second before continuing)
same = n,Answer(1000)

same = n,NoOp(the system plays the first part of the audio file "welcome to ...")
same = n(first),Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-welcome-sound)

same = n,NoOp(variable "counter" is set to 0)
same = n(beginning),Set(counter=0)

same = n,NoOp(variable "counter" is incremented and the label "start" is defined)
same = n(start),Set(counter=$[${counter} + 1])

same = n,NoOp(counter variable is now = ${counter})
same = n,NoOp(waiting for 1 second before reading the message that indicate all␣
↪choices)
same = n,Wait(1)
same = n,NoOp(play the message ivr-example-choices that contain all choices)
same = n,Background(${GV_DIRECTORY_SOUNDS}/ivr-example-choices)
same = n,NoOp(waiting for DTMF during 5s)
same = n,Waitexten(5)

;##### CHOICE 1 #####
exten = 1,1,NoOp(pressed digit is 1, redirect to 8000 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 1,n,Goto(${IVR_DESTINATION_CONTEXT},8000,1)

;##### CHOICE 2 #####
exten = 2,1,NoOp(pressed digit is 2, redirect to 8833 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 2,n,Goto(${IVR_DESTINATION_CONTEXT},8833,1)

;##### CHOICE 3 #####
exten = 3,1,NoOp(pressed digit is 3, redirect to 8547 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 3,n,Goto(${IVR_DESTINATION_CONTEXT},8547,1)

;##### CHOICE 4 #####
exten = 4,1,NoOp(pressed digit is 4, redirect to start label in this context)
exten = 4,n,Goto(s,start)

;##### TIMEOUT #####
exten = t,1,NoOp(no digit pressed for 5s, process it like an error)
exten = t,n,Goto(i,1)

;##### INVALID CHOICE #####
exten = i,1,NoOp(if counter variable is 3 or more, then goto label "error")
exten = i,n,GotoIf($[${counter}>=3]?error)
exten = i,n,NoOp(pressed digit is invalid and less than 3 errors: the guide ivr-
↪exemple-invalid-choice is now played)
exten = i,n,Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-invalid-choice)
exten = i,n,Goto(s,start)
exten = i,n(error),Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-error)
exten = i,n,Hangup()
```

IVR external dial

To call the script dp-ivr-example from an external phone, you must create an incoming call and redirect the call to the script dp-ivr-example with the command :

```
Goto(dp-ivr-example,s,1)
```



IVR internal dial

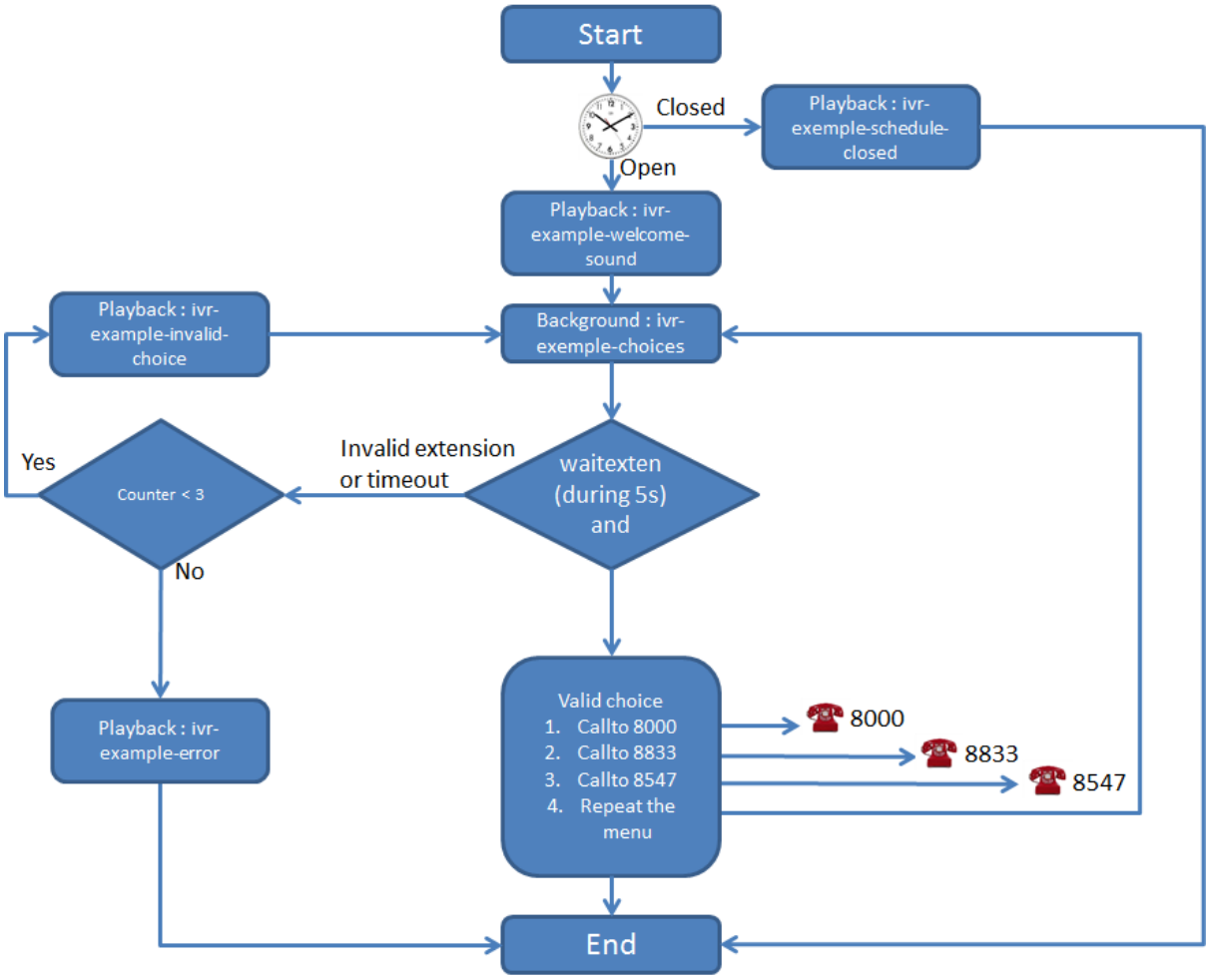
To call the script dp-ivr-example from an internal phone you must create an entry in the default context (`xivo-extrafeatures` is included in `default`). The best way is to add the extension in the file `xivo-extrafeatures.conf`.

```
exten => 8899,1,Goto(dp-ivr-example,s,1)
```

5.19.3 Use Case: IVR with a schedule

In many cases, you need to associate your IVR to a schedule to indicate when your company is closed.

Flowchart



Create Schedule

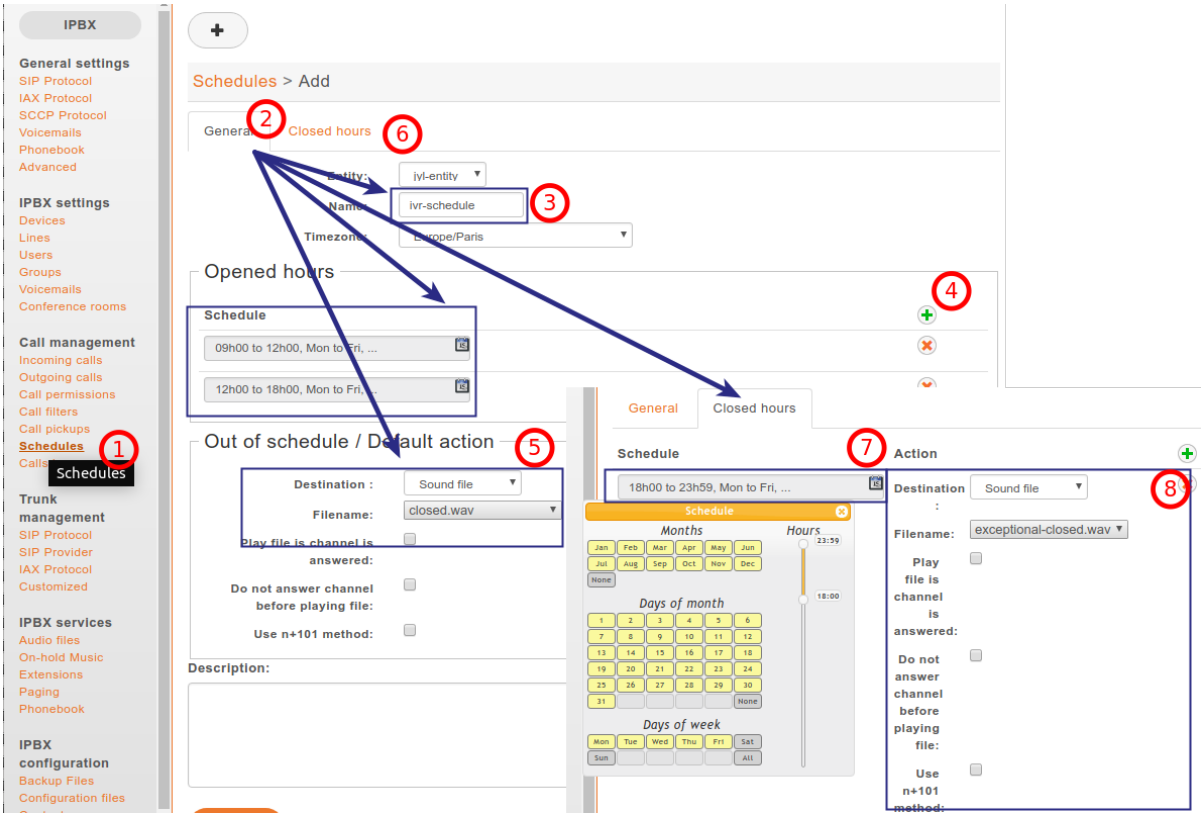
First step, create your schedule (1) from the menu *Call management* → *Schedules*. In the General tab, give a name (3) to your schedule and configure the open hours (4) and select the sound which is played when the company is closed.

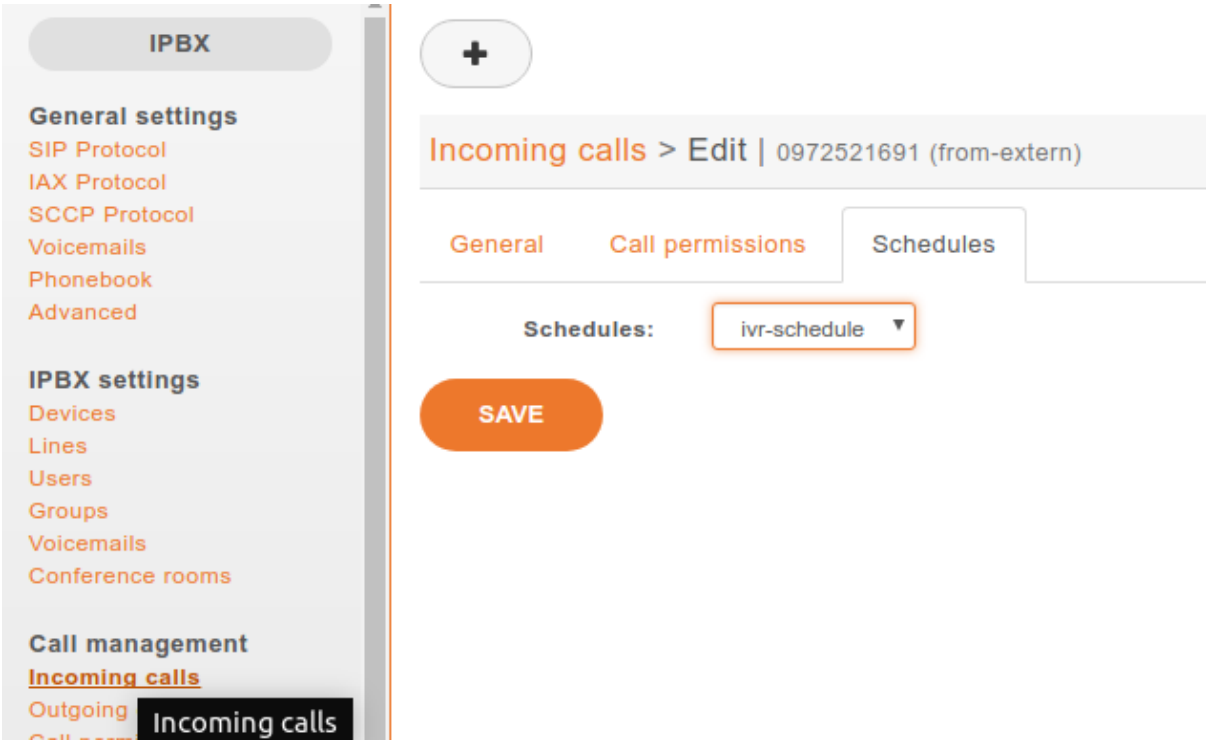
In the Closed hours tab (6), configure all special closed days (7) and select the sound that indicate to the caller that the company is exceptionally closed.

The IVR script is now only available during workdays.

Assign Schedule to Incall

Return editing your Incall (*Call management* → *Incoming calls*) and assign the newly created schedule in the "Schedules" tab


IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
Outgoing calls
Call permissions
Call filters
Call pickups
Schedules
Trunk management
SIP Protocol
SIP Provider
IAX Protocol
Customized
IPBX services
Audio files
On-hold Music
Extensions
Paging
Phonebook
IPBX configuration
Backup Files
Configuration files
Schedules
Schedules > Add
General
Closed hours
Entity:
ivl-entity
Name:
ivr-schedule
Timezone:
Europe/Paris
Opened hours
Schedule
09h00 to 12h00, Mon to Fri, ...
12h00 to 18h00, Mon to Fri, ...
Out of schedule / Default action
Destination :
Sound file
Filename:
closed.wav
Play file is channel is answered:
Do not answer channel before playing file:
Use n+101 method:
Description:
General
Closed hours
Schedule
18h00 to 23h59, Mon to Fri, ...
Months
Hours
Days of month
Days of week
Action
Destination :
Sound file
Filename:
exceptional-closed.wav
Play file is channel is answered:
Do not answer channel before playing file:
Use n+101 method:
1
2
3
4
5
6
7
8


IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
Outgoing
Incoming calls
Incoming calls > Edit | 0972521691 (from-extern)
General
Call permissions
Schedules
Schedules:
ivr-schedule
SAVE

5.19.4 Use Case: IVR with submenu

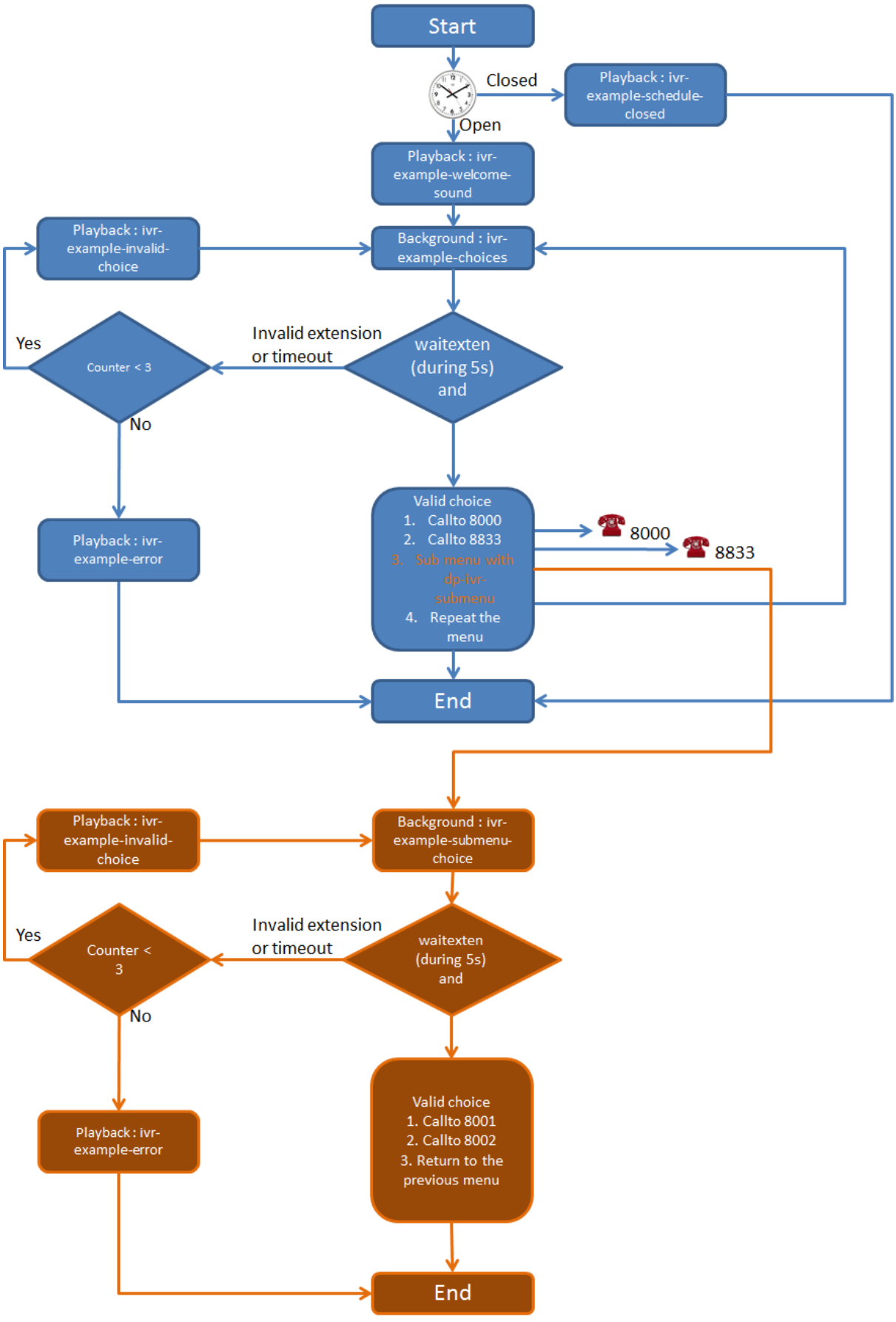
Flowchart

Configuration File and Dialplan

Copy all these lines (2 contexts) in a configuration file on your XiVO server :

```
[dp-ivr-example]

exten = s,1,NoOp(### dp-ivr-example.conf ###)
same = n,NoOp(Set the context containing your ivr destinations.)
same = n,Set(IVR_DESTINATION_CONTEXT=my-ivr-destination-context)
same = n,NoOp(Set the directory containing your ivr sounds.)
same = n,Set(GV_DIRECTORY_SOUNDS=/var/lib/xivo/sounds/ivr-sounds)
same = n,NoOp(the system answers the call and waits for 1 second before continuing)
same = n,Answer(1000)

same = n,NoOp(the system plays the first part of the audio file "welcome to ...")
same = n(first),Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-welcome-sound)

same = n,NoOp(variable "counter" is set to 0)
same = n(beginning),Set(counter=0)

same = n,NoOp(variable "counter" is incremented and the label "start" is defined)
same = n(start),Set(counter=$[${counter} + 1])

same = n,NoOp(counter variable is now = ${counter})
same = n,NoOp(waiting for 1 second before reading the message that indicate all␣
↪choices)
same = n,Wait(1)
same = n,NoOp(play the message ivr-example-choices that contain all choices)
same = n,Background(${GV_DIRECTORY_SOUNDS}/ivr-example-choices)
same = n,NoOp(waiting for DTMF during 5s)
same = n,Waitexten(5)

;##### CHOICE 1 #####
exten = 1,1,NoOp(pressed digit is 1, redirect to 8000 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 1,n,Goto(${IVR_DESTINATION_CONTEXT},8000,1)

;##### CHOICE 2 #####
exten = 2,1,NoOp(pressed digit is 2, redirect to 8833 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 2,n,Goto(${IVR_DESTINATION_CONTEXT},8833,1)

;##### CHOICE 3 #####
exten = 3,1,NoOp(pressed digit is 3, redirect to the submenu dp-ivr-submenu)
exten = 3,n,Goto(dp-ivr-submenu,s,1)


;##### CHOICE 4 #####
exten = 4,1,NoOp(pressed digit is 4, redirect to start label in this context)
exten = 4,n,Goto(s,start)

;##### TIMEOUT #####
exten = t,1,NoOp(no digit pressed for 5s, process it like an error)
exten = t,n,Goto(i,1)

;##### INVALID CHOICE #####
exten = i,1,NoOp(if counter variable is 3 or more, then goto label "error")
exten = i,n,GotoIf($[${counter}>=3]?error)
```

(continues on next page)


Start
Closed
Playback : ivr-example-schedule-closed
Open
Playback : ivr-example-welcome-sound
Playback : ivr-example-invalid-choice
Background : ivr-example-choices
Yes
Counter < 3
Invalid extension or timeout
waitexten (during 5s) and
No
Playback : ivr-example-error
Valid choice
1. Callto 8000
2. Callto 8833
3. Sub menu with dp-ivr-submenu
4. Repeat the menu
8000
8833
End
Playback : ivr-example-invalid-choice
Background : ivr-example-submenu-choice
Yes
Counter < 3
Invalid extension or timeout
waitexten (during 5s) and
No
Playback : ivr-example-error
Valid choice
1. Callto 8001
2. Callto 8002
3. Return to the previous menu
End

(continued from previous page)

```
exten = i,n,NoOp(pressed digit is invalid and less than 3 errors: the guide ivr-
↪exemple-invalid-choice is now played)
exten = i,n,Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-invalid-choice)
exten = i,n,Goto(s,start)
exten = i,n(error),Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-error)
exten = i,n,Hangup()


[dp-ivr-submenu]

exten = s,1,NoOp(### dp-ivr-submenu ###)
same = n,NoOp(the system answers the call and waits for 1 second before continuing)
same = n,Answer(1000)

same = n,NoOp(variable "counter" is set to 0)
same = n(beginning),Set(counter=0)

same = n,NoOp(variable "counter" is incremented and the label "start" is defined)
same = n(start),Set(counter=$[${counter} + 1])

same = n,NoOp(counter variable is now = ${counter})
same = n,NoOp(waiting for 1 second before reading the message that indicate all␣
↪choices)
same = n,Wait(1)
same = n,NoOp(play the message ivr-example-choices that contain all choices)
same = n,Background(${GV_DIRECTORY_SOUNDS}/ivr-example-submenu-choices)
same = n,NoOp(waiting for DTMF during 5s)
same = n,Waitexten(5)

;##### CHOICE 1 #####
exten = 1,1,NoOp(pressed digit is 1, redirect to 8000 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 1,n,Goto(${IVR_DESTINATION_CONTEXT},8000,1)

;##### CHOICE 2 #####
exten = 2,1,NoOp(pressed digit is 2, redirect to 8001 in ${IVR_DESTINATION_CONTEXT}
↪ context)
exten = 2,n,Goto(${IVR_DESTINATION_CONTEXT},8001,1)

;##### CHOICE 3 #####
exten = 3,1,NoOp(pressed digit is 3, redirect to the previous menu dp-ivr-example)
exten = 3,n,Goto(dp-ivr-example,s,beginning)


;##### TIMEOUT #####
exten = t,1,NoOp(no digit pressed for 5s, process it like an error)
exten = t,n,Goto(i,1)


;##### INVALID CHOICE #####
exten = i,1,NoOp(if counter variable is 3 or more, then goto label "error")
exten = i,n,GotoIf($[${counter}>=3]?error)
exten = i,n,NoOp(pressed digit is invalid and less than 3 errors: the guide ivr-
↪exemple-invalid-choice is now played)
exten = i,n,Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-invalid-choice)
exten = i,n,Goto(s,start)
exten = i,n(error),Playback(${GV_DIRECTORY_SOUNDS}/ivr-example-error)
exten = i,n,Hangup()
```

5.20 Monitoring

The Monitoring section gives an overview of a XiVO system's status and of all monitored processes. It is divided into 6 sections :

- *System*
- *Device*
- *CPU*
- *Network*
- *Memory*
- *Other Services*

5.20.1 System

Displays generic information about the operating system, network addresses, uptime and load average. Read only.

5.20.2 Device

Displays free/used space on physical storage partitions. Read only.

5.20.3 CPU

Monitors the CPU usage. Read only.

5.20.4 Network

Displays network interfaces and corresponding network traffic. Read only.

5.20.5 Memory

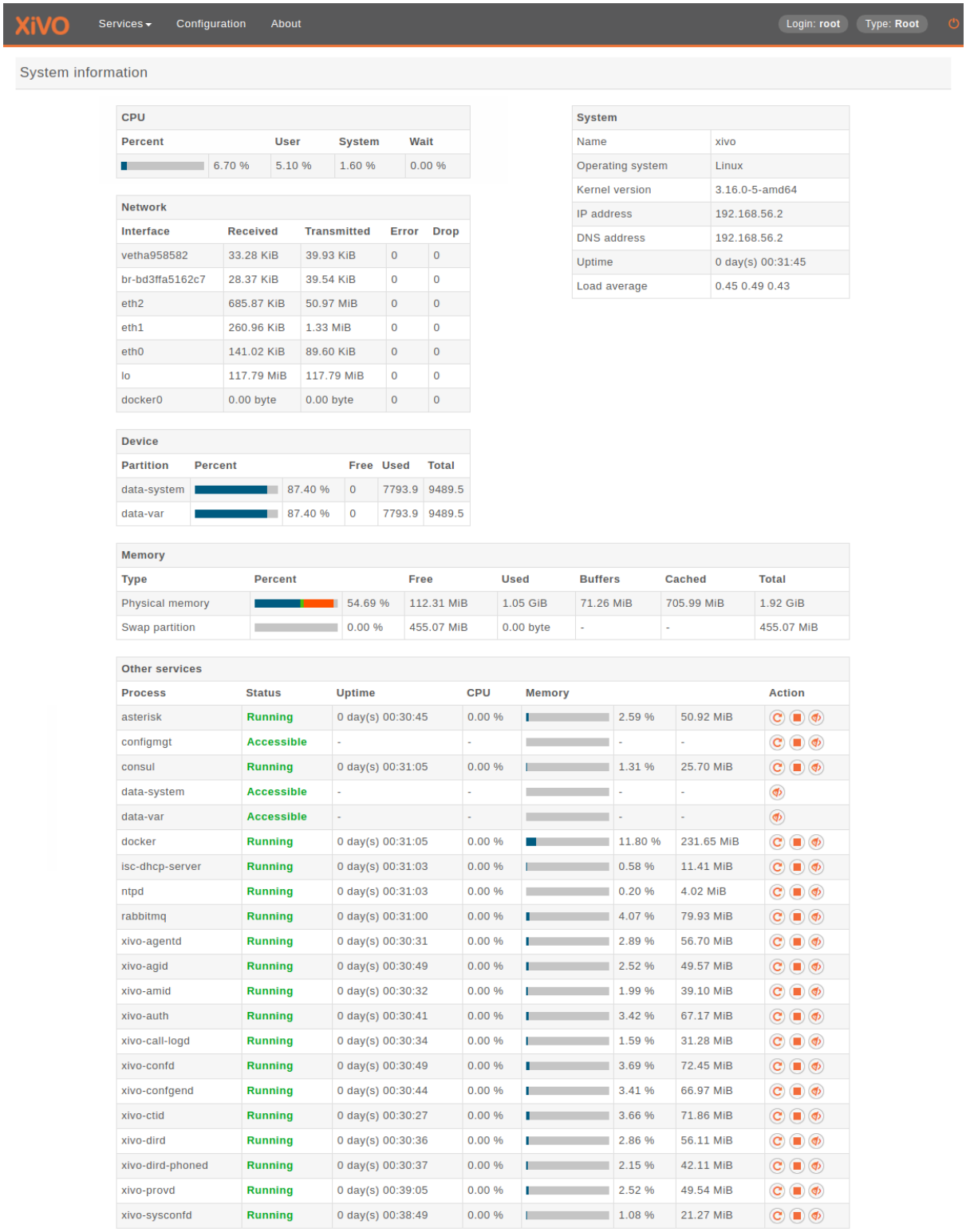
Displays Physical and swap memory usage. Read only.

5.20.6 Other Services

Lists XiVO related processes (most of which are daemons) with their corresponding status, uptime, resource usage and controls to restart service, stop service and stop monitoring service.

5.21 Music on Hold

The menu *Services* → *IPBX* → *IPBX services* → *On-hold Music* leads to the list of available on-hold musics.


XiVO
Services
Configuration
About
Login: root
Type: Root
System information
CPU
Percent
User
System
Wait
6.70 %
5.10 %
1.60 %
0.00 %
Network
Interface
Received
Transmitted
Error
Drop
vetha958582 33.28 KiB 39.93 KiB 0 0
br-bd3ffa5162c7 28.37 KiB 39.54 KiB 0 0
eth2 685.87 KiB 50.97 MiB 0 0
eth1 260.96 KiB 1.33 MiB 0 0
eth0 141.02 KiB 89.60 KiB 0 0
lo 117.79 MiB 117.79 MiB 0 0
docker0 0.00 byte 0.00 byte 0 0
System
Name xivo
Operating system Linux
Kernel version 3.16.0-5-amd64
IP address 192.168.56.2
DNS address 192.168.56.2
Uptime 0 day(s) 00:31:45
Load average 0.45 0.49 0.43
Device
Partition
Percent
Free
Used
Total
data-system 87.40 % 0 7793.9 9489.5
data-var 87.40 % 0 7793.9 9489.5
Memory
Type
Percent
Free
Used
Buffers
Cached
Total
Physical memory 54.69 % 112.31 MiB 1.05 GiB 71.26 MiB 705.99 MiB 1.92 GiB
Swap partition 0.00 % 455.07 MiB 0.00 byte - - 455.07 MiB
Other services
Process
Status
Uptime
CPU
Memory
Action
asterisk Running 0 day(s) 00:30:45 0.00 % 2.59 % 50.92 MiB
configmgt Accessible - - - -
consul Running 0 day(s) 00:31:05 0.00 % 1.31 % 25.70 MiB
data-system Accessible - - - -
data-var Accessible - - - -
docker Running 0 day(s) 00:31:05 0.00 % 11.80 % 231.65 MiB
isc-dhcp-server Running 0 day(s) 00:31:03 0.00 % 0.58 % 11.41 MiB
ntpd Running 0 day(s) 00:31:03 0.00 % 0.20 % 4.02 MiB
rabbitmq Running 0 day(s) 00:31:00 0.00 % 4.07 % 79.93 MiB
xivo-agentd Running 0 day(s) 00:30:31 0.00 % 2.89 % 56.70 MiB
xivo-agid Running 0 day(s) 00:30:49 0.00 % 2.52 % 49.57 MiB
xivo-amid Running 0 day(s) 00:30:32 0.00 % 1.99 % 39.10 MiB
xivo-auth Running 0 day(s) 00:30:41 0.00 % 3.42 % 67.17 MiB
xivo-call-logd Running 0 day(s) 00:30:34 0.00 % 1.59 % 31.28 MiB
xivo-confd Running 0 day(s) 00:30:49 0.00 % 3.69 % 72.45 MiB
xivo-confgend Running 0 day(s) 00:30:44 0.00 % 3.41 % 66.97 MiB
xivo-ctid Running 0 day(s) 00:30:27 0.00 % 3.66 % 71.86 MiB
xivo-dird Running 0 day(s) 00:30:36 0.00 % 2.86 % 56.11 MiB
xivo-dird-phoned Running 0 day(s) 00:30:37 0.00 % 2.15 % 42.11 MiB
xivo-provd Running 0 day(s) 00:39:05 0.00 % 2.52 % 49.54 MiB
xivo-sysconfd Running 0 day(s) 00:38:49 0.00 % 1.08 % 21.27 MiB

5.21.1 Categories

Available categories are:

- files: play sound files. Formats supported:

Format Name	Filename Extension
G.719	.g719
G.723	.g723 .g723sf
G.726	.g726-40 .g726-32 .g726-24 .g726-16
G.729	.g729
GSM	.gsm
iLBC	.ilbc
Ogg Vorbis	.ogg (only mono files sampled at 8000 Hz)
G.711 A-law	.alaw .al .alw
G.711 μ-law	.pcm .ulaw .ul .mu .ulw
G.722	.g722
Au	.au
Siren7	.siren7
Siren14	.siren14
SLN	.raw .sln .sln12 .sln16 .sln24 .sln32 .sln44 .sln48 .sln96 .sln192
VOX	.vox
WAV	.wav .wav16
WAV GSM	.WAV .wav49

  Only 1 audio channel must be present per file, i.e. files must be in mono.

  If your music on hold files don't seem to work, you should look for errors in the asterisk logs.

  The on-hold music will always play from the start.

- mp3: play MP3 files.

  The on-hold music will play from an arbitrary position on the track, it will not play from the start.

- custom: do not play sound files. Instead, run an external process. That process must send on stdout the same binary format than WAV files.

  Example process: `/usr/bin/mpg123 -s --mono -y -f 8192 -r 8000 http://streaming.example.com/stream.mp3`

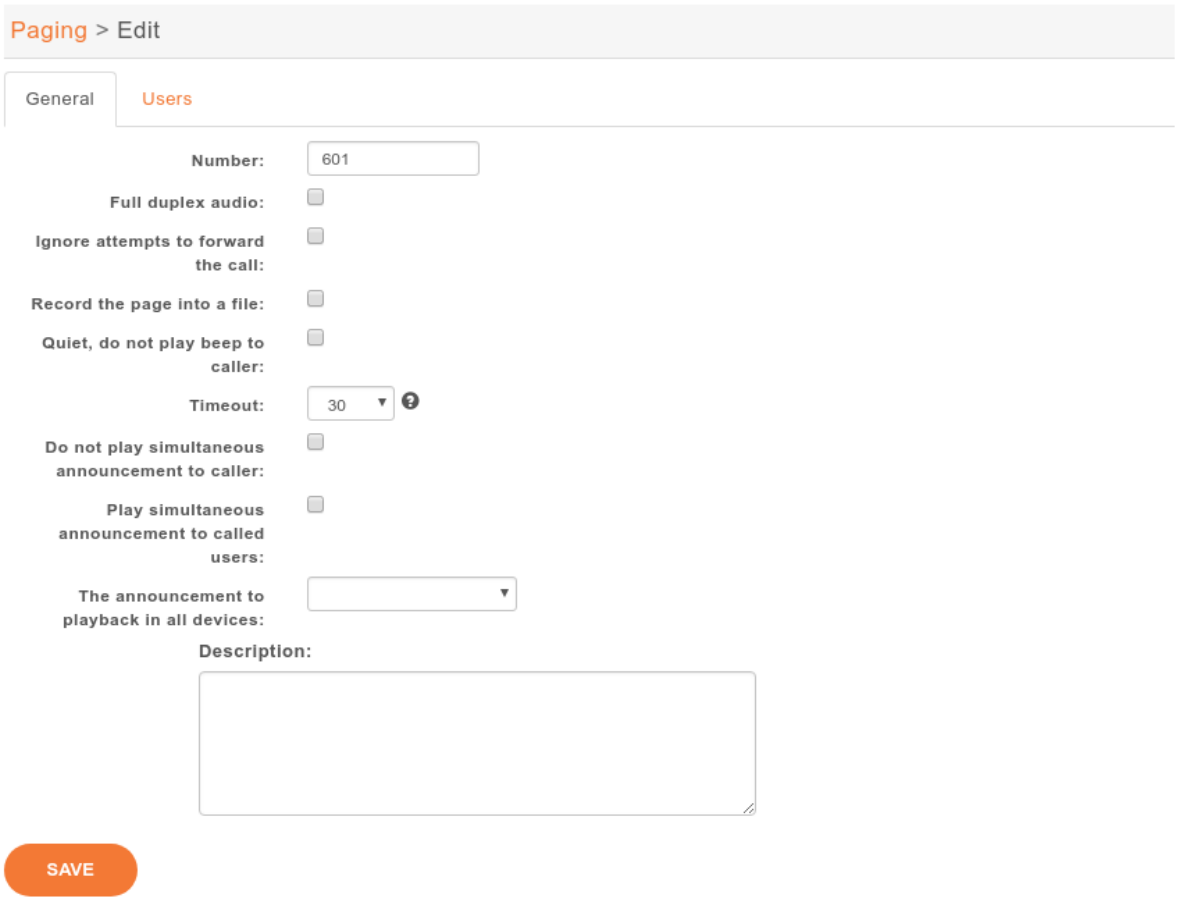
Note: Processes run by custom categories are started as soon as the category is created and will only stop when the category is deleted. This means that on-hold music fed from online streaming will constantly be receiving network traffic, even when there are no calls.

5.22 Paging

With XiVO, you can define paging (i.e. intercom) extensions to page a group of users. When calling a paging extension, the phones of the specified users will auto-answer, if they support it.

You can manage your paging extensions via the *Services* → *IPBX* → *Paging* page.

When adding a new paging extension, the number can be any numeric value; to call it, you just need to prefix the paging number with `*11`.



5.23 Parking

With XiVO it is possible to park calls, the same way you may park your car in a car parking. If you define supervised keys on a phone set for all the users of a system, when a call is parked, all the users are able to see that some one is waiting for an answer, push the phone key and get the call back to the phone.

There is a default parking number, 700, which is already configured when you install XiVO, but you may change the default configuration by editing the parking extension in menu *Service* → *IPBX* → *IPBX Services* → *Extensions* → *Parking*

Using this extension, you may define the parking number used to park call, the parking lots, wether the sytem is rotating over the parking lots to park the calls, enable parking hint if you want to be able to supervise the parking using phone keys and other system default parameters.

You have two options in case of parking timeout :

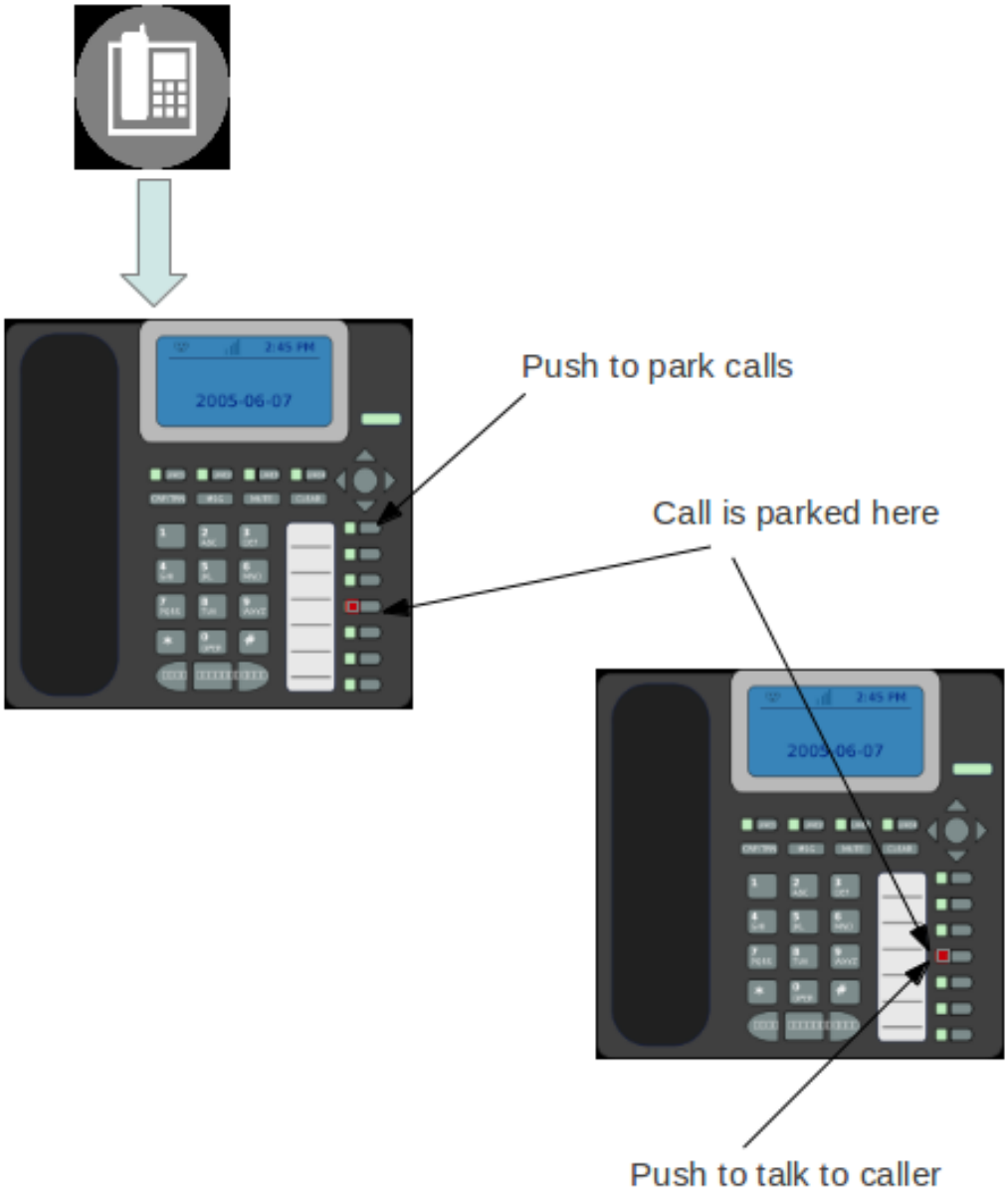
- Callback the peer that parked this call

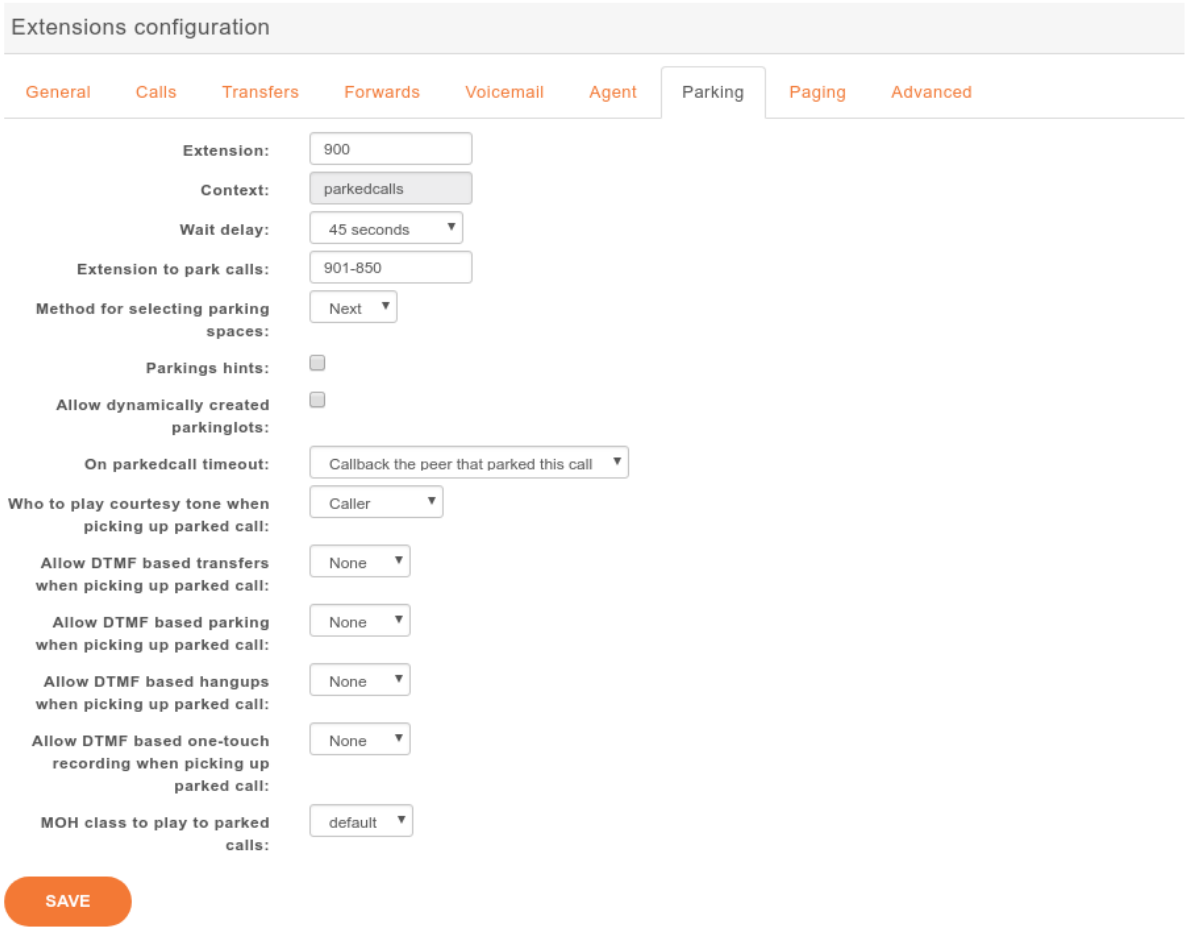
  In this case the call is sent back to the user who parked the call.

- Send park call to the dialplan

  In case you don't want to call back the user who parked the call, you have the option to send the call to any other extension or application. If the parking times out, the call is sent back to the dialplan in context `[parkedcallstimeout]`. You can define this context in a dialplan configuration file *Service* → *IPBX* → *Configuration Files* where you may define this context with dialplan commands.

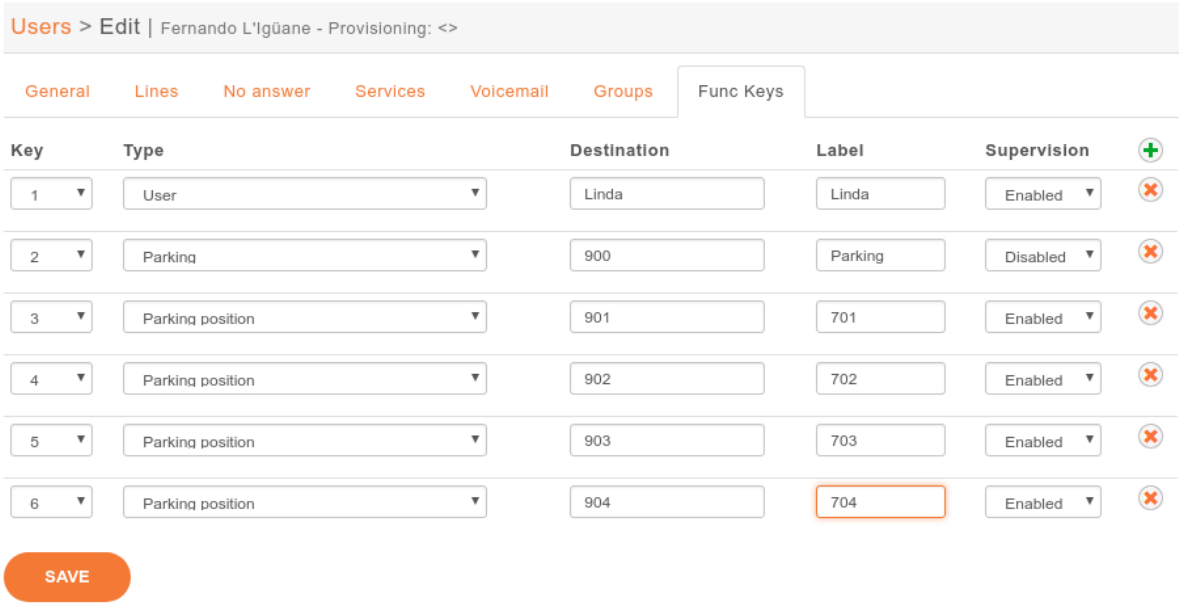
  Example:


Push to park calls
Call is parked here
Push to talk to caller


Extensions configuration
General
Calls
Transfers
Forwards
Voicemail
Agent
Parking
Paging
Advanced
Extension:
900
Context:
parkedcalls
Wait delay:
45 seconds
Extension to park calls:
901-850
Method for selecting parking spaces:
Next
Parkings hints:
Allow dynamically created parkinglots:
On parkedcall timeout:
Callback the peer that parked this call
Who to play courtesy tone when picking up parked call:
Caller
Allow DTMF based transfers when picking up parked call:
None
Allow DTMF based parking when picking up parked call:
None
Allow DTMF based hangups when picking up parked call:
None
Allow DTMF based one-touch recording when picking up parked call:
None
MOH class to play to parked calls:
default
SAVE

```
[parkedcallstimeout]
exten = s,1,Noop('park call time out')
same  =   n,Playback(hello-world)
same  =   n,Hangup()
```

It is also usual to define supervised phone keys to be able to park and unpark calls as in the example below.



5.24 Phonebook

A global phone book can be defined in *Services* → *IPBX* → *IPBX Services* → *Phonebook*. The phone book can be used from the XiVO Client, from the phones directory look key if the phone is compatible and are used to set the Caller ID for incoming calls.

You can add entries one by one or you can mass-import from a CSV file.

Note: To configure phonebook, see *Directories*.

5.24.1 Mass-import contacts

Go in the *Services* → *IPBX* → *IPBX Services* → *Phonebook* section and move your mouse cursor on the + button in the upper right corner. Select *Import a file*.

The file to be imported must be a CSV file, with a pipe character | as field delimiter. The file must be encoded in UTF-8 (without an initial BOM).

Mandatory headers are :

- title (possible values : “mr”, “mrs”, “ms”)
- displayname

Optional headers are :

- firstname
- lastname

- society
- mobilenumber[1]
- email
- url
- description
- officenumber[1]
- faxnumber[1]
- officeaddress1
- officeaddress2
- officecity
- officestate
- officezipcode
- officecountry[2]
- homenumber[1]
- homeaddress1
- homeaddress2
- homecity
- homestate
- homezipcode
- homecountry[2]
- othernumber[1]
- otheraddress1
- otheraddress2
- othercity
- otherstate
- otherzipcode
- othercountry[2]

5.25 Provisioning

XiVO supports the auto-provisioning of a large number of telephony devices, including SIP phones, SIP ATAs, and even softphones.

5.25.1 Introduction

The auto-provisioning feature found in XiVO make it possible to provision, i.e. configure, a lots of telephony devices in an efficient and effortless way.

[1] These fields must contain only numeric characters, no space, point, etc.
[2] These fields must contain ISO country codes. The complete list is described here.

How it works

Here's a simplified view of how auto-provisioning is supported on a typical SIP hardphone:

1. The phone is powered on
2. During its boot process, the phone sends a DHCP request to obtain its network configuration
3. A DHCP server replies with the phone network configuration + an HTTP URL
4. The phone use the provided URL to retrieve a common configuration file, a MAC-specific configuration file, a firmware image and some language files.

Building on this, configuring one of the supported phone on XiVO is as simple as:

1. *Configuring the DHCP Server*
2. *Installing the required provd plugin*
3. Powering on the phone
4. Dialing the user's provisioning code from the phone

And *voila*, once the phone has rebooted, your user is ready to make and receive calls. No manual editing of configuration files nor fiddling in the phone's web interface.

Limitations

- Device synchronisation does not work in the situation where multiple devices are connected from behind a NAPT network equipment. The devices must be resynchronised manually.

External links

- Introduction to provd plugin model
- HTTP/TFTP requests processing in provd - part 1
- HTTP/TFTP requests processing in provd - part 2

5.25.2 Basic Configuration

You have two options to get your phone to be provisioned:

- Set up a DHCP server
- Tell manually each phone where to get the provisioning informations

You may want to manually configure the phones if you are only trying XiVO or if your network configuration does not allow the phones to access the XiVO DHCP server.

You may want to set up a DHCP server if you have a significant number of phones to connect, as no manual intervention will be required on each phone.

Configuring the DHCP Server

XiVO includes a DHCP server that facilitate the auto-provisioning of telephony devices. It is *not* activated by default.

There's a few things to know about the peculiarities of the included DHCP server:

- it only answers to DHCP requests from supported devices.
- it only answers to DHCP requests coming from the VoIP subnet (see *network configuration*).

This means that if your phones are on the same broadcast domain than your computers, and you would like the DHCP server on your XiVO to handle both your phones and your computers, that won't do it.

The DHCP server is configured via the *Configuration* → *Network* → *DHCP* page:

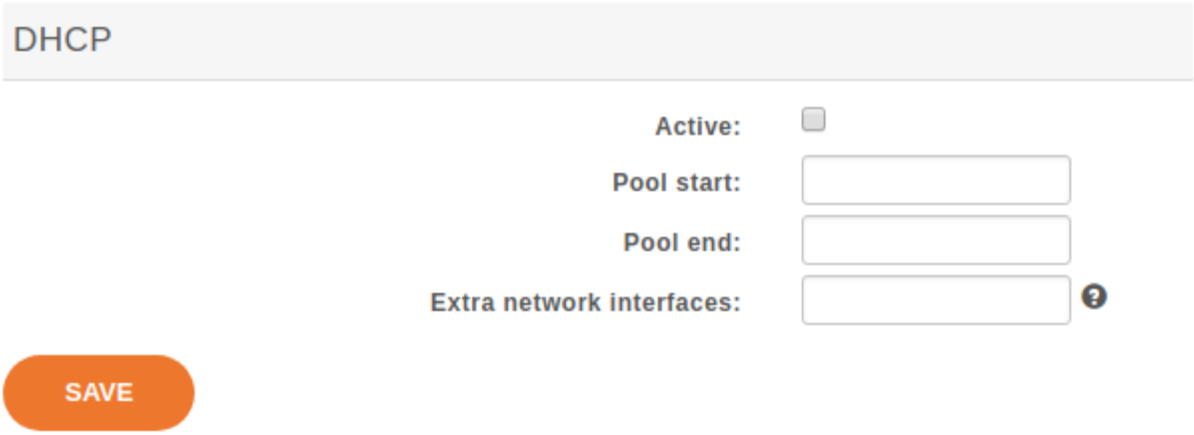


Fig. 33: *Configuration* → *Network* → *DHCP*

Active Activate/desactivate the DHCP server.

Pool start The lower IP address which will be assigned dynamically. This address should be in the VoIP subnet. Example: `10.0.0.10`.

Pool end The higher IP address which will be assigned dynamically. This address should be in the VoIP subnet. Example: `10.0.0.99`.

Extra network interfaces A list of space-separated network interface name. Example: `eth0`.

Useful if you have done some custom configuration in the `/etc/dhcp/dhcpd_extra.conf` file. You need to explicitly specify the additional interfaces the DHCP server should listen on.

After saving your modifications, you need to click on *Apply system configuration* for them to be applied.

Installing `provd` Plugins

The installation and management of `provd` plugins is done via the *Configuration* → *Provisioning* → *Plugin* page.

The page shows the list of both the installed and installable plugins. You can see if a plugin is installed or not by looking at the *Action* column.

Here's the list of other things that can be done from this page:

- update the list of installable plugins, by clicking on the top left icon (1). On a fresh XiVO installation, this is the first thing to do.
- install a new plugin (2)
- edit an installed plugin (3), i.e. install/uninstall optional files that are specific to each plugin, like firmware or language files
- upgrade an installed plugin (4)
- uninstall an installed plugin (5)

After installing a new plugin, you are automatically redirected to its edit page. You can then download and install optional files specific to the plugin. You are strongly advised to install firmware and language files for the phones you'll use although it's often not a strict requirement for the phones to work correctly.

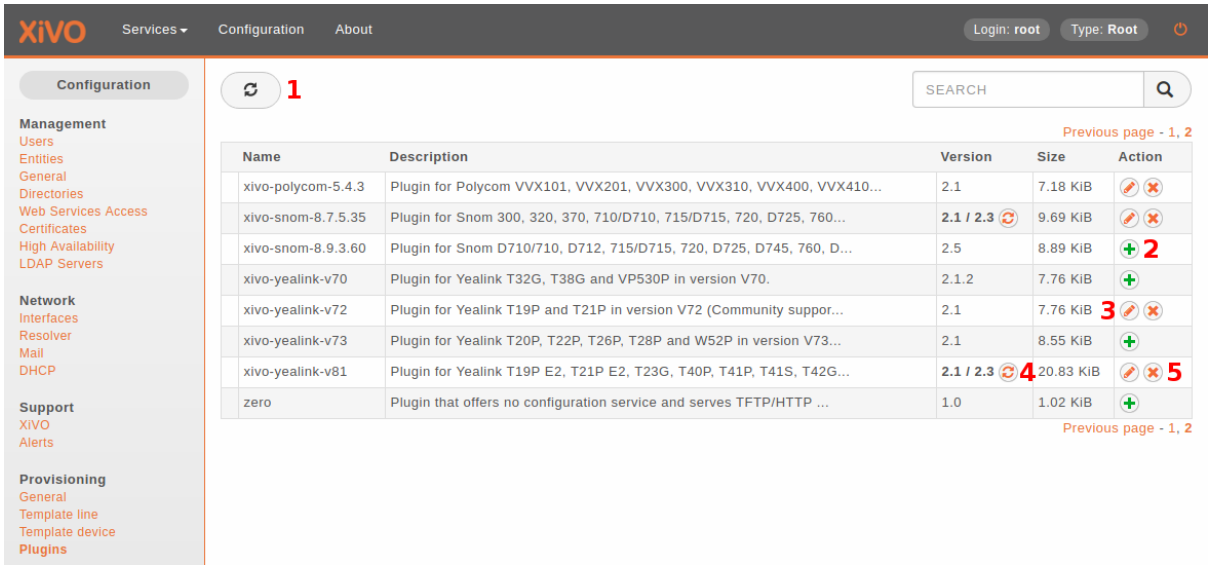


Fig. 34: *Configuration → Provisioning → Plugin*

Warning: If you uninstall a plugin that is used by some of your devices, they will be left in an unconfigured state and won't be associated to another plugin automatically.

The search box at the top comes in handy when you want to find which plugin to install for your device. For example, if you have a Cisco SPA508G, enter "508" in the search box and you should see there's 1 plugin compatible with it.

Note: If your device has a number in its model name, you should use only the number as the search keyword since this is what usually gives the best results.

It's possible there will be more than 1 plugin compatible with a given device. In these cases, the difference between the two plugins is usually just the firmware version the plugins target. If you are unsure about which version you should install, you should look for more information on the vendor website.

It's good practice to only install the plugins you need and no more.

Alternative plugins repository

By default, the list of plugins available for installation are the stable plugins for the officially supported devices.

This can be changed in the *Configuration → Provisioning → General* page, by setting the *URL* field to one of the following value:

- `http://provd.xivo.solutions/plugins/1/stable/` – officially supported devices "stable" repository (*default*)
- `http://provd.xivo.solutions/plugins/1/testing/` – officially supported devices "testing" repository
- `http://provd.xivo.solutions/plugins/1/archive/` – officially supported devices "archive" repository
- `http://provd.xivo.solutions/plugins/1/addons/stable/` – community supported devices "stable" repository
- `http://provd.xivo.solutions/plugins/1/addons/testing/` – community supported devices "testing" repository

The difference between the stable and testing repositories is that the latter might contain plugins that are not working properly or are still in developement.

The archive repository contains plugins that were once in the stable repository.

After setting a new URL, you must refresh the list of installable plugins by clicking the update icon of the *Configuration → Provisioning → Plugin* page.

How to manually tell the phones to get their configuration

If you have set up a DHCP server on XiVO and the phones can access it, you can skip this section.

The according provisioning plugins must be installed.

Aastra

On the web interface of your phone, go to *Advanced settings → Configuration server*, and enter the following settings:

Polycom

On the phone, go to *Menu → Settings → Advanced → Admin Settings → Network configuration → Server Menu* and enter the following settings:

- Server type: HTTP
- Server address: `http://<XiVO IP address>:8667/000000000000.cfg`

Then save and reboot the phone.

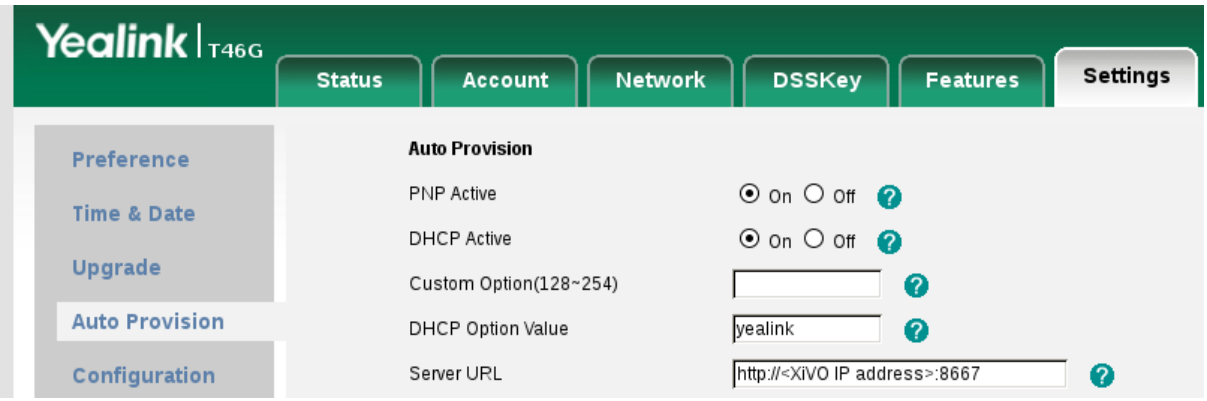
Snom

On the web interface of your phone, go to *Setup* → *Advanced* → *Update* and enter the following settings:

Yealink

On the web interface of your phone, go to *Settings* → *Auto Provision*, and enter the following settings:

- Server URL: `http://<XiVO IP address>:8667`



Save the changes by clicking on the *Confirm* button and then click on the *Autoprovision Now* button.

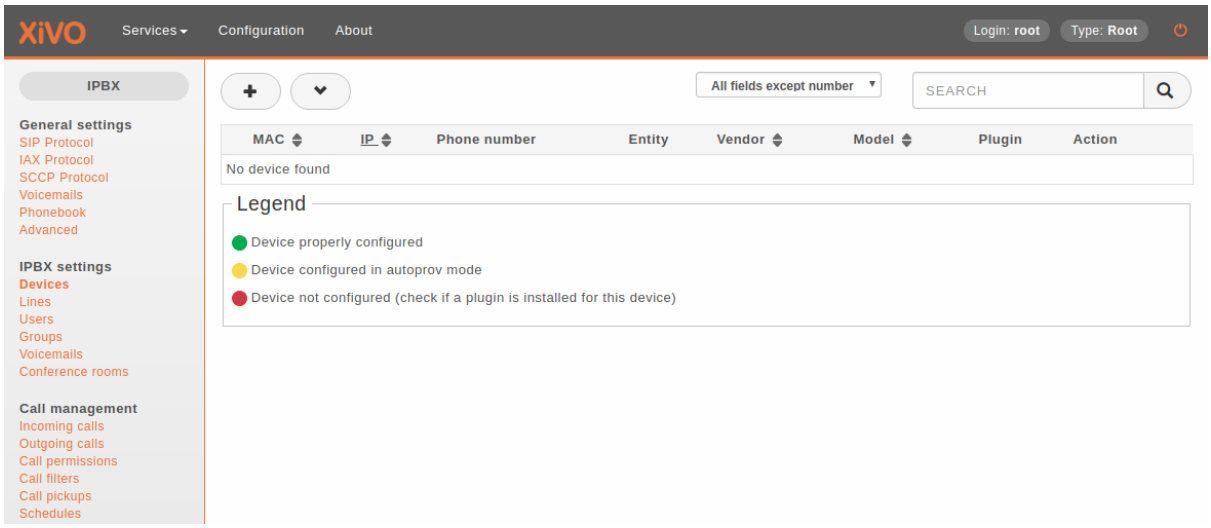
Autoprovisioning a Device

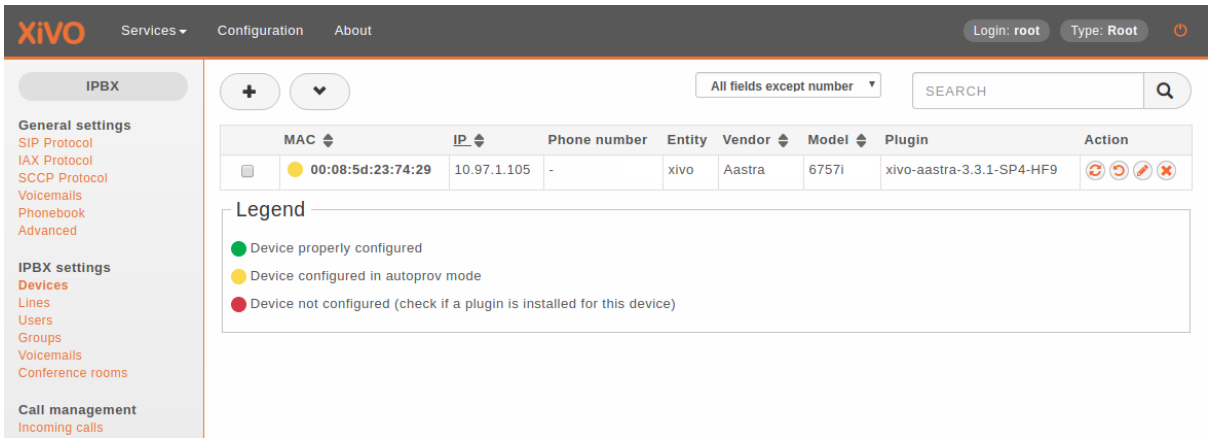
Once you have installed the proper provd plugins for your devices and setup correctly your DHCP server, you can then connect your devices to your network.

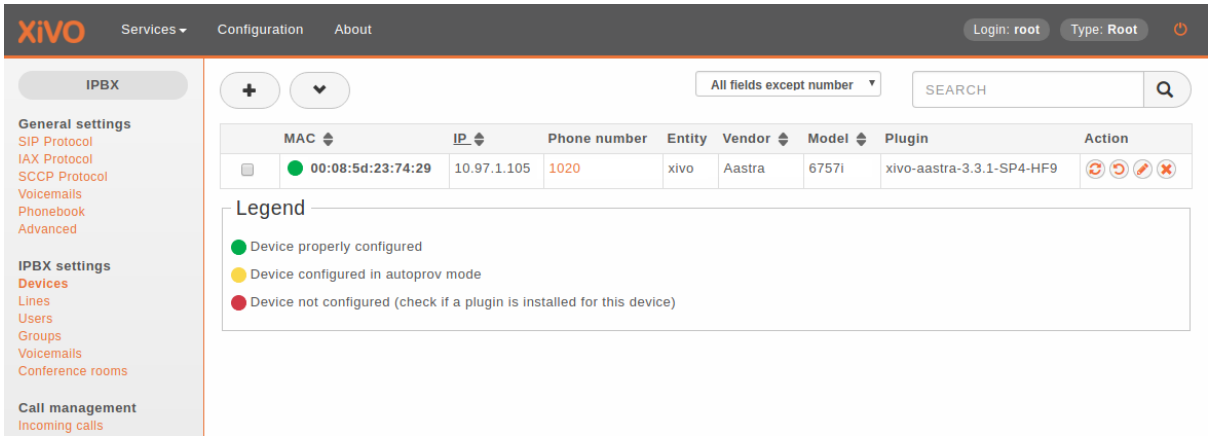
But first, go to *Services* → *IPBX* → *Devices* page. You will then see that no devices are currently known by your XiVO:

You can then power on your devices on your LAN. For example, after you power on an Aastra 6757i and give it the time to boot and maybe upgrade its firmware, you should then see the phone having its first line configured as 'autoprov', and if you refresh the devices page, you should see that your XiVO now knows about your 6757i:

You can then dial from your Aastra 6757i the provisioning code associated to a line of one of your user. You will hear a prompt thanking you and your device should then reboot in the next few seconds. Once the device has rebooted, it will then be properly configured for your user to use it. And also, if you update the device page, you'll see that the icon next to your device has now passed to green:


XiVO
Services
Configuration
About
Login: root
Type: Root
IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
Outgoing calls
Call permissions
Call filters
Call pickups
Schedules
All fields except number
SEARCH
MAC
IP
Phone number
Entity
Vendor
Model
Plugin
Action
No device found
Legend
Device properly configured
Device configured in autoprov mode
Device not configured (check if a plugin is installed for this device)


XiVO
Services
Configuration
About
Login: root
Type: Root
IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
All fields except number
SEARCH
MAC
IP
Phone number
Entity
Vendor
Model
Plugin
Action
00:08:5d:23:74:29
10.97.1.105
-
xivo
Aastra
6757i
xivo-aastra-3.3.1-SP4-HF9
Legend
Device properly configured
Device configured in autoprov mode
Device not configured (check if a plugin is installed for this device)


XiVO
Services
Configuration
About
Login: root
Type: Root
IPBX
General settings
SIP Protocol
IAX Protocol
SCCP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
Voicemails
Conference rooms
Call management
Incoming calls
All fields except number
SEARCH
MAC
IP
Phone number
Entity
Vendor
Model
Plugin
Action
00:08:5d:23:74:29
10.97.1.105
1020
xivo
Aastra
6757i
xivo-aastra-3.3.1-SP4-HF9
Legend
Device properly configured
Device configured in autoprov mode
Device not configured (check if a plugin is installed for this device)

Resetting a Device

From the Device List in the Webi

To remove a phone from XiVO or enable a device to be used for another user there are two different possibilities :

- click on the `reset to autoprov` button on the web interface

The phone will restarts and display autoprov, ready to be used for another user.

From the User Form in the Webi

Device With one User Only Associated

Edit the user associated to the device and put the device field to null.

- click on the `Save` button on the web interface

The phone doesn't restart and the phone is in autoprov mode in the device list.

You can synchronize the device to reboot it.

Device with Several Users Associated

Edit the primary user associated to the terminal (one with the line 1) and put the device field to null.

- click on the `Save` button on the web interface

The primary line of the phone has been removed, so the device will lose its funckeys associated to primary user but there others lines associated to the device will stay provisionned.

The phone doesn't restart and the phone is in autoprov mode in the device list.

You can synchronize the device for reboot it.

From a Device

- Dial ***guest** (*48378) on the phone dialpad followed by **xivo** (9486) as a password

The phone restarts and display autoprov, ready to be used for another user.

5.25.3 Advanced Configuration

DHCP Integration

If your phones are getting their network configuration from your XiVO's DHCP server, it's possible to activate the DHCP integration on the *Configuration* → *Provisioning* → *General* page.

What DHCP integration does is that, on every DHCP request made by one of your phones, the DHCP server sends information about the request to `provd`, which can then use this information to update its device database.

This feature is useful for phones which lack information in their TFTP/HTTP requests. For example, without DHCP integration, it's impossible to extract model information for phones from the Cisco 7900 series. Without the model information extracted, there's chance your device won't be automatically associated to the best plugin.

This feature can also be useful if your phones are not always getting the same IP addresses, for one reason or another. Again, this is useful only for some phones, like the Cisco 7900; it has no effect for Aastra 6700.

Creating Custom Templates

Custom templates comes in handy when you have some really specific configuration to make on your telephony devices.

Templates are handled on a per plugin basis. It's not possible for a template to be shared by more than one plugin since it's a design limitation of the plugin system of `provd`.

Note: When you install a new plugin, templates are not migrated automatically, so you must manually copy them from the old plugin directory to the new one. This does not apply for a plugin upgrade.

Let's suppose we have installed the `xivo-aastra-3.3.1-SP2` plugin and want to write some custom templates for it.

First thing to do is to go into the directory where the plugin is installed:

```
cd /var/lib/xivo-provd/plugins/xivo-aastra-3.3.1-SP2
```

Once you are there, you can see there's quite a few files and directories:

```
tree
.
+-- common.py
+-- entry.py
+-- pkgs
|   +-- pkgs.db
+-- plugin-info
+-- README
+-- templates
|   +-- 6730i.tpl
|   +-- 6731i.tpl
|   +-- 6739i.tpl
|   +-- 6753i.tpl
|   +-- 6755i.tpl
|   +-- 6757i.tpl
|   +-- 9143i.tpl
|   +-- 9480i.tpl
|   +-- base.tpl
+-- var
    +-- cache
    +-- installed
    +-- templates
    +-- tftpboot
        +-- Aastra
            +-- aastra.cfg
```

The interesting directories are:

templates This is where the original templates lies. You *should not* edit these files directly but instead copy the one you want to modify in the var/templates directory.

var/templates This is the directory where you put and edit your custom templates.

var/tftpboot This is where the configuration files lies once they have been generated from the templates. You should look at them to confirm that your custom templates are giving you the result you are expecting.

Warning: When you uninstall a plugin, the plugin directory is removed altogether, including all the custom templates.

A few things to know before writing your first custom template:

- templates use the Jinja2 template engine.
- when doing an `include` or an `extend` from a template, the file is first looked up in the `var/templates` directory and then in the `templates` directory.
- device in autoprov mode are affected by templates, because from the point of view of `provd`, there's no difference between a device in autoprov mode or fully configured. This means there's usually no need to modify static files in `var/tftpboot`. And this is a bad idea since a plugin upgrade will override these files.

Custom template for every devices

```
cp templates/base.tpl var/templates
vi var/templates/base.tpl
xivo-provd-cli -c 'devices.using_plugin("xivo-aastra-3.3.1-SP2").reconfigure()'
```

Once this is done, if you want to synchronize all the affected devices, use the following command:

```
xivo-provd-cli -c 'devices.using_plugin("xivo-aastra-3.3.1-SP2").synchronize()'
```

Custom template for a specific model

Let's supose we want to customize the template for our 6739i:

```
cp templates/6739i.tpl var/templates
vi var/templates/6739i.tpl
xivo-provd-cli -c 'devices.using_plugin("xivo-aastra-3.3.1-SP2").reconfigure()'
```

Custom template for a specific device

To create a custom template for a specific device you have to create a device-specific template named `<device_specific_file_with_extension>.tpl` in the `var/templates/` directory :

- for an Aastra phone, if you want to customize the file `00085D2EECFB.cfg` you will have to create a template file named `00085D2EECFB.cfg.tpl`,
- for a Snom phone, if you want to customize the file `000413470411.xml` you will have to create a template file named `000413470411.xml.tpl`,
- for a Polycom phone, if you want to customize the file `0004f2211c8b-user.cfg` you will have to create a template file named `0004f2211c8b-user.cfg.tpl`,
- and so on.

Here, we want to customize the content of a device-specific file named `00085D2EECFB.cfg`, we need to create a template named `00085D2EECFB.cfg.tpl`:

```
cp templates/6739i.tpl var/templates/00085D2EECFB.cfg.tpl
vi var/templates/00085D2EECFB.cfg.tpl
xivo-provd-cli -c 'devices.using_mac("00085D2EECFB").reconfigure()'
```

Note: The choice to use this syntax comes from the fact that `provd` supports devices that do not have MAC addresses, namely softphones.

Also, some devices have more than one file (like Snom), so this way make it possible to customize more than 1 file.

The template to use as the base for a device specific template will vary depending on the need. Typically, the model template will be a good choice, but it might not always be the case.

Changing the Plugin Used by a Device

From time to time, new firmwares are released by the devices manufacturer. This sometimes translate to a new plugin being available for these devices.

When this happens, it almost always means the new plugin obsoletes the older one. The older plugin is then considered "end-of-life", and won't receive any new updates nor be available for new installation.

Let's suppose we have the old `xivo-aastra-3.2.2.1136` plugin installed on our xivo and want to use the newer `xivo-aastra-3.3.1-SP2` plugin.

Both these plugins can be installed at the same time, and you can manually change the plugin used by a phone by editing it via the *Services* → *IPBX* → *Devices* page.

If you are using custom templates in your old plugin, you should copy them to the new plugin and make sure that they are still compatible.

Once you take the decision to migrate all your phones to the new plugin, you can use the following command:

```
xivo-provd-cli -c 'helpers.mass_update_devices_plugin("xivo-aastra-3.2.2.1136",
↪"xivo-aastra-3.3.1-SP2")'
```

Or, if you also want to synchronize (i.e. reboot) them at the same time:

```
xivo-provd-cli -c 'helpers.mass_update_devices_plugin("xivo-aastra-3.2.2.1136",
↪"xivo-aastra-3.3.1-SP2", synchronize=True)'
```

You can check that all went well by looking at the *Services* → *IPBX* → *Devices* page.

NAT

The provisioning server has partial support for environment where the telephony devices are behind a NAT equipment.

By default, each time the provisioning server receives an HTTP/TFTP request from a device, it makes sure that only one device has the source IP address of the request. This is not a desirable behaviour when the provisioning server is used in a NAT environment, since in this case, it's normal that more than 1 devices have the same source IP address (from the point of view of the server).

If *all* your devices used on your XiVO are behind a NAT, you should disable this behaviour by setting the `NAT` option to 1 via the *Configuration* → *Provisioning* → *General* page.

Enabling the NAT option will also improve the performance of the provisioning server in this scenario.

If you have many devices behind a NAT equipment, you should also check the *security* section to make sure the IP address of your NAT equipment doesn't get banned unintentionally.

Limitations

- You must only have phones of the following brands:
  - Aastra
  - Cisco SPA
  - Yealink
- All your devices must be behind a NAT equipment (the devices may be grouped behind different NAT equipments, not necessarily the same one)
- You must provision the devices via the Web interface, i.e. associate the devices from the user form. Using the 6-digit provisioning code on the phone will produce unexpected results (i.e. the wrong device will be provisioned)

Security

By design, the auto-provisioning process is vulnerable to:

- Leakage of sensitive information: some files that are served by the provisioning server contains sensitive information, e.g. SIP credentials that are used by SIP phones to make calls. Depending on your network configuration and the amount of information an attacker has on your telephony ecosystem (phone vendor, MAC address, etc.), he could retrieve the content of some files containing sensitive information.
- Denial-of-service attack: in its default configuration, each time the provisioning server identify a request coming from a new device, it creates a new device object in its database. An attacker could spoof requests to the provisioning server to create a huge amount of devices, creating a denial-of-service condition.

That said, starting from XiVO 16.08, XiVO adds Fail2ban support to the provisioning server to drastically lower the likelihood of such attacks. Every time a request for a file potentially containing sensitive information is requested, a log line is appended to the `/var/log/xivo-provd-fail2ban.log` file, which is monitored by fail2ban. The same thing happens when a new device is automatically created by the provisioning server.

The fail2ban configuration for the provisioning server is located at `/etc/fail2ban/jail.d/xivo.conf`. You may want to adjust the `findtime` / `maxretry` value if you have special requirements. In particular, if you have many phones behind a NAT equipment, you'll probably have to adjust these values, since every request coming from your phones behind your NAT will appear to the provisioning server as coming from the same source IP address, and this IP address will then be more likely to get banned promptly if you, for example, reboot all your phones at the same time. Another solution would be to add your IP address to the list of ignored IP address of fail2ban. See the fail2ban(1) man page for more information.

System Requirements

XiVO 16.08 or later is required. You also need to use compatible xivo-provd plugins. Here's the list of official plugins which are compatible:

Plugin family	Version
xivo-aastra	>= 1.6
xivo-cisco-sccp	>= 1.1
xivo-cisco-spa	>= 1.0
xivo-digium	>= 1.0
xivo-polycom	>= 1.7
xivo-snom	>= 1.6
xivo-yealink	>= 1.26

5.25.4 Remote directory

If you have a phone provisioned with XiVO and its one of the supported ones, you'll be able to search in your XiVO directory and place call directly from your phone.

See the list of supported devices to know if a model supports the XiVO directory or not.

Configuration

For the remote directory to work on your phones, the first thing to do is to go to the *Services → IPBX → (General settings) Phonebook* page.

You then have to add the range of IP addresses that will be allowed to access the directory. So if you know that your phone's IP addresses are all in the 192.168.1.0/24 subnet, just click on the small "+" icon and enter "192.168.1.0/24", then save.

Once this is done, on your phone, just click on the "remote directory" function key and you'll be able to do a search in the XiVO directory from it.

5.25.5 Jitsi

Jitsi (http://jitsi.org/) is an opensource softphone (previously SIP Communicator).

XiVO now support Jitsi sofphones provisioning. Here are the steps to follow :

Requirements

This how to needs :

1. Jitsi installed,
2. SIP line created

Add Jitsi plugin on XiVO

Open XiVO Web interface, and go to Configuration tab, Then chose *Provisioning → Plugins menu*, Install the Jitsi plugin you want to use : e.g.:

```
xivo-jitsi-1
```

You can now launch your Jitsi softphone

Configuring Jitsi

1. Launch Jitsi,
2. If you don't have any accounts configured Jitsi will launch a windows and you can click
3. Use online provisioning. Otherwise go to Tools -> Options -> Advanced -> Provisioing, Click on Enable provisioning
4. Select Manually specify a provisioning URI,
5. Enter the folowing URI where <provd_ip> is the VoIP interface IP address of your XiVO and <provd_port> is the provd port (default : 8667)

   ```
   http://<provd_ip>:<provd_port>/jitsi?uuid=${uuid}
   ```

6. When done, quit Jitsi,

7. Launch Jitsi again,
   - You should now be connected with in autoprov mode,
   - You could see a new device in the devices list,
8. You can now provision the phones by typing the provisioning code (you get it in the Lines list),
9. Quit Jitsi again (configuration syncing is not available with the Jitsi plugin)
10. And launch Jitsi again : you should now be connected with you phone account

5.26 SCCP Configuration

5.26.1 Provisioning

To be able to provision SCCP phones you should :

- activate the *DHCP Server*,
- activate the *DHCP Integration*,

Then install a plugin for SCCP Phone: *Configuration → Provisioning → Plugins*

Name	Description	Version	Size	Action
null	Plugin that offers no configuration service and rejects TFTP/HTTP...	1.0	955.00 bytes	
xivo-aastra-3.3.1-SP2	Plugin for Aastra 6730i, 6731i, 6735i, 6737i, 6739i, 6753i, 6755i...	0.5.9	-	
xivo-aastra-3.3.1-SP4-HF9	Plugin for Aastra/Mitel 67XXi in version 3.3.1 SP4 HF9.	1.8	8.25 KiB	
xivo-aastra-4.3.0	Plugin for Aastra/Mitel 6863i, 6865i, 6867i and 6869i in version ...	1.8	7.73 KiB	
xivo-aastra-switchboard	Plugin for Aastra 6731i, 6755i, 6757i in version 3.2.2.2112.	0.5.1	-	
xivo-cisco-pap2t-5.1.6	Plugin for Cisco PAP2T in version 5.1.6.	1.0	12.48 KiB	
xivo-cisco-sccp-8.5.2	Plugin for Cisco 7906G, 7911G, 7931G, 7941G, 7941G-GE, 7942G, 794...	1.1	6.63 KiB	
xivo-cisco-sccp-9.0.3	Plugin for Cisco 7906G, 7911G, 7941G, 7941G-GE, 7942G, 7961G and ...	0.5	-	
xivo-cisco-sccp-9.4	Plugin for Cisco 7906G, 7911G, 7931G, 7941G, 7941G-GE, 7942G, 796...	1.1	6.78 KiB	
xivo-cisco-sccp-cipc-2.1....	Plugin for Cisco IP Communicator in version 2.1.2.	1.1	5.59 KiB	
xivo-cisco-sccp-legacy	Plugin for Cisco 7905G, 7912G, 7920, 7940G and 7960G in different...	1.1.1	6.86 KiB	
xivo-cisco-sccp-wireless-...	Plugin for Cisco 7921G in version 1.4.5 of the SCCP software.<br/...	1.1	6.05 KiB	
xivo-cisco-spa-7.5.5	Plugin for Cisco SMB SPA301, 303, 501G, 502G, 504G, 508G, 509G, 5...	1.0	13.58 KiB	

Fig. 35: Installing xivo cisco-sccp plugin

At this point you should have a fully functional DHCP server that provides IP address to your phones. Depending on what type of CISCO phone you have, you need to install the plugin sccp-legacy, sccp-9.4 or both.

Note: Please refer to the Provisioning page for more information on how to install CISCO firmwares.

Once your plugin is installed, you'll be able to edit which firmwares and locales you need. If you are unsure, you can choose all without any problem.

Now if you connect your first SCCP phone, you should be able to see it in the device list.

Listing the detected devices: *Services → IPBX → IPBX settings → Devices*

When connecting a second SCCP phone, the device will be automatically detected as well.

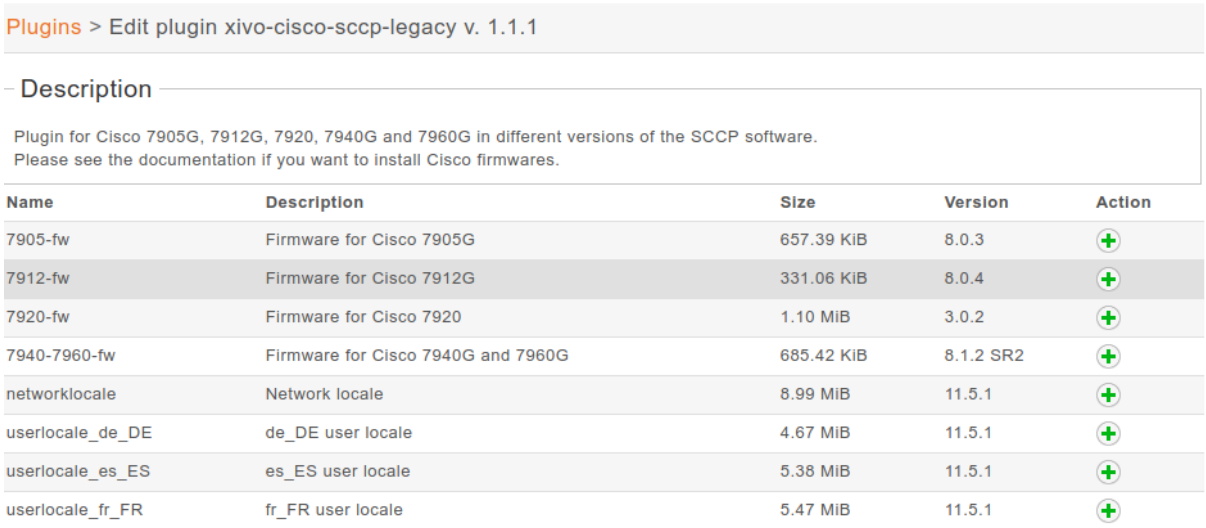


Fig. 36: Editing the xivo-cisco-sccp-legacy plugin

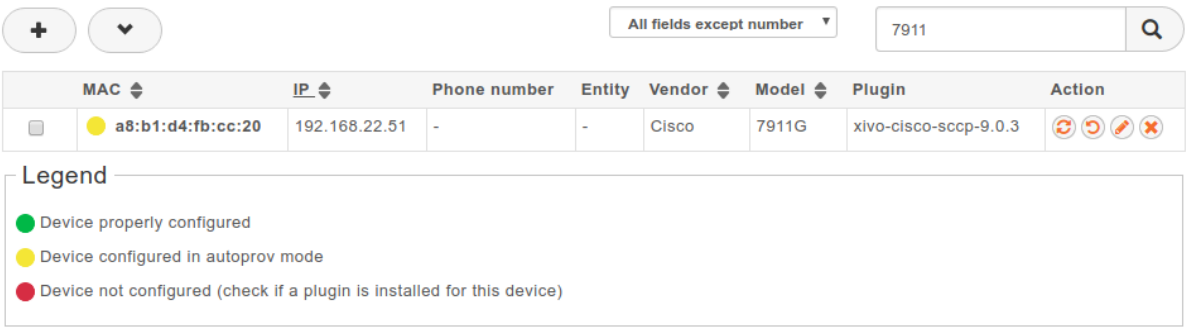


Fig. 37: Device list

5.26.2 SCCP General Settings

Review SCCP general settings: *Services → IPBX → IPBX settings → SCCP general settings*

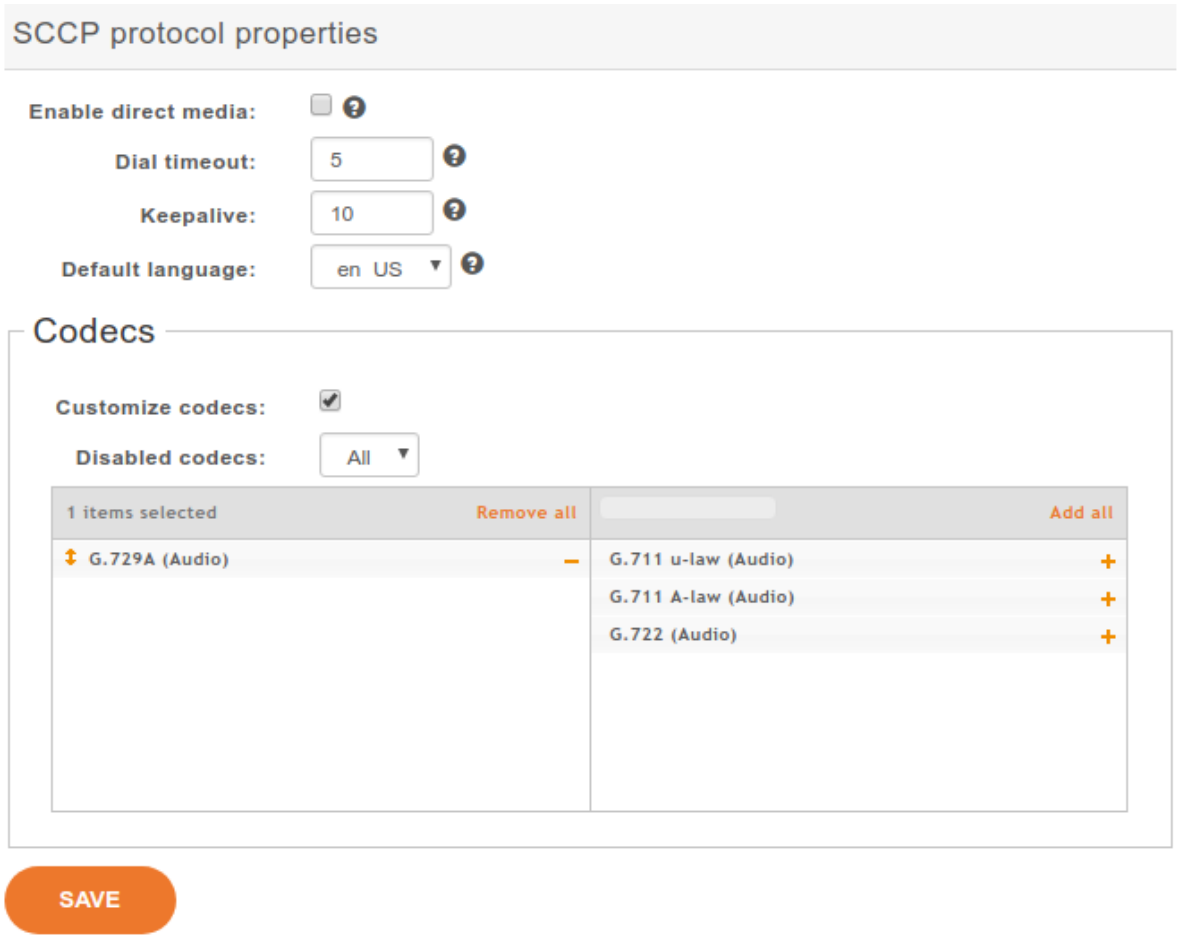


Fig. 38: SCCP general settings

5.26.3 User creation

The last step is to create a user with a **SCCP line**.

Creating a user with a SCCP line: *Services → IPBX → IPBX settings → Users*

Before saving the newly configured user, you need to select the *Lines* menu and add a SCCP line. Now, you can save your new user.

Congratulations ! Your SCCP phone is now ready to be called !

5.26.4 Function keys

With SCCP phones, the only function keys that can be configured are:

- *Key*: Only the order is important, not the number
- *Type*: `Customized`; Any other type doesn't work
- *Destination*: Any valid extension
- *Label*: Any label

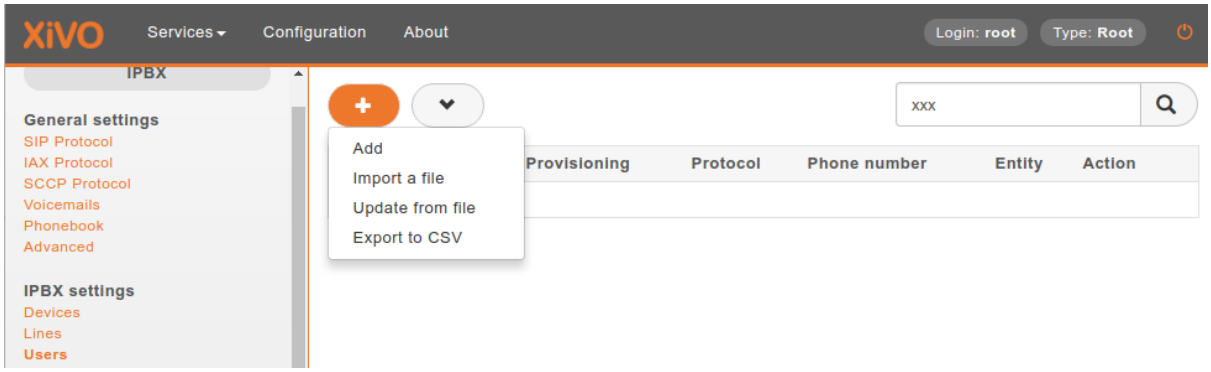


Fig. 39: Add a new user

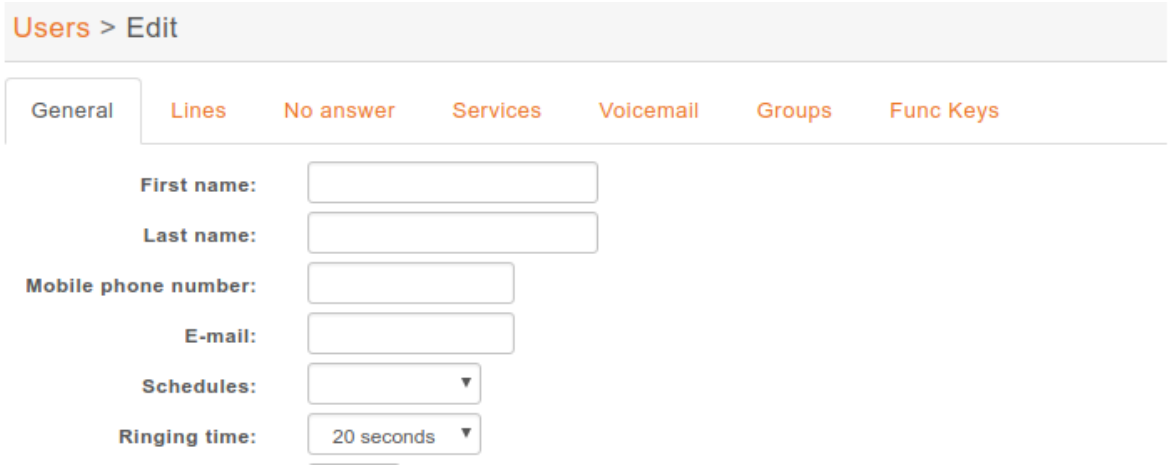


Fig. 40: Edit user informations

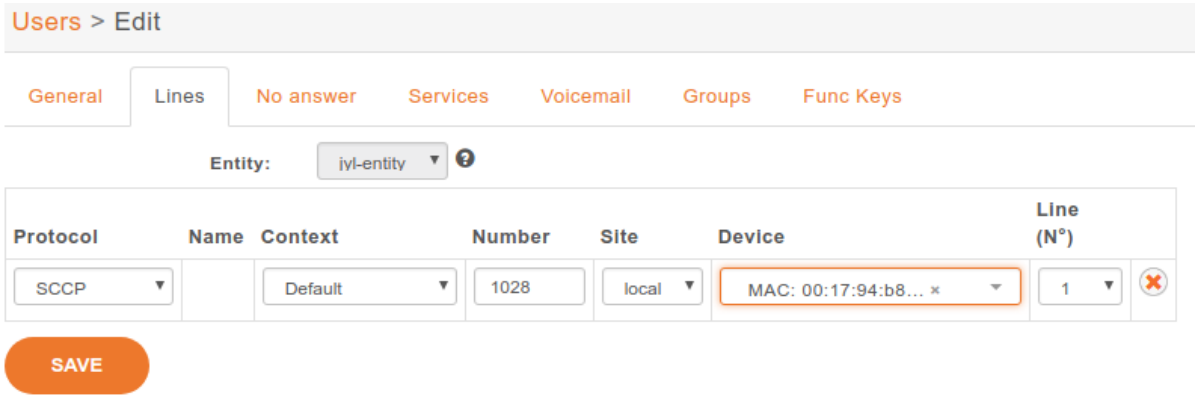


Fig. 41: Add a line to a user

- *Supervision*: `Enabled` or `Disabled`

5.26.5 Direct Media

SCCP Phones support directmedia (direct RTP). In order for SCCP phones to use directmedia, one must enable the directm
Services → IPBX → IPBX settings → SCCP general settings

5.26.6 Features

Features	Supported
Receive call	Yes
Initiate call	Yes
Hangup call	Yes
Transfer call	Yes
Congestion Signal	Yes
Autoanswer (custom dialplan)	Yes
Call forward	Yes
Multi-instance per line	Yes
Message waiting indication	Yes
Music on hold	Yes
Context per line	Yes
Paging	Yes
Direct RTP	Yes
Redial	Yes
Speed dial	Yes
BLF (Supervision)	Yes
Resync device configuration	Yes
Do not disturb (DND)	Yes
Group listen	Yes
Caller ID	Yes
Connected line ID	Yes
Group pickup	Yes
Auto-provisioning	Not yet
Multi line	Not yet
Codec selection	Yes
NAT traversal	Not yet
Type of Service (TOS)	Manual

5.26.7 Telephone

Device type	Supported	Firmware version	Timezone aware
7905	Yes	8.0.3	No
7906	Yes	SCCP11.9-4-2SR1-1	Yes
7911	Yes	SCCP11.9-4-2SR1-1	Yes
7912	Yes	8.0.4(080108A)	No
7920	Yes	3.0.2	No
7921	Yes	1.4.5.3	Yes
7931	Yes	SCCP31.9-4-2SR1-1	Yes
7937	Testing		
7940	Yes	8.1(SR.2)	No
7941	Yes	SCCP41.9-4-2SR1-1	Yes
7941GE	Yes	SCCP41.9-4-2SR1-1	Yes
7942	Yes	SCCP42.9-4-2SR1-1	Yes
7945	Testing		
7960	Yes	8.1(SR.2)	No
7961	Yes	SCCP41.9-4-2SR1-1	Yes
7962	Yes	SCCP42.9-4-2SR1-1	Yes
7965	Testing		
7970	Testing		
7975	Testing		
CIPC	Yes	2.1.2	Yes

Models not listed in the table above won't be able to connect to Asterisk at all. Models listed as "Testing" are not yet officially supported in XiVO: use them at your own risk.

The "Timezone aware" column indicates if the device supports the timezone tag in its configuration file, i.e. in the file that the device request to the provisioning server when it boots. If you have devices that don't support the timezone tag and these devices are in a different timezone than the one of the XiVO, you can look at the issue #5161 for a potential solution.

5.27 Schedules

Schedules are specific time frames that can be defined to open or close a service. Within schedules you may specify opening days and hours or close days and hours.

A default destination as user, group ... can be defined when the schedule is in closed state.

Schedules can be applied to :

- Users
- Groups
- Inbound calls
- Outbound calls
- Queues

5.27.1 Creating Schedules

A schedule is composed of a name, a timezone, one or more opening hours or days that you may setup using a calendar widget, a destination to be used when the schedule state is closed.

With the calendar widget you may select months, days of month, days of week and opening time.

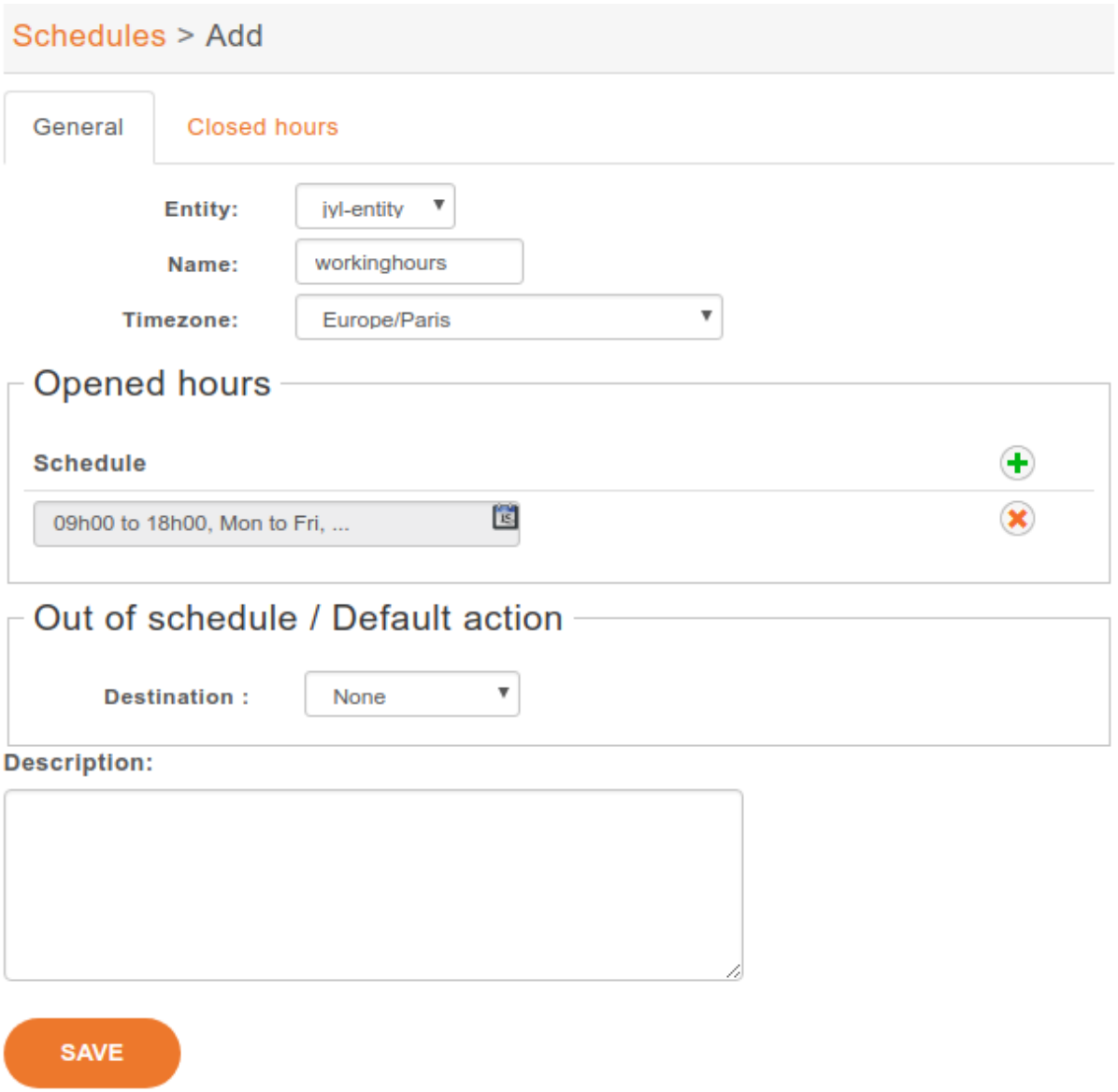


Fig. 42: Creating a schedule

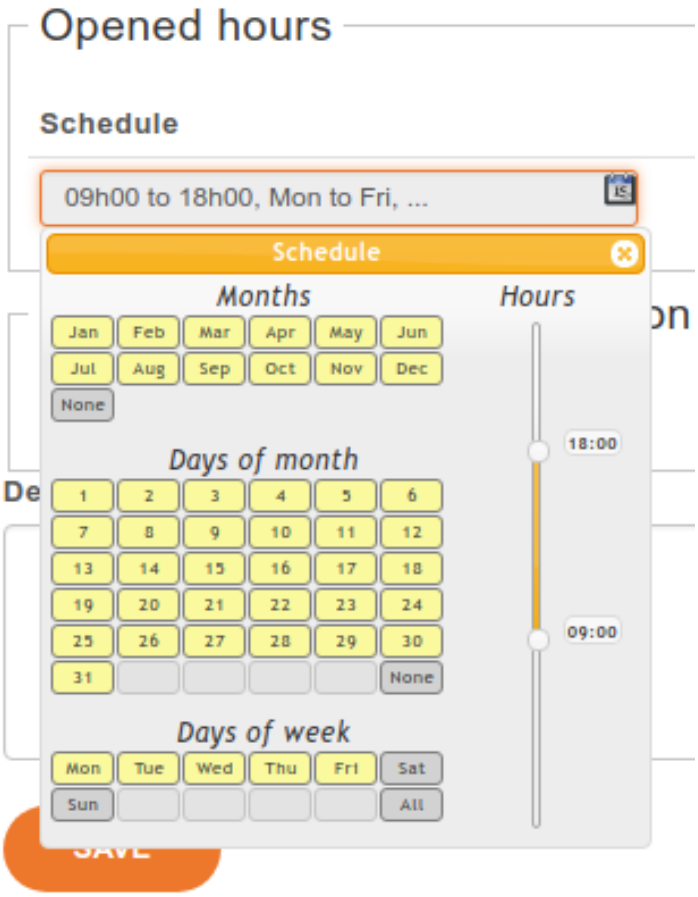


Fig. 43: Schedule calendar widget

You may also optionaly select closed hours and destination to be applied when period is inside the main schedule. For example, your main schedule is opened between 08h00 and 18h00, but you are closed between 12h00 and 14h00.

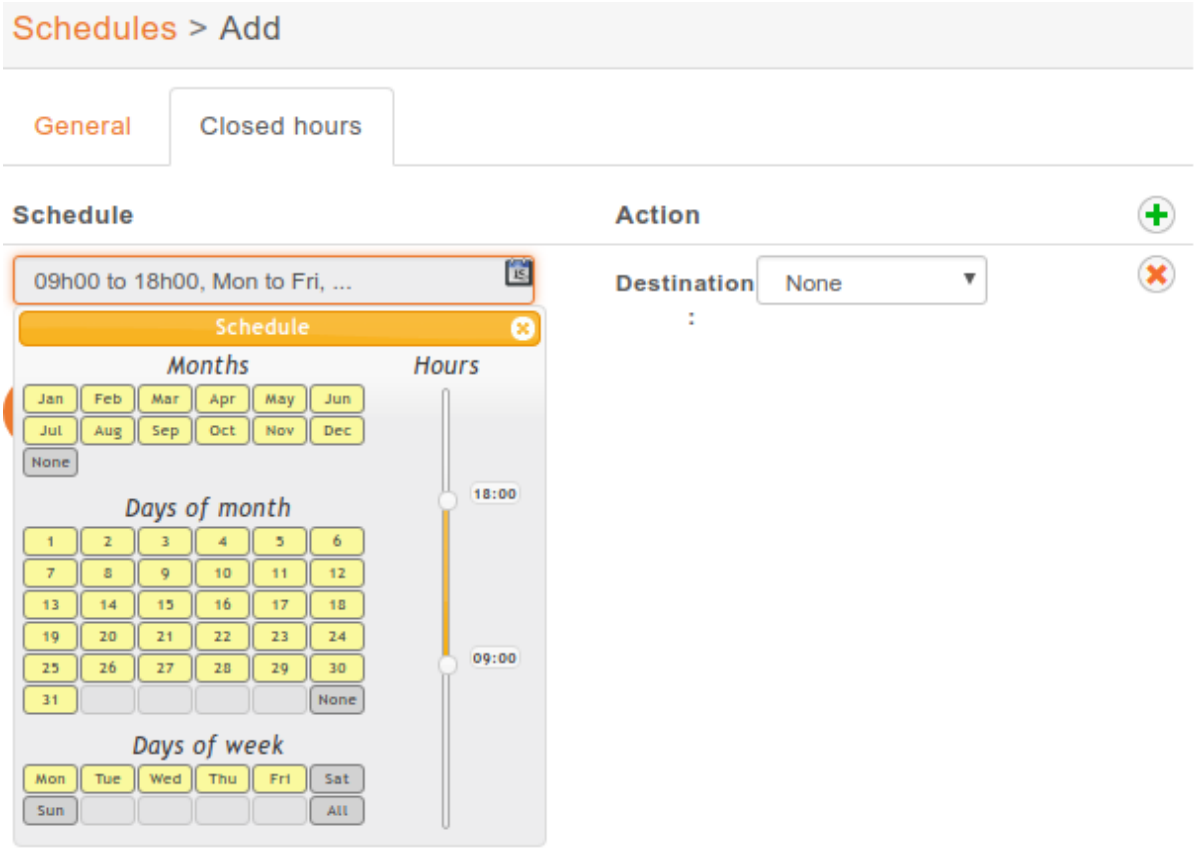


Fig. 44: Schedule closed hours

5.27.2 Using Schedule on Users

When you have a schedule associated to a user, if this user is called during a closed period, the caller will first hear a prompt saying the call is being transferred before being actually redirected to the closed action of the schedule.

If you don't want this prompt to be played, you can change the behaviour by:

1. editing the `/etc/xivo/asterisk/xivo_globals.conf` file and setting the `XIVO_FWD_SCHEDULE_OUT_ISDA` to `1`
2. reloading the asterisk dialplan with an `asterisk -rx "dialplan reload"`.

5.28 Sound Files

5.28.1 Add Sounds Files

On a fresh install, only en_US and fr_FR sounds are installed. Canadian French and German are available too.

To install Canadian French sounds you have to execute the following command:

```
apt-get install asterisk-sounds-wav-fr-ca xivo-sounds-fr-ca
```

To install German sounds you have to execute the following command:

```
apt-get install asterisk-sounds-wav-de-de xivo-sounds-de-de
```

Now you may select the newly installed language for your users.

5.28.2 Convert Your Wav File

Asterisk will read natively WAV files encoded in wav 8kHz, 16 bits, mono.

The following command will return the encoding format of the <file>

```
$ file <file>
RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
```

The following command will re-encode the <input file> with the correct parameters for asterisk and write into the <output file>:

```
sox <input file> -b 16 -c 1 -t wav <output file> rate -I 8000
```

5.29 Switchboard

This page describes the configuration needed to have a switchboard on your XiVO.

5.29.1 Overview

Switchboard functionality is available in the XiVO client. The goal of this page is to explain how to configure your switchboard and how to use it.

The switchboard xlet and profile allow an operator to view incoming calls, answer them, put calls on hold, view the calls on hold and pick up the calls on hold.

5.29.2 Limitations

Note: The shortcut keys of the switchboard do not work on the Mac version of the XiVO client.

Note: The enter shortcut to answer a call will not work if the focus is currently on a widget that will consume the key press. ie: a text field, a drop down

Note: Attended transfers to the switchboard cannot be managed with the switchboard xlets depending on the moment at which the call was completed.

5.29.3 Table of Contents

Switchboard Configuration

Be sure to read the *limitations* before configuring a switchboard.

Server configuration

Quick Summary

In order to configure a switchboard on your XiVO, you need to:

- Create a queue for your switchboard
- Create a queue for your switchboard's calls on hold
- Create the users that will be operators
- Activate the switchboard option for your phone
- Create an agent for your user
- Assign the incoming calls to the switchboard queue
- For each operator, add a function key for logging in or logging out from the switchboard queue.
- Set "no answer" destinations on the switchboard queue

Supported Devices

The supported phones for the switchboard are:

Brand	Model	XiVO version	Plugin version
Aastra	6755i	>= 14.07	>= xivo-aastra-3.3.1-SP2, v1.0
Aastra	6757i	>= 14.07	>= xivo-aastra-3.3.1-SP2, v1.0
Aastra	6735i	>= 14.07	>= xivo-aastra-3.3.1-SP2, v1.2
Aastra	6737i	>= 14.07	>= xivo-aastra-3.3.1-SP2, v1.2
Polycom	VVX 400	>= 15.11	>= xivo-polycom-5.3.0, v1.3
Polycom	VVX 410	>= 15.11	>= xivo-polycom-5.3.0, v1.3
Snom	720	>= 14.14	>= xivo-snom-8.7.3.25.5, v1.0
Snom	D725	>= 14.14	>= xivo-snom-8.7.5.17, v1.4
Yealink	T46G	>= 15.01	>= xivo-yealink-72.0, v1.22.1

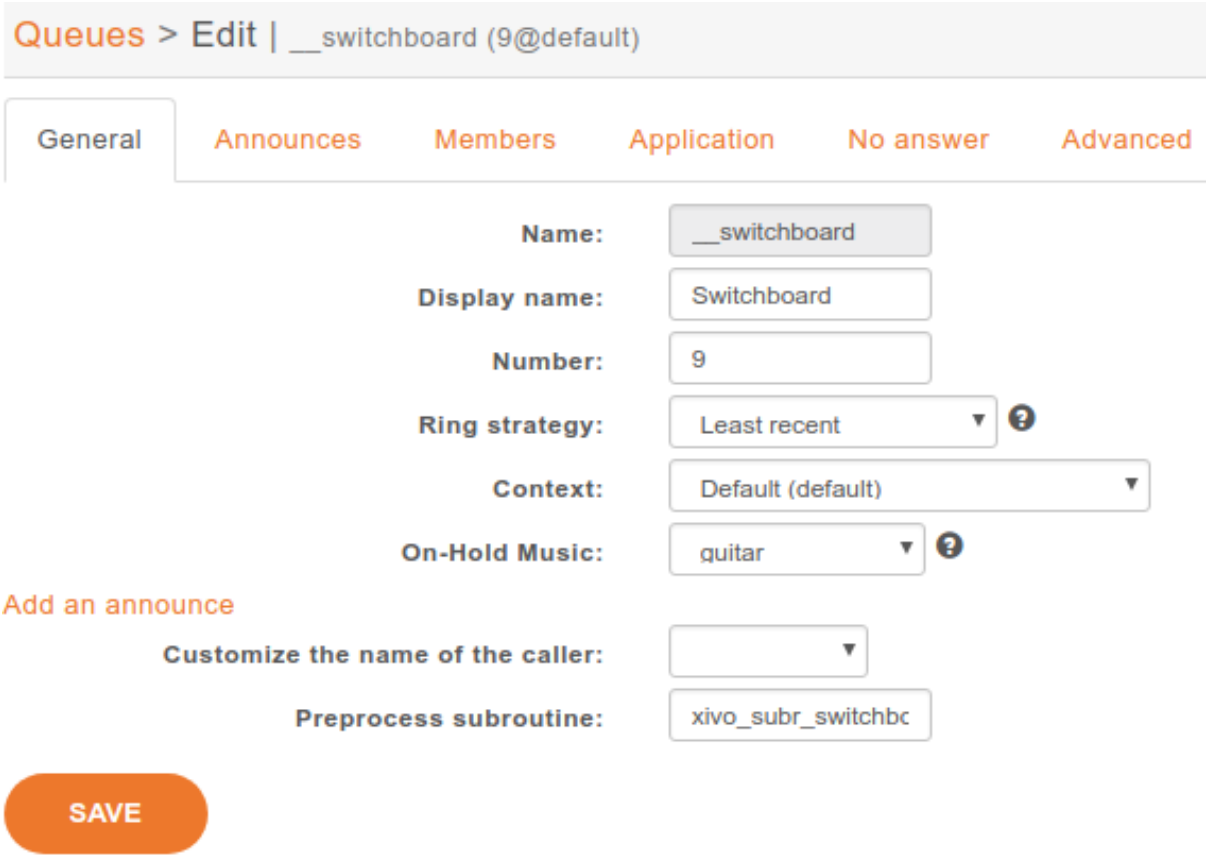
Create a Queue for Your Switchboard

All calls to the switchboard will first be distributed to a switchboard queue.

To create this queue, go to *Services* → *Call center* → *Queues* and click the add button.

The following configuration is mandatory

- The *General* → *Name* field has to be *__switchboard*
- The *General* → *Ring strategy* field has to be *Ring all*
- The *General* → *Preprocess subroutine* field has to be *xivo_subr_switchboard*
- The *Application* → *Allow caller to hang up call* option has to be *enabled*
- The *Application* → *Allow callee to transfer the call* option has to be *enabled*
- The *Advanced* → *Member reachability timeout* option has to be *disabled*
- The *Advanced* → *Time before retrying a call to a member* option has to be *1 second*
- The *Advanced* → *Delay before reassigning a call* option has to be *disabled*
- The *Advanced* → *Call a member already on* option has to be *disabled*
- The *Advanced* → *Autopause agents* option has to be *No*



Other important fields

- The *General* → *Display name* field is the name displayed in the XiVO client xlets and in the statistics
- The *General* → *Number* field is the number that will be used to reach the switchboard internally (typically *9*)

Create a Queue for Your Switchboard on Hold

The switchboard uses a queue to track its calls on hold.

To create this queue, go to *Services* → *Call center* → *Queues* and click the add button.

The following configuration is mandatory

- The *General* → *Name* field has to be *__switchboard_hold*
- The *General* → *Number* field has to be a valid number in a context reachable by the switchboard
- The *Advanced* → *Join an empty queue* option list has to be empty
- The *Advanced* → *Remove callers if there are no agents* option list has to be empty

Other important fields

- The *General* → *Display name* field is the name displayed in the XiVO client xlets and in the statistics

Warning: This queue MUST have **NO** members

Create the Users that Will be Operators

Each operator needs to have a user configured with a line. The XiVO client profile has to be set to *Switchboard*.

The following configuration is mandatory for switchboard users

- The *General* → *First name* field has to be set
- The *General* → *Enable XiVO Client* option has to be *enabled*
- The *General* → *Login* field has to be set
- The *General* → *Password* field has to be set
- The *General* → *Profile* field has to be set to *Switchboard*
- The *Lines* → *Number* field has to have a valid extension
- The *Lines* → *Device* field has to be a *supported device*
- The *Services* → *Enable call transfer* option has to be *enabled*
- The *Services* → *Enable supervision* option has to be *enabled*

Activate the Switchboard Option for your Phone

The switchboard option must be activated on the phone. It's possible to activate this option only on *supported phones* and plugins.

- Edit device associated to your user in *Services* → *Devices*
- Check the switchboard checkbox and save
- Synchronize your phone to apply the changes

Polycom Phones

To be able to use a Polycom phone for the switchboard, the XiVO must be able to do HTTP requests to the phone. This might be problematic if there's a NAT between your XiVO and your phone.

It's possible to configure the Polycom switchboard via the *configuration files* of xivo-ctid. The following options are available:

```
switchboard_polycom:
    username: xivo_switchboard
    password: xivo_switchboard
    answer_delay: 0.5
```

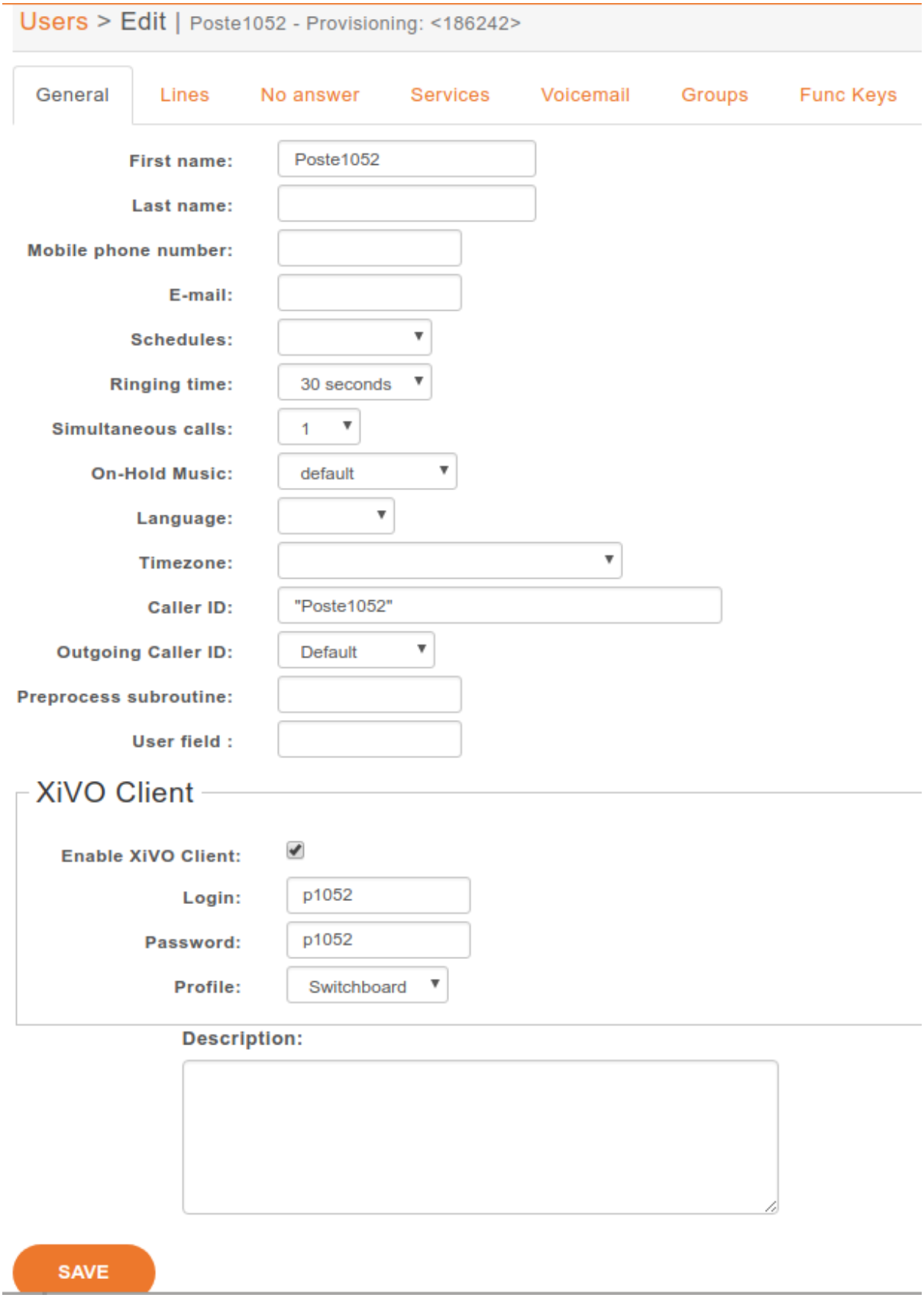
You will also need to change the XML API username/password by creating a *custom template* for your phone.

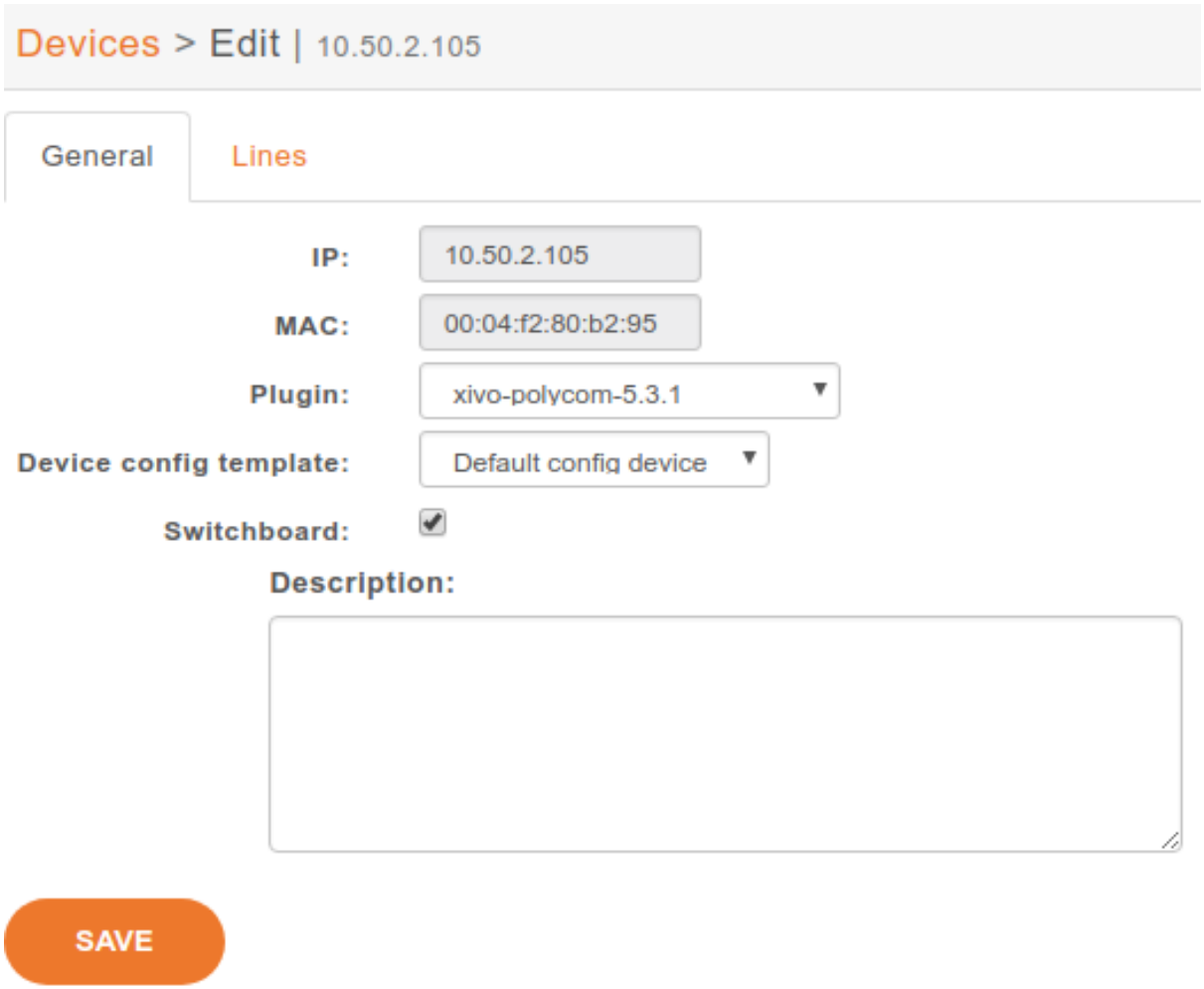
Snom Phones

When using a Snom switchboard, you must not configure a function key on position 1.

To be able to use a Snom phone for the switchboard, the XiVO must be able to do HTTP requests to the phone. This might be problematic if there's a NAT between your XiVO and your phone. The following command should work from your XiVO's bash command line `wget http://guest:guest@<phone IP address>/command.htm?key=SPEAKER`. If this command does not activate the phone's speaker, your network configuration will have to be *fixed* before you can use the Snom switchboard.

It's possible to configure the Snom switchboard via the *configuration files* of xivo-ctid. The following options are available:


Users > Edit | Poste1052 - Provisioning: <186242>
General
Lines
No answer
Services
Voicemail
Groups
Func Keys
First name:
Poste1052
Last name:
Mobile phone number:
E-mail:
Schedules:
Ringing time:
30 seconds
Simultaneous calls:
1
On-Hold Music:
default
Language:
Timezone:
Caller ID:
"Poste1052"
Outgoing Caller ID:
Default
Preprocess subroutine:
User field :
XiVO Client
Enable XiVO Client:
Login:
p1052
Password:
p1052
Profile:
Switchboard
Description:
SAVE


Devices > Edit | 10.50.2.105
General
Lines
IP:
10.50.2.105
MAC:
00:04:f2:80:b2:95
Plugin:
xivo-polycom-5.3.1
Device config template:
Default config device
Switchboard:
Description:
SAVE

```
switchboard_snom:
    username: guest
    password: guest
    answer_delay: 0.5
```

You have to change the username and password option if you have changed the administrator username or administrator password for your phone in *Configuration* → *Provisioning* → *Template Device*.

Yealink Phones

When using a Yealink switchboard, you must not configure a function key on position 1.

Create an Agent for the Operator

Each operator needs to have an associated agent.

Warning: Each agent MUST ONLY be a member of the Switchboard queue

To create an agent:

- Go to *Services* → *Call center* → *Agents*
- Click on the group *default*
- Click on the *Add* button
- Associate the user to the agent in the *Users* tab
- Assign the Agent to the *Switchboard* Queue (**and ONLY to the Switchboard queue**)

Send Incoming Calls to the *Switchboard* Queue

Incoming calls must be sent to the *Switchboard* queue to be distributed to the operators. To do this, we have to change the destination of our incoming call for the switchboard queue.

In this example, we associate our incoming call (DID *444*) to our *Switchboard* queue:

Set "No Answer" Destinations on the *Switchboard* Queue

When there are no operators available to answer a call, "No Answer" destinations should be used to redirect calls towards another destination.

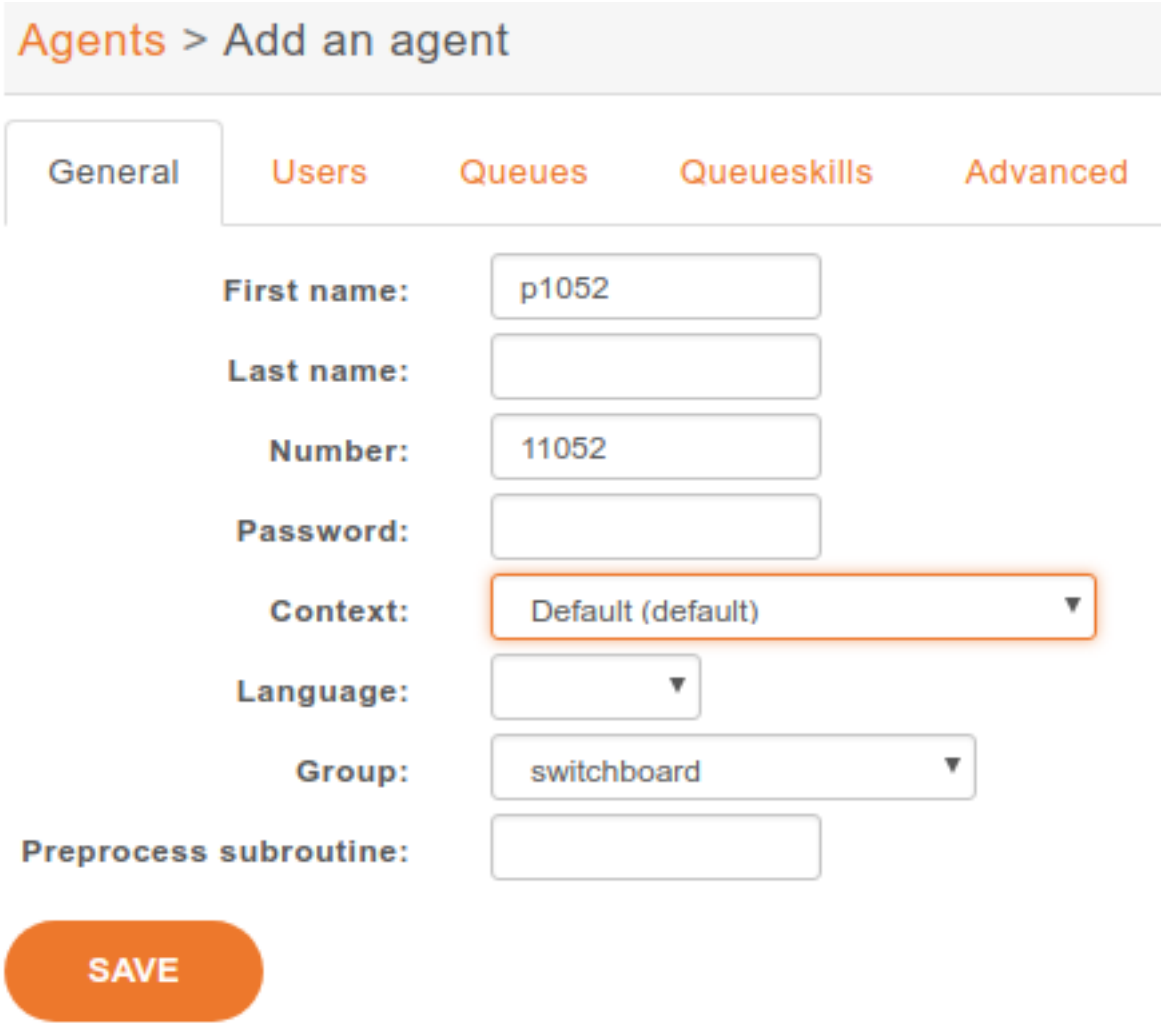
You also need to set the timeout of the Switchboard queue to know when calls will be redirected.

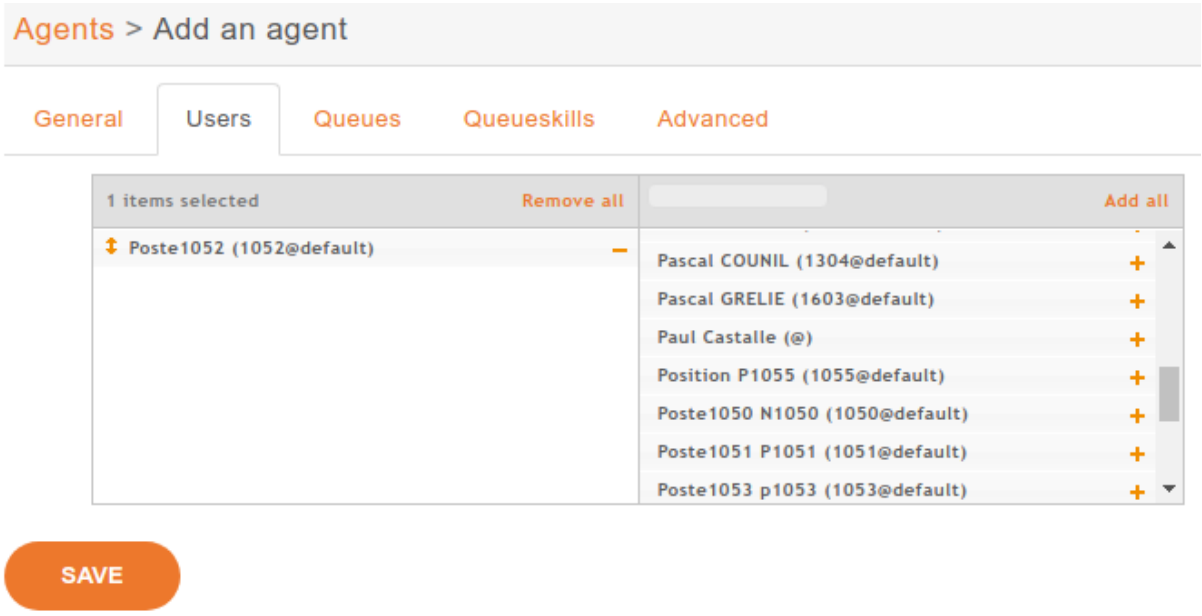
The reachability timeout must not be disabled nor be too short.

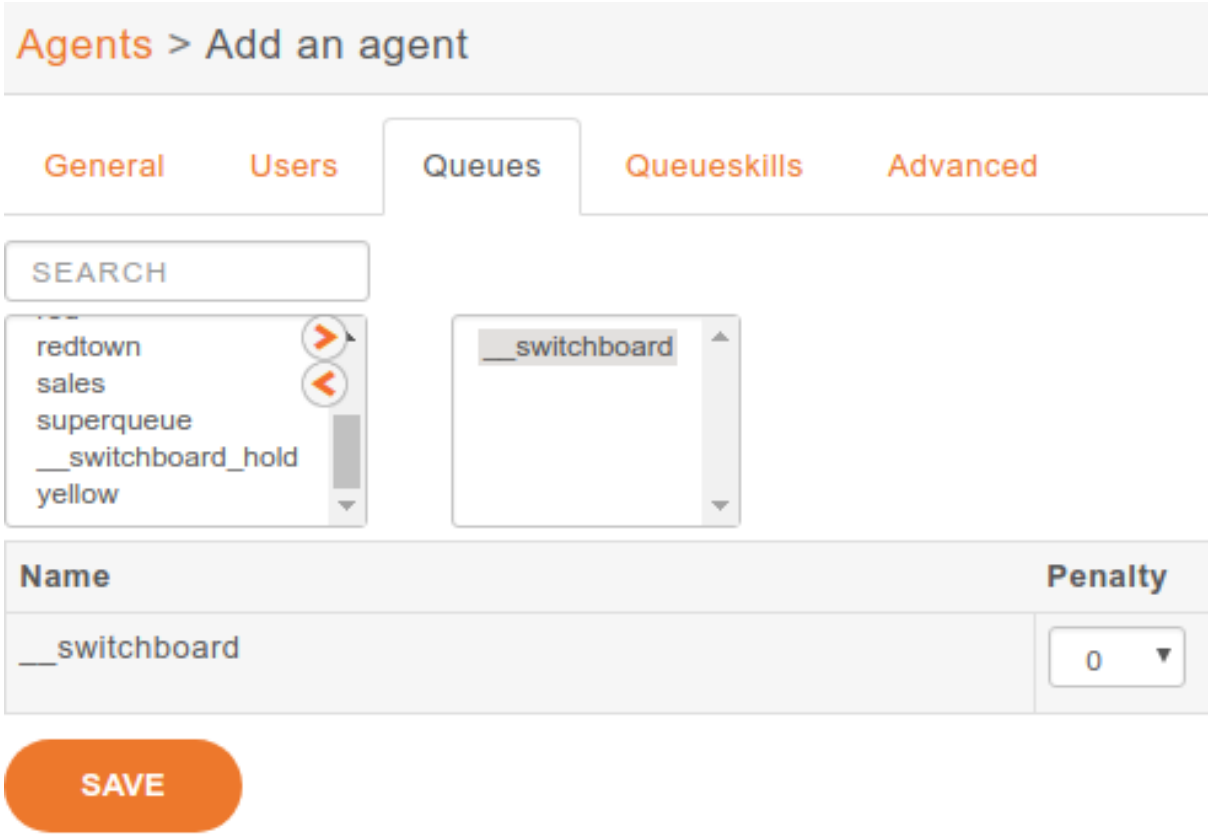
The time before retrying a call to a member should be as low as possible (1 second).

In this example we redirect "No Answer", "Busy" and "Congestion" calls to the *everyone* group and "Fail" calls to the *guardian* user.

You can also choose to redirect all the calls to another user or a voice mail.


Agents > Add an agent
General
Users
Queues
Queueskills
Advanced
First name:
p1052
Last name:
Number:
11052
Password:
Context:
Default (default)
Language:
Group:
switchboard
Preprocess subroutine:
SAVE


Agents > Add an agent
General
Users
Queues
Queueskills
Advanced
1 items selected
Remove all
Add all
Poste1052 (1052@default)
Pascal COUNIL (1304@default)
Pascal GRELIE (1603@default)
Paul Castalle (@)
Position P1055 (1055@default)
Poste1050 N1050 (1050@default)
Poste1051 P1051 (1051@default)
Poste1053 p1053 (1053@default)
SAVE



XiVO Client configuration

Directory xlet

The transfer destination is chosen in the Directory xlet. You **must** follow the *Directory Xlet* section to be able to use it.

Configuration for multiple switchboards

The above documentation can be used for multiple switchboards on the same XiVO by replacing the *__switchboard* and *__switchboard_hold* queues name and configuring the operators XiVO client accordingly in the *XiVO Client* → *Configure* → *Functions* → *Switchboard* window.

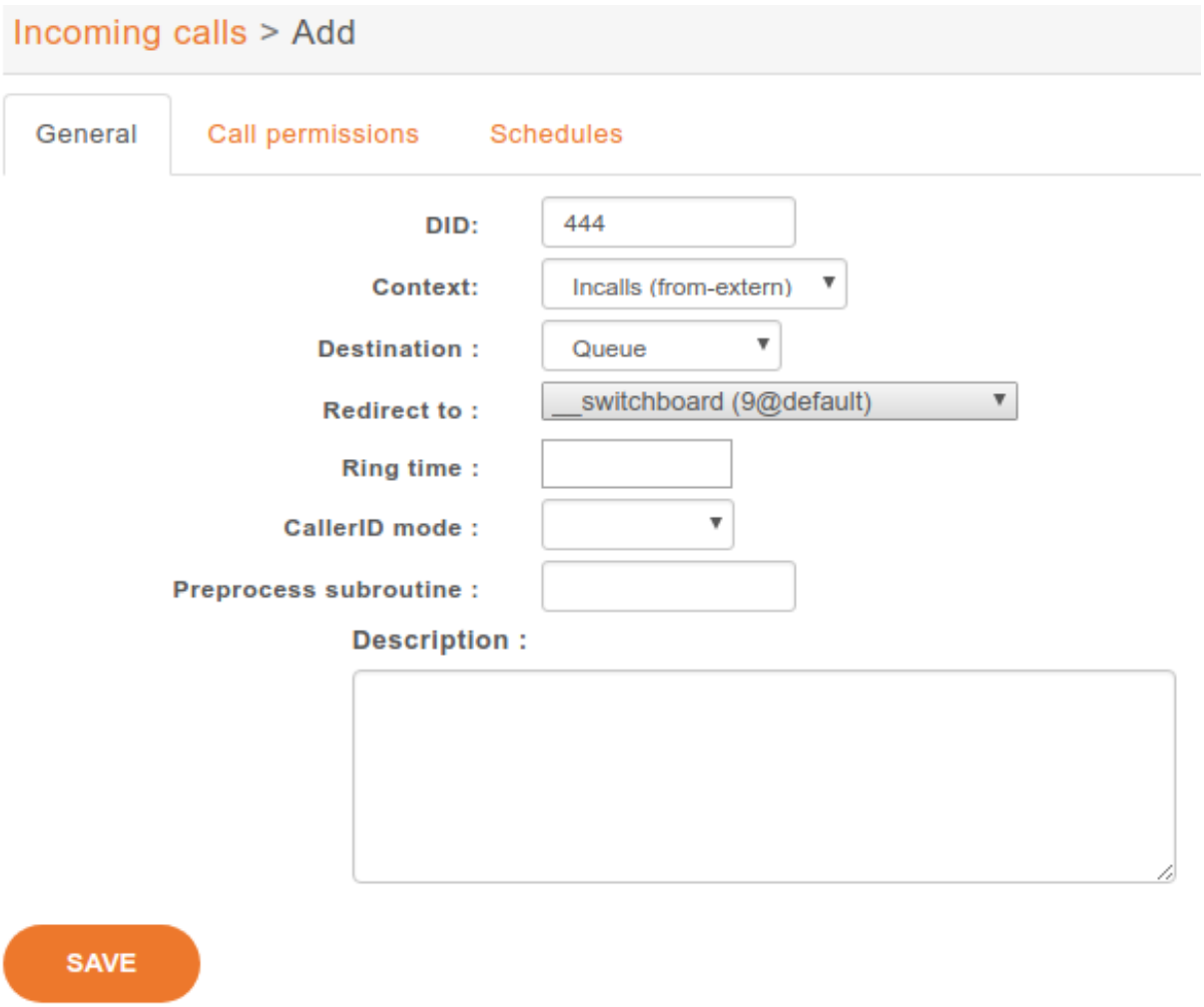
All switchboard queues should be added to the xivo-ctid configuration. New queues can be added by adding a file in `/etc/xivo-ctid/conf.d`. For example, the following content should be used for a new switchboard queue names *__switchboard_two* and an hold queue names *__switchboard_hold_two*.

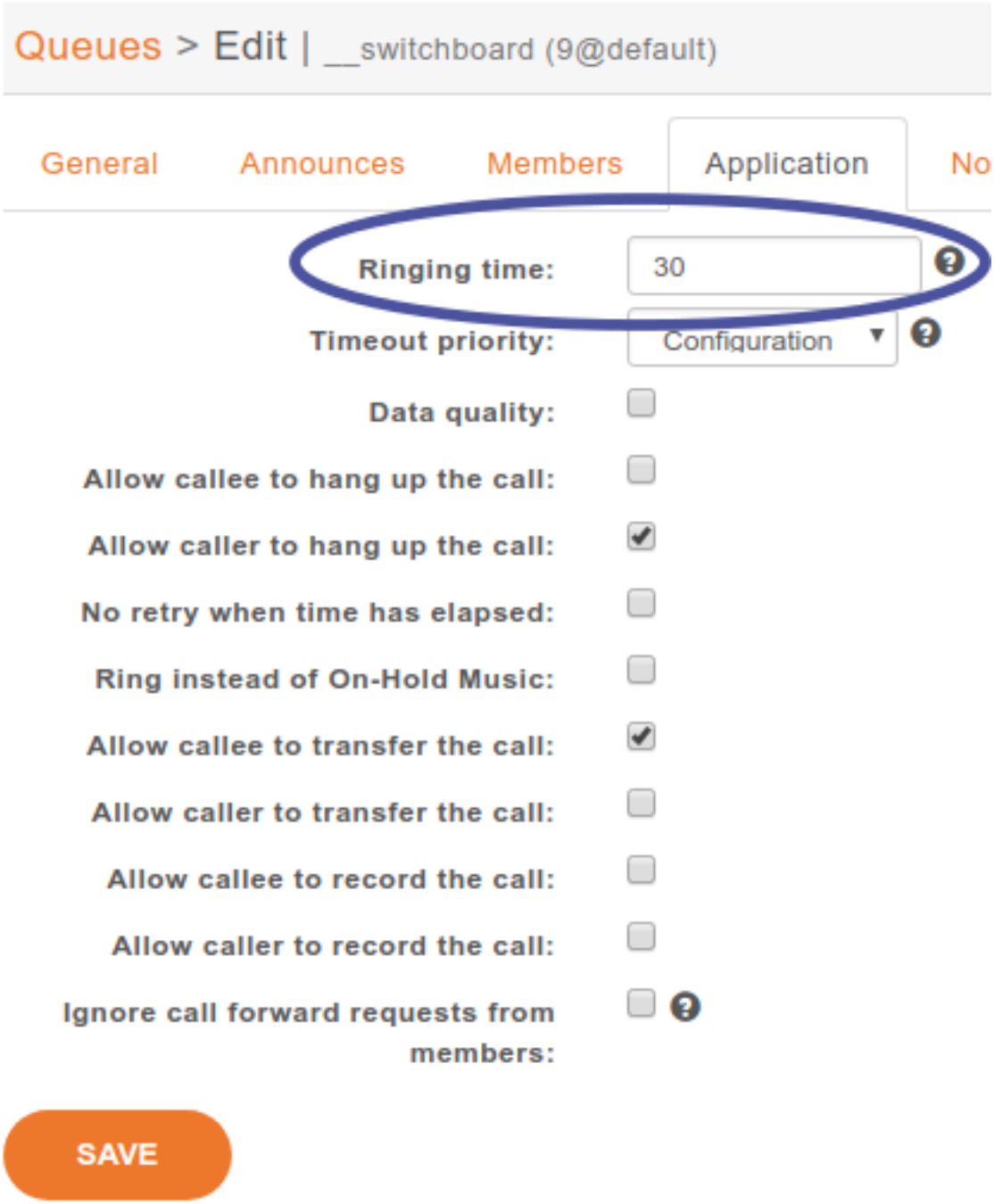
```
{"switchboard_queues": {"__switchboard_two": true},
 "switchboard_hold_queues": {"__switchboard_hold_two": true}}
```

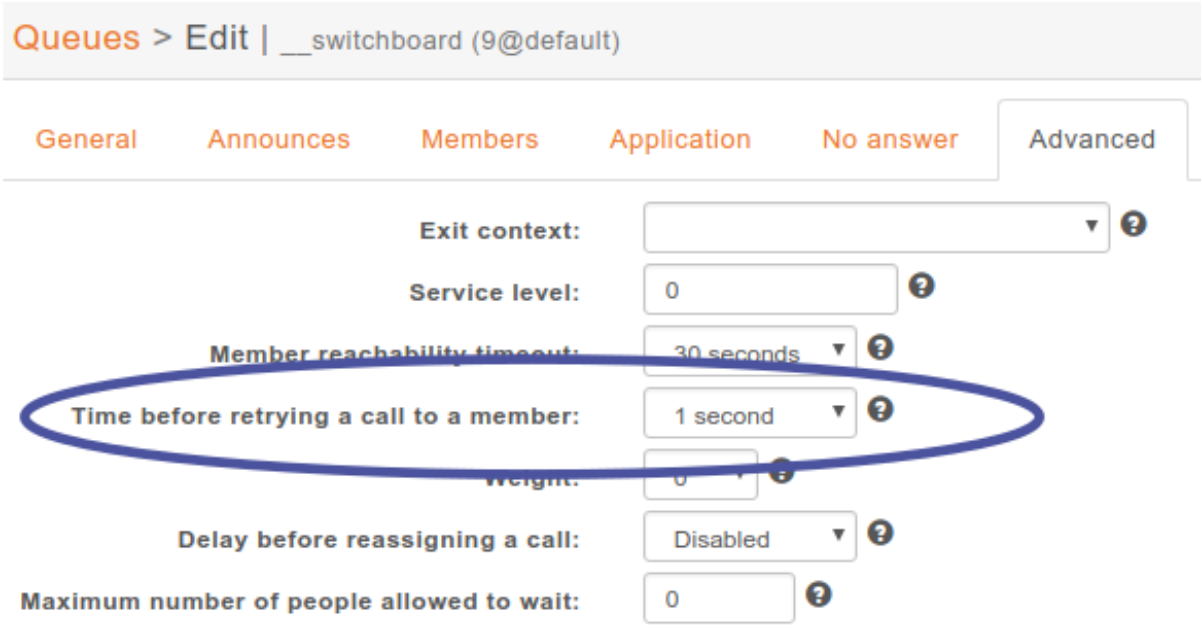
Switchboard Usage

Warning: The *switchboard configuration* must be completed before using the switchboard. This includes :

- Device, User, Agent and Queues configuration (see above),
- Directory xlet configuration (see *Directory Xlet*)


Incoming calls > Add
General
Call permissions
Schedules
DID:
444
Context:
Incalls (from-extern)
Destination :
Queue
Redirect to :
__switchboard (9@default)
Ring time :
CallerID mode :
Preprocess subroutine :
Description :
SAVE


Queues > Edit | __switchboard (9@default)
General
Announces
Members
Application
No
Ringing time:
30
Timeout priority:
Configuration
Data quality:
Allow callee to hang up the call:
Allow caller to hang up the call:
No retry when time has elapsed:
Ring instead of On-Hold Music:
Allow callee to transfer the call:
Allow caller to transfer the call:
Allow callee to record the call:
Allow caller to record the call:
Ignore call forward requests from members:
SAVE



If it's not the case, the user must disconnect his XiVO client and reconnect.

Be sure to read the *limitations* before using the switchboard.

The XiVO Client Switchboard Profile

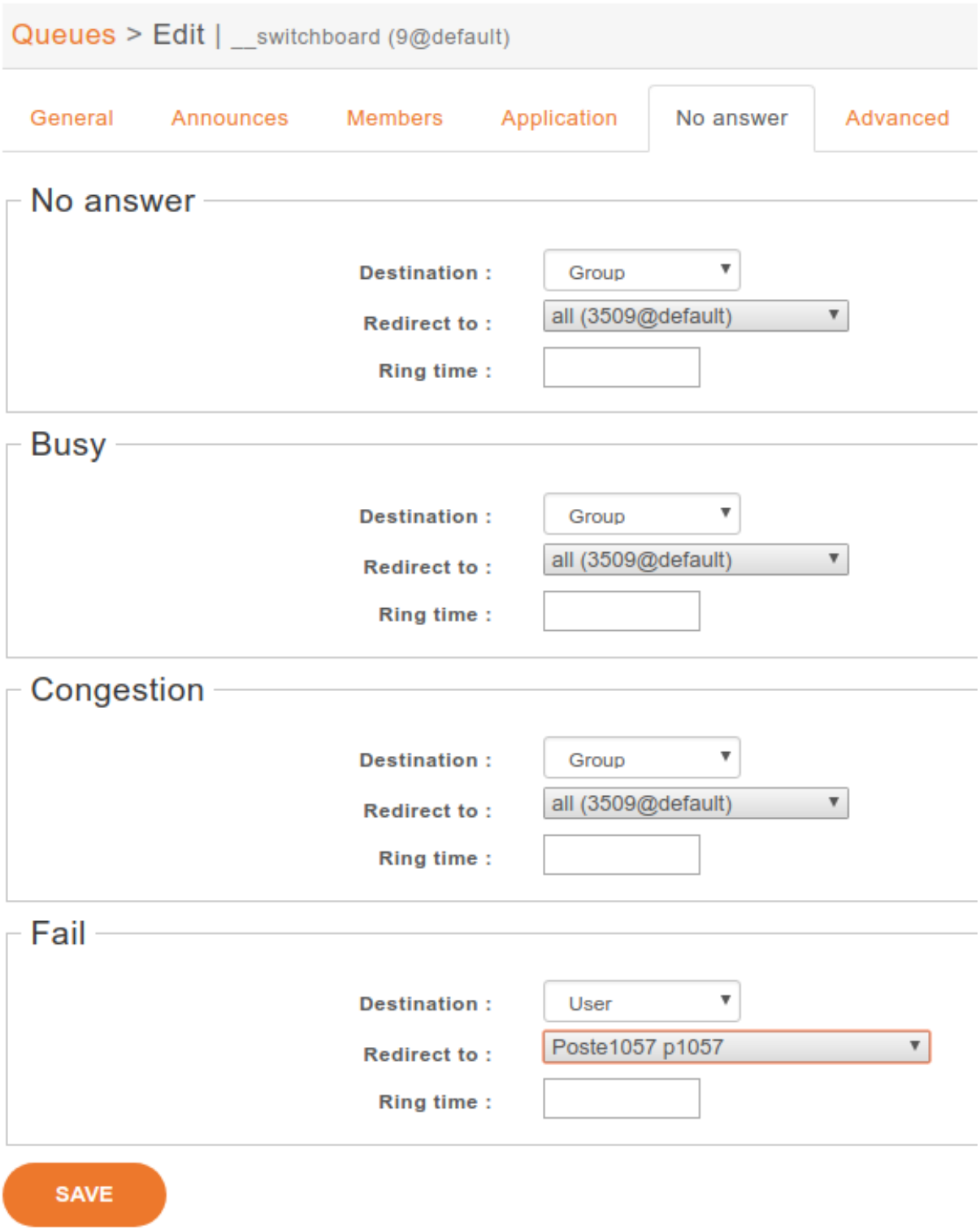
When the user connects with his XiVO Client, he gets the Switchboard profile.

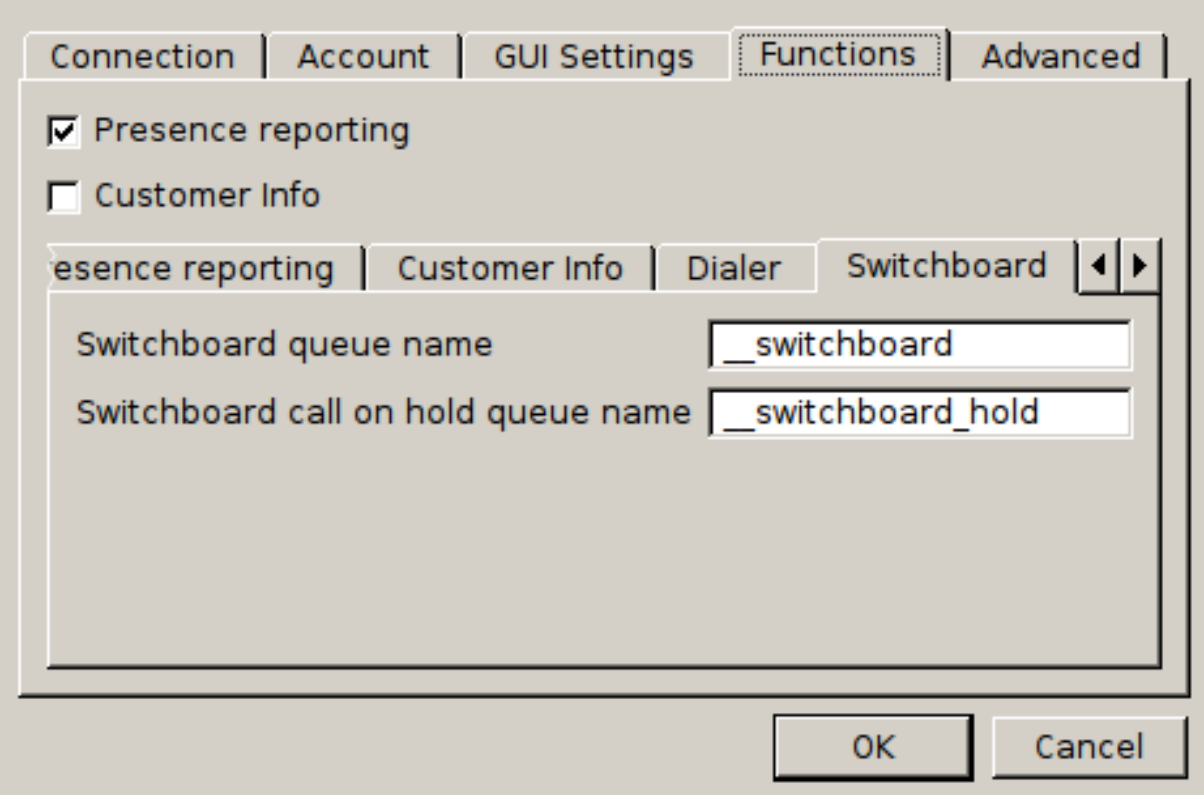
1. *Current Call* frame
2. *Answer* button
3. *Call* button
4. *Blind transfer* button
5. *Attended transfer* button
6. *Hold* button
7. *Hangup* button
8. *Incoming Calls* list
9. *Waiting Calls* list
10. *Directory* Xlet
11. *Dial* Xlet

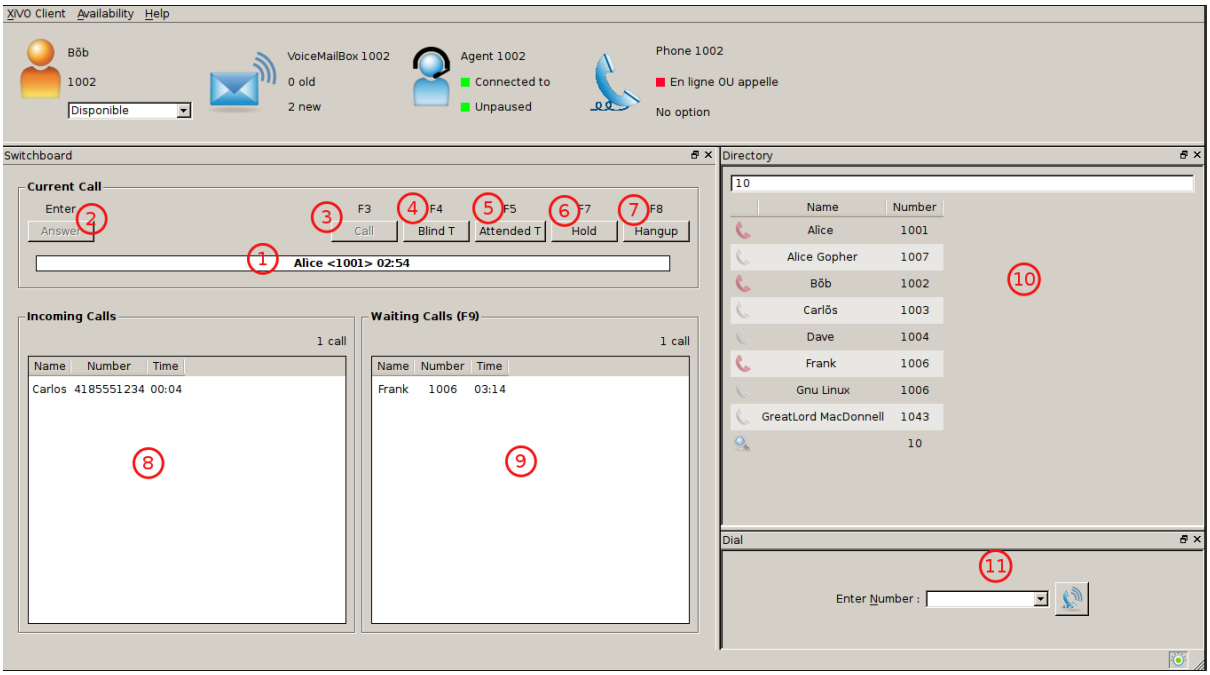
Note: If you don't see the Switchboard Xlet, right-click on the grey bar at the right of the *Help* menu and check *Switchboard*:

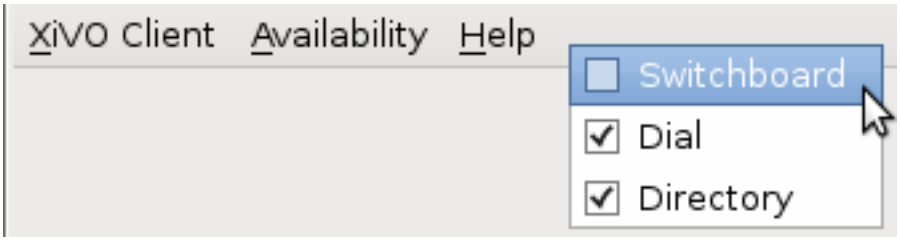
The operator can login his agent using a function key or an extension to start receiving calls.

Call flow


Queues > Edit | __switchboard (9@default)
General
Announces
Members
Application
No answer
Advanced
No answer
Destination :
Group
Redirect to :
all (3509@default)
Ring time :
Busy
Destination :
Group
Redirect to :
all (3509@default)
Ring time :
Congestion
Destination :
Group
Redirect to :
all (3509@default)
Ring time :
Fail
Destination :
User
Redirect to :
Poste1057 p1057
Ring time :
SAVE


Connection
Account
GUI Settings
Functions
Advanced
Presence reporting
Customer Info
esence reporting
Customer Info
Dialer
Switchboard
Switchboard queue name
__switchboard
Switchboard call on hold queue name
__switchboard_hold
OK
Cancel


XiVO Client Availability Help
Bõb
1002
Disponible
VoiceMailBox 1002
0 old
2 new
Agent 1002
Connected to
Unpaused
Phone 1002
En ligne OU appelle
No option
Switchboard
Current Call
Enter
Answer
F3
Call
F4
Blind T
F5
Attended T
F7
Hold
F8
Hangup
Alice <1001> 02:54
Incoming Calls
1 call
Name Number Time
Carlos 4185551234 00:04
Waiting Calls (F9)
1 call
Name Number Time
Frank 1006 03:14
Directory
Name Number
Alice 1001
Alice Gopher 1007
Bõb 1002
Carlõs 1003
Dave 1004
Frank 1006
Gnu Linux 1006
GreatLord MacDonnell 1043
10
Dial
Enter Number :


XiVO Client
Availability
Help
Switchboard
Dial
Directory

Answering an incoming call

When the switchboard receives a call, the new call is added to the *Incoming Calls* list on the left and the phone starts ringing. The user can answer this call by:

- clicking on any call in the list
- clicking the *Answer* button
- pressing the *Enter* key

Note: The XiVO Client must be the active window for the keyboard shortcuts to be handled

---

The operator can select which call to answer by:

- clicking directly on the incoming call
- pressing *F6* to select the incoming calls frame and pressing the up and down arrow keys

Selecting a call to answer while talking will not answer the call.

Once the call has been answered, it is removed from the incoming calls list and displayed in the *Current Call* frame.

Making a Call

The switchboard operator can do the following operations:

- Press the *Call* button or press *F3*
- Search for the call destination in the directory xlet
- Press to confirm the selection and start the call

Hanging Up a Call

The switchboard operator can hang up its current call by either:

- Clicking the *Hangup* button
- Pressing the *F8* key

If the operator has placed a new call via the *Directory* or *Dial* xlet and that call has not yet been answered, he can cancel it in the same way.

Distributing a call

Once the call has been answered and placed in the current call frame, the operator has 3 choices:

- transfer the call to another user
  - using the *Blind transfer* button or the *F4* key.
  - using the *Attended transfer* button or the *F5* key
- put the call on hold using the *Hold* button or the *F7* key
- end the call using the *Hangup* button or the *F8* key.

Transferring a call

Transfer buttons allow the operator to select towards which destination he wishes to transfer the call. This is made through the *Directory* xlet. For defails about the xlet *Directory* usage and configuration see *Directory Xlet*.

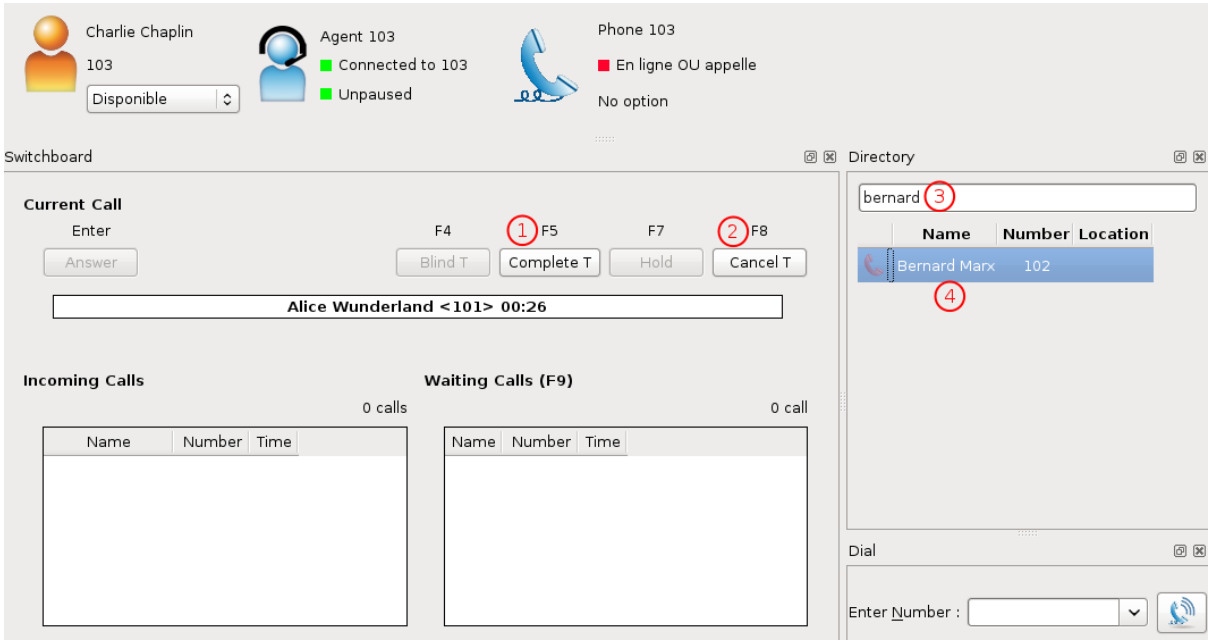
Once the destination name has been entered, press *Enter*. If multiple destinations are displayed, you can choose by:

- double-clicking on the destination
- using *Up*/*Down* arrows then:
  - pressing *Enter*
  - pressing the transfer button again

Blind transfers are straightforward: once the call is transferred, the operator is free to manage other calls.

Attended transfers are a bit more complicated: the operator needs to wait for the transfer destination to answer before completing the transfer.

In this example, the operator is currently asking *Bernard Marx* if he can transfer *Alice Wonderland* to him.



1. *Complete transfer* button
2. *Cancel transfer* button
3. Transfer destination filtering field (xlet *Directory*)
4. Transfer destination list (xlet *Directory*)

Once the destination has answered, you can:

- cancel the transfer with *F8* key
- complete the transfer with *F5* key

Note: The operator can not complete an attended transfer while the transfer destination is ringing. In this case, the operator must cancel the attended transfer and use the *Blind transfer* action.

Putting a call on hold

If the user places the call on hold, it will be removed from the *Current call* frame and displayed in the *Waiting calls* list. The time counter shows how long the call has been waiting, thus it will be reset each time the call returns in the *Waiting calls* list. The calls are ordered from the oldest to the newest.

Retrieving a call on hold

Once a call has been placed on hold, the operator will most certainly want to retrieve that call later to distribute it to another destination.

To retrieve a call on hold:

- click the desired call in the *Waiting calls* list
- with the keyboard:
  - move the focus to the *Waiting calls* list (*F9* key)
  - choose the desired call with the arrow keys
  - press the *Enter* key.

Once a call has been retrieved from the *Waiting calls* list, it is moved back into the *Current Call* frame, ready to be distributed.

Switchboard Statistics

Limitations

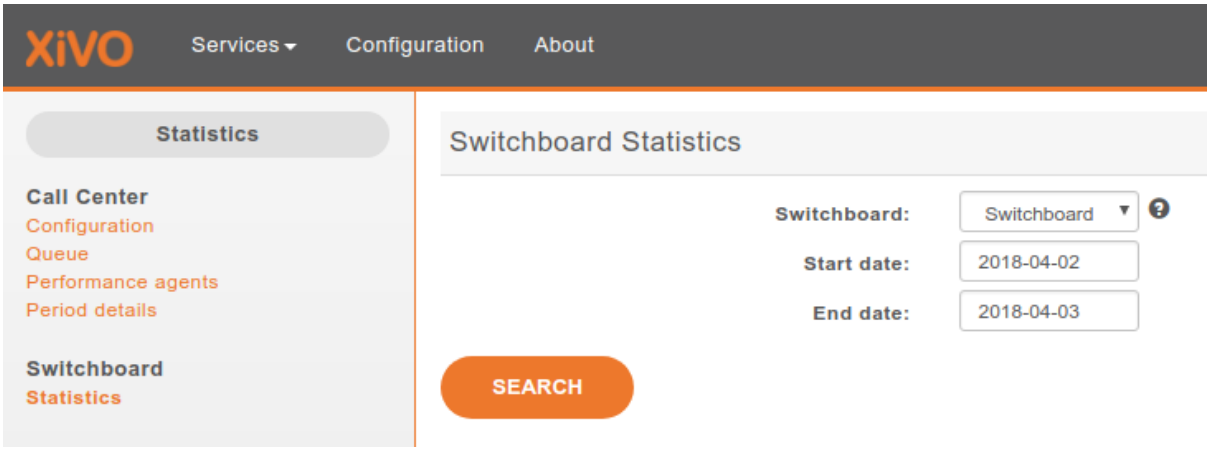
Note: Statistics are produced by *xivo-ctid*. If a call is received when *xivo-ctid* is stopped, no statistics will be produced for that call.

Note: Statistics are only generated for calls answered in XiVO 16.03.

Note: Statistics are only available for existing switchboard queues, i.e. deleting a queue will also delete the associated statistics.

Daily Statistics

Switchboard statistics can be retrieved in CSV format via the web interface in *Services* → *Statistics* → *Switchboard* → *Statistics*.



- Start date: when empty, the result will contain statistics from the beginning
- End date: when empty, the result will contain statistics until the current time

Note: Switchboard statistics older than a year are automatically removed. See *Purge Logs* for more details.

Report

The generated CSV report includes the following columns:

- date: The date at which the calls were received
- entered: The number of calls to the switchboard for the given date excluding calls when the switchboard was closed (e.g. with a *schedule*)
- answered: The number of calls that have been answered by the operator and then transferred or completed by the operator
- transferred: The number of calls that have been transferred by the switchboard operator to another destination
- abandoned: The number of calls that have been abandoned in the switchboard queue or while waiting in the hold queue
- forwarded: The number of calls that have been forwarded to another destination:
  - a call reaching a full queue
  - a call waiting until the max ring time is reached
  - a call forwarded because of a diversion rule
  - a call forwarded because of a leave empty condition
- waiting_time_average: The average time spent in the switchboard and hold queue for all calls that entered the switchboard

Events

Switchboard statistic events are published on the bus to be consumed by collectd.

Configuration

In order to process these events, you need:

- collectd installed on your XiVO: `apt-get install collectd`

- In `/etc/collectd/collectd.conf.d/amqp.conf`, configure collectd to read events from the bus (RabbitMQ):

```
LoadPlugin amqp

<Plugin "amqp">
  <Subscribe "xivo">
    Host "127.0.0.1"
    Port "5672"
    VHost "/"
    User "guest"
    Password "guest"
    Exchange "collectd"
    ExchangeType "topic"
    RoutingKey "collectd.#"
  </Subscribe>
</Plugin>
```

- another service receiving events from collectd, e.g. logstash, graphite, another collectd.

Event description

The collectd events have the following attributes:

- host: the UUID of the XiVO.
- plugin: `switchboard`
- plugin_instance: the name (not the display name) of the queue for incoming calls of the switchboard.
- type: `counter` or `gauge`.
- type_instance: the following values.

entered This event is produced when a call enters the switchboard on an open schedule. Calls that did not enter the queue, if the queue was full for example, will also generate an entered event.

abandoned This event is produced when the called hangs up while waiting in the incoming queue or in the hold queue.

transferred This event is produced when a call is transferred from the switchboard by the operator. For attended transfers, the event is sent when the transfer is completed.

forwarded This event is produced when a call is redirected to another destination under certain conditions. This include:

- When the queue is full
- When the queue timeout is reached
- When no agent are logged with a join empty configuration
- When a divertion occured

completed This event is produced when a call was answered by the operator without being transferred to another destination.

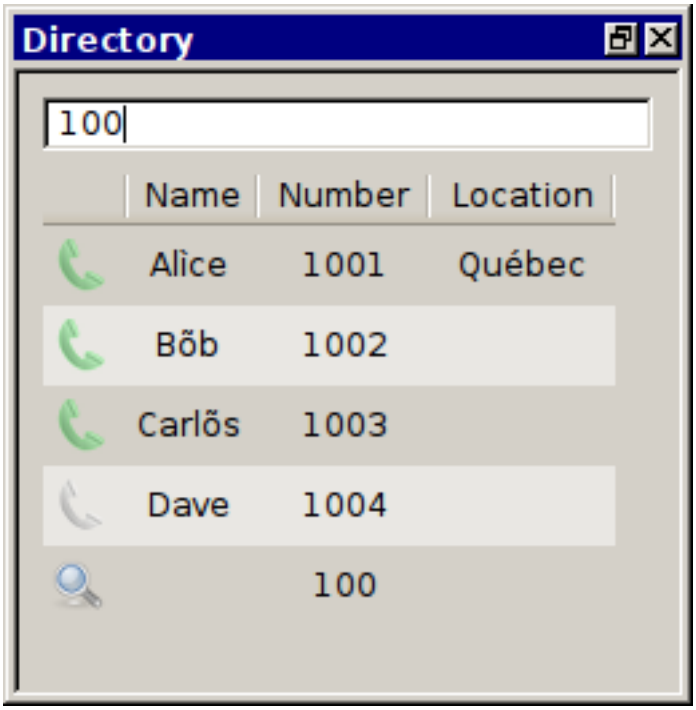
wait_time This event is produced when a call is completed, its value is the sum of all times spent in the hold queue and the time spent in the incoming queue before being answered.

Directory Xlet

Overview

Warning: This xlet should only be used with a Switchboard profile. It is not meant to be used alone.

The goal of the directory xlet is to allow the user to search through XiVO users, directory entries and arbitrary numbers to be able to call and transfer calls to these destinations.



Usage

The list of entries in the xlet is searched using the top field. Entries are filtered by column content. The entry list will initially appear as empty.

If the current search term is a valid number, it will be displayed in the result list with no name to allow transfer to numbers that are not currently in the phonebook or configured on the XiVO.

Legend

- Users available
- Users ringing
- Users talking
- Users
- Mobile phone

- External contacts
- Current search (not a contact)

Phonebook

Phonebook searches are triggered after the user has entered 3 characters. Results from remote directories will appear after 1 second.

If a directory entry as the same number as a mobile or a phone configured on the XiVO, it's extra columns will be added to the corresponding entry instead of creating a new line in the search result.

For example:

If *User 1* has number *1000* and is also in a configured LDAP with a location in "Québec", if the display filter contains the *Location* column, the entry for *User 1* will show "Québec" in the *Location* column after the search results are received.

Configuration

Context

The directory xlet needs a special context named *__switchboard_directory*. In *Services* → *IPBX* → *IPBX configuration* → *Contexts* add a new context with the following parameters :

- Name : `__switchboard_directory`
- Type of context : **Other**
- Display name : Switchboard

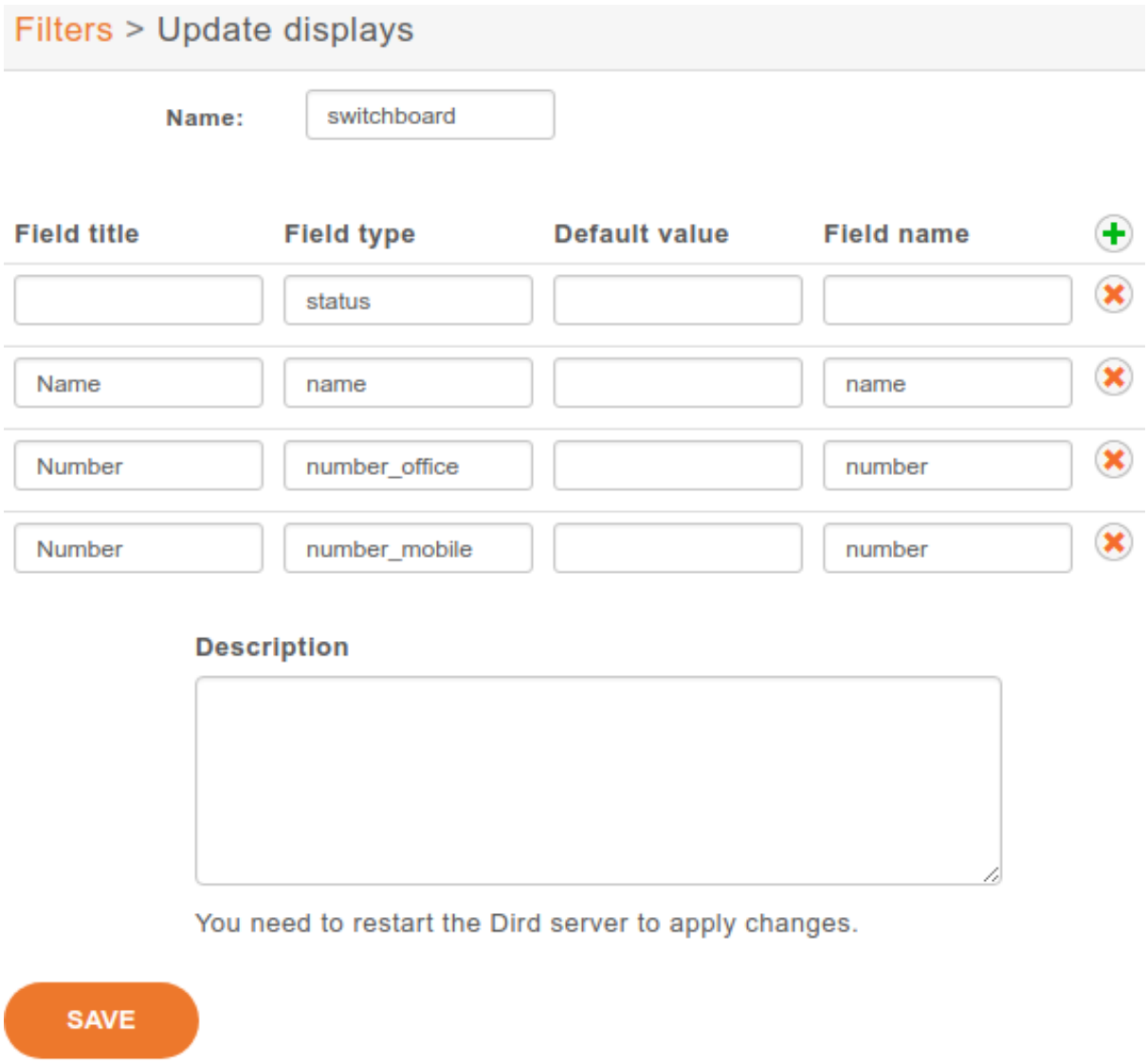
	Name	Displayed name	Context type	Entity	Action
☐	cha	cha10x	Internal	jyl-entity (jyl-entity)	✎
☐	cof	cof11x	Internal	jyl-entity (jyl-entity)	✎
☐	default	Default	Internal	jyl-entity (jyl-entity)	✎
☐	exit_with_pound	exit_with_pound	Internal	jyl-entity (jyl-entity)	✎
☐	from-extern	Incalls	Incall	jyl-entity (jyl-entity)	✎
☐	michel	michel	Internal	jyl-entity (jyl-entity)	✎
☐	routage	routage	Outcall	jyl-entity (jyl-entity)	✎
☐	__switchboard_directory	Switchboard	Other	-	✎ ✖

Display filter

A new display filter must be created for the directory xlet.

The following fields must be configured with the correct value for the *Field type* column in order for entries to be displayed in the xlet:

1. *status* is the column that will be used to display the status icon, the title can be empty
2. *name* is displayed in the *Name* column of the xlet


Filters > Update displays
Name:
switchboard
Field title
Field type
Default value
Field name
status
Name
name
name
Number
number_office
number
Number
number_mobile
number
Description
You need to restart the Dird server to apply changes.
SAVE

3. *number_office* is displayed in the *Number* column with a phone icon in the xlet
4. *number_mobile* is displayed in the *Number* column with a mobile icon in the xlet
5. *number_...* any other field starting with *number_* will be displayed in the *Number* column of the xlet with a generic directory icon
6. Any other field will be displayed in their own column of the directory xlet

The values in the *Field name* column must contain values that were created in the *Directory definition*.

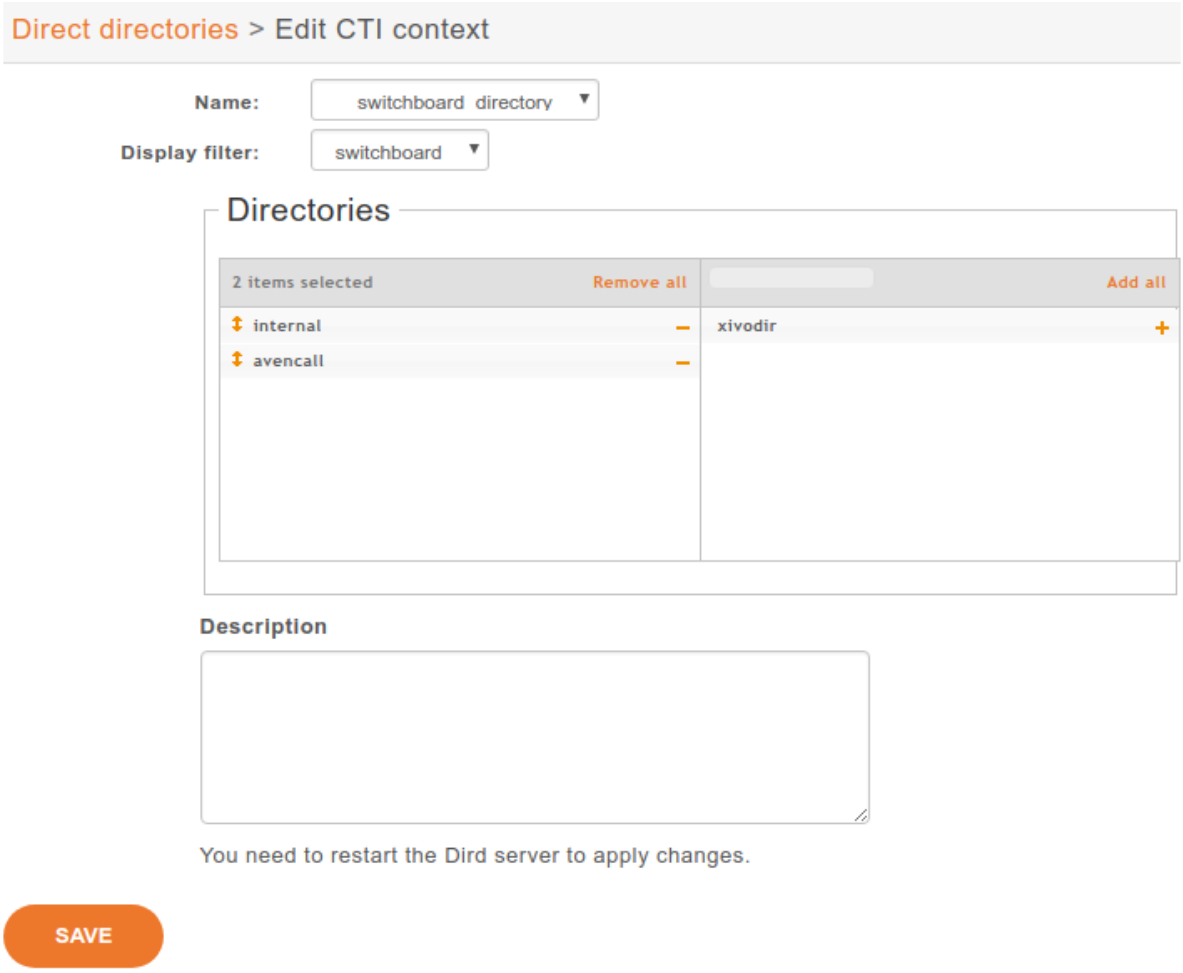
The title used for the *Number* column is the title of the first field whose type starts with *number_*.

Note: The field title of the first number column will be used for the header title in the xlet.

Warning: Make sure that the fields entered in the display format are also available in the directory definition, otherwise the filter will not return any results

Context and filter association

The new *Display filter* has to be assigned to the *__switchboard_directory* context



You can then choose which directories will be searched by the Xlet.

Warning: You must **not select internal** directory, as it is already handled.

LDAP Configuration

To search in ldap directories, you must have an LDAP server configured. See *LDAP* for more details.

LDAP filter

If you already have an LDAP filter configured for the *Remote directory* Xlet, you can use it.

If not, please refer to *Add a LDAP Filter*. configuration.rst

Include the new directory for lookup

You must use the new LDAP filter in the *Context and filter association* step.

5.30 Users

Users Configuration.

5.30.1 User Import and Export

CSV Import

Users can be imported and associated to other resources by use of a CSV file. CSV Importation can be used in situations where you need to modify many users at the same in an efficient manner, or for migrating users from one system to another. A CSV file can be created and edited by spreadsheet tools such as Excel, LibreOffice/OpenOffice Calc, etc.

CSV file

The first line of a CSV file contains a list of field names (also sometimes called "columns"). Each new line afterwards are users to import. CSV data must respect the following conditions:

- Files must be encoded in UTF-8
- Fields must be separated with a `,`
- Fields can be optionally quoted with a `"`
- Double-quotes can be escaped by writing them twice (e.g. `Robert ""Bob"" Jenkins`)
- Empty fields or headers that are not defined will be considered null.
- Fields of type *bool* must be either `0` for false, or `1` for true.
- Fields of type *int* must be a positive number

In the following tables, columns have been grouped according to their resource. Each resource is created and associated to its user when all required fields for that resource are present.

User

Field	Type	Required	Values	Description
entity_id	int	Yes		Entity ID (Defined in menu *Configuration* → *Management* → *Entities*)
firstname	string	Yes		User's firstname
lastname	string			User's lastname
email	string			User's email
language	string		de_DE, en_US, es_ES, fr_FR, fr_CA	User's language
mobile_phone_number	string			Mobile phone number
outgoing_caller_id	string			Customize outgoing caller id for this user
enabled	bool			Enable/Disable the user
supervision_enabled	bool			Enable/Disable supervision
call_transfer_enabled	bool			Enable/Disable call transfers by DTMF
dtmf_hangup_enabled	bool			Enable/Disable hangup by DTMF
simultaneous_calls	int			Number of calls a user can have on his phone simultaneously
ring_seconds	int		Must be a multiple of 5	Number of seconds a call will ring before ending
call_permission_password	string			Overwrite all passwords set in call permissions associated to the user

CTI Profile

Field	Type	Required	Values	Description
cti_profile_enabled	bool	No		Activates the XiVO Client account for this user
username	string	Yes, if profile enabled		XiVO Client username
password	string	Yes, if profile enabled		XiVO Client password
cti_profile_name	string	Yes, if profile enabled		XiVO Client profile (Defined in menu *Services* → *CTI server* → *Profiles*)

Phone

Field	Type	Required	Values	Description
exten	string	Yes		Number for calling the user. The number must be inside the range of acceptable numbers defined for the context
context	string	Yes		Context
line_protocol	string	Yes	sip, sccp	Line protocol
sip_username	string			SIP username
sip_secret	string			SIP secret

Incoming call

Field	Type	Required	Values	Description
incall_exten	string	Yes		Number for calling the user from an incoming call (i.e outside of XiVO). The number must be inside the range of acceptable numbers defined for the context.
incall_context	string	Yes		context used for calls coming from outside of XiVO
incall_ring_seconds	int			Number of seconds a call will ring before ending

Voicemail

Field	Type	Required	Values	Description
voicemail_name	string	Yes		Voicemail name
voicemail_number	string	Yes		Voicemail number
voicemail_context	string	Yes		Voicemail context
voicemail_password	string		A sequence of digits or #	Voicemail password
voicemail_email	string			Email for sending notifications of new messages
voicemail_attach_audio	bool			Enable/Disable attaching audio files to email message
voicemail_delete_messages	bool			Enable/Disable deleting message after notification is sent
voicemail_ask_password	bool			Enable/Disable password checking

Call permissions

Field	Type	Required	Values	Description
call_permissions	string		list separated by semicolons (;)	Names of the call permissions to assign to the user

Importing a file

Once your file is ready, you can import it via *Services* → *IPBX* → *IPBX settings* → *Users*. At the top of the page there is a plus button. A submenu will appear when the mouse is on top. Click on Import a file.

Examples

The following example defines 3 users who each have a phone number. The first 2 users have a SIP line, where as the last one uses SCCP:

```
entity_id,firstname,lastname,exten,context,line_protocol
1,John,Doe,1000,default,sip
1,George,Clinton,1001,default,sip
1,Bill,Bush,1002,default,sccp
```

Fig. 45: Import Users

The following example imports a user with a phone number and a voicemail:

```
entity_id,firstname,lastname,exten,context,line_protocol,voicemail_name,voicemail_
↪number,voicemail_context
1,John,Doe,1000,default,sip,Voicemail for John Doe,1000,default
```

The following exmple imports a user with both an internal and external phone number (e.g. incoming call):

```
entity_id,firstname,lastname,exten,context,line_protocol,incall_exten,incall_
↪context
1,John,Doe,1000,default,sip,2050,from-extern
```

CSV Update

The field list for an update is the same as for an import with the addition of the column uuid, which is mandatory. For each line in the CSV file, the updater goes through the following steps:

1. Find the user, using the uuid
2. For each resource (line, voicemail, extension, etc) find out if it already exists.
3. If an existing resource was found, associate it with the user. Otherwise, create it.
4. Update all remaining fields

The following restrictions must also be respected during update:

- Columns that are not included in the CSV header will not be updated.
- A field that is empty (i.e, “”) will be converted to NULL, which will unset the value.
- A line’s protocol cannot be changed (i.e you cannot go from “sip” to “sccp” or vice-versa).
- An incall cannot be updated if the user has more than one incall associated.

Updating is done through the same menu as importing (*Services → IPBX → IPBX settings → Users*). A submenu will appear when the mouse is on top. Click on *Update from file* in the submenu.

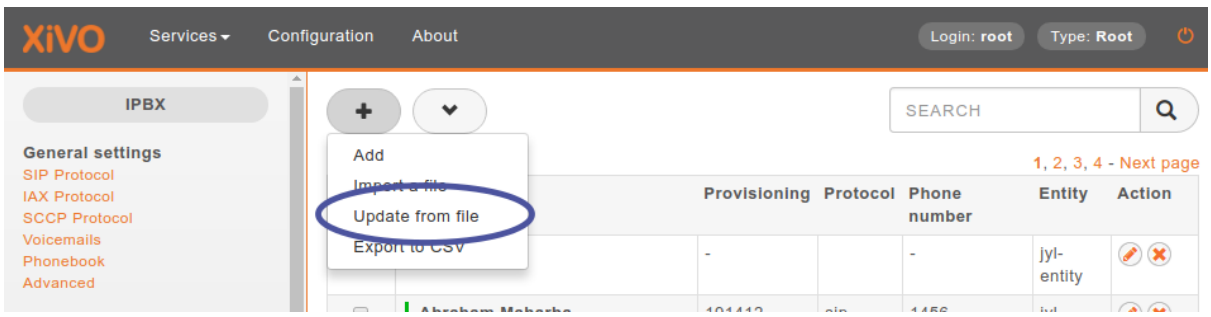


Fig. 46: *Services → IPBX → IPBX settings → Users → Update from file*

CSV Export

CSV exports can be used as a scaffold for updating users, or as a means of importing users into another system. An export will generate a CSV file with the same list of columns as an import, with the addition of uuid and provisioning_code.

Exports are done through the same menu as importing (*Services* → *IPBX* → *IPBX settings* → *Users*). Click on *Export to CSV* in the submenu. You will be asked to download a file.

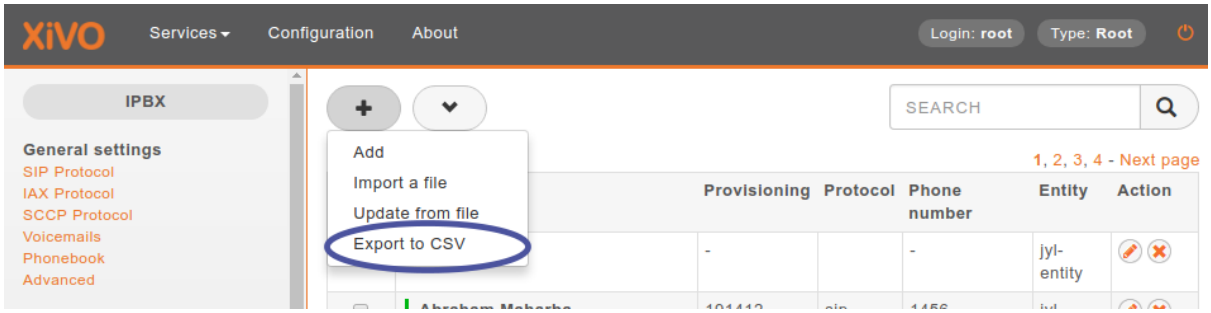
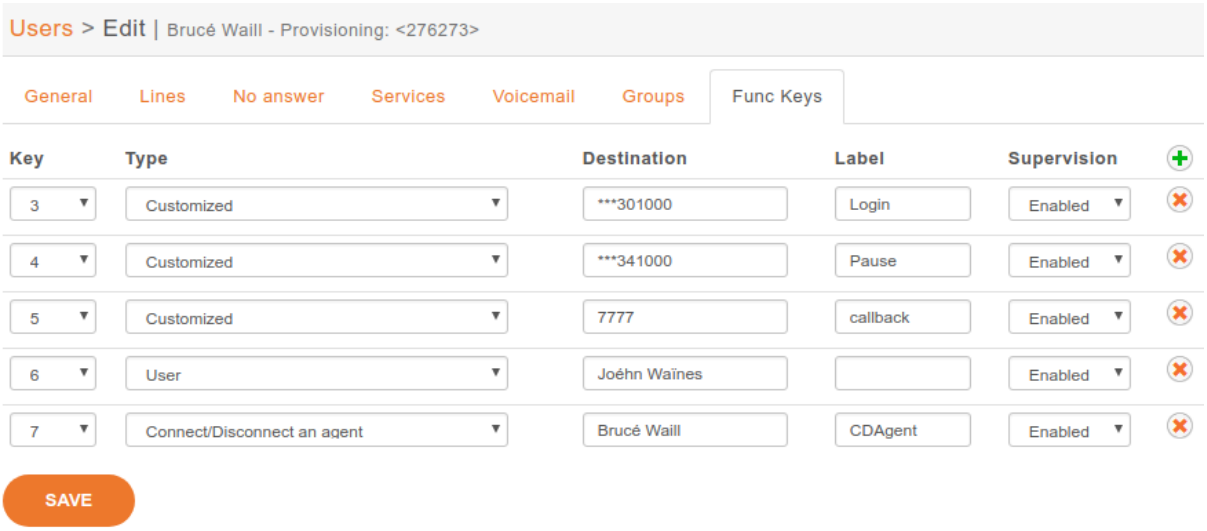


Fig. 47: *Services* → *IPBX* → *IPBX settings* → *Users* → *Export to CSV*

5.30.2 Function keys

Function keys can be configured to customize the user's phone keys. Key types are pre-defined and can be browsed through the Type drop-down list. The Supervision field allows the key to be supervised. A supervised key will light up when enabled. In most cases, a user cannot add multiple times exactly the same function key (example : two user function keys pointing to the same user). Adding the same function key multiple times can lead to undefined behavior and generally will delete one of the two function keys.

Warning: SCCP device only supports type "Customized".



For User keys, start to key in the user name in destination, XiVO will try to complete with the corresponding user.

If the forward unconditional function key is used with no destination the user will be prompted when the user presses the function key and the BLF will monitor *ALL* unconditional forward for this user.

5.30.3 Extensions

*3 (online call recording)

To enable online call recording, you must check the "Enable online call recording" box in the user form.

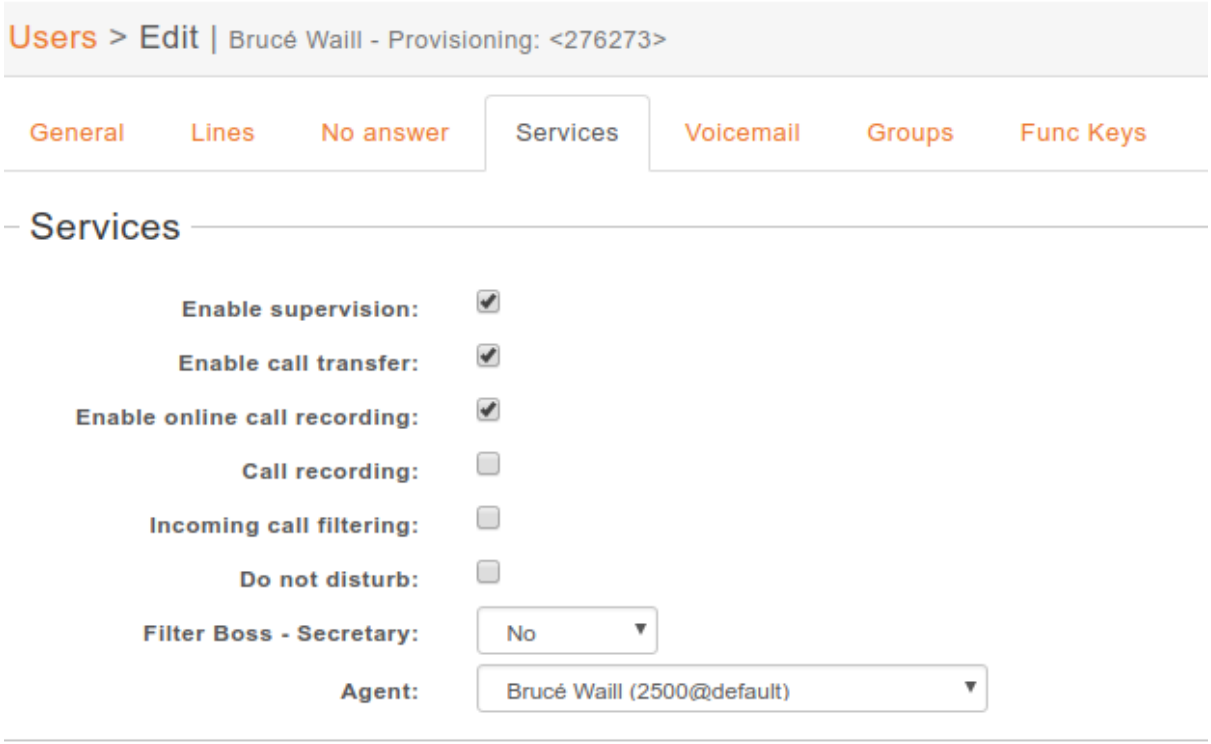


Fig. 48: Users Services

When this option is activated, the user can press `*3` during a conversation to start/stop online call recording. The recorded file will be available in the `monitor` directory of the *Services* → *IPBX* → *Audio files* menu.

*26 (call recording)

You can enable/disable the recording of all calls for a user in 2 different way:

1. By checking the "Call recording" box of the user form.
2. By using the extension **26* from your phone (the "call recording" option must be activated in *Services* → *IPBX* → *Extensions*).

When this option is activated, all calls made to or made by the user will be recorded in the `monitor` directory of the *Services* → *IPBX* → *Audio files* menu.

*55 (echo test)

To test your microphone and speaker, you can call the echo test application. The application will echo everything what you speak.

1. Dial the **55* number from your phone or application.
2. You should hear "Hello World" followed by the "Echo" announcement.
3. After the announcement, you will hear everything what you say.
4. Press # or hangup to exit the echo test application.

Using this application you may also get the latency between you and the server running the echo test.

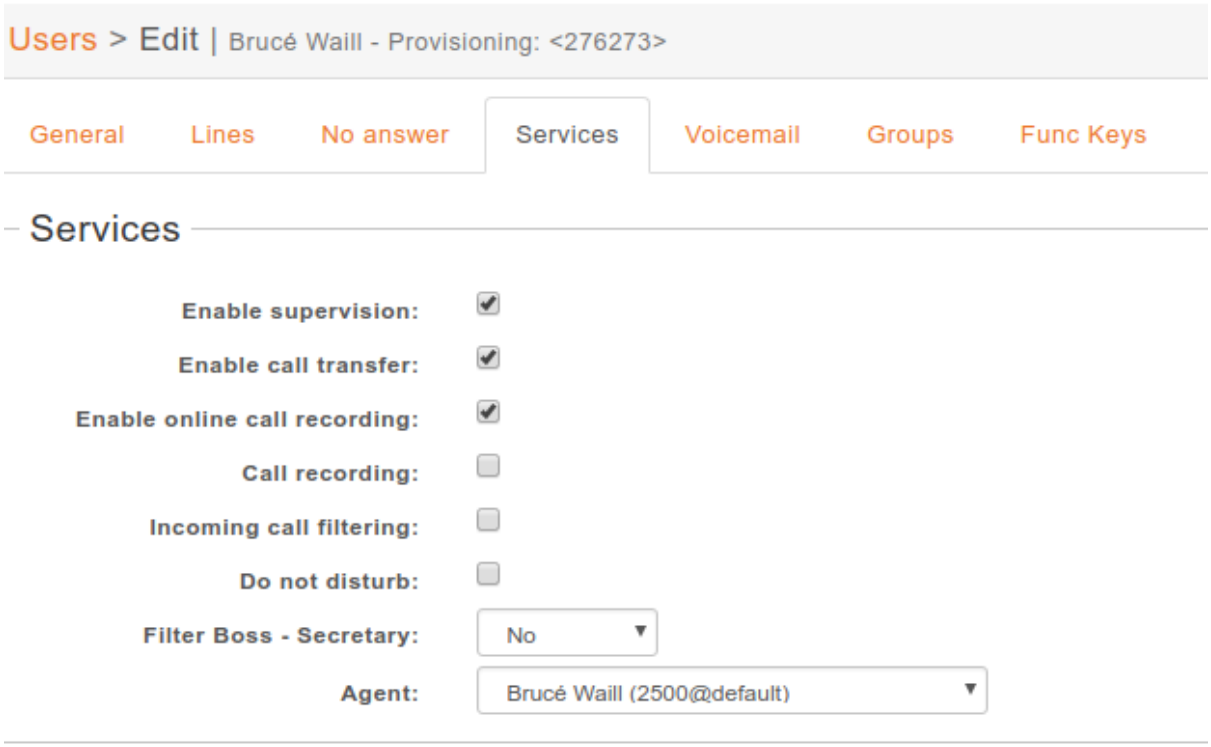


Fig. 49: Users Services

5.31 Voicemail

Voicemail Configuration.

5.31.1 General Configuration

The global voicemail configuration is located under *Services → IPBX → General Settings → Voicemails*.

5.31.2 Adding voicemails

There are 2 ways to add a voicemail:

- *Using Services → IPBX → IPBX settings → Voicemails*
- *Using the user's configuration*

Using *Services → IPBX → IPBX settings → Voicemails*

New voicemails can be added using the + button.

Once your voicemail is configured, you have to edit the user configuration and search the voicemail previously created and then associate it to your user.

Using the user's configuration

The other way is to add the voicemail from user's configuration in the 'voicemail' tab by

1. Clicking the + button
2. Filling the voicemail form

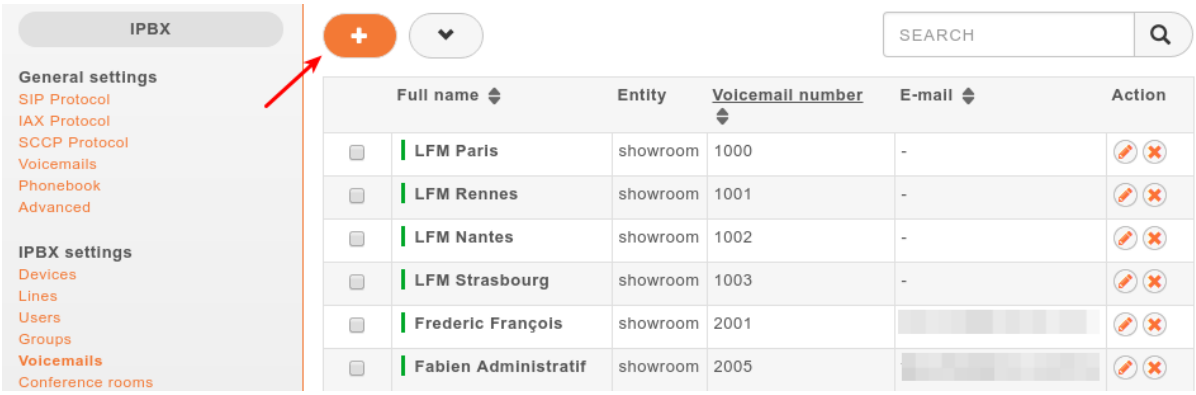


Fig. 50: Add voicemails from voicemail menu

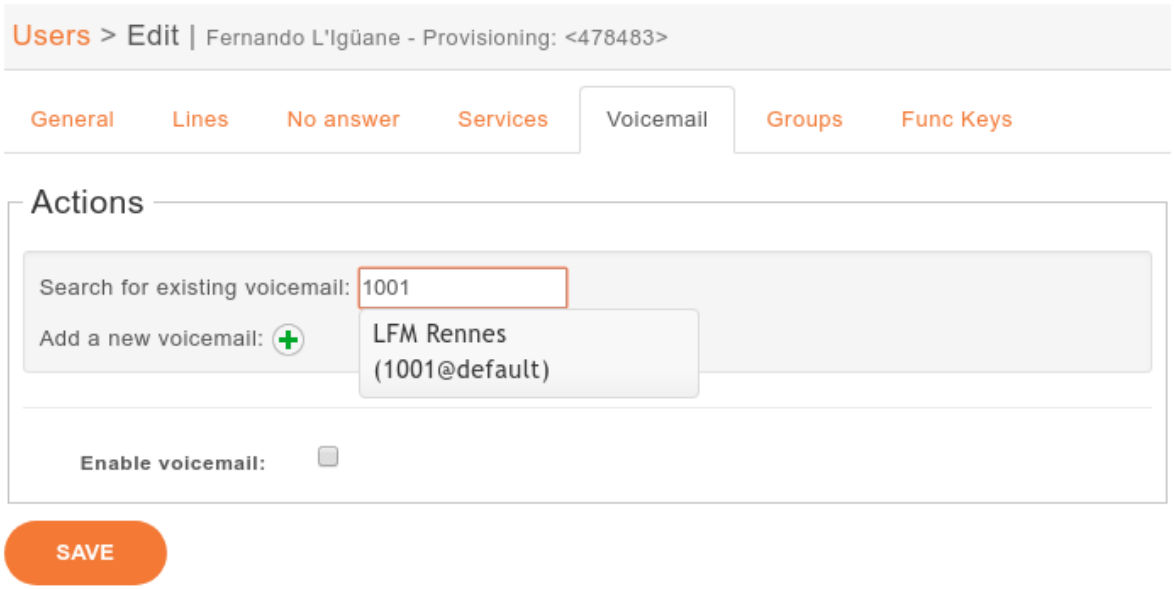


Fig. 51: Search for a voicemail in the user's configuration

3. Saving

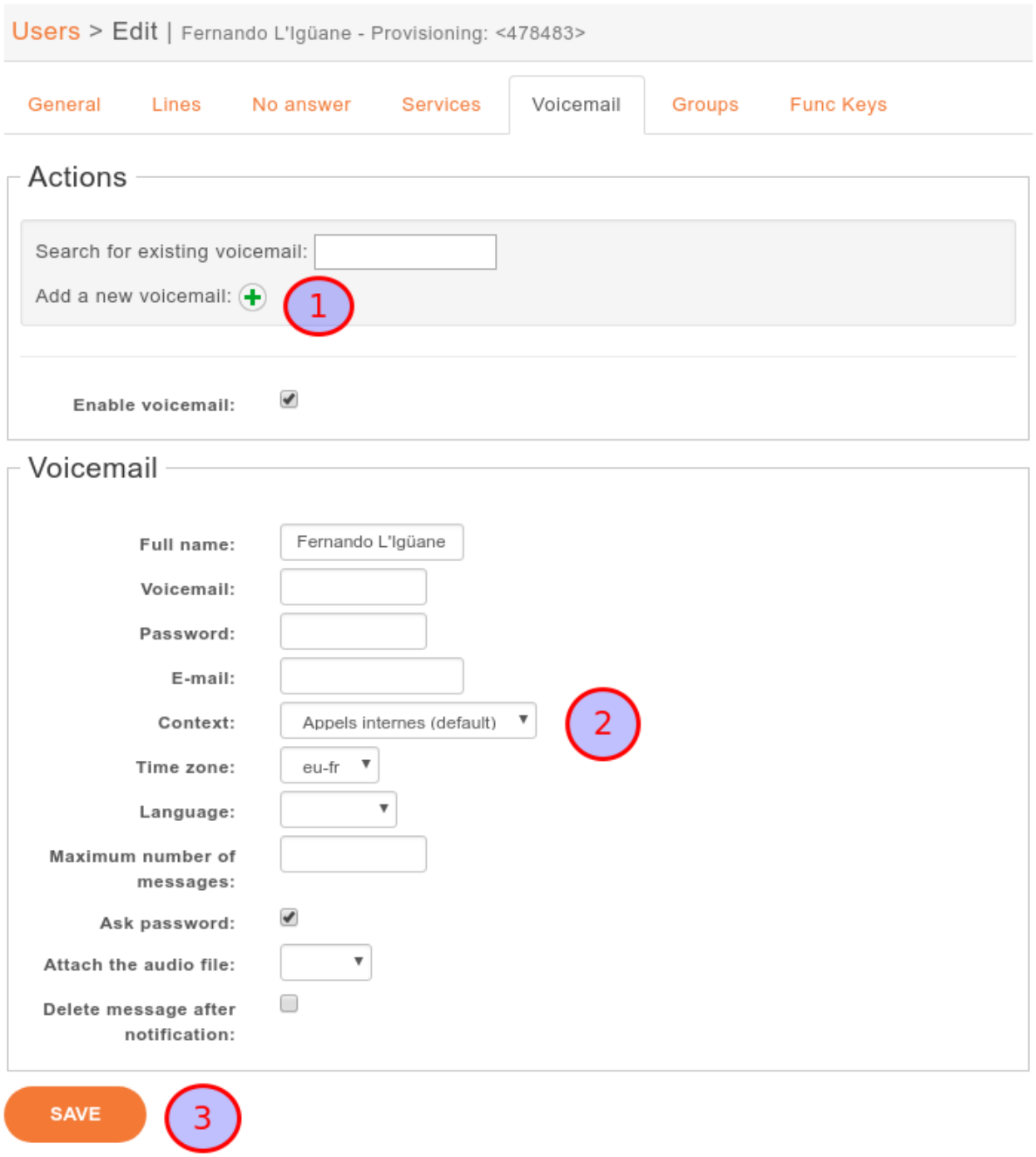


Fig. 52: Add a voicemail from the user's configuration

Note: The user's language *must* be set in the *general* tab

5.31.3 Disabling a voicemail

You can disable a user's voicemail by un-checking the 'Enable voicemail' option on the Voicemail tab from user's configuration.

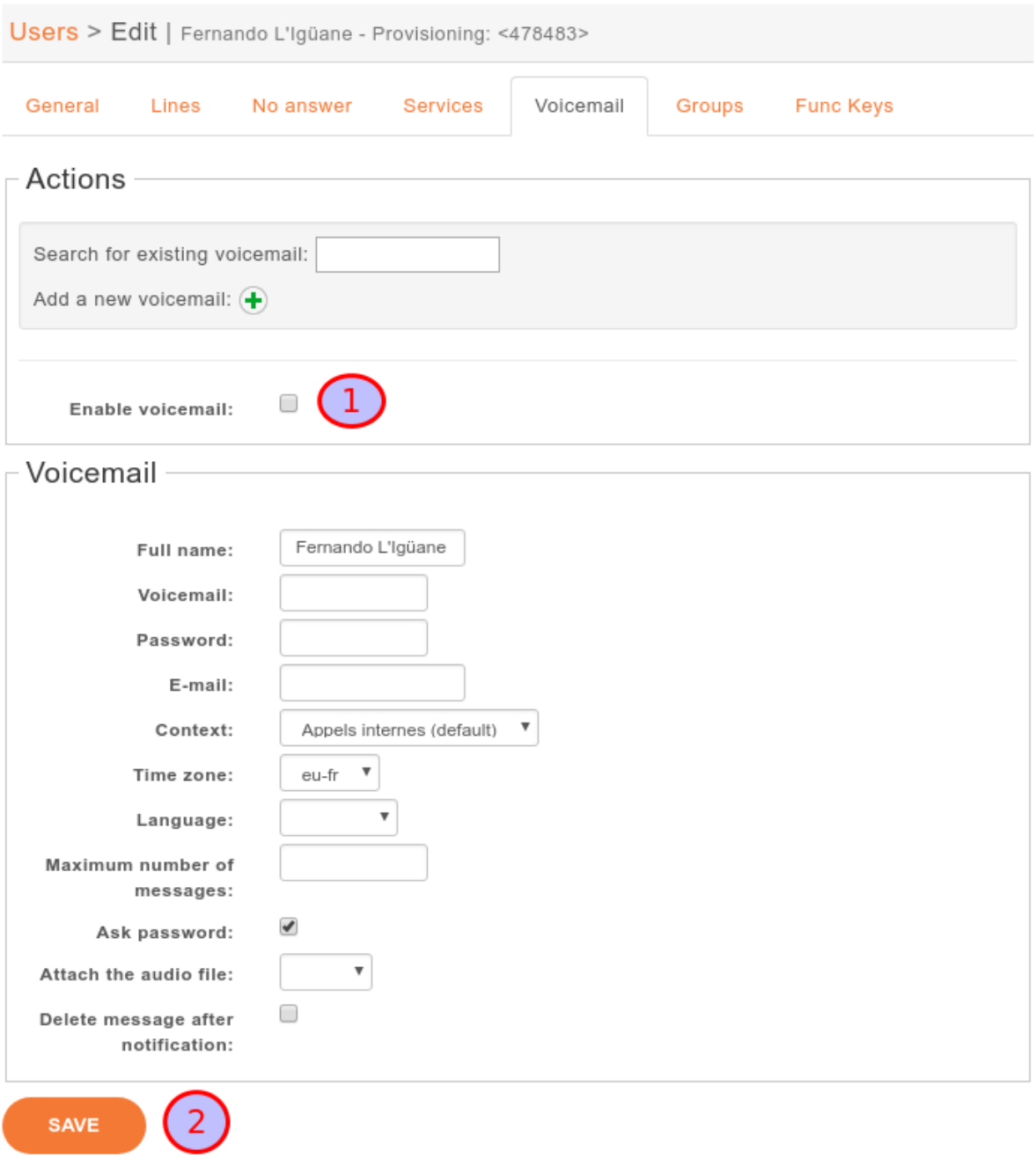


Fig. 53: Deactivate user's voicemail

5.31.4 Deleting a voicemail

Delete voicemail is done on *Services* → *IBX* → *IPBX settings* → *Voicemails* or from the user's *voicemail* tab.

Note:

- Deleting a voicemail is irreversible. It deletes all messages associated with that voicemail.
- If the voicemail contains messages, the message waiting indication on the phone will not be deactivated until the next phone reboot.

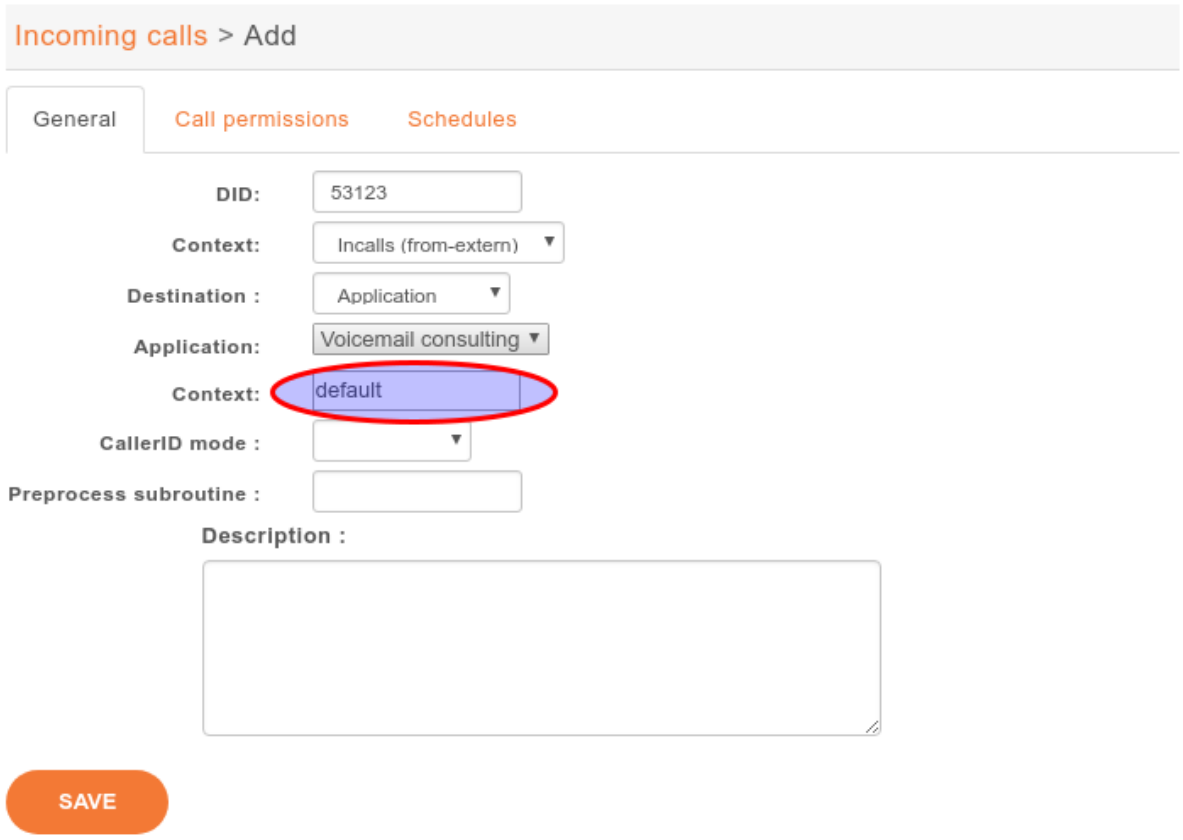
5.31.5 Disable password checking

Unchecking the option `Ask password` allows you to skip password checking for the voicemail only when it is consulted from an internal context.

- when calling the voicemail with *98
- when calling the voicemail with *99<voicemail number>

Warning: If the the *99 extension is enabled and a user does not have a password on its voicemail, anyone from the same context will be able to listen to its messages, change its password and greeting messages.

However, the password will be asked when the voicemail is consulted through an incoming call. For instance, let's consider the following incoming call:



With such a configuration, when calling this incoming call from the outside, we will be asked for:

- the voicemail number we want to consult
- the voicemail password, **even if the "Disable password checking option" is activated**

And then, we will be granted access to the voicemail.

Take note that the second "context" field contains the context of the voicemail. Voicemails of other contexts will not be accessible through this incoming call.

Warning: For security reasons, such an incoming call should be avoided if a voicemail in the given context has no password.

5.31.6 E-mail notification

E-mail message can be configured to be sent to user of the voicemail :

- configure mail server in *Configuration* → *Network* → *Mail*
- set e-mail address of the user in configuration of his voicemail

Default message can be set in *Services* → *IBX* → *General Settings* → *Voicemails* and it can be customized from tab E-mail in *Services* → *IBX* → *IPBX settings* → *Voicemails*. Letters `;` and \ must be preceded with backslash and `,` (comma) can be written only in the default message.

5.31.7 Advanced configuration

Remote *xivo-confd*

If *xivo-confd* is on a remote host, *xivo-confd-client* configuration will be required to be able to change the voicemail passwords using a phone.

This configuration should be done:

```
mkdir -p /etc/systemd/system/asterisk.service.d
cat >/etc/systemd/system/asterisk.service.d/remote-confd-voicemail.conf <<EOF
[Service]
Environment=CONFD_HOST=localhost
Environment=CONFD_PORT=9486
Environment=CONFD_HTTPS=true
Environment=CONFD_USERNAME=<username>
Environment=CONFD_PASSWORD=<password>
EOF
systemctl daemon-reload
```

5.32 WebRTC

5.32.1 General notes

Note: added in version 2016.04

XiVO comes with a WebRTC lines support, you can use in with XiVO *UC Assistant* and *Desktop Assistant*. Before starting, please check the *WebRTC Environment*.

Current WebRTC implementation requires following configuration steps:

- configure *asterisk to accept websocket connections*,

- and create user *with one line configured for WebRTC*. To have user with both SIP and WebRTC line is not supported.

5.32.2 Configuration of XiVO PBX for WebRTC

Warning: Security warning: when enabling WebRTC you need to ensure that you do it securely:

1. by securing the access to the ARI,
2. and by securing (e.g. via an external firewall) the access to the asterisk HTTP server (which listens on port 5039).

First, secure ARI connection (this step is **very important** otherwise your *XiVO PBX* won't be secure) :

- Generate a password (e.g. with `pwgen -s 16`)
- Edit file `/etc/asterisk/ari.conf`
- replace:

```
password = Nasheow8Eag
```

by:

```
password = <YOUR_GENERATED_PASSWORD>
```

Then, open asterisk HTTP server to accept outside websocket connections:

- Edit file `/etc/asterisk/http.conf`
- replace:

```
bindaddr=127.0.0.1
```

by:

```
bindaddr=0.0.0.0
```

By default, asterisk HTTP server has a limit of 100 websocket connections. You can change this limit in the `/etc/asterisk/http.conf` file:

```
sessionlimit=200
```

Restart XiVO PBX services to apply the new settings:

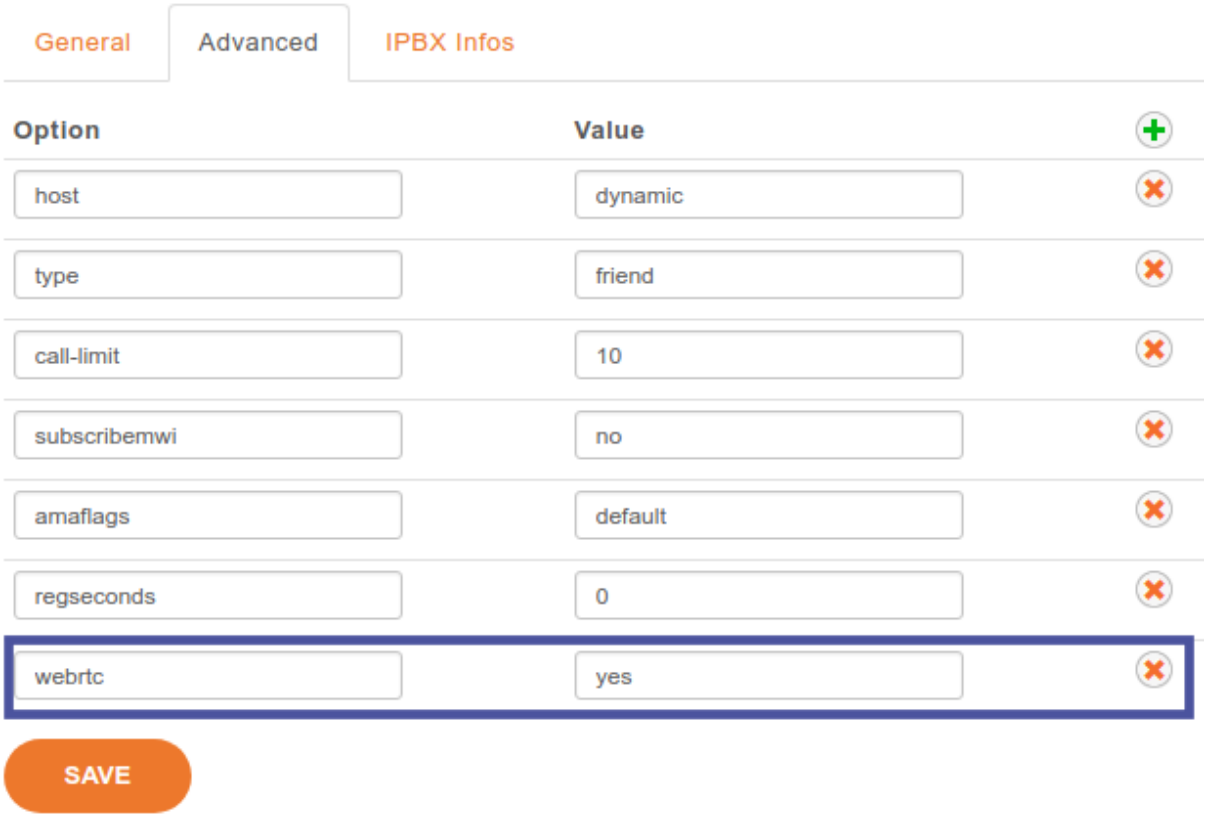
```
xivo-service restart
```

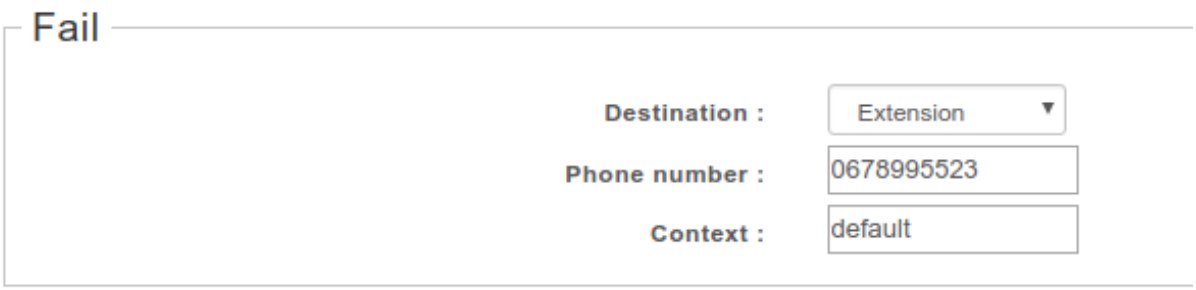
5.32.3 Configuration of user with WebRTC line

1. Create user
2. Add line to user without any device
3. Edit the line created and, in the *Advanced* tab, add *webrtc=yes* options:

5.32.4 Fallback Configuration

When the user is not connected to its WebRTC line, or disconnect from the assistant, you can route the call to a default number as for example the user mobile number. Update the fail option on the No Answer user tab configuration, and add an extension to the appropriate context.


General
Advanced
IPBX Infos
Option
Value
host
dynamic
type
friend
call-limit
10
subscribemwi
no
amaflags
default
regseconds
0
webrtc
yes
SAVE


Fail
Destination :
Extension
Phone number :
0678995523
Context :
default

5.32.5 Experimental video call feature

Note: Since 2018.02 as an experimental feature, susceptible to change or removal at any moment.

Since this feature is experimental, it is disabled by default. Video calls are enabled by setting the environment variable `ENABLE_VIDEO` to `true` in the `/etc/docker/compose/custom.env` file:

```
...
ENABLE_VIDEO=true
```

Then you need to recreate the xucmgt container with `xivocc-dcomp up -d xucmgt`.

WebRTC users are automatically configured with VP8 video codec to allow video calls, you must disable all other video codecs. Please avoid any specific codec configuration for WebRTC peers and be sure to remote all video codecs from the `SIP General settings`, tab `Signalling`, section `Codecs`. Either do not customize codecs, either do not select any video codecs.

5.32.6 Manual configuration of user with WebRTC line

For the records

WebRTC manual configuration

Note: This is the manual way to configure a WebRTC line. It is here *for the record*. You should follow the *Configuration of user with WebRTC line* instead.

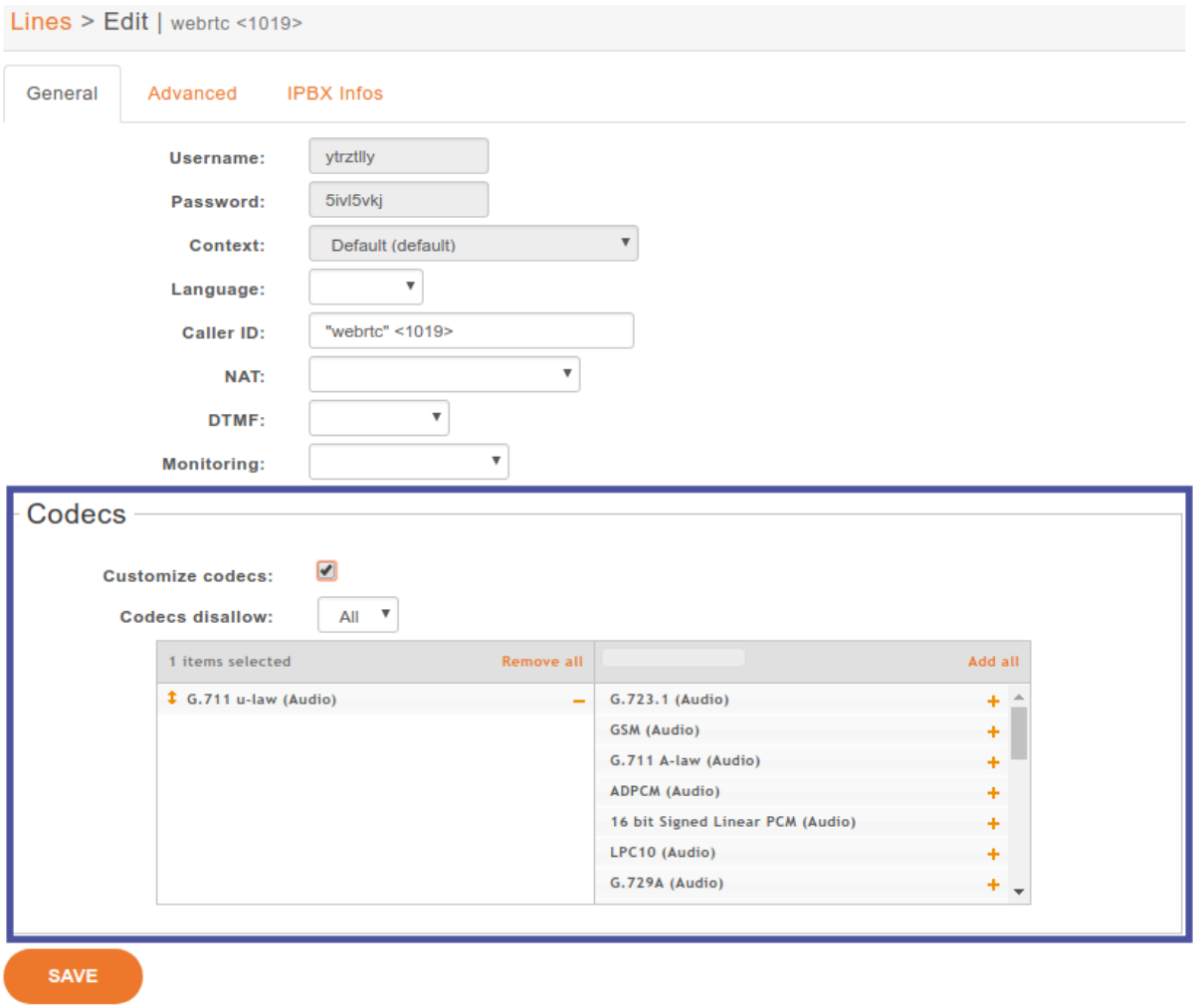
1. Create user
2. Optional: set codec to ulaw
3. Add line to user without any device
4. Configure Advanced Line options, so that it is usable with the softphone WebRTC

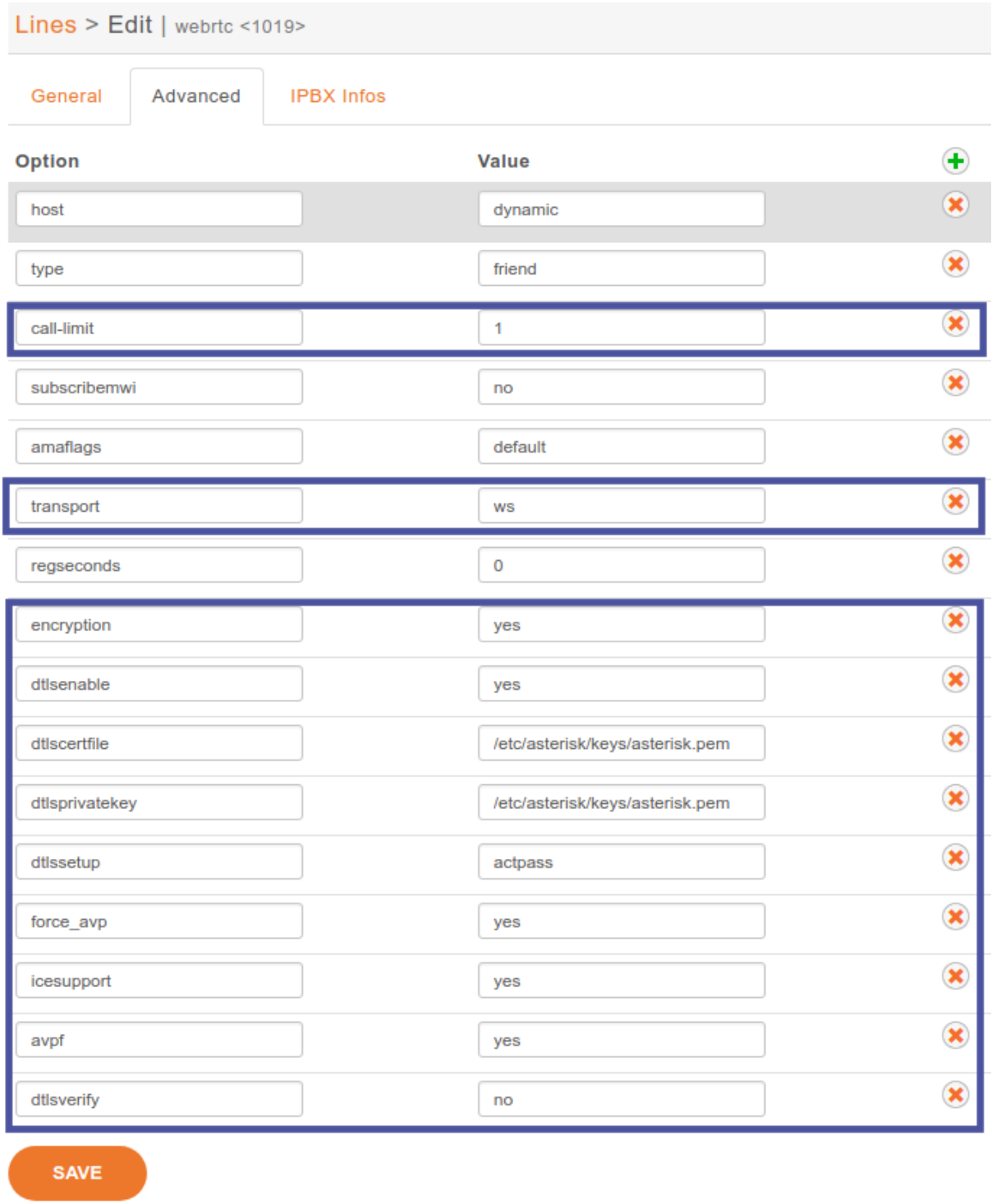
```
avpf = yes
call-limit = 1 ; use 2 from 2017.11.03
dtlsenable = yes ; Tell Asterisk to enable DTLS for this peer
dtlsverify = no ; Tell Asterisk to not verify your DTLS certs
dtlscertfile=/etc/asterisk/keys/asterisk.pem ; Tell Asterisk where your DTLS cert␣
↪file is
dtlsprivatekey = /etc/asterisk/keys/asterisk.pem ; Tell Asterisk where your DTLS␣
↪private key is
dtlssetup = actpass ; Tell Asterisk to use actpass SDP parameter when setting up␣
↪DTLS
encryption = yes
force_avp = yes
icesupport = yes
transport = ws
```

5.33 Web Services Access

You may configure Web Services / REST API permissions in *Configuration* → *Management* → *Web Services Access*.

Web services access may have two different meanings:


Lines > Edit | webrtc <1019>
General
Advanced
IPBX Infos
Username:
ytrztlly
Password:
5ivl5vkj
Context:
Default (default)
Language:
Caller ID:
"webrtc" <1019>
NAT:
DTMF:
Monitoring:
Codecs
Customize codecs:
Codecs disallow:
All
1 items selected
Remove all
Add all
G.711 u-law (Audio)
G.723.1 (Audio)
GSM (Audio)
G.711 A-law (Audio)
ADPCM (Audio)
16 bit Signed Linear PCM (Audio)
LPC10 (Audio)
G.729A (Audio)
SAVE


Lines > Edit | webrtc <1019>
General
Advanced
IPBX Infos
Option
Value
host
dynamic
type
friend
call-limit
1
subscribemwi
no
amaflags
default
transport
ws
regseconds
0
encryption
yes
dtlsenable
yes
dtlscertfile
/etc/asterisk/keys/asterisk.pem
dtlsprivatekey
/etc/asterisk/keys/asterisk.pem
dtlssetup
actpass
force_avp
yes
icesupport
yes
avpf
yes
dtlsverify
no
SAVE

- Who may access REST APIs of various XiVO daemons, and which resources in those REST APIs?
- Who may access PHP web services under `https://xivo.example.com/xivo/configuration/json.php/*`?

5.33.1 REST API access and permissions

Those REST API interfaces are documented on http://<youxivo>.api. They all require an authorization token, obtained by giving valid credentials to the REST API of xivo-auth. The relevant settings are:

- Login/Password: the xivo-auth credentials (for the xivo-auth *backend* `xivo_service`)
- ACL: The list of authorized REST API resources. See *REST API Permissions*.

Unlike PHP web services, there is no host-based authorization, so the `Host` setting is not relevant.

A few REST API access are automatically generated during the installation of XiVO, so that XiVO services may authenticate each other.

You will probably only need to create such a REST API access when you want another non-XiVO service to communicate with XiVO via REST API.

5.33.2 PHP web services

Warning: DEPRECATED

Those web services are deprecated. There is no documentation about their usage, and the goal is to remove them.

They are still protected with HTTP authentication, requiring a login and password. The relevant settings are:

- Login/Password: the HTTP authentication credentials
- Host: the authorized hosts that are allowed to make HTTP requests:
  - Empty value: HTTP authentication
  - Non-empty value: no HTTP authentication, all requests coming from this host will be accepted. Valid hosts may be: a hostname, an IP address, a CIDR block.

There is no fine-grained permissions: either the user has access to every PHP web services, or none.

5.33.3 xivo-confd

Warning: DEPRECATED

There is also a special case for authentication with xivo-confd. See *XiVO REST Api* for more details.

CHAPTER 6

Contact Center

In XiVO, the contact center is implemented to fulfill the following objectives :

- Call routing

  Includes basic call distribution using call queues and skills-based routing

- Agent and Supervisor workstation.

  Provides the ability to execute contact center actions such as: agent login, agent logout and to receive real time statistics regarding contact center status

- Statistics reporting

  Provides contact center management reporting on contact center activities

- Advanced functionalities

  Call recording

- Screen Pop-up

6.1 Agents

6.1.1 Introduction

> *A call center agent is the person who handles incoming or outgoing customer calls for a business. A call center agent might handle account inquiries, customer complaints or support issues. Other names for a call center agent include customer service representative (CSR), telephone sales or service representative (TSR), attendant, associate, operator, account executive or team member.*
>
> —SearchCRM

In this respect, agents in XiVO have no fixed line and can login from any registered device.

6.1.2 Getting Started

- Create a user with a SIP line and a provisioned device.
- Create agents.
- Create a queue adding created agent as member of queue.

6.1.3 Creating agents

Service > Call center > Agents > General

These settings are specific for a given agent.

Service > Call center > Agents > Users

These settings are specific for a given agent.

Service > Call center > Agents > Queues

These settings are specific for a given agent.

Service > Call center > Agents > Advanced

These settings are specific for a given agent.

Service > IPBX > General settings > Advanced > Agent

These settings are global for all agents.

6.1.4 Agent with external line

XiVO system agents can be external to the system, the agent can use it's personal PSTN line, mobile phone or a terminal connected to some other PBX system. We call these remote lines external line. The same agent can login to a standard line, or to an external line. The choice is done via the line number on the login page.

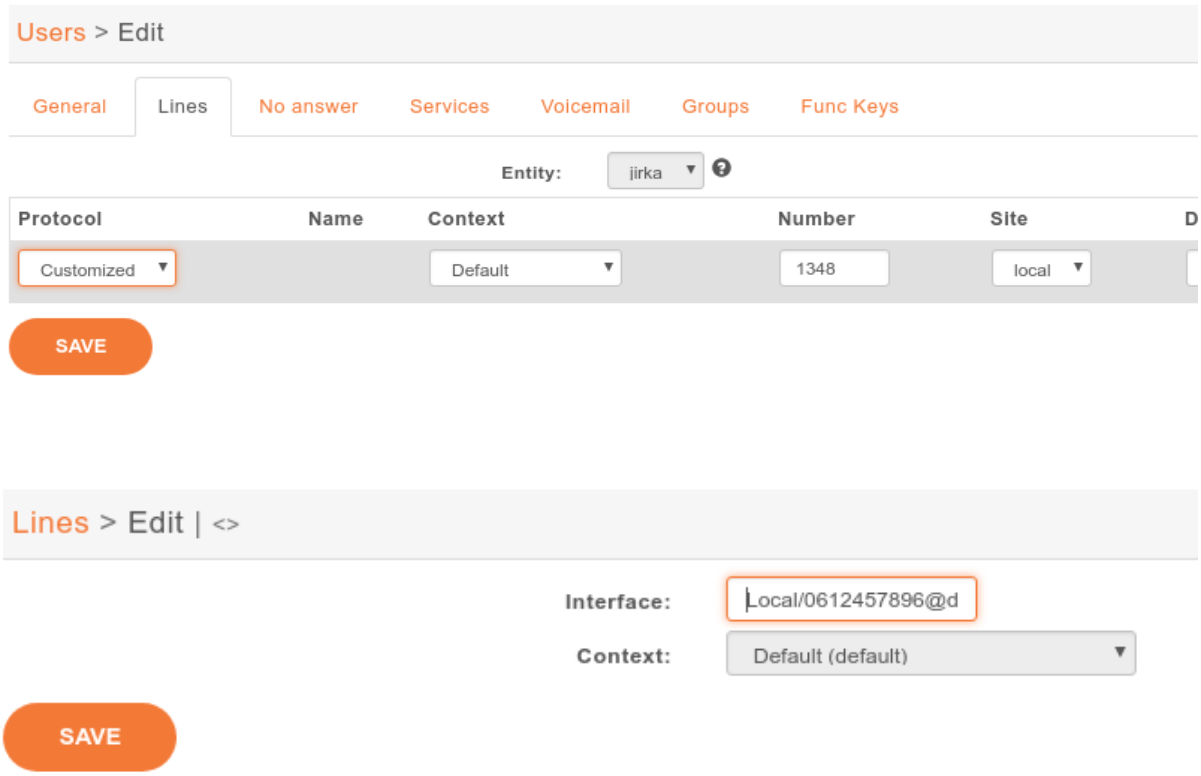
Creating agents with external line

Agent settings are the same, the only difference is in the line which is used by the agent. You must create a user with a custom line:

- Start by creating a standard user, when creating a line pick up a line number and choose Customized line protocol.
- Then save the user, go to lines listing and edit the created custom line.
- Replace the line Interface by a string with following format:

  Local/EXTERNAL_LINE_NUMBER@default/n

where EXTERNAL_LINE_NUMBER@default is the number and context which can be used to join the remote phone, for a french mobile phone it would usually be *0xxxxxxxx@default*.



Usage

Tha agent has to login using the custom line, the standard ccagent features are available. However, due to external line some phone control features are not available - the agent can't answer, put on hold or transfer calls from the ccagent interface. On the XiVO side, please ensure that the call distributed to the agent is canceled if the agent doesn't answer before the call is answered by for example the voicemail.

6.2 Queues

Call queues are used to distribute calls to the agents subscribed to the queue. Queues are managed on the *Services → Call Center → Queues* page.

A queue can be configured with the following options:

- Name: used as an unique id, cannot be `general`
- Display name: Displayed on the supervisor screen
- On-Hold music: The music the caller will hear. The music is played when waiting and when the call is on hold.

A ring strategy defines how queue members are called when a call enters the queue. A queue can use one of the following ring strategies:

- Linear: For each call, in the same order, starting from the same member
  - For agents: In login order
  - For static members: In definition order
- Least recent: call the member who least recently hung up a call
- Fewest calls: call the member with the fewest completed calls

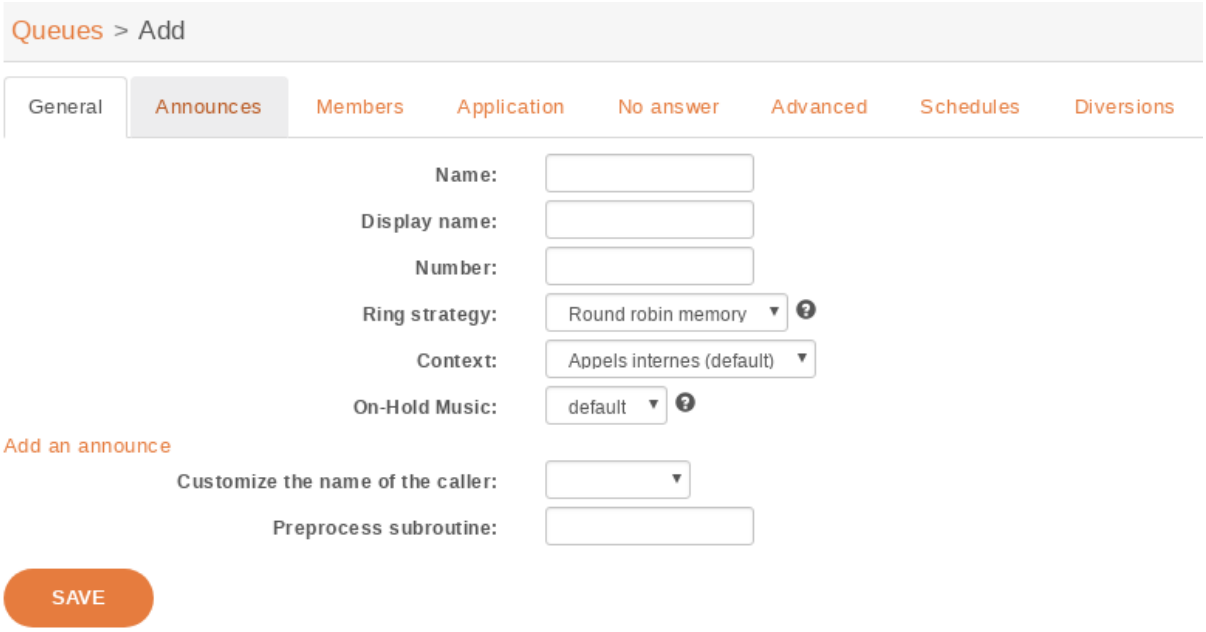


Fig. 1: *Services → Call Center → Queues → Add*

- Round robin memory: call the "next" member after the one who answered last
- Random: call a member at random
- Weight random: same as random, but taking the member penalty into account

> **Warning:** When editing a queue, you can't change the ring strategy to linear. This is due to an asterisk limitation. Unfortunately, if you want to change the ring strategy of a queue to linear, you'll have to delete it first and then create a new queue with the right strategy.

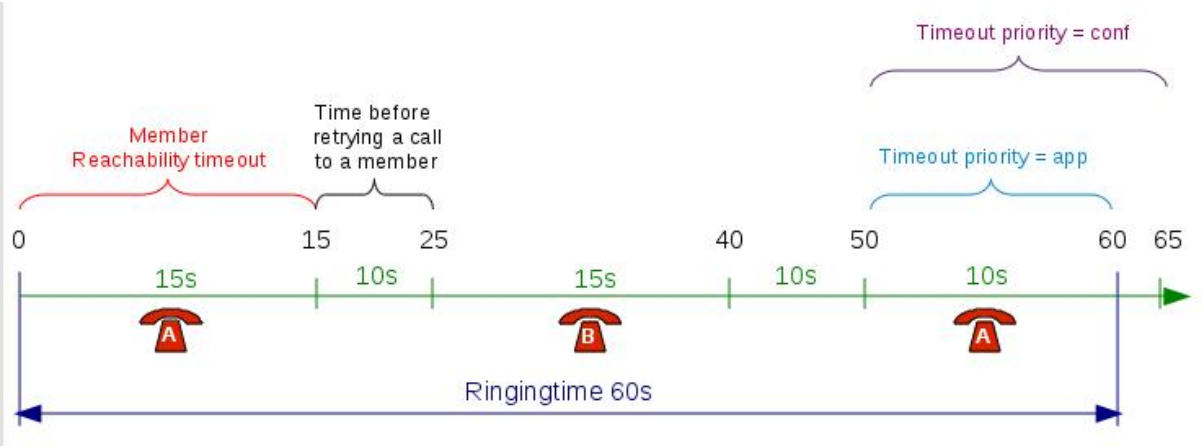
Note: When an agent is a member of many queues, configured with the **same** weight, the order of call distribution between multiple queues is nondeterministic and cannot be configured.

In order to have a deterministic behavior, you MUST *configure* different weight on each queues.

6.2.1 Timers

You may control how long a call will stay in a queue using different timers:

- Member reachabillity time out (Advanced tab): Maximum number of seconds a call will ring on an agent's phone. If a call is not answered within this time, the call will be forwarded to another agent.
- Time before retrying a call to a member (Advanced tab): Used once a call has reached the "Member reachability time out". The call will be put on hold for the number of seconds alloted before being redirected to another agent.
- Ringing time (Application tab): The total time the call will stay in the queue.
- Timeout priority (Application tab): Determines which timeout to use before ending a call. When set to "configuration", the call will use the "Member reachability time out". When set to "dialplan", the call will use the "Ringing time".



6.2.2 No Answer

Calls can be diverted on no answer:

- No answer: The call reached the "Ringing time" in Application tab and no agent answered the call
- Congestion: The number of calls waiting has reached the "Maximum number of people allowed to wait" limit specified on the advanced tab
- Fail: No agent was available to answer the call when the call entered the queue ("Join an empty queue" condition on the advanced tab) or the call was queued and no agents were available to answer ("Remove callers if there are no agents" on the advanced tab)

6.2.3 Advanced

- **Weight: Give the queue a priority to others queues (if agents belong to two or more queues). Check *weight warning* fo**

> **Warning:** When configuring a queue with a higher weight, all the calls in this queue will be prioritized over the calls of other queues if they they have the same set of members.

Here, the red call must wait end of orange and green calls. Even if the red calls ring agent first, if for some reasons agent did not answer red call, the red call will have to wait for orange and green. If some others orange or green calls come after, red call will also have to wait.

Actually, red call must wait that queue of weight 2 and queue of weight 1 be completely empty.

6.2.4 Diversions

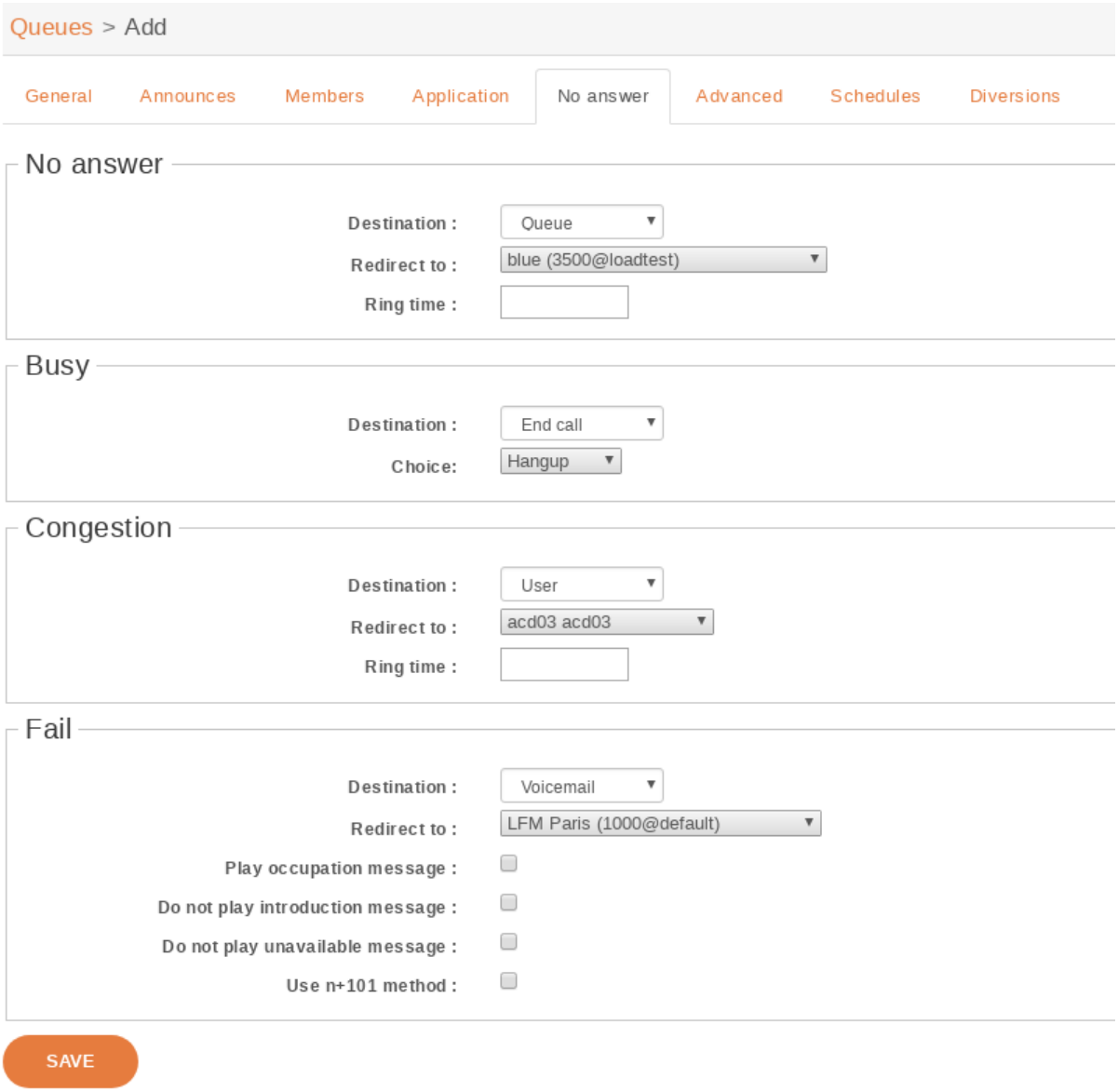
Diversions can be used to redirect calls to another destination when a queue is very busy. Calls are redirected using one of the two following scenarios:

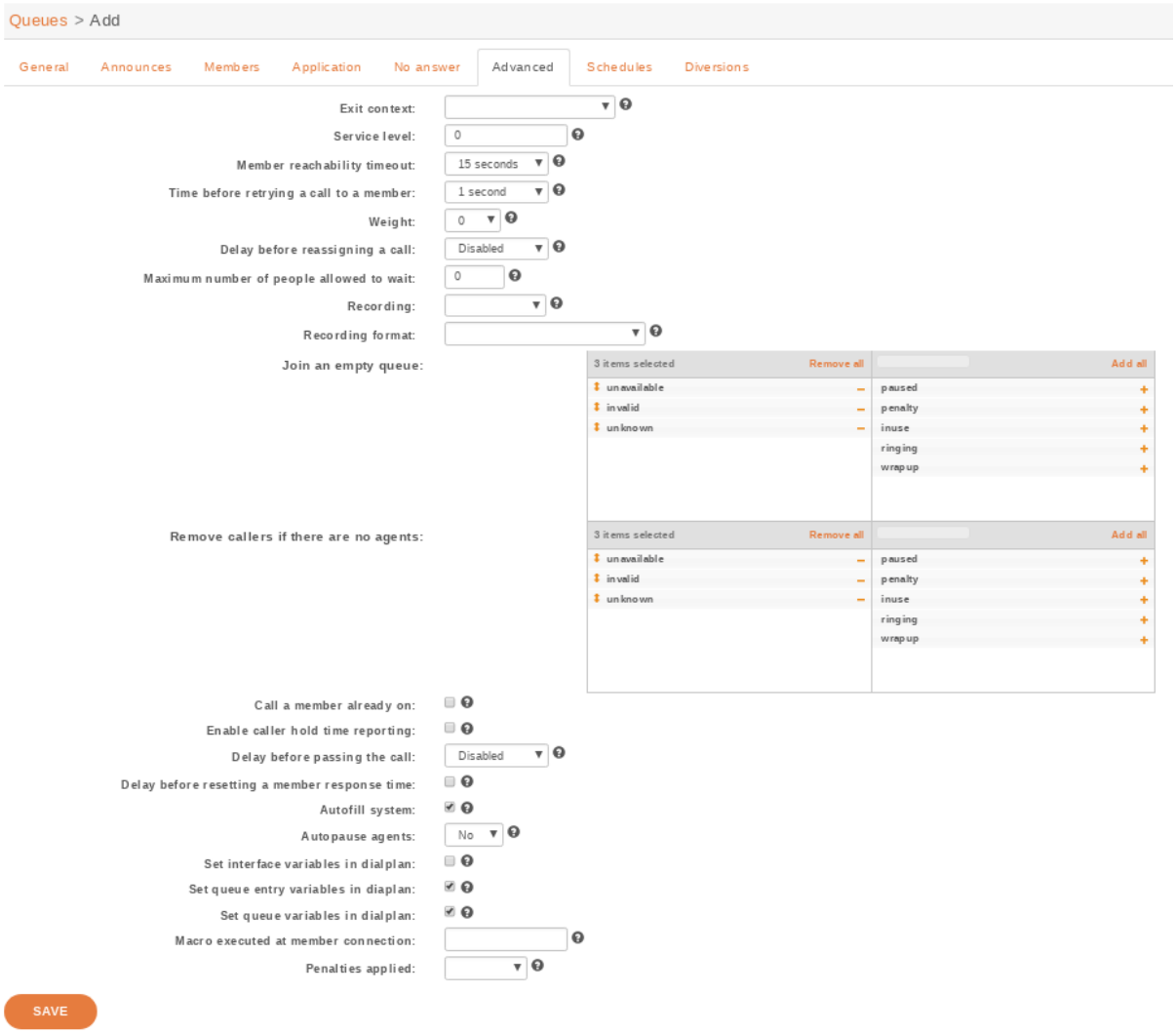
The diversion check is done only once per call, before the *preprocess subroutine* is executed and before the call enters the queue.

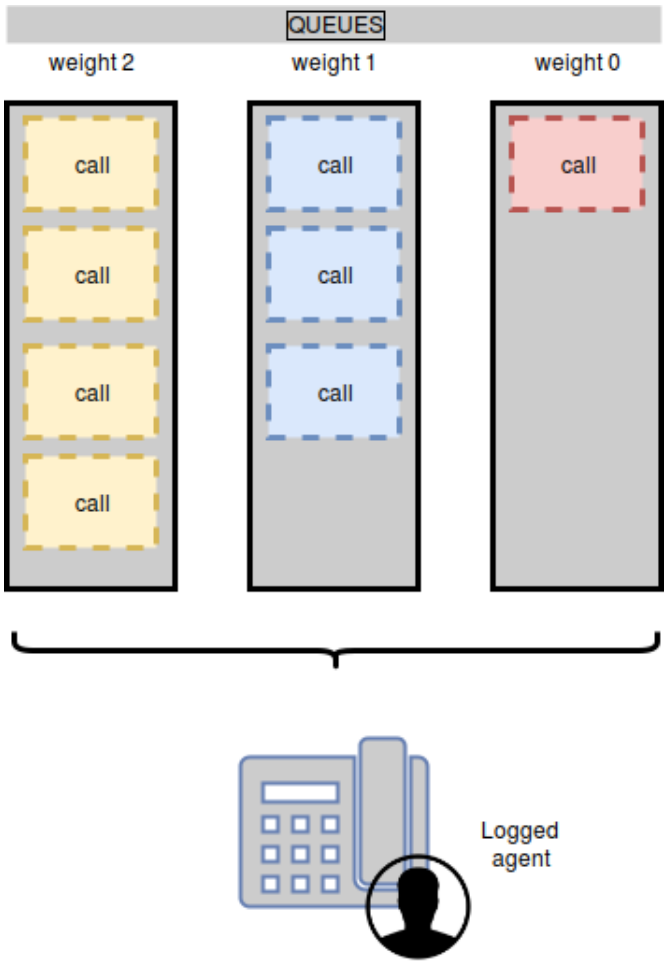
In the following sections, a waiting call is a call that has entered the queue but has not yet been answered by a queue member.

Estimated Wait Time Overrun

When this scenario is used, the administrator can set a destination for calls to be sent to when the estimated waiting time is over the threshold.


Queues > Add
General
Announces
Members
Application
No answer
Advanced
Schedules
Diversions
No answer
Destination :
Queue
Redirect to :
blue (3500@loadtest)
Ring time :
Busy
Destination :
End call
Choice:
Hangup
Congestion
Destination :
User
Redirect to :
acd03 acd03
Ring time :
Fail
Destination :
Voicemail
Redirect to :
LFM Paris (1000@default)
Play occupation message :
Do not play introduction message :
Do not play unavailable message :
Use n+101 method :
SAVE


Queues > Add
General
Announces
Members
Application
No answer
Advanced
Schedules
Diversions
Exit context:
Service level:
0
Member reachability timeout:
15 seconds
Time before retrying a call to a member:
1 second
Weight:
0
Delay before reassigning a call:
Disabled
Maximum number of people allowed to wait:
0
Recording:
Recording format:
Join an empty queue:
3 items selected
Remove all
Add all
unavailable
invalid
unknown
paused
penalty
inuse
ringing
wrapup
Remove callers if there are no agents:
3 items selected
Remove all
Add all
unavailable
invalid
unknown
paused
penalty
inuse
ringing
wrapup
Call a member already on:
Enable caller hold time reporting:
Delay before passing the call:
Disabled
Delay before resetting a member response time:
Autofill system:
Autopause agents:
No
Set interface variables in dialplan:
Set queue entry variables in diaplan:
Set queue variables in dialplan:
Macro executed at member connection:
Penalties applied:
SAVE



Queues > Add

General | Announces | Members | Application | No answer | Advanced | Schedules | Diversions

On estimated wait time overrun

Maximum estimated wait time:	5 minutes
Destination :	End call
Choice:	Busy
Delay before hangup:	

On number of waiting calls per logged-in agent overrun

Maximum number of waiting calls per logged-in agent:	1
Destination :	End call
Choice:	Busy
Delay before hangup:	

SAVE

Note that if a new call arrives when there are no waiting calls in the queue, the call will **always** be allowed to enter the queue.

Note:

- this *estimated* waiting time is computed from the **actual hold time** of all **answered** calls in the queue (since last asterisk restart) according to an exponential smoothing formula
- the estimated waiting time of a queue is updated only when a queue member answers a call.

Number of Waiting Calls per Logged-In Agent Overrun

When this scenario is used, the administrator can set a destination for calls to be sent to when the number of waiting calls per logged-in agent is over the threshold.

The number of waiting calls includes the call for which the check is currently being performed.

The number of logged-in agents is the sum of user members and currently logged-in agent members. An agent only needs to be logged in and a member of the queue to participate towards the count of logged-in agents, regardless of whether he is available, on call, on pause or on wrapup.

The maximum number of waiting calls per logged-in agent can have a fractional part.

Here are a few examples:

```
Maximum number of waiting calls per logged-in agent: 1
Current number of waiting calls: 2
Current number of logged-in agents: 2
Number of waiting calls per logged-in agent when a new call arrives: 3 / 2 = 1.5
Call will be redirected

Maximum number of waiting calls per logged-in agent: 0.5
Number of waiting calls: 5
Number of logged-in agents: 12
Number of waiting calls per logged-in agent when a new call arrives: 6 / 12 = 0.5
Call will not be redirected
```

Note that if a new call arrives when there are no waiting calls in the queue, the call will **always** be allowed to enter the queue. For example, in the following scenario:

```
Maximum number of waiting calls per logged-in agent: 0.5
Current number of waiting calls: 0
Current number of logged-in agents: 1
Number of waiting calls per logged-in agent when a new call arrives: 1 / 1 = 1
```

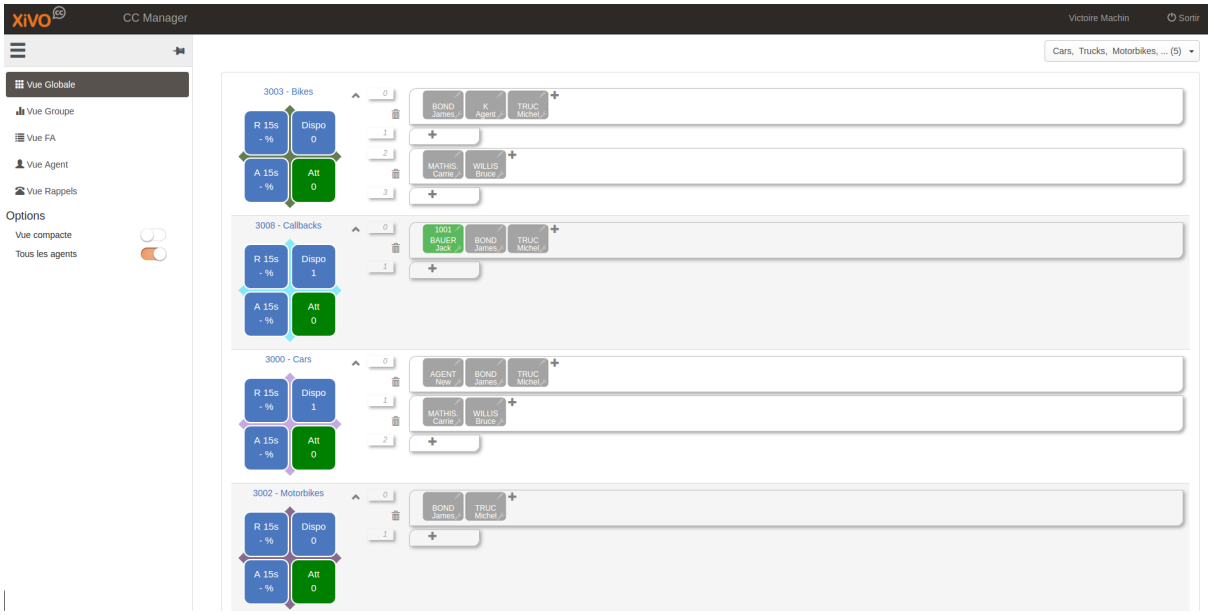
Even if the number of waiting calls per logged-in agent (1) is greater than the maximum (0.5), the call will still be accepted since there are currently no waiting calls.

6.3 Contact Center Management

6.3.1 Introduction

CCmanager is a web application to manage and supervise a contact center, different menus are available from hamburger icon with following features:

1. **Global view**: Queues and penalties with real time activity of each agent, possible options are
    a. Enable Compact view (remove queue statistics)
    b. Show/Hide agents that are not logged in



2. **Group view**: Distribution of agents per queues
3. **Queue view**: Activity per queue
4. **Agent view**: Activity per agent, possible actions are[1]
    a. Login / Logout
    b. Pause / Available
    c. Listen[2]
    d. Call[2]
5. **Callback view**: List of callbacks
6. **Qualification view**: To export calls qualification

Start the application : http://<xucmgt:port>/ccmanager

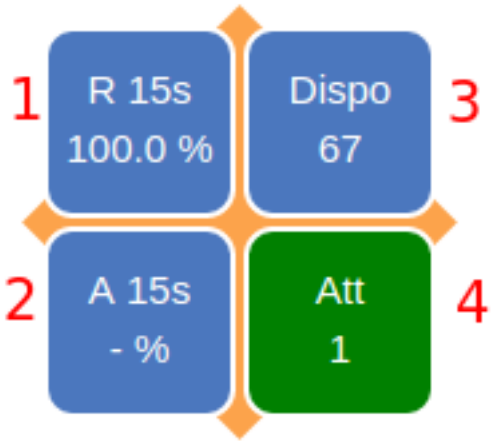
6.3.2 Authorizations and Access Control

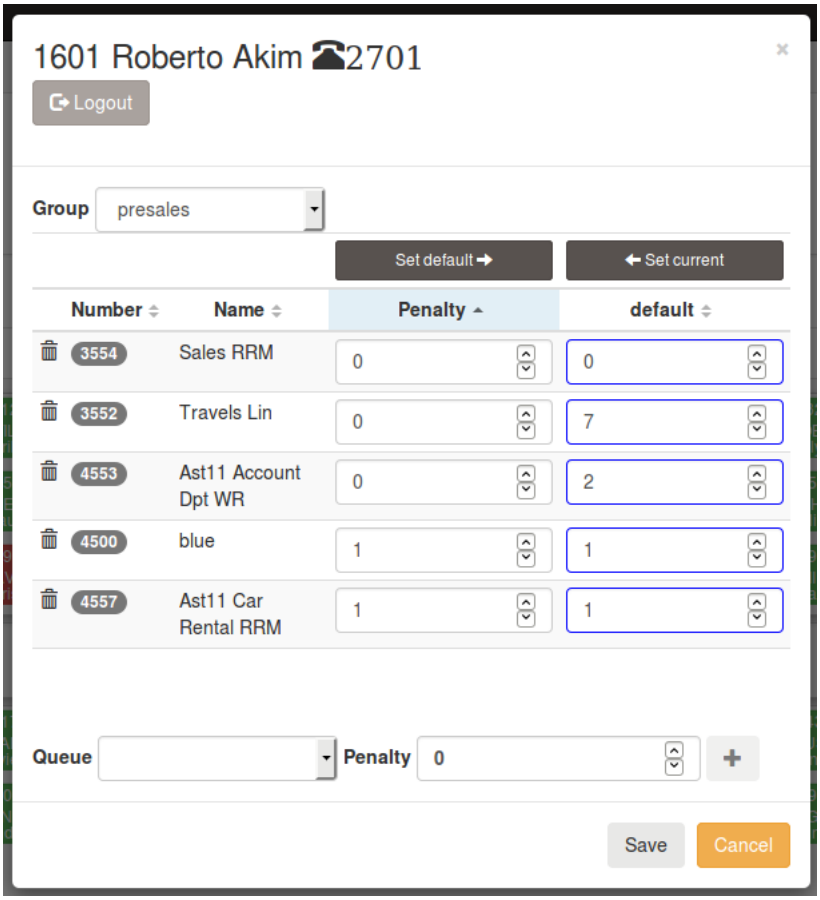
Access to the application is restricted to authorized users (see *Access authorizations in CCManager*).

6.3.3 Queue statistics

This diagram shows some aggregation about statistics collected from queue activity

1. Percentage of calls answered before 15s
2. Percentage of calls abandoned after 15s
3. Number of available agents to take calls
4. Number of pending calls not answered yet


1
R 15s
100.0 %
Dispo
67
3
2
A 15s
- %
Att
1
4


1601 Roberto Akim ☎2701
Logout
Group
presales
Set default
Set current
Number
Name
Penalty
default
3554
Sales RRM
0
0
3552
Travels Lin
0
7
4553
Ast11 Account Dpt WR
0
2
4500
blue
1
1
4557
Ast11 Car Rental RRM
1
1
Queue
Penalty
0
Save
Cancel

6.3.4 Editing Agent Configuration

This interface allows a user to change queue assignement and the associated penalty. The queue table display the following columns

- "Number": The queue number
- "Name": The queue name
- "Penalty": The active penalty for the corresponding queue
- "default": The default penalty for the corresponding queue

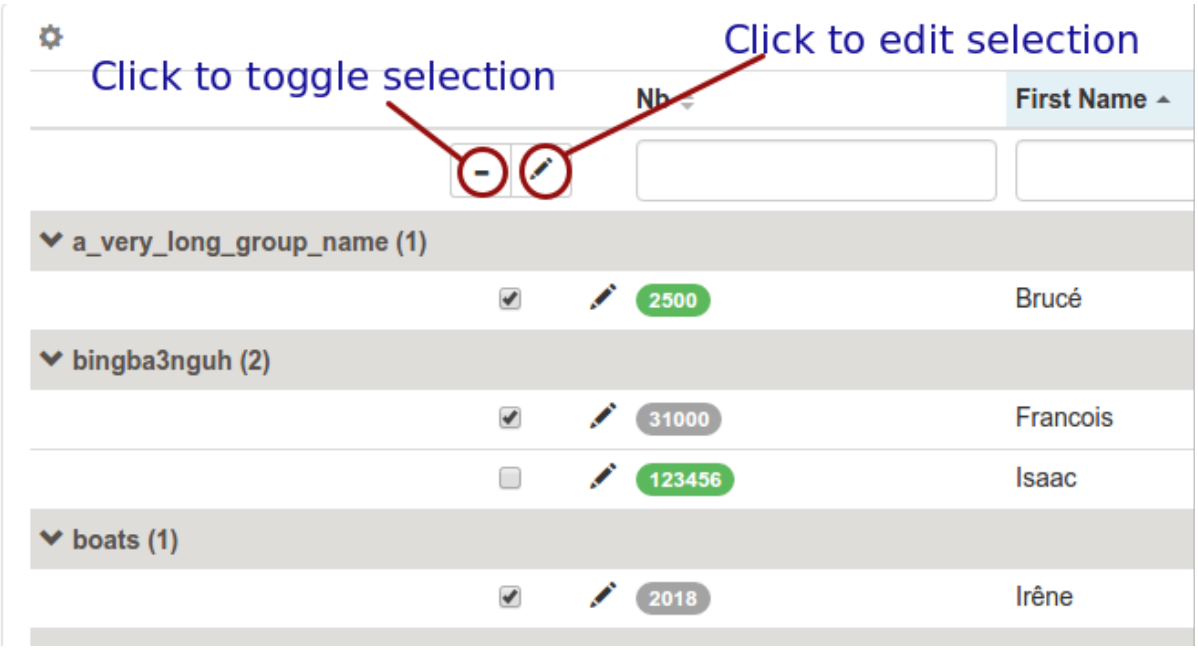
The queue/active penalty couples can be saved as default configuration by clicking the "Set default" button, then "Save". The queue/default penalty couples can be saved as active configuration by clicking the "Set current" button, then "Save".

Note: Removing an agent from a queue

- Emptying the penalty textbox and saving will remove the queue from the active configuration for the agent.
- Emptying the default textbox and saving will remove the queue from the default configuration for the agent.

6.3.5 Multiple Agent Selection

From agent view you are able to add or remove more than one agent at the same time.

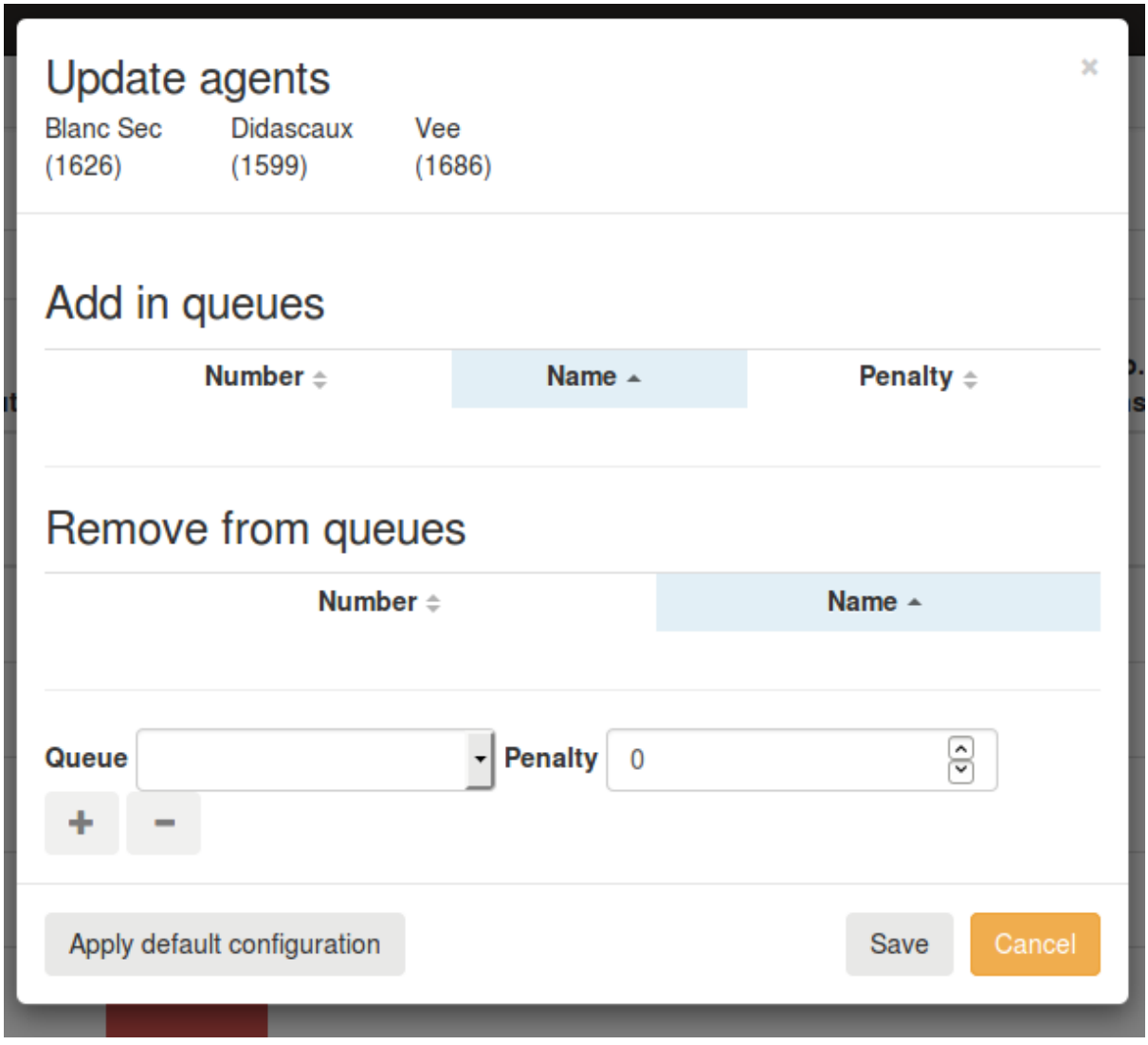


Once the agent selection is done, click on the edit button to display the configuration window

Click on the plus button to add a queue for selection, click on the minus button to remove a queue to the selection. Once queue to add or removed are choosen, click on save button to apply your configuration change.

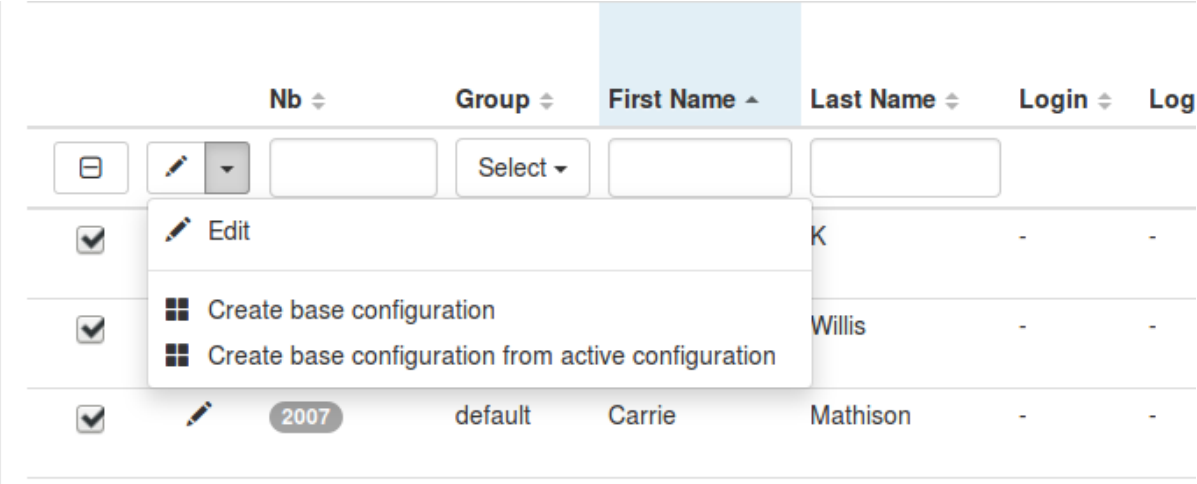
Click on "Apply default configuration" to apply existing default configuration to all selected users and make it the active configuration. This action only affects users with an existing default configuration, agents whithout default configuration remain unchanged.

[1] Available actions depend on the state of the agent
[2] Only supervisors which have their own line can listen to or call agents, not supported for mobile supervisors, a line has to be affected to supervisors in xivo


Update agents
Blanc Sec
(1626)
Didascaux
(1599)
Vee
(1686)
Add in queues
Number
Name
Penalty
Remove from queues
Number
Name
Queue
Penalty
0
Apply default configuration
Save
Cancel

6.3.6 Agent Base Configuration

From the agent view, after selecting one or more agents, you can create a base configuration by clicking on one of the menu item in the following drop down:



- 'Create base configuration' will allow you to create a base configuration from scratch for all the selected agents.
- 'Create base configuration from active configuration' will allow you to create a base configuration using the selected agents active configuration. The queue membership and penalty populated will be built based on the merged membership of all the selected agents. In case of conflict, the lowest penalty will be used.

In both cases, you will be able to review your changes before applying them. The 'Create base configuration' popup is similar to the single agent edition popup:

The queue table display the following columns:

- "Number": The queue number
- "Name": The queue name
- "Penalty": The active penalty for the corresponding queue

Click on the plus button to add a queue for selection. Once your configuration is complete, click on save button to apply your configuration change.

Applying Default Configuration

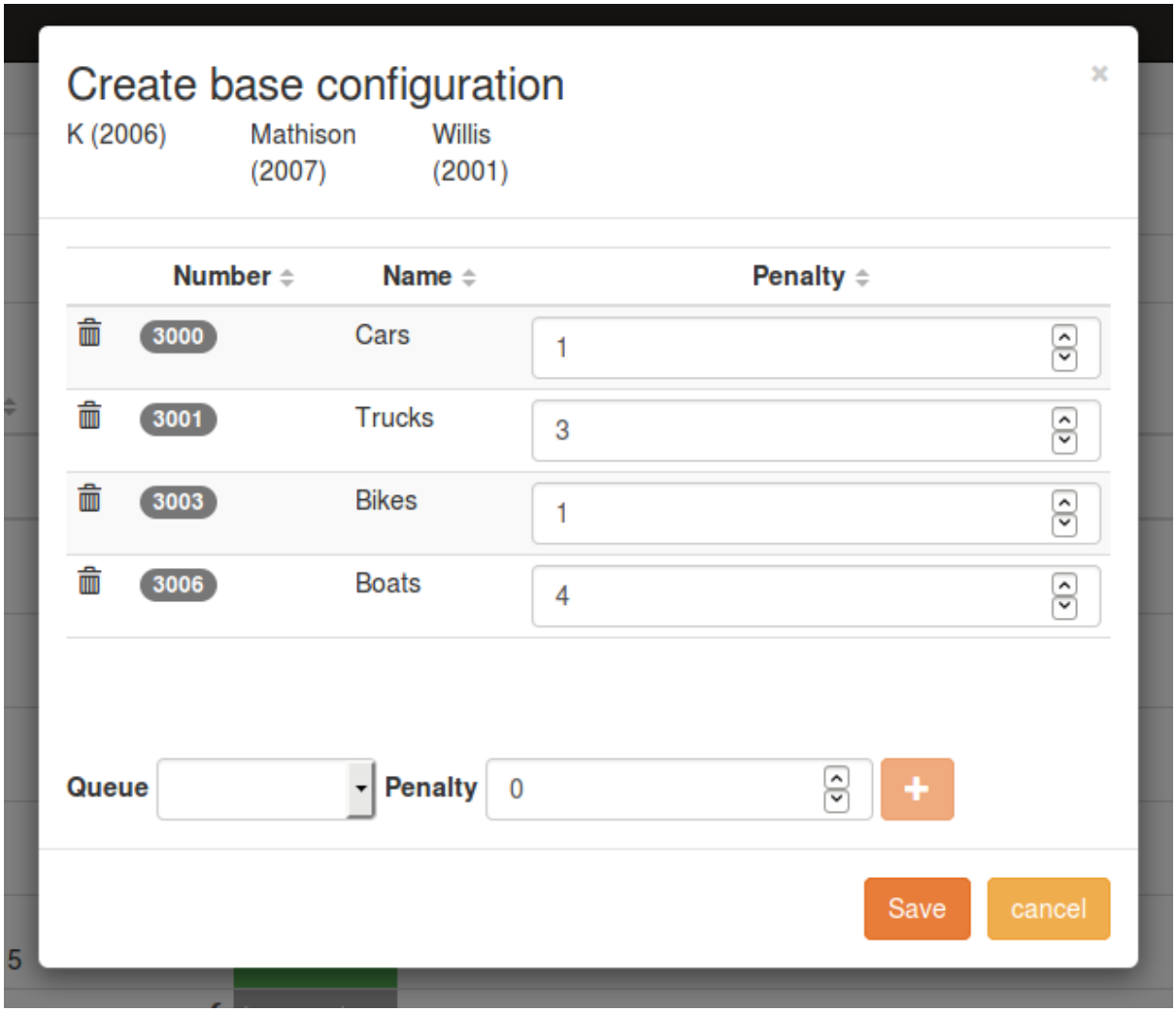
In order to re-apply or apply a default configuration, you may select agent whose base configuration is different from active configuration.

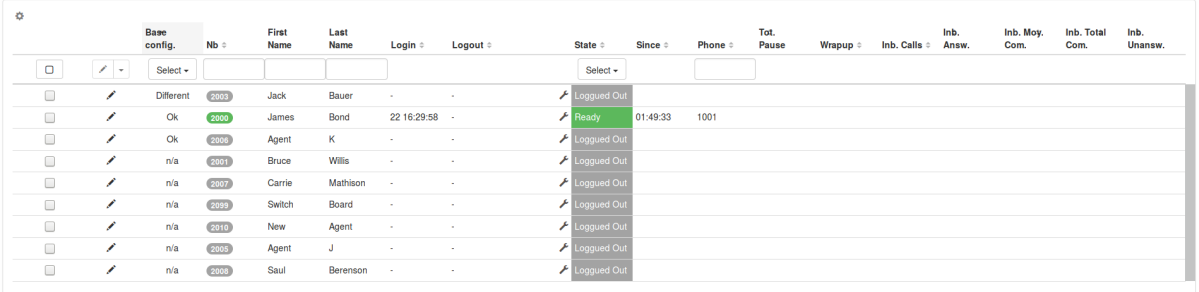
In the agent view, you will find a new column (Base config.) displaying if the base configuration is different from the active one:

The possible values for this field are:

- "n/a": The base configuration is not available for this agent
- "Ok" : The base configuration match the active configuration
- "Different": The base configuration **does not** match the active configuration

You can use this column to filter agent whose base configuration is different from the active one and then apply the default configuration by using the "Edit agent" option.

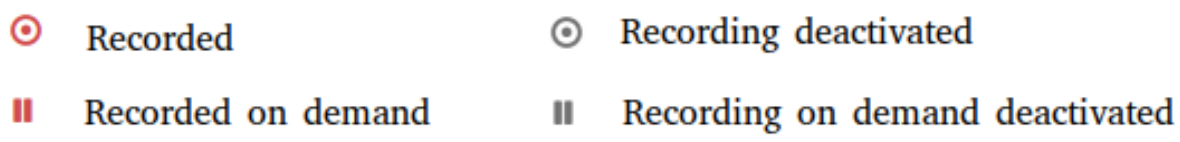

Create base configuration
K (2006)
Mathison (2007)
Willis (2001)
Number
Name
Penalty
3000
Cars
1
3001
Trucks
3
3003
Bikes
1
3006
Boats
4
Queue
Penalty
0
Save
cancel


Base config.
Nb
First Name
Last Name
Login
Logout
State
Since
Phone
Tot. Pause
Wrapup
Inb. Calls
Inb. Answ.
Inb. Moy. Com.
Inb. Total Com.
Inb. Unansw.
Different 2003 Jack Bauer Loggued Out
Ok 2000 James Bond 22 16:29:58 Ready 01:49:33 1001
Ok 2006 Agent K Loggued Out
n/a 2001 Bruce Willis Loggued Out
n/a 2007 Carrie Mathison Loggued Out
n/a 2099 Switch Board Loggued Out
n/a 2010 New Agent Loggued Out
n/a 2005 Agent J Loggued Out
n/a 2008 Saul Berenson Loggued Out

6.3.7 Queue Recording

Description

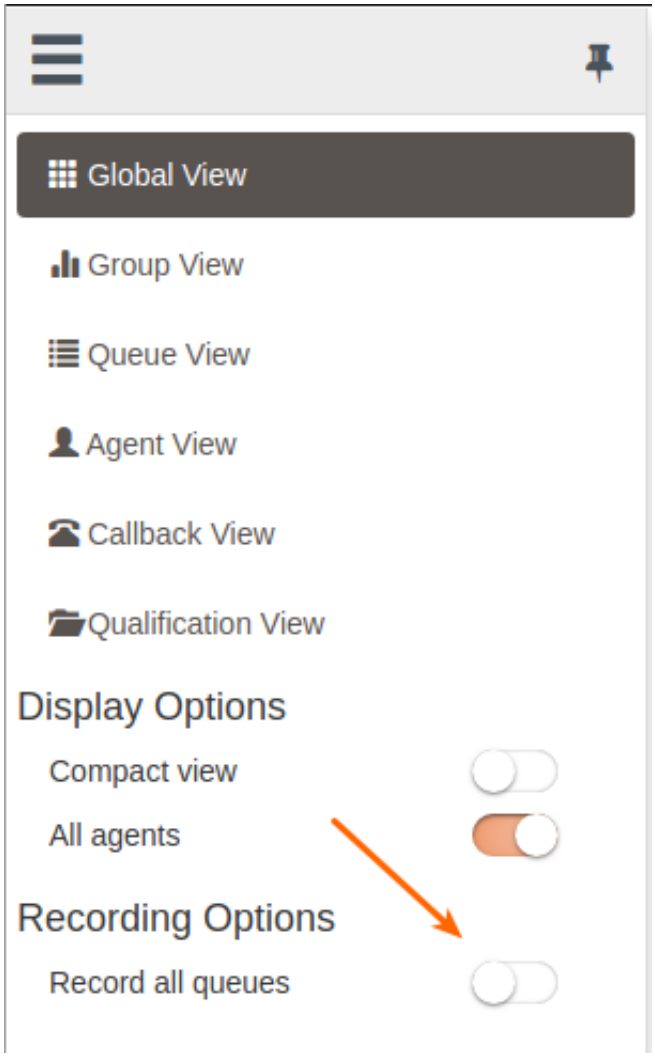
Once you setup queue recording in XiVO (see *Enable recording in the Queue configuration*), visual indicators are displayed next to queue name in **Global view**. Respectively following icons represents **recording mode** set on the queue.



Furthermore shortcut action is displayed in the left menu to control the activation / deactivation of **all** recorded queues. The switch will change its position in case the queue's recording status is activated / deactivated through XiVO Web Interface accordingly.

Global recording activation

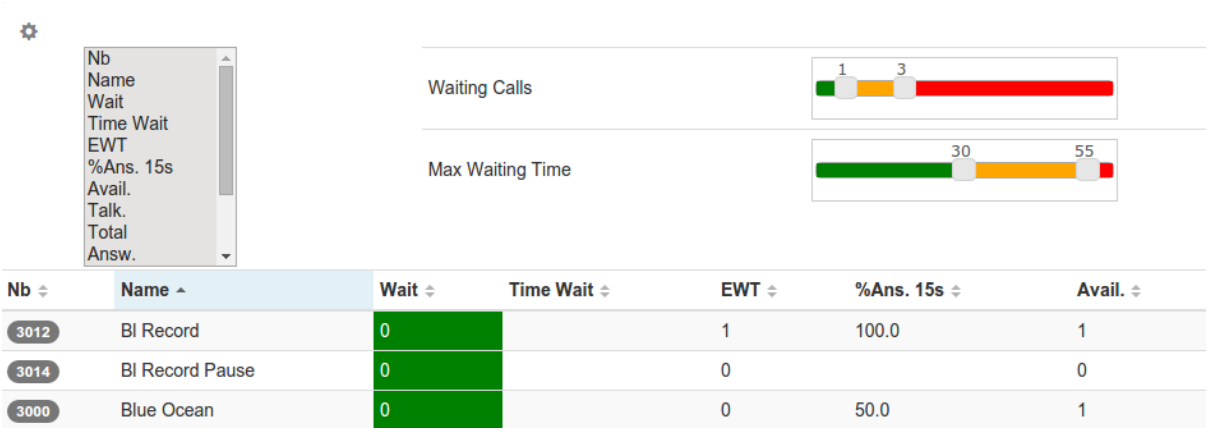
The switch button will either activate all queues configured with **recording mode** set (*Recorded* or *Recorded on demand*), or stop the recording feature for all queues.



Note: Action is applied only for next calls. Ongoing call recording is not started nor stopped when switch is triggered.

6.3.8 Thresholds

Color thresholds can be defines for the waiting calls counter and the maximum waiting time counter



Applies to the queue view and the global view

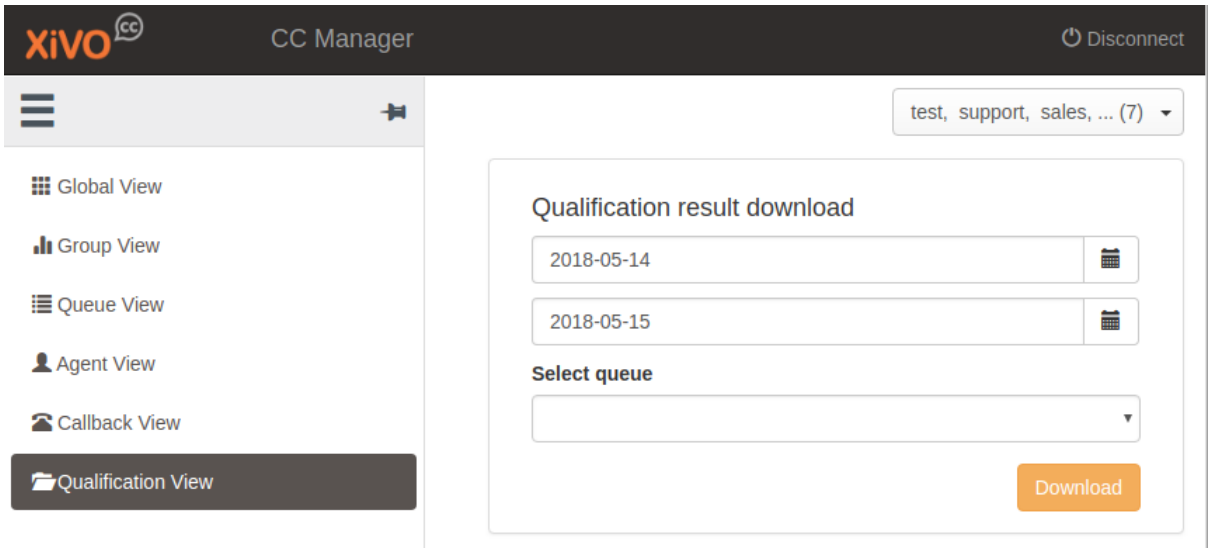
6.3.9 Callbacks

This view allows to manage callback request see *Managing Callbacks Using CCManager* for details.

6.3.10 Exporting qualifications

This view allows to export calls qualification see *Call Qualifications*.

To export the qualification answers, open **Qualifiification View** page.



Select the date from, date to and queue. Then click to Download button. This will open new page with CSV file to download.

Warning: When exporting data from the same day, select **date to** to be +1 day of **date from**.

6.4 CC Agent Environment

Note: This section describes the CC Agent application features. It is available as a web application from your Web Browser. It is also available as a *desktop application* with these additionnal features:

- show integrated popup when receiving call
- get keyboard shortcut to answer/hangup and make call using *Select2Call feature*
- *handle callto: and tel: links*
- be able to *minimize the application to a side bar*

To install the *desktop application*, see *the desktop application installation* page.

What is the CC Agent application ?

CC Agent is a Web application for contact center operators. Some parameters for Recording, Callbacks, Queue control, Pause statuses and Sheet popup may be configured. Instructions can be found in the *configuration section*.

From the interface you will be able to :

- Manage your activities you are subscribed to receive calls.
- See the customer history inside your organization when a call is coming
- Interact with your phone from the call control panel
- Get some real time statistics of your session

The web application can either be displayed in a minimal bar or be extended as seen in screen shot below when launched as standalone application. See *desktop application*.

Warning: The application offers support for the WebRTC lines, currently there's a limitation on complementary services like conference or second call which are partially supported.

6.4.1 Login

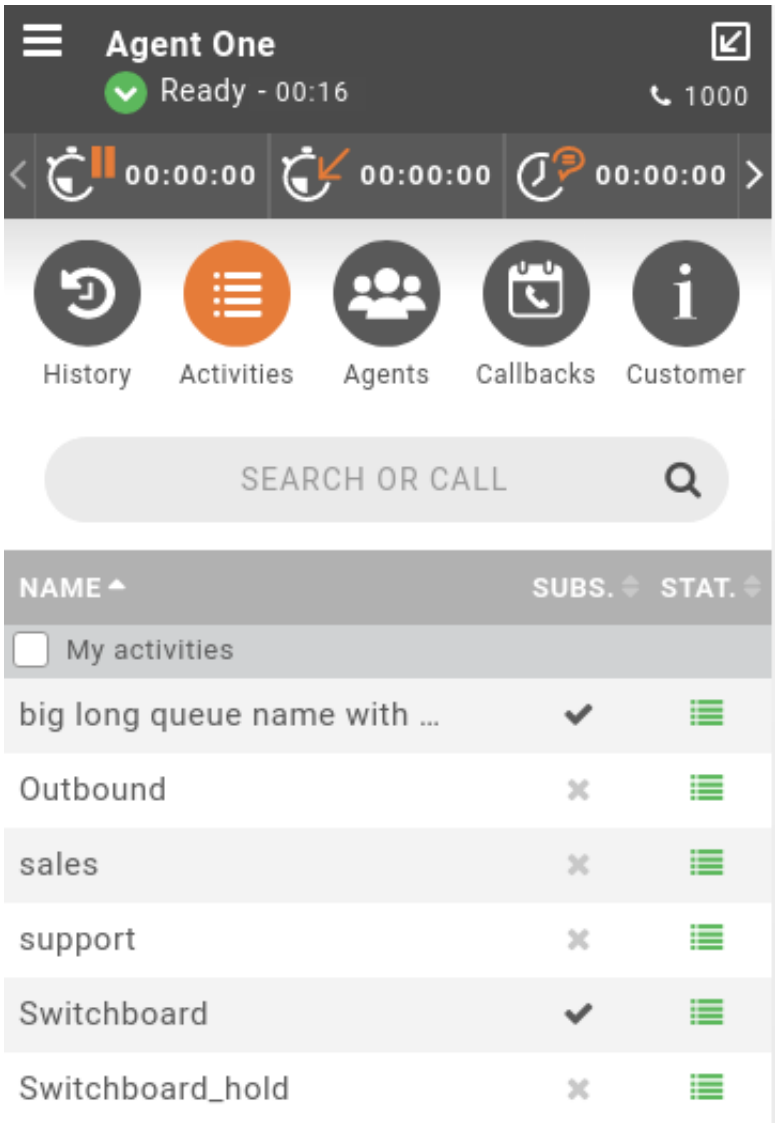
Enter your XiVO client username, password and phone set you want to logged to on the login page.

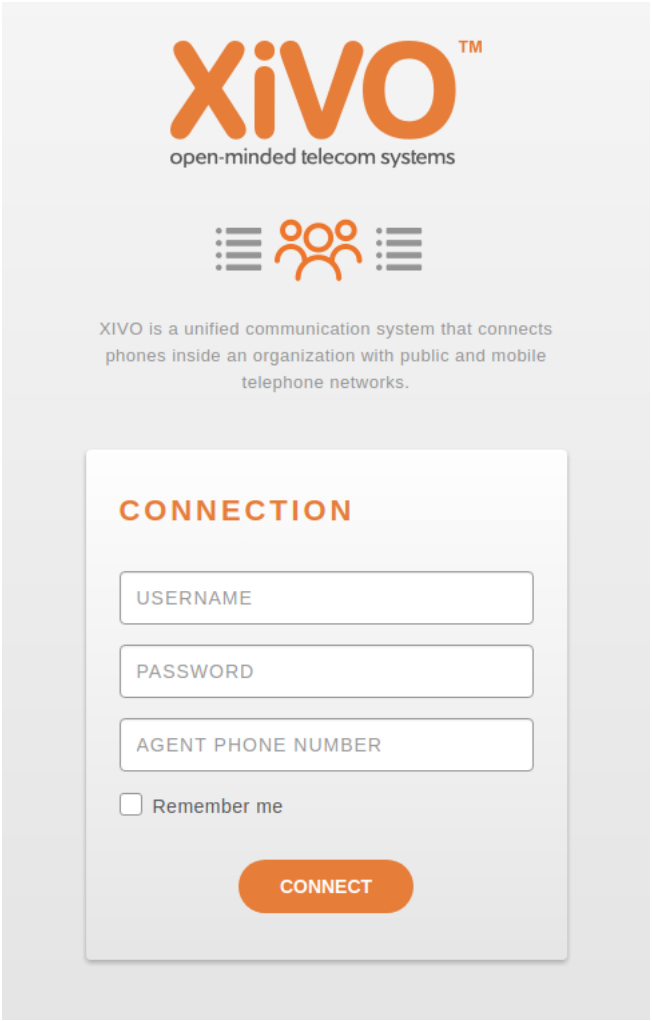
If you are using Kerberos authentication and enabled SSO (see *Kerberos Authentication*), then you only have to set your phone set number, the authentication and login will be done automatically:

6.4.2 Statistics

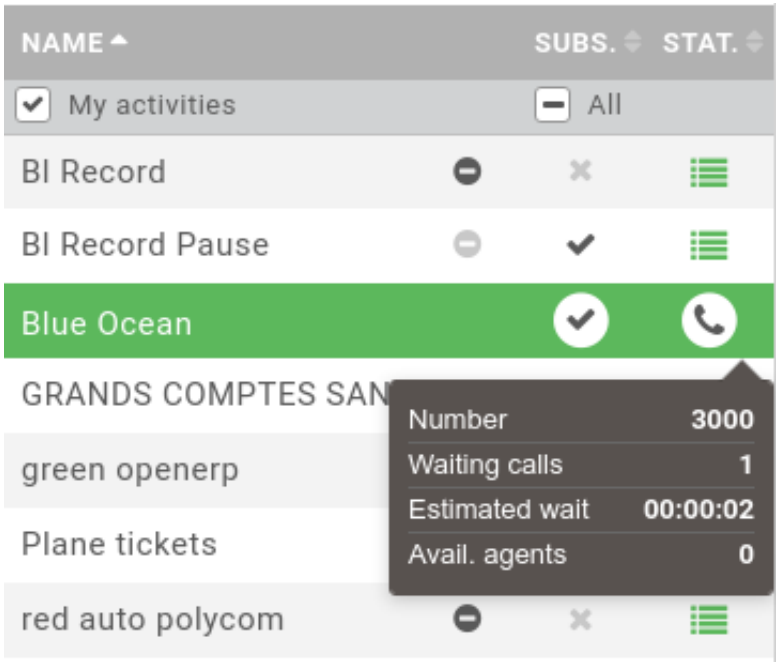
At the top of the application, *computed statistics* from XUC are displayed to monitor the agent activity. Simply hover the icon to know its definition.

They are updated once current action is over.


Agent One
Ready - 00:16
1000
00:00:00
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
NAME
SUBS.
STAT.
My activities
big long queue name with ...
Outbound
sales
support
Switchboard
Switchboard_hold


XiVO™
open-minded telecom systems
XIVO is a unified communication system that connects phones inside an organization with public and mobile telephone networks.
CONNECTION
USERNAME
PASSWORD
AGENT PHONE NUMBER
Remember me
CONNECT

00:00:00
00:00:00
00:00:00


NAME
SUBS.
STAT.
My activities
All
BI Record
BI Record Pause
Blue Ocean
GRANDS COMPTES SAN
green openerp
Plane tickets
red auto polycom
Number 3000
Waiting calls 1
Estimated wait 00:00:02
Avail. agents 0

6.4.3 Activities

Once logged in you are automatically redirected to `activities` view, this view contains the list of activities you are registered in.

Hovering an activity triggers a popover which displays some real-time statistics about call distribution in this queue. You may also call or transfer a call to an activity using the displayed phone icon

It is possible to filter on *favorite* activities just by clicking `My activities` check box.

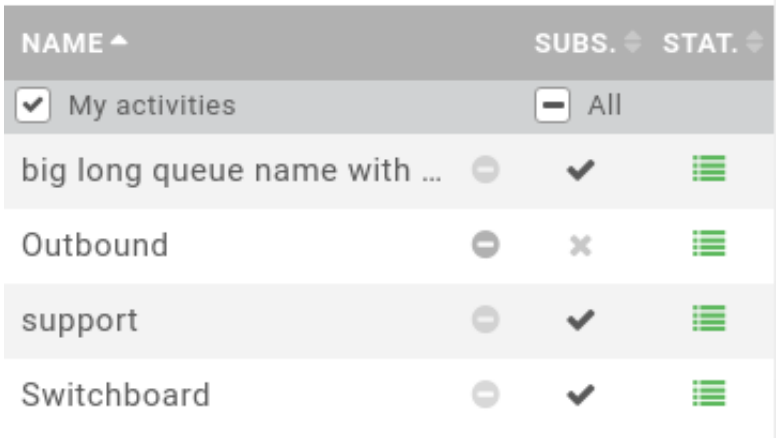
Activity Management

You can manage `subscription` if allowed globally for the application (see *CC Agent configuration*). If enabled you will be able to enter/quit an activity just by clicking on checkbox associated to it.

Each time you enter an activity, it is automatically added to your favorites. At any time you can remove it by clicking on *minus* sign next to the name in `My activities` view.

Note: You can remove an activity if and only if you are not already registered in.

It's also possible to enter all your favorites activities just by one click on `All` checkbox.



Activity Colors

Activity color changes depending on call waiting and agent status:

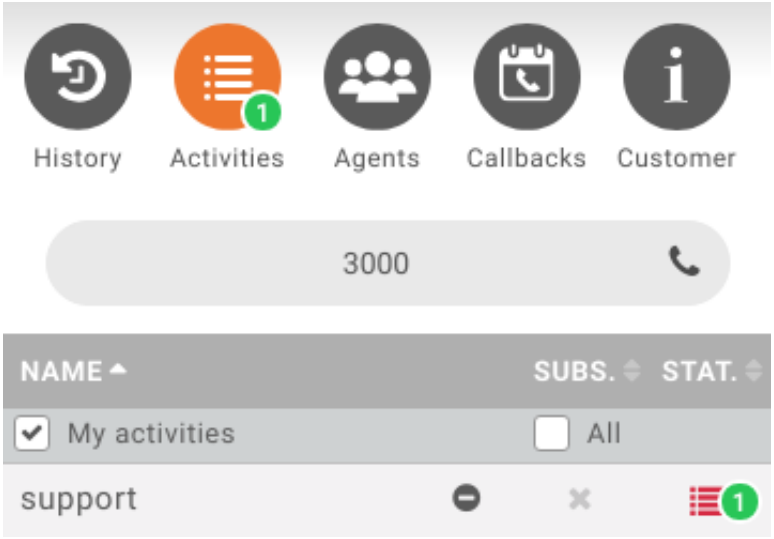
- **Grey**: No call, No agents logged in this activity
- **Green**: At least one agent logged and available in this activity
- **Orange**: At least one agent logged, but no agents available in this activity
- **Red**: No agents logged in this activity, but one or more waiting calls in this activity

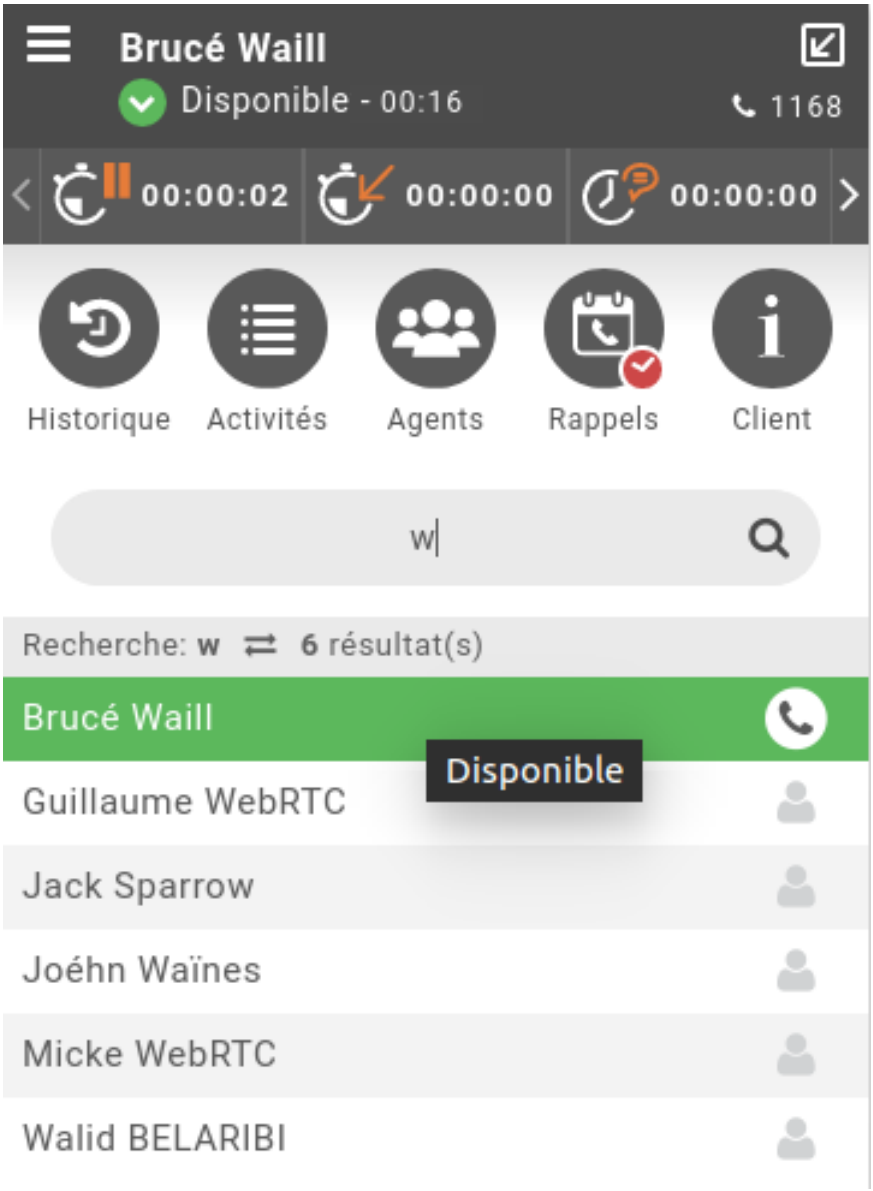
Activity Waiting Calls

A little badge displays the number of waiting calls in each activity. The sum of all waiting calls in the agent activities is displayed in top menu and refreshed in real time.

6.4.4 User directory search

When using the Agent interface, you can at any time search for a user existing in your directory:


History
Activities
Agents
Callbacks
Customer
3000
NAME
SUBS.
STAT.
My activities
All
support

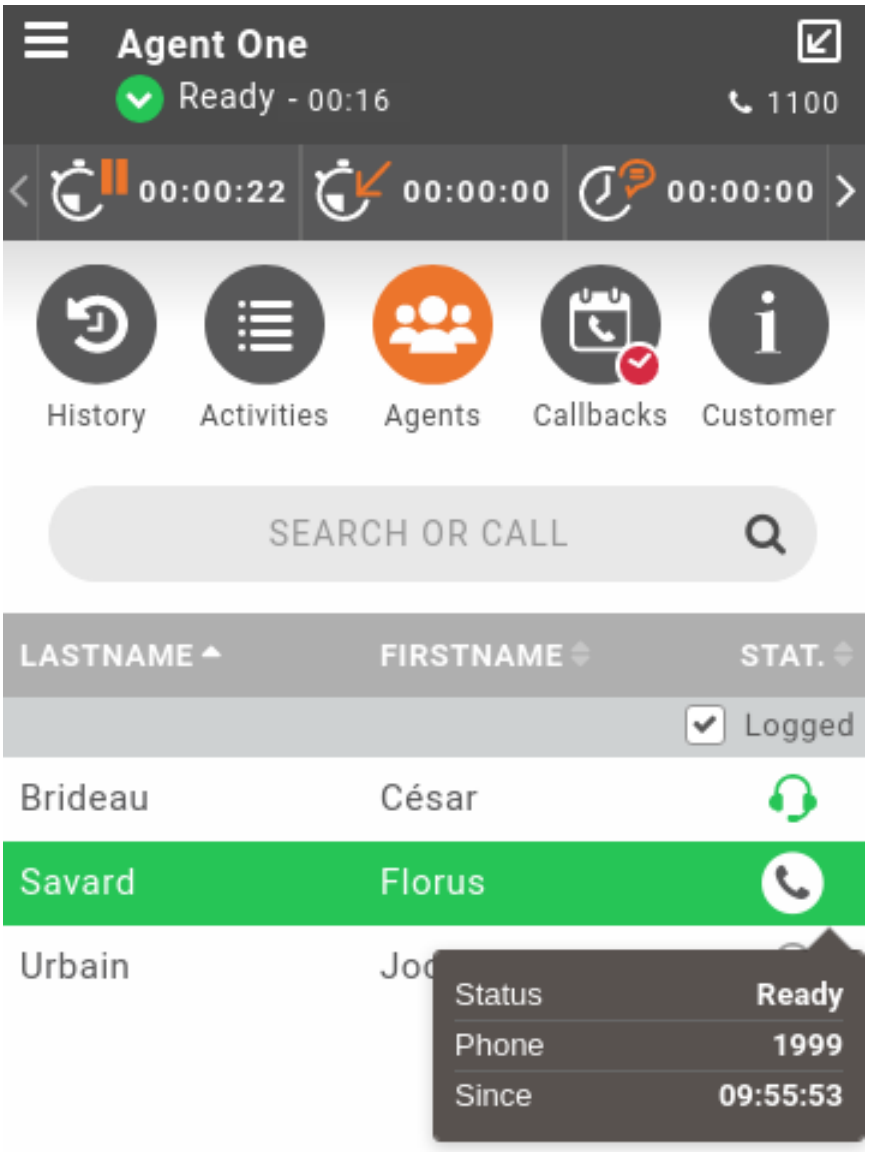

Brucé Waill
Disponible - 00:16
1168
00:00:02
00:00:00
00:00:00
Historique
Activités
Agents
Rappels
Client
w
Recherche: w
6 résultat(s)
Brucé Waill
Disponible
Guillaume WebRTC
Jack Sparrow
Joéhn Waïnes
Micke WebRTC
Walid BELARIBI

Known limitations

- The search support is limited so far to simple words without spaces and simple characters.

6.4.5 Agent list

When clicking on the `Agents` menu, you will see all the agents of **your group**. By hovering one of them, you will quickly find his current state (ready, calling, in pause...)

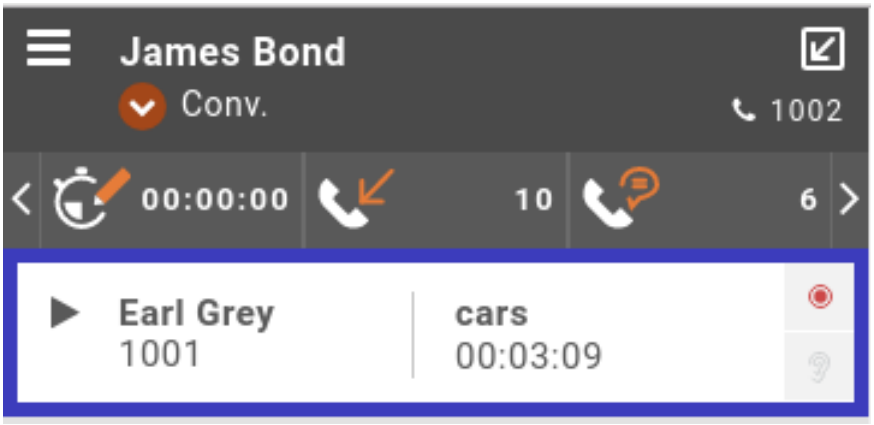


A simple click on the phone icon when agent is hovered will trigger a call to its phone number associated.

By default, agents are shown only if they are logged in (checkbox *Logged* checked). By unchecking the checkbox *Logged*, you will see all the agents of your group even if they are logged out.

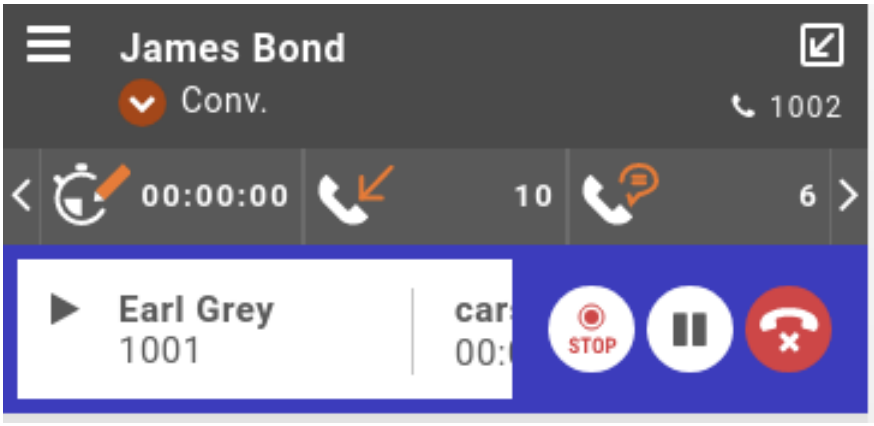
6.4.6 Call tracking & control

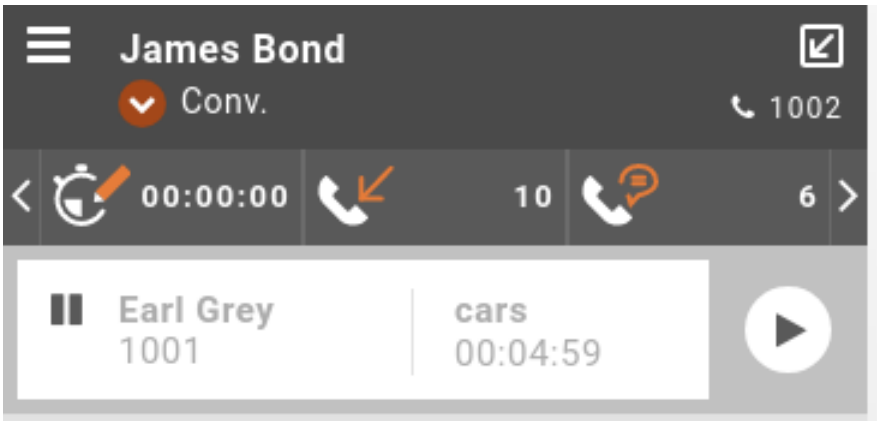
When using the Agent interface, you will see your current calls at the top of the screen:



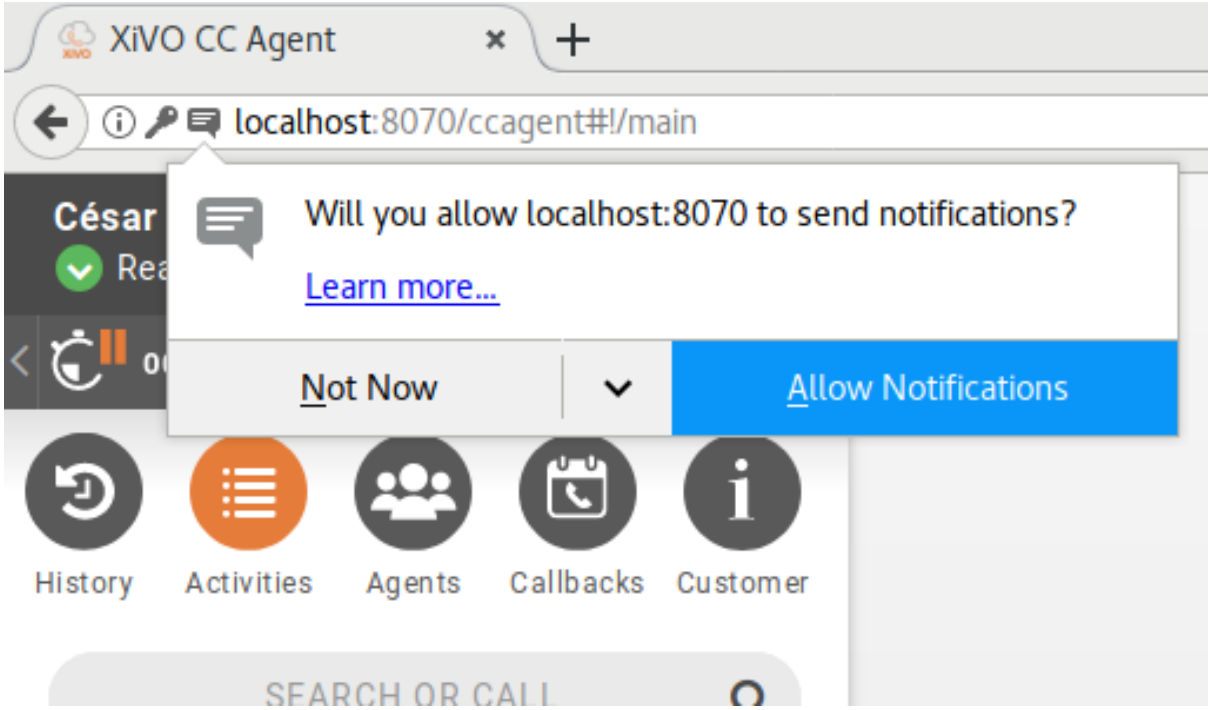
This panel will display the current caller name & number and also the associated activity if the call came from one. You also have two indicator on the right side letting you know if the call is currently recorded and if the call is currently listened by a supervisor.

By hovering your mouse on the call line, an action pane will slide to display action button on the related call. The available buttons depend on the call state.





The Agent interface use Desktop notification for incoming calls and notify long calls on hold, but this feature needs to be enabled from the browser window when logging in:



On hold notifications

You can be notified if you forget a call in hold for a long time, see *configuration section*.

Known limitations

- The transfer support is limited, to transfer a call you need to write the number to the search field, then click the attended transfer button and then you can complete the transfer by the complete transfer button.
- Conference and second call except the attended transfer are not supported for agents with webRTC lines.

Also see the phone integration *Known limitations*.

6.4.7 Agent Call history

First menu `History` tab is displaying the call history of connected agent.

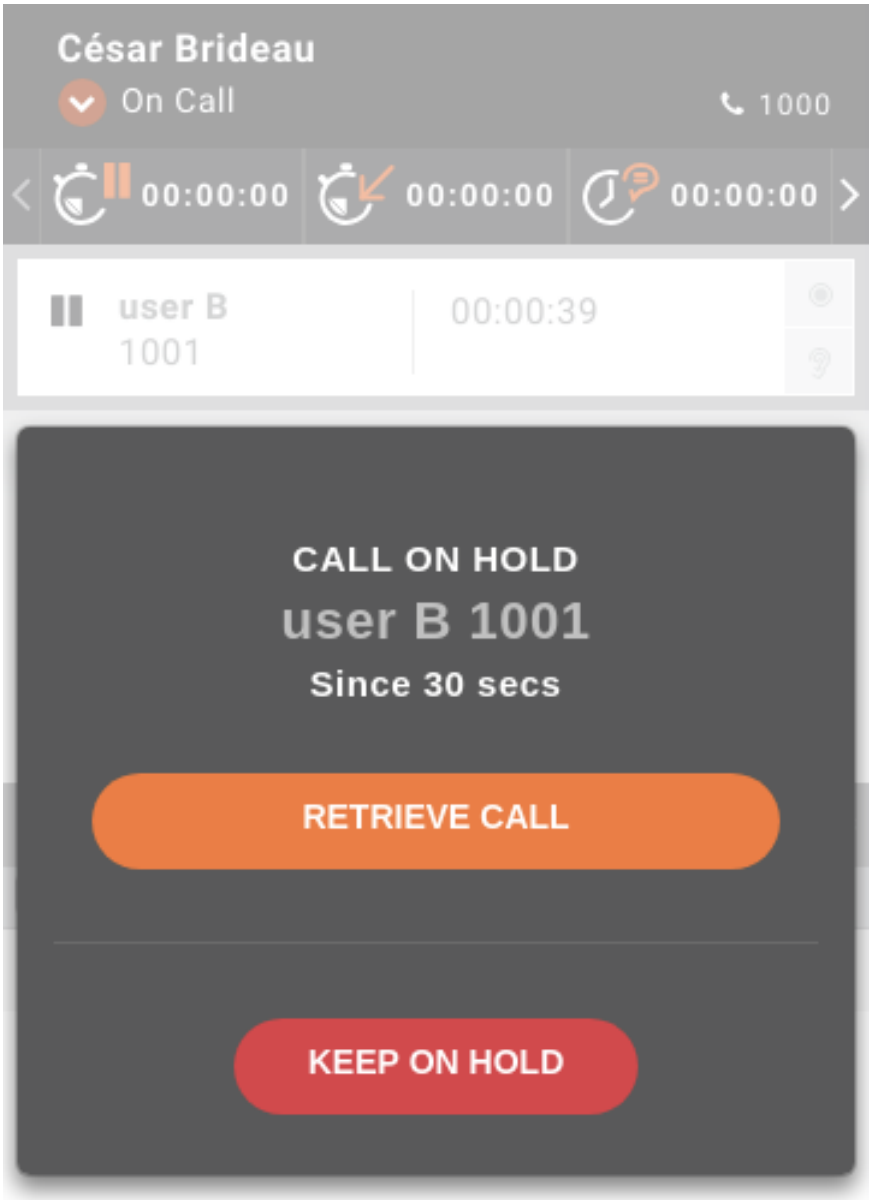
Information shown in the call list are :

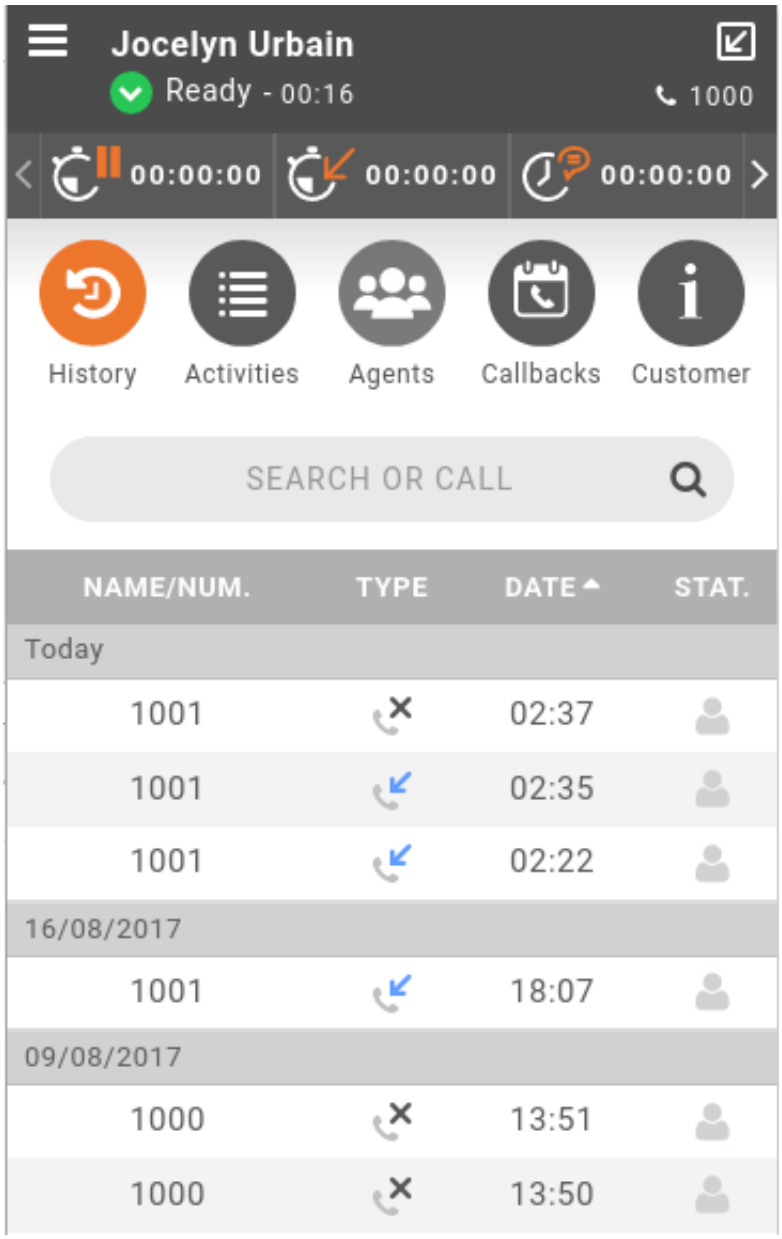
- *destination number* or *name* of callee if call is *emitted*
- *source number* or *name* of caller if a call is *received* or *missed*
- Call icon status and call start date

By clicking on phone icon you will be able to call back if needed.

Warning: Pay attention that agent history is **not** the phone device history, but his call activity independently the device he is connected to. Actually when agent is logged out, if a call is received on his last used phone, nothing will be shown in his history.

Note: A call answered by another agent from the queue, will appear as answered in the history of the first agent.


César Brideau
On Call
1000
00:00:00
00:00:00
00:00:00
user B
1001
00:00:39
CALL ON HOLD
user B 1001
Since 30 secs
RETRIEVE CALL
KEEP ON HOLD


Jocelyn Urbain
Ready - 00:16
1000
00:00:00
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
NAME/NUM.
TYPE
DATE
STAT.
Today
1001 02:37
1001 02:35
1001 02:22
16/08/2017
1001 18:07
09/08/2017
1000 13:51
1000 13:50

6.4.8 Customer Call History

Meanwhile phone is ringing or discussion is ongoing, it is possible to have a quick overview of the customer call history of the caller just by clicking `information` menu.

The customer history is displayed from most recent to last one with an icon to know quickly waiting time of current or previous call :

- agent with grey *play* icon states for current call
- agent with red *bubble* icon states for an unanswered call
- agent with green *bubble* icon states for an answered call

Note: If a call is answered (*green icon*), hovering the line will give the name of the agent who took the call.

HOUR	ACTIVITY	WAIT	ANSWER
Today			
17:16	support	00:00:28	
17:16	support	00:00:05	
17:15	support	00:00:13	
15:33	support	00:00:29	
15:30	support	00:02:06	
15:13	support	00:00:53	

6.4.9 Customer Call Context

Second tab of `information` menu displays all attached data enriched to the ongoing call or display Sheet fields if you are using *Sheet Configuration*.

The customer history is displayed from most recent to last one with an icon to know quickly waiting time of current or previous call :

It's also possible to trigger either to open a web page, see *Screen Popup* or completely integrate a third party application while agent is having calls, see *Configuration*.

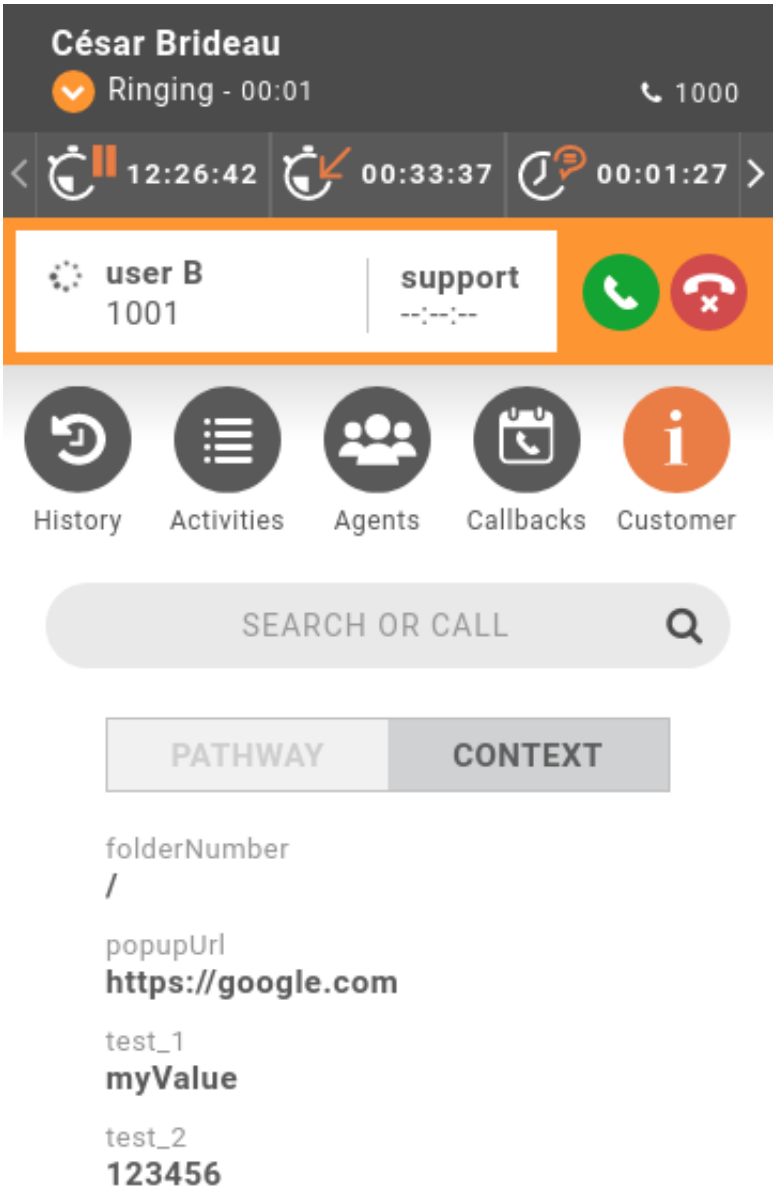
6.4.10 Callbacks

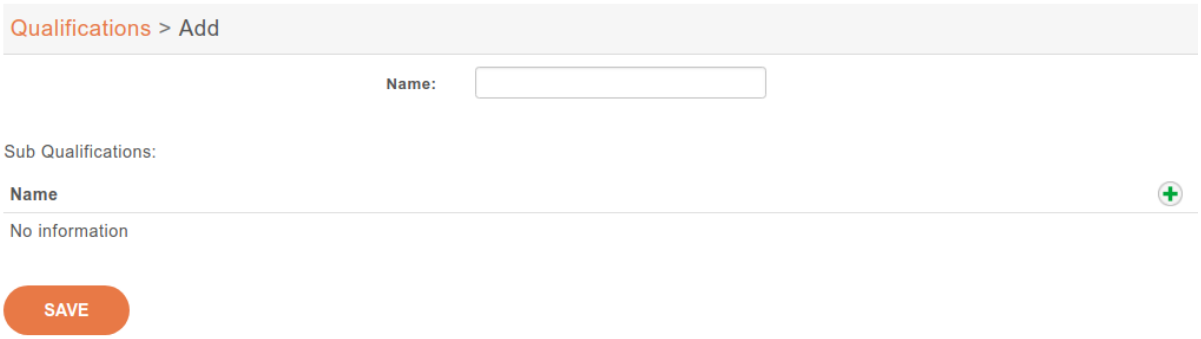
This view allows to manage callback request see *Processing Callbacks with CCAgent* for details.

6.5 Call Qualifications

Call qualifications are used to describe a call in a queue with a certain qualification and its related sub qualification. For example, a call can be qualified as a qualification **Sales** and a sub qualification **Hardware**.

Qualifications are managed on the *Services* → *Call Center* → *Qualifications* page.


César Brideau
Ringing - 00:01
1000
12:26:42
00:33:37
00:01:27
user B
1001
support
--:--:--
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
PATHWAY
CONTEXT
folderNumber
/
popupUrl
https://google.com
test_1
myValue
test_2
123456


Qualifications > Add
Name:
Sub Qualifications:
Name
No information
SAVE

Services → *Call Center* → *Qualifications* → *Add*

6.5.1 Qualification

Qualification is a main description to which a sub qualification belongs.

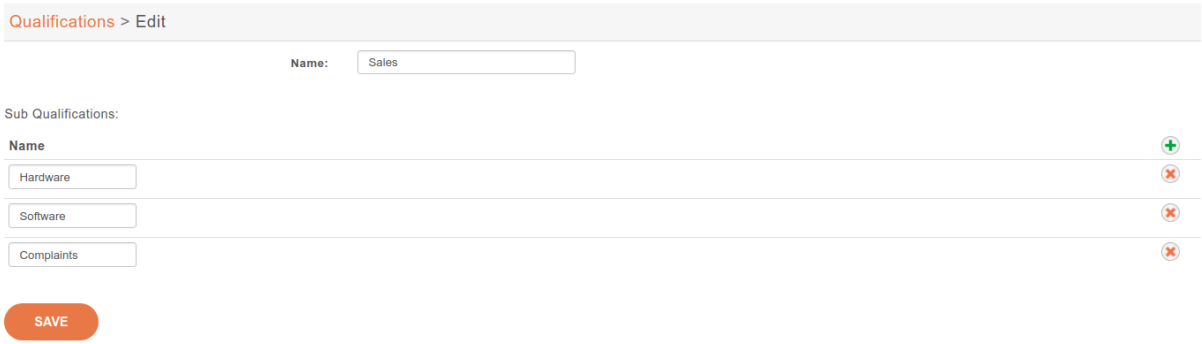
A qualification can be configured with the following options:

- Name: used as name for the qualification
- Sub Qualifications: used as name for the sub qualification

6.5.2 Sub Qualification

A sub qualification can be configured with the following options:

- Name: used as name for the qualification

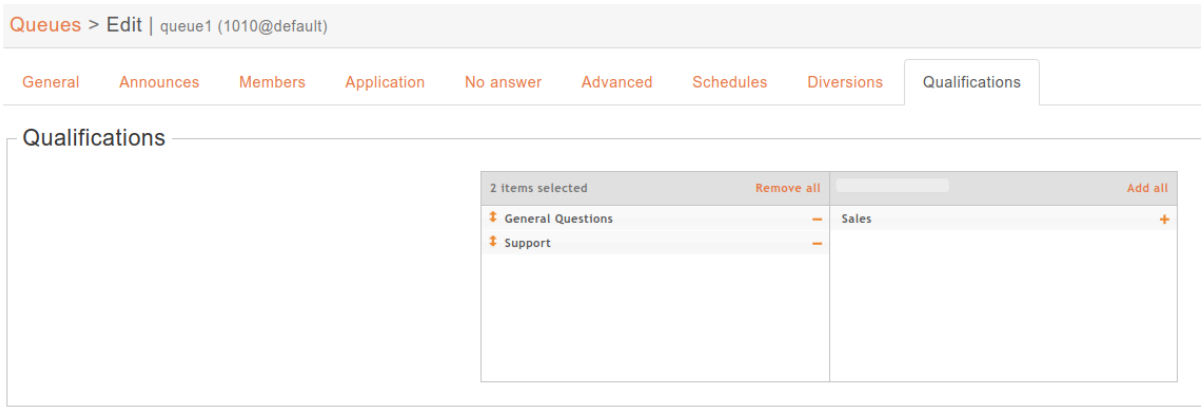


6.5.3 Assign qualification to queue

In order to retrieve all qualifications for a certain queue, the qualification must be added to that queue. A queue can be assigned with multiple qualifications and a qualification can be assigned to multiple queues.

To assign a qualification or multiple qualifications to the queue, open

Services → *Call Center* → *Queues* → *Edit* → *Qualifications tab* page.



6.5.4 Removing qualifications

Removing a qualification will remove also all related sub qualifications. The sub qualifications can be removed by opening the qualification edit page.

Note: The (sub)qualification will not be deleted from the database, but will be marked as inactive. This is due to the situation in which a call has been already qualified with now removed qualification. Therefore the information won't be lost.

6.5.5 Qualification answers

A call in a queue can be qualified by the qualification assigned to this queue.

6.5.6 Exporting qualification answers

See *Exporting qualifications* feature in CCmanager.

6.6 Recording

XiVO CC includes a call recording feature: recording is done on *XiVO PBX* and recorded files are sent to the *XiVO CC* Recording server. It's possible then to search for recordings and download recorded files from *Recording server*. If connected user is an administrator, he will be able to download also an access log file that track all action made on files by all users having access to recording server.

6.6.1 Recording

For configuring this feature, see *Recording configuration*.

Description

Recording feature records the conversation between caller and callee and stores it on dedicated recording server. All the files are then available for download only for predefined *granted* users. Recording is done on *XiVO PBX* and sent to *Recording server*. If recorded file can't be synchronized on *XiVO CC* Recording Server, files might be found on *XiVO PBX* in

```
ls -al /var/spool/xivocc-recording/failed
```

These files will be automatically resynchronized from *XiVO PBX* to *XiVO CC* Recording server each night.

You can then do following actions:

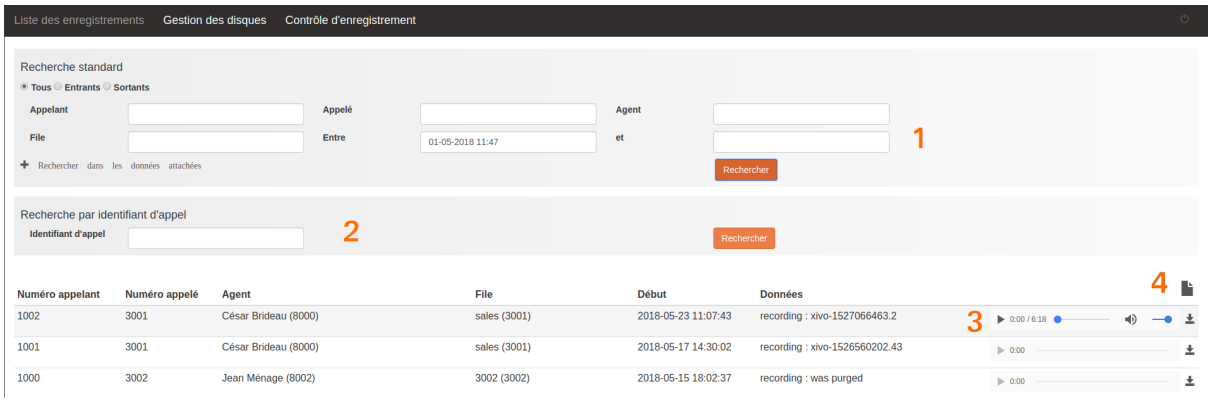
1. Find by caller, callee, agent number etc.
2. Find by callid
3. Listen or download the recording
4. Download access log to recording files

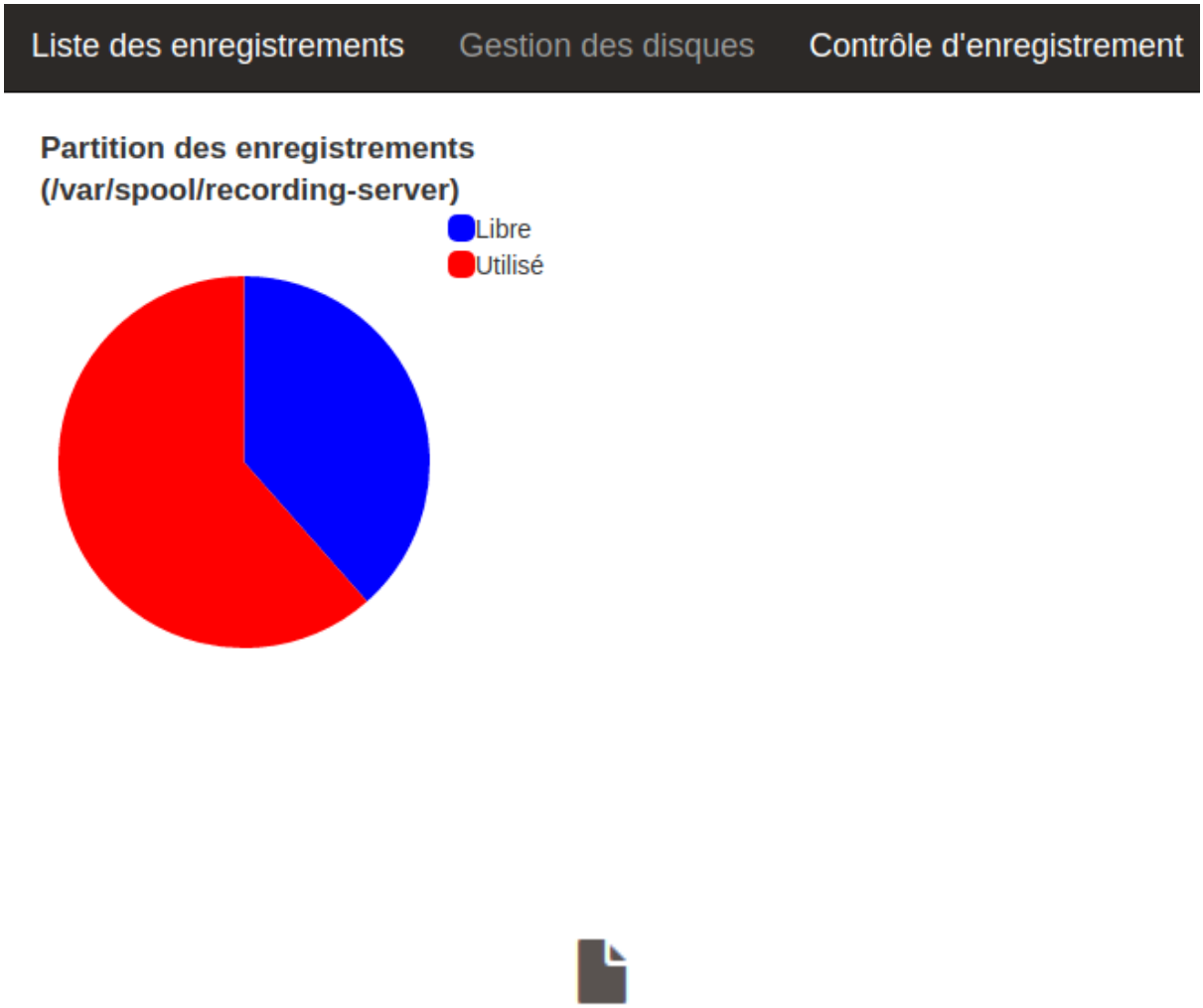
On *Recording server*, one can monitor the space used by the audio files stored in `Contrôle d'enregistrement` menu.

Access logs

Any action made from the UI on a recording file (`result`, `listen` or `download`) is tracked in access log that can be downloaded by clicking on following icon if you have administrator role.

Note: This access log is defined to track information for 6 months by default.


Liste des enregistrements
Gestion des disques
Contrôle d'enregistrement
Recherche standard
Tous Entrants Sortants
Appelant
Appelé
Agent
File
Entre
01-05-2018 11:47
et
Rechercher dans les données attachées
Rechercher
1
Recherche par identifiant d'appel
Identifiant d'appel
2
Rechercher
4
Numéro appelant
Numéro appelé
Agent
File
Début
Données
1002
3001
César Brideau (8000)
sales (3001)
2018-05-23 11:07:43
recording : xivo-1527066463.2
3
0:00 / 6:18
1001
3001
César Brideau (8000)
sales (3001)
2018-05-17 14:30:02
recording : xivo-1526560202.43
0:00
1000
3002
Jean Ménage (8002)
3002 (3002)
2018-05-15 18:02:37
recording : was purged
0:00


Liste des enregistrements
Gestion des disques
Contrôle d'enregistrement
Partition des enregistrements
(/var/spool/recording-server)
Libre
Utilisé

Transferred calls recordings

Search and Display

In case an agent has transferred a call to another queue, which was answered by the agent available in that queue, the recording feature will display both agents (name and number) in column **Agent** and both queues (name and number) under column **File** in one recording.

Numéro appelant	Numéro appelé	Agent	File	Début	Durée	Données
1002	1010	User Three (1003), User One (1001)	queue2 (1012), queue1 (1010)	2017-08-28 11:15:49	00:00:16	recording : xivo-1503911749.129
1002	1010	User Three (1003), User One (1001)	queue2 (1012), queue1 (1010)	2017-08-28 11:09:19	00:00:11	recording : xivo-1503911359.119
1002	1010	User Three (1003), User One (1001)	queue2 (1012), queue1 (1010)	2017-08-28 11:07:54	00:00:16	recording : xivo-1503911274.109
1002	1010	User Three (1003), User One (1001)	queue2 (1012), queue1 (1010)	2017-08-28 11:07:05	00:00:23	recording : xivo-1503911225.98
1002	1010	User Three (1003), User One (1001)	queue2 (1012), queue1 (1010)	2017-08-28 11:03:29	00:00:18	recording : xivo-1503911009.84
1002	1010	User Three (1003), User One (1001)	queue2 (1012), queue1 (1010)	2017-08-28 10:42:32	00:00:23	recording : xivo-1503909752.74

Stop/Start Recording

When a call enters a queue, the recording will be started (or not) according to the queue Recording mode (see *Enable recording in the Queue configuration*). If this call is transferred to another queue, it will stop or start the recording according to the following table:

Action on Recording		*Recording mode of 'transferred to' Queue*		
		Recorded	*Recorded On Demand*	*Not Recorded*
Current Call Recording State	*Active*			Stop recording
	Paused	Unpause recording		Stop recording
	Disabled	Stop recording	Stop recording	

Stop recording upon external transfer

Recording is stopped when both parties of the call are external.

For example if an external incoming call is recorded and if, at some point, it is transferred to an external party, as soon as the transfer is completed, the recording of the incoming call will be stopped.

In the same way if an internal user makes an outgoing call which is recorded and then transfers it to another external party, as soon as the transfer is completed, the recording of the outgoing call will be stopped.

This behavior can be deactivated, see *configuration section*.

6.6.2 Recording filtering

For configuring this feature, see *Recording filtering configuration*.

Description

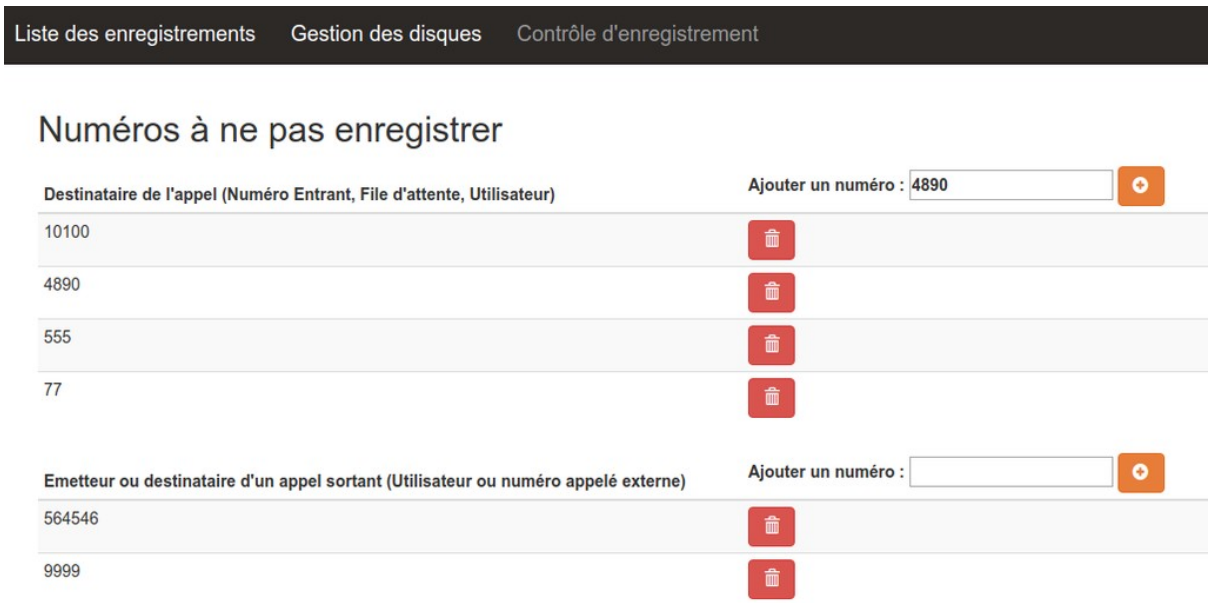
Recording server allows to prevent some numbers not to be recorded.

You can deactivate recording either

- for specific called numbers (called Incalls or called Queues or called Users),
- or, on outgoing calls, for calling Users internal numbers

Note: This feature is designed to activate recording globally (e.g. for every Queues) and then deactivate it for some Queues (e.g. for queue 1001)

To do so, navigate to : http://XIVOCC_IP:9400/recording_control and in page `Contrôle d'enregistrement` you can add or remove the number to disable recording on this number.



Limitations

- Recording can only be disabled for specific Incalls, Queues, Users and External called numbers.
- Recording can't be disabled for one Agent.
- Recording can only be disabled by the object number (to disable recording for one queue it must have a number).

6.7 Callbacks

6.7.1 Introduction

The goal of the callback system is to be able to perform scheduled outgoing calls. These requests can be completed with specific information such as a description or a personal name.

The core object of the callback system is the **callback request**. A callback request is made of the following fields:

- First name of person to call
- Last name
- Phone number
- Mobile phone number
- Company name
- Description
- Due date

Each callback request is associated to a predefined **callback period**, which represents the preferred interval of the day in which the call should be performed.

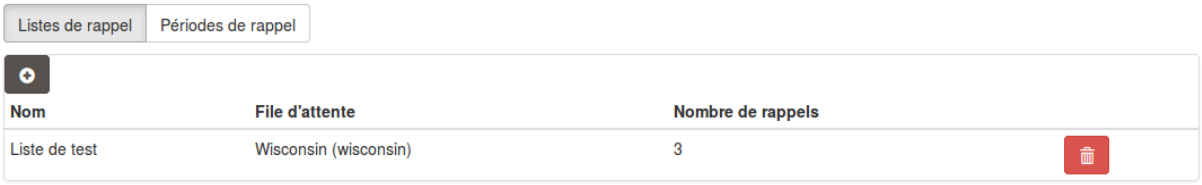
A callback request cannot exist on its own: it must be stored in a **callback list**, which is itself associated to a queue.

Once a callback request has been performed, it generates a **callback ticket**. This ticket sums up the original information of the callback request, but adding some new fields:

- Start date: date at which the callback request was actually performed
- Last update: date of the last modification of the ticket
- Comment
- Status : the result of the callback
- Agent: the Call Center agent who performed the callback

6.7.2 Callback Lists

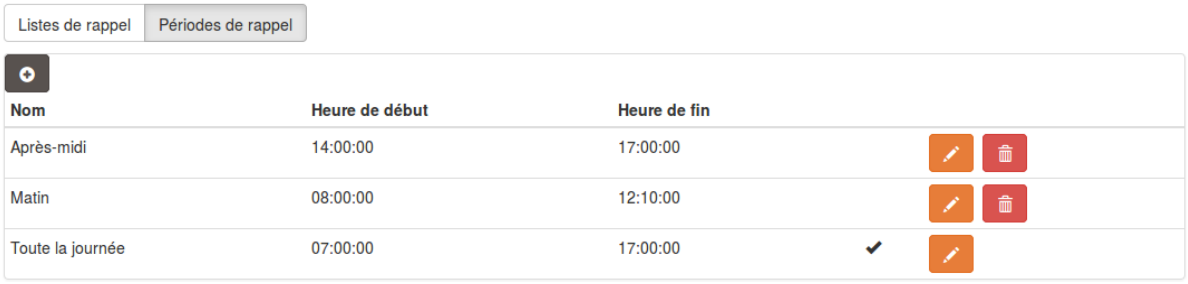
A callback list is an object which will contain callback request. It is associated to a queue, and several callback lists can be associated to the same queue. Callback lists are created using the **configuration manager**



Once created, a list can be populated whether through the *Callbacks tab* of the CCManager, or programmatically through the web services of the configuration server.

6.7.3 Callback Periods

A callback period represents an interval of the day, bounded by a start date and an end date. It can be set as the default interval, so that a newly created callback request will be associated to this period if none is specified.



6.7.4 Managing Callbacks Using CCManager

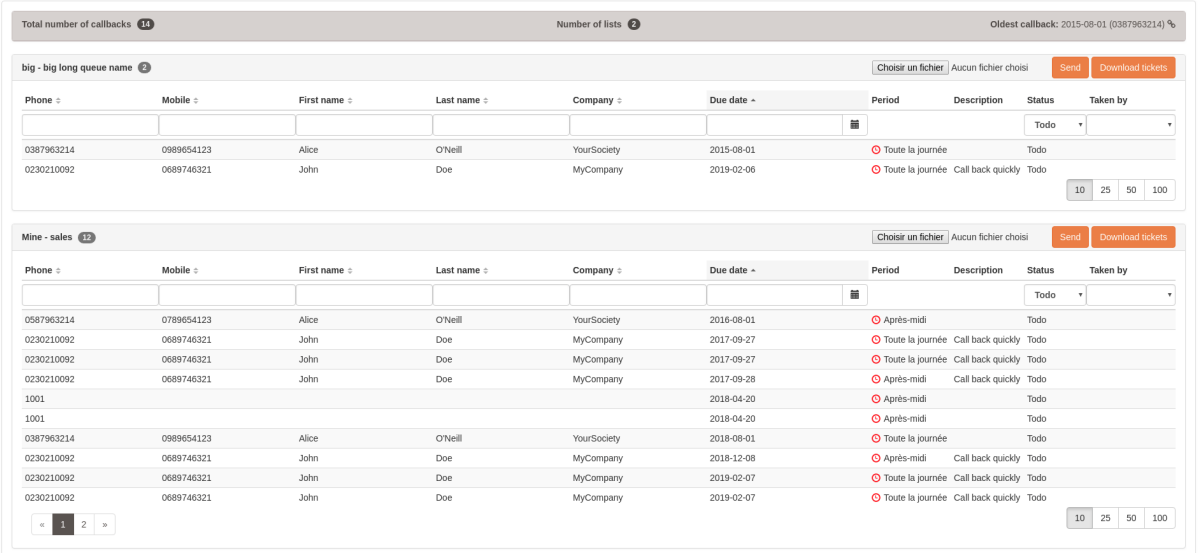
Using the CCManager callback view you may import a list of callbacks, monitor callback completion and download the associated tickets.

Importing Callbacks

Callbacks can be imported from a CSV file into a *callback list*.

Line delimiter must be a new line character and column separator must be one of: ‘|’ or ‘,’ or ‘;’. Columns can be optionaly enclosed by double-quote ‘”’.

The file must look like the following:




```
phoneNumber|mobilePhoneNumber|firstName|lastName|company|description|dueDate|period
0230210092|0689746321|John|Doe|MyCompany|Call back quickly||
0587963214|0789654123|Alice|O'Neill|YourSociety||2016-08-01|Afternoon
```

The header line must contain the exact field named described below:

Field Name	Description
phoneNumber	The number to call (at least either phoneNumber or mobilePhoneNumber is required)
mobilePhoneNumber	Alternate number to call
firstName	The contact first name (optional)
lastName	The contact last name (optional)
company	The contact company (optional)
description	A text that will appear on the agent callback pane
dueDate	The date when to callback, using ISO format: YYYY-MM-DD, ie. 2016-08-01 for August, 1st, 2016. If not present the next day will be used as dueDate (optional)
period	The name of the period as defined in *callback list*. If not present, the default period will be used (optional)

When an agent takes a callback, the column `Taken by` is updated with the number of the agent. The callback disappears when it is processed.

Exporting Callbacks Tickets

The tickets of the processed callbacks can be downloaded by clicking on the `Download tickets` button.

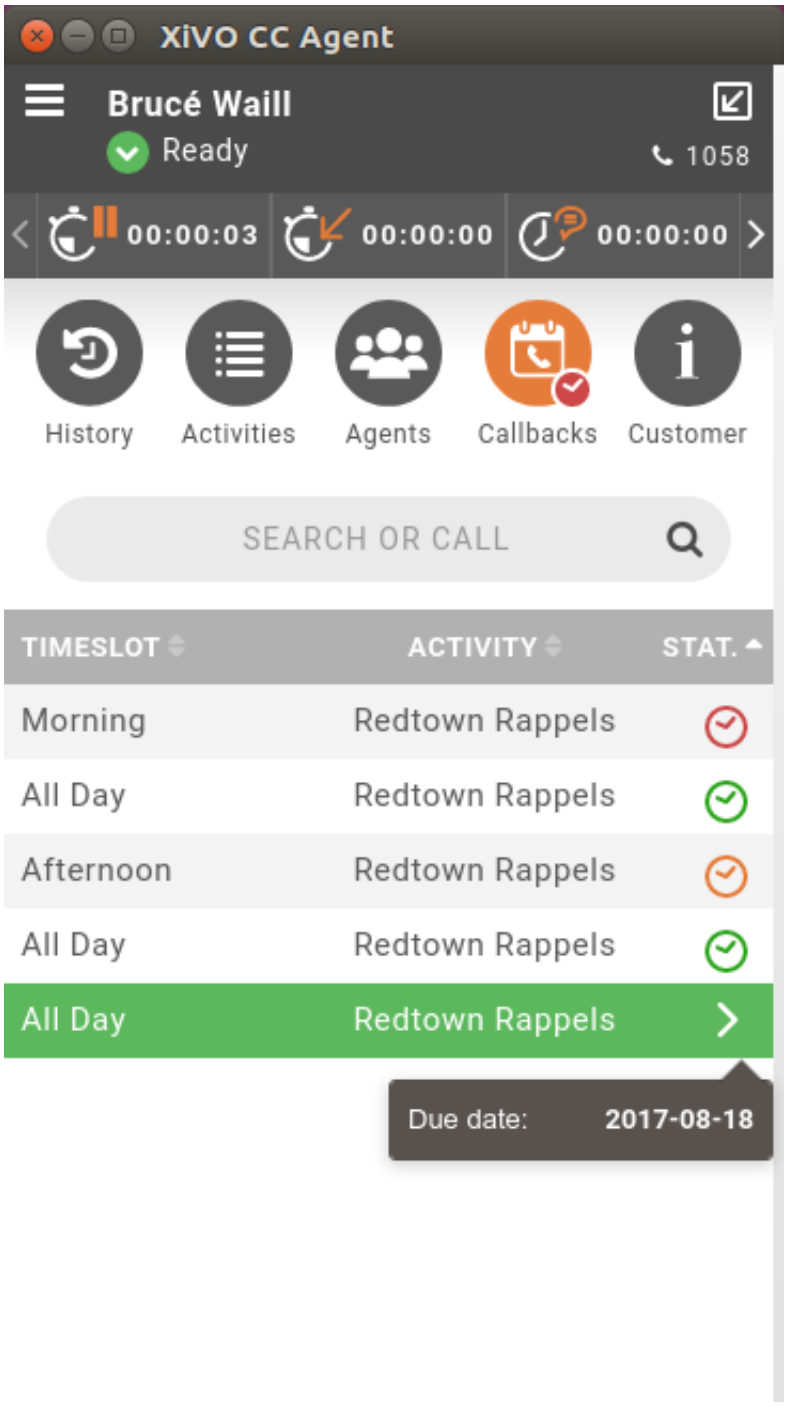
The downloaded file is a csv file with the comma ',' as delimiter.

6.7.5 Processing Callbacks with CCAgent

The agent can see the *callbacks* related to the queues he is logged on. They are available in the `Callbacks` menu. A notification is also shown to display pending callbacks in menu to know status at any time and from any other screen of the application.

On this page, the agent only has access to basic information about the callback: activity and due date, On the left of each callback line, a colored clock indicates the temporal status of this callback:

- *yellow* if the callback is to be processed later
- *green* if we are currently inside the callback period
- *red* if the callback period is over

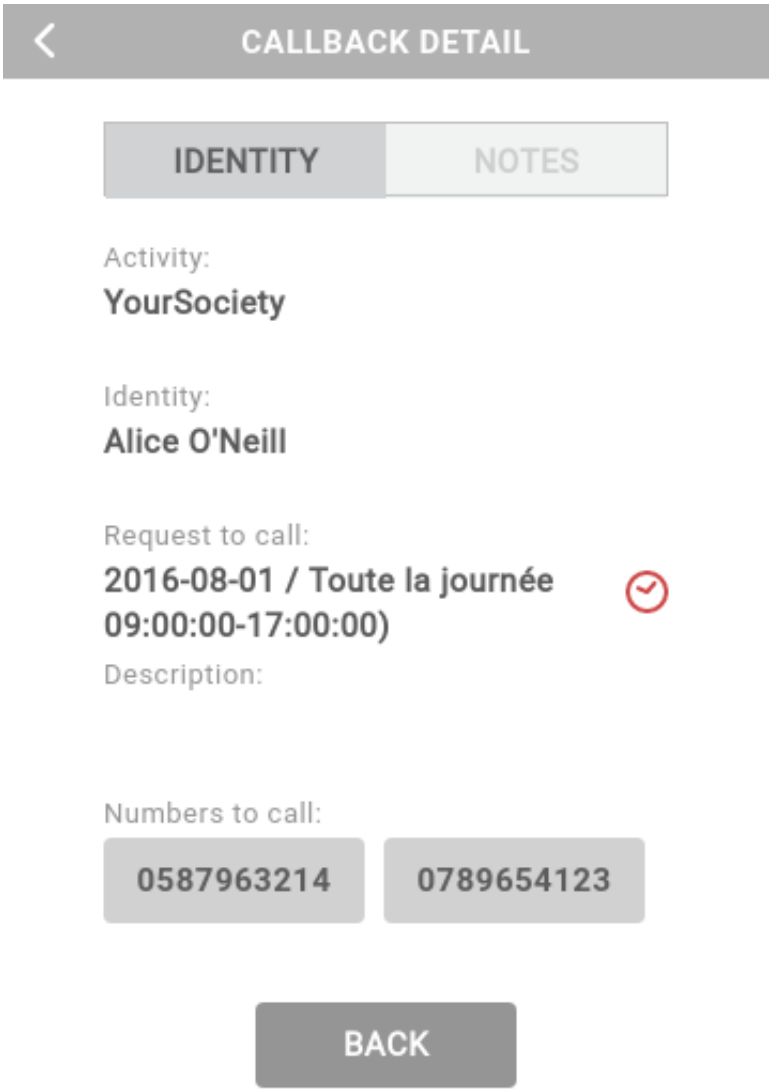


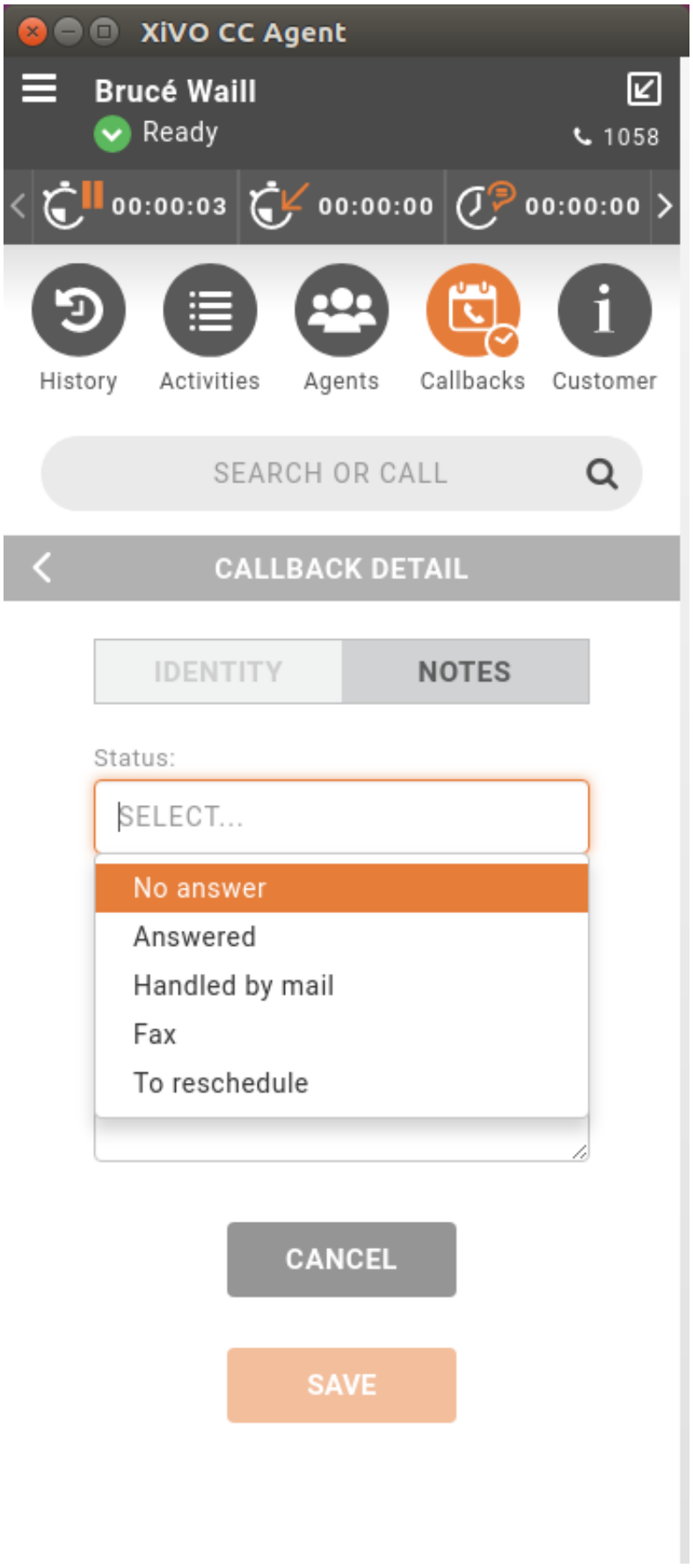
To process one of these callbacks, the agent must click on one of the callbacks line.

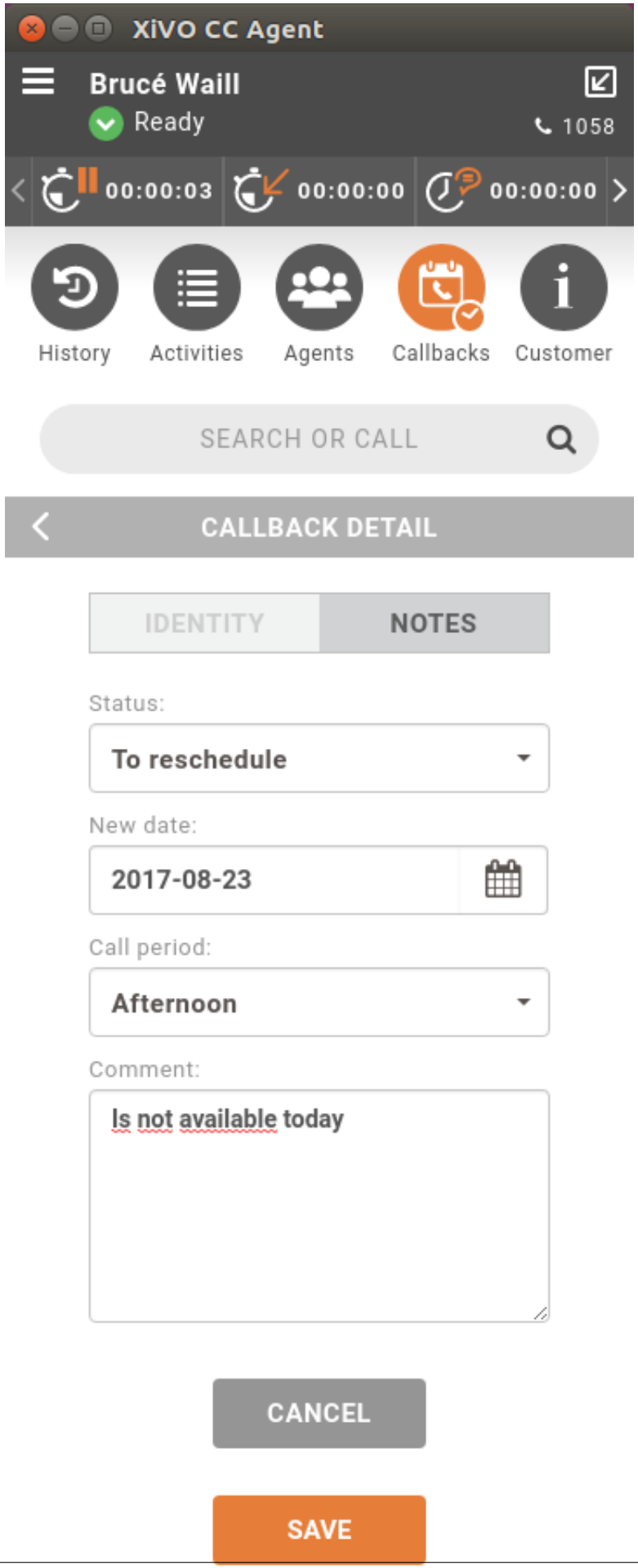
To launch the call, the agent must click on one of the available phone numbers.

Once the callback is launched, the status can be changed and a comment can be added.

If you set 'To reschedule' as status, the callback can be rescheduled at a later time and another period:


CALLBACK DETAIL
IDENTITY
NOTES
Activity:
YourSociety
Identity:
Alice O'Neill
Request to call:
2016-08-01 / Toute la journée
09:00:00-17:00:00)
Description:
Numbers to call:
0587963214
0789654123
BACK


XiVO CC Agent
Brucé Waill
Ready
1058
00:00:03
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
CALLBACK DETAIL
IDENTITY
NOTES
Status:
SELECT...
No answer
Answered
Handled by mail
Fax
To reschedule
CANCEL
SAVE


XiVO CC Agent
Brucé Waill
Ready
1058
00:00:03
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
CALLBACK DETAIL
IDENTITY
NOTES
Status:
To reschedule
New date:
2017-08-23
Call period:
Afternoon
Comment:
Is not available today
CANCEL
SAVE

Other statuses are available to be set and will close callback once saved :

- *Answered* if caller accepted the call
- *NoAnswer* if caller were unreachable
- *Fax* if callback has been resolved thanks to a Fax message
- *Handled by mail* if callback has been resolved thanks to an E-mail

Clicking on the calendar icon next to the "New due date" field, will popup a calendar to select another callback date.

6.8 Profile Management

With what is called the Configuration Management Server one can specify the profile of a user.

To do that one has to:

- log in the Configuration management server accessible at `https://XIVO_PBX/configmgt`,
- as *Superadmin* (whose login is `avencall`)

When logged in as *Superadmin* one can:

- attribute a profile (see *Profiles Definition*) to a user (note that the user must have a CTI login to be listed),
- create callback lists (see *Callback Lists*)
- and create callback periods (see *Callback Periods*)

6.8.1 Profiles Definition

Profiles and their rights are summed up in the following table:

Profile	Application			
	CC Manager		**Recording**	
	Access	*Actions*	*Access*	*Actions*
Administrator	Yes	All	Yes	• all recording • recordings filtering
Supervisor[4] • w/ *Recording*	Yes	• All on its queues[1] • Recording switch	Yes	Its queues'[1] recordings
Supervisor[4] • wo/ *Recordings*		• All on its queues[1] • No recording switch	No	N.A.[2]
Teacher	No	N.A.[2]	Yes	• Its queues'[1] recordings • During configured period
No profile	No[3]	N.A.[2]	No	N.A.

Supervisor Profile Specificities

Compared to other profiles, supervisor profile has specific rights. That is, when logged in the Configuration Management Server, a *Supervisor** can:

- attribute a profile **Teacher** to another user (to a subset of its own attributed queues),
- and create callbacks list (on its own attributed queues)

6.8.2 Impact on CC Agent

Any agent can log in the CC Agent application.

Then, when logged in the CC Agent, this agent can see all the queues in the Activities view.

Important: **But** if an agent *is also a Supervisor*. When this agent is logged in the CC Agent, it will see **only** the set of queues and agents as it was given him in the *Supervisor* profile via the Configuration Management Server.

[4] See *Supervisor Profile Specificities* section
[1] i.e. attributed queues to the user via the Config Mgt. Note also that agents groups should be attributed accordingly.
[2] Not Applicable
[3] Depending on the *Access authorizations in CCManager* configuration

6.9 Skills-Based Routing

6.9.1 Introduction

> *Skills-based routing (SBR), or Skills-based call routing, is a call-assignment strategy used in call centres to assign incoming calls to the most suitable agent, instead of simply choosing the next available agent. It is an enhancement to the Automatic Call Distributor (ACD) systems found in most call centres. The need for skills-based routing has arisen, as call centres have become larger and dealt with a wider variety of call types.*
>
> —Wikipedia

In this respect, skills-based routing is also based on call distribution to agents through waiting queues, but one or many skills can be assigned to each agent, and call can be distributed to the most suitable agent.

In skills-based routing, you will have to find a way to be able to tag the call for a specific skill need. This can be done for example by entering the call distribution system using different incoming call numbers, using an IVR to let the caller do his own choice, or by requesting to the information system database the customer profile.

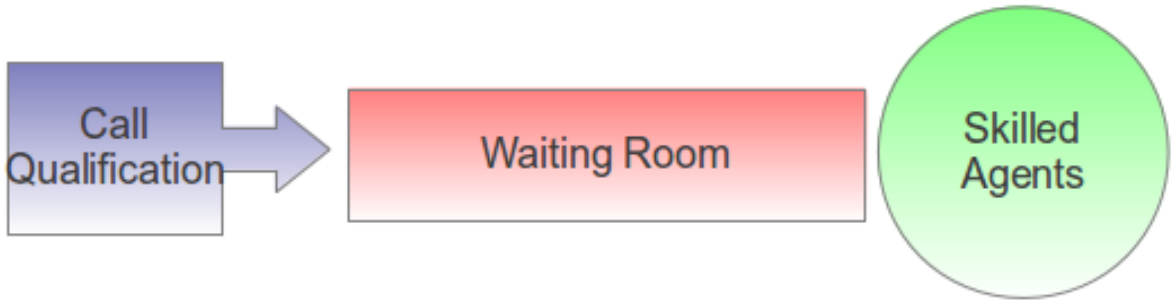


Fig. 2: Skills-Based Routing

6.9.2 Getting Started

- Create the skills
- Apply the skills to the agents
- Create the skill rule sets
- Assign the skill rule sets using a configuration file
- Apply the skill rule sets to call qualification, i.e. incoming calls by using the preprocess subroutine field

Note that you shouldn't use skill based routing on a queue with queue members of type user because the behaviour is not defined and might change in a future XiVO version.

6.9.3 Skills

Skills are created using the menu *Services* → *Call center* → *Skills*. Each skill belongs to a category. First create the category, and in this category create different skills.

Note that a skill name can't contain upper case letters and must be globally unique (i.e. the same name can't be used in two different categories).

Once all the skills are created you may apply them to agents. Agents may have one or more skills from different categories.

It is typical to use a value between 0 and 100 inclusively as the weight of a skill, although any integer is accepted.

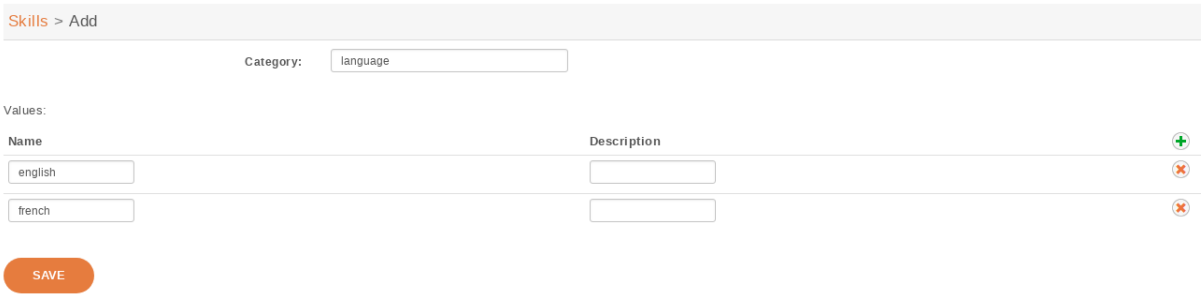


Fig. 3: Skills Creation

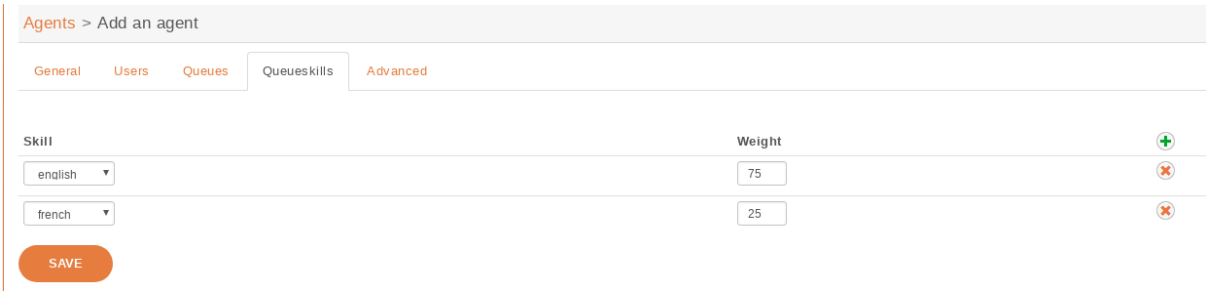


Fig. 4: Apply Skills to Agents

6.9.4 Skill Rule Sets

Once skills are created, rule sets can be defined.

A rule set is a list of rules. Here's an example of a rule set containing 2 rules:

1. WT < 60, english > 50
2. english > 0

The first rule of this rule set can be read as:

> If the caller has been waiting for less than 60 seconds (WT < 60), only try to call agents which have the skill "english" set to a value higher than 50; otherwise, go to the next rule.

And the second rule can be read as:

> Only try to call agents which have the skill "english" set to a value higher than 0.

Let's examine some simple scenarios, because there's actually some subtilities on how calls are distributed. We will suppose that we have a queue with the default settings and the following members:

- Agent A, with skill english set to 75
- Agent B, with skill english set to 25

Scenario 1

Given:

- Agent A is logged and not in use
- Agent B is logged and not in use
- There is no call in the queue

When a new call enters the queue, then it is distributed to Agent A. As long as Agent A is available and doesn't answer the call, the call will never be distributed to Agent B, even after 60 seconds of waiting time.

When another call enters the queue, then after 60 seconds of waiting time, this call will be distributed to Agent B (and the first call will still be distributed only to Agent A).

The reason is that there's a difference between a call that is being distributed (i.e. that is making agents ring) and a call that is waiting for being distributed. When a call is being distributed to a set of members, no other rule is tried as long as there's at least 1 of these members available.

Scenario 2

Given:

- Agent A is not logged
- Agent B is logged and not in use
- There is no call in the queue

When a new call enters the queue, then it is *immediately* distributed to Agent B.

The reason is that when there's no logged agent matching a rule, the next rule is immediately tried.

Rules

Each rule set is composed of rules, and each rule has two parts, separated by a comma:

- the first part (optional) is the *"dynamic part"*
- the second part is the *"skill part"*

Each part contains an expression composed of operators, variables and integer constants.

Operators

The following operators can be used inside rules:

Comparison operators:

- operand1 ! operand2 (is not equal)
- operand1 = operand2 (is equal)
- operand1 > operand2 (is greater than)
- operand1 < operand2 (is lesser than)

Logical operators:

- operand1 & operand2 (both are true)
- operand1 | operand2 (at least one of them are true)

'!' is the operator with the higher priority, and '|' the one with the lower priority. You can use parentheses '()' to change the priority of operations.

Dynamic Part

The dynamic part can reference the following variables:

- WT
- EWT

The waiting time (WT) is the elapsed time since the call entered the queue. The time the call pass in an IVR or another queue is not taken into account.

The estimated waiting time (EWT) has never fully worked. It is mentioned here only for historical reason. You should not use it. It might be removed in a future XiVO version.

Examples

- WT < 60

Skill Part

The skill part can reference any skills name as variables.

You can also use meta-variables, starting with a '$', to substitute them with data set on the Queue() call. For example, if you call Queue() with the skill rule set argument equal to:

```
select_lang(lang=german)
```

Then every `$lang` occurrence will be replaced by 'german'.

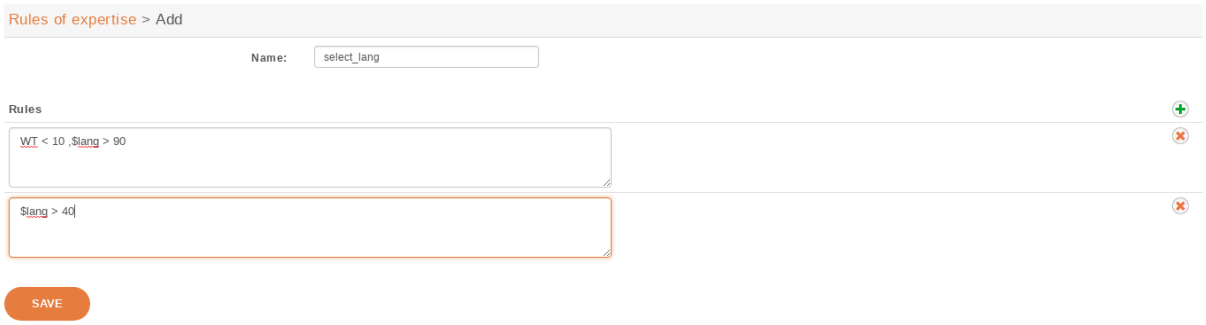


Fig. 5: Create Skill Rule Sets

Examples

- english > 50
- technic ! 0 & ($os > 29 & $lang > 39 | $os > 39 & $lang > 19)

Evaluation

Note that the expression:

english | french

is equivalent to:

english ! 0 | french ! 0

Sometimes, a rule references a skill which is not defined for every agent. For example, given the following rule:

english > 0 | english < 1

Then, for an agent which has the skill english defined, the result of this expression is always true. For an agent which does not have the skill english defined, the result of this expression is always false.

Said differently, an agent without a skill X is not the same as an agent with the skill X set to the value 0.

Technically, this is what is happening when evaluating the rule "english > 0" for an agent without the skill english:

```
  english > 0
=     <Substituing english with the agent value>
  "undefined" > 0
=     <A comparison with "undefined" in at least one operand yields undefined>
  "undefined"
=     <In a boolean context, "undefined" is equal to false>
  false
```

This behaviour applies to every comparison operators.

Also, the syntax that is currently accepted for comparison is always of the form:

```
variable cmp_op constant
```

Where "variable" is a variable name, "cmp_op" is a comparison operator and "constant" is an integer constant. This means the following expressions are not accepted:

- 10 < english (but english > 10 is accepted)
- english < french (the second operand must be a constant)
- 10 < 11 (the first operand must be a variable name)

6.9.5 Apply Skill Rule Sets

A skill rule set is attached to a call using a bit of dialplan. This dialplan is stored in a configuration file you may edit using menu *Services* → *IPBX* → *Configuration Files*.

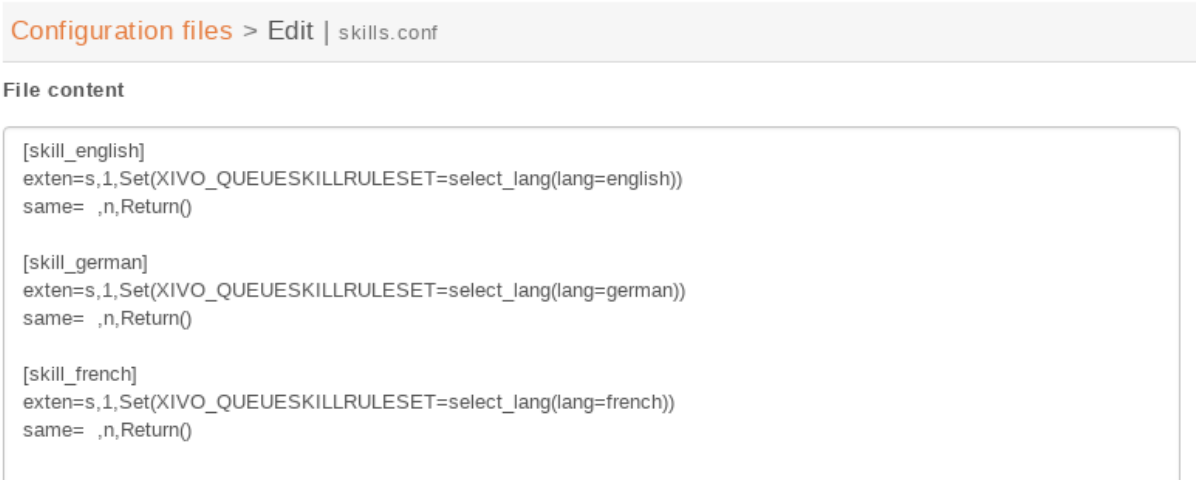


Fig. 6: Use Rule Set In Dialplan

In the figure above, 3 different languages are selected using three different subroutines.

Each of this different selections of subroutines can be applied to the call qualifying object. In the following example language selection is applied to incoming calls.

Example

Configuration file for simple skill selection :

```
[simple_skill_english]
exten = s,1,Set(XIVO_QUEUESKILLRULESET=english_rule_set)
same  =   n,Return()

[simple_skill_french]
exten = s,1,Set(XIVO_QUEUESKILLRULESET=french_rule_set)
same  =   n,Return()
```

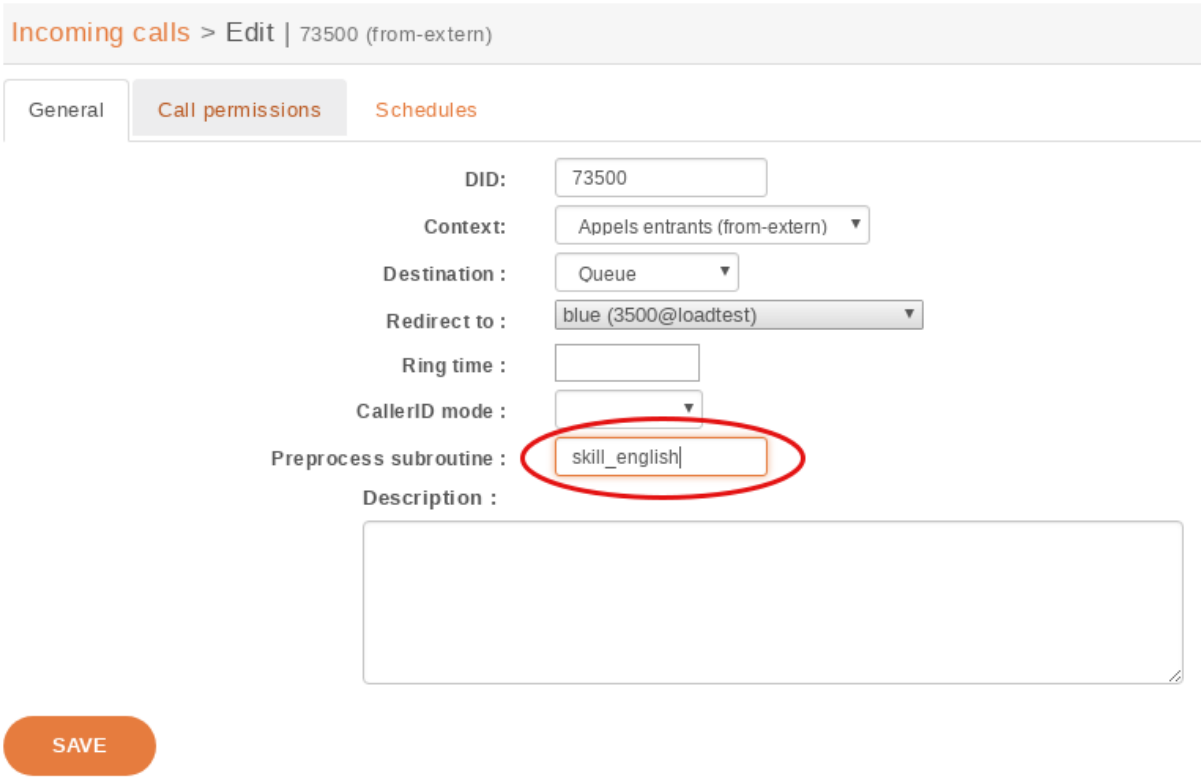


Fig. 7: Apply Rule Set to Incoming Call

6.9.6 Monitoring

You may monitor your waiting calls with skills using the asterisk CLI and the command `queue show <queue_name>`:

```
xivo-jylebleu*CLI> queue show services
services has 1 calls (max unlimited) in 'ringall' strategy (0s holdtime, 2s␣
↪talktime), W:0, C:1, A:10, SL:0.0% within 0s
  Members:
     Agent/2000 (Not in use) (skills: agent-1) has taken no calls yet
     Agent/2001 (Unavailable) (skills: agent-4) has taken no calls yet
  Virtual queue english:
  Virtual queue french:
     1. SIP/jyl-dev-assur-00000017 (wait: 0:05, prio: 0)
  Callers:
```

You may monitor your skills groups with the command `queue show skills groups <agent_name>`:

```
xivo-jylebleu*CLI> queue show skills groups <PRESS TAB>
agent-2   agent-3   agent-4   agent-48  agent-7   agent-1
xivo-jylebleu*CLI> queue show skills groups agent-1
Skill group 'agent-1':
  - bank            : 50
  - english         : 100
```

You may monitor your skills rules with the command `queue show skills rules <rule_name>`:

```
xivo-jylebleu*CLI> queue show skills rules <PRESS TAB>
english      french       select_lang
xivo-jylebleu*CLI> queue show skills rules english
Skill rules 'english':
```

(continues on next page)

(continued from previous page)

```
  => english>90
```

6.10 Reporting and statistics

6.10.1 Introduction

Pack reporting is a part of the XivoCC. It aims at computing historical statistics, which are stored in the **xivo_stats** database. Sample reports based on them are accessible in **SpagoBI**.

6.10.2 Known limitations

- Queue members should only be agents. If users are members of a queue, their statistics will be incomplete.
- Configuration modifications on the XiVO (such as an agent deletion) are replicated on the statistics server, and their previous value is not kept. However, statistics history is preserved.
- POPC statistics are not accurate.
- Calls longer than 4 hours are considered as unterminated calls and therefore communication time is set to 0 for these calls.
- If two agents are associated to the same call, they will have the same hold time for this call.
- Transfer statistics limitations :
  - Given two queues Q1 and Q2, two agents A1 and A2, and an external caller C.
    - C calls Q1 and A1 answers
    - A1 transfers to Q2 and A2 answers
    - A2 transfers to the outside

  Then the second transfer is seen as a transfer to the outside.
  - Given one queue Q1, Two agents A1 and A2 and an external caller C.
    - C calls Q1 and A1 answers
    - A1 transfers to A2 through internal phone number
    - A1 completes the transfer

  Then the call between A2 and C is not computed at all in statistics.

6.10.3 Reporting Architecture

6.10.4 Attached Data

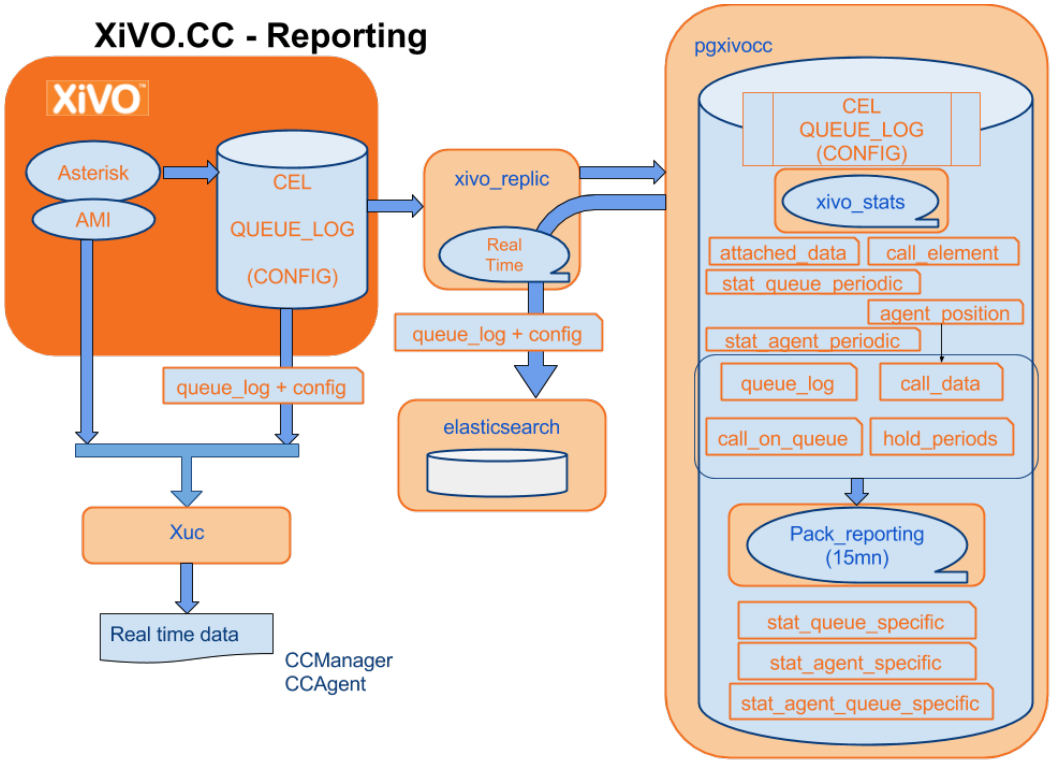
The pack reporting allows to attach as mush data as wished to a given call, in order to find them in the reporting database for future use. This data must be in the form of a set of key-value pairs.

To attach data to a call, you must use the dialplan's **CELGenUserEvent** application:

```
exten = s,n,CELGenUserEvent(ATTACHED_DATA,my_key=my_value)
```

This will insert the following tuple in the **attached_data** table:

key	value
my_key	my_value



6.10.5 Using SpagoBi

Important: All report samples (see *Upload Sample Reports*) bundled with XiVOCC are aimed to work with french date format. By default dates are picked in french format (day/month/year). Either you need to type them explicitly in your locale format either you can set Locale to french thanks to Flag menu in Spago. See #1662

Scheduling reports

Due to a SpagoBi limitation, when scheduling reports using string parameters, you need to enter manualy the parameter using comma and quotes to separate values. Example for queues support, technical and sales, the parameters of the schedulers must be filled as follow:

'support','technical','sales'

You may use Jaspersoft® Studio to design you own reports.

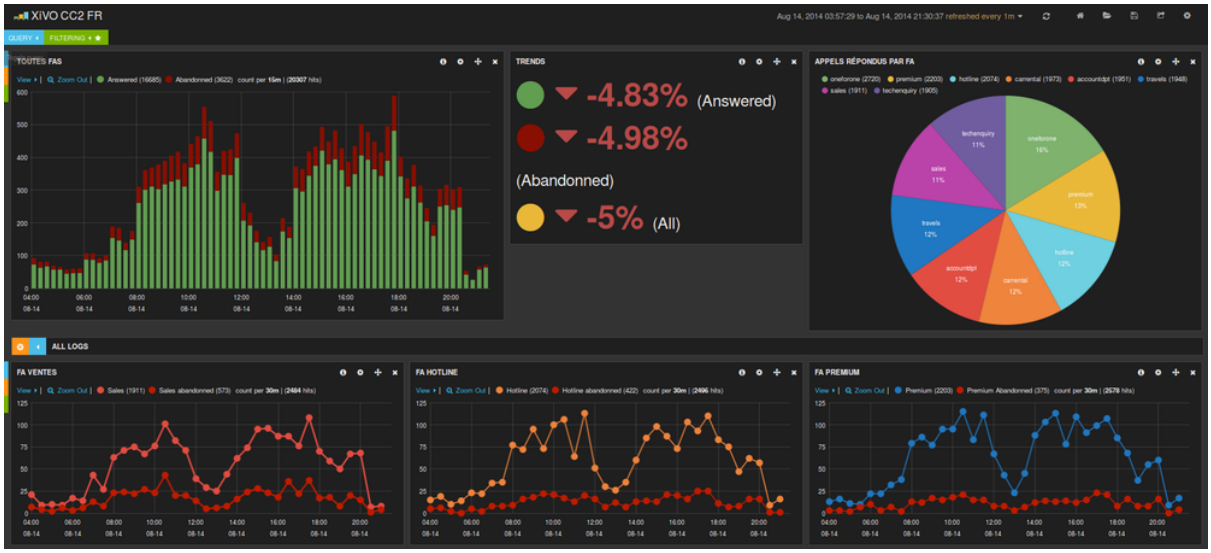
Warning: Due to a limitation in SpagoBi do not use single quotes in your reports name, otherwise you will not be able to schedule your report (see #213).

6.10.6 Using Kibana

Kibana is a web tool used to compute statistics based on Elasticsearch content. The reports packaged with the Pack reporting give you an outline of your recent call center activity. Here is a Kibana sample panel:

Graphs are based on the queue_log table, enriched with agent names and agent groups, and inserted into an Elasticsearch index. It contains avents about calls placed on queues, and events about agent presences.

For each entry in the queue_log index, the following attributes are available:



- queudis'downloads page playname : Queue display name
- data1: basic queue_log data, with a different meaning according to the event
- callid : Call unique identifier, generated by Asterisk
- event : Call or agent status event - please see below
- agentnumber: Agent number
- queuename : Technical queue name
- groupname : Agent group name
- queuetime: Time of the event
- agentname : Name of the agent, if available

The event can be one of the following (for a detailed explanation, please refer to https://wiki.asterisk.org/wiki/display/AST/Queue+Logs):

- Call events:
  - FULL
  - CONNECT
  - EXITEMPTY
  - CLOSED
  - EXITWITHTIMEOUT
  - JOINEMPTY
  - ABANDON
  - ENTERQUEUE
  - TRANSFER
  - COMPLETEAGENT
  - COMPLETECALLER
  - RINGNOANSWER
- Agent or queue event:
  - ADDMEMBER
  - PAUSEALL

- PAUSE
- WRAPUPSTART
- UNPAUSE
- UNPAUSEALL
- PENALTY
- CONFIGRELOAD
- AGENTCALLBACKLOGIN
- AGENTCALLBACKLOGOFF
- REMOVEMEMBER
- PRESENCE
- QUEUESTART

6.10.7 Database schema

Glossary

uniqueId :

- unqiueId are Ids generated by asterisk for each call leg.
- When **A** calls **B** we get a call leg for **A** and one for **B**, each one is a separate uniqueId.
- in *cel* asterisk table you can find all event linked to a call leg by looking for it in the column *uniqueId*.

linkedId :

- the column *linkedId* in *cel* Asterisk table contains a uniqueId which links the two legs of the call.
- The leg which initiates the call (**A** leg) will have the same value for both *uniqueId* and *linkedId* columns.
- The other leg (**B** leg) will have the uniqueId of **B** in *uniqueId* column and will have the uniqueId of **A** in *linkedId* one.

call_data

Each line in *call_data*, correspond to a unique call and contains only one uniqueId of one call leg. This table is aggregated in near real time with the data of *cel* Asterisk table.

Important: End of a call is considered when phone is hung up, we don't consider the time of transfers that can be done besides initial call.

Column	Type	Description
id	INTEGER	
uniqueid	VARCHAR	Call unique reference, generated by Asterisk
dst_num	VARCHAR	Called number
start_time	TIMESTAMP	Call start time
answer_time	TIMESTAMP	Call answer time
end_time	TIMESTAMP	Call end time (phone is hung up)
status	status_type	Call status. Beware: only *answered* is properly filled.
ring_duration_on_answer	INTEGER	Ring time of the endpoint answering the call, in seconds
transfered	BOOLEAN	True if the call has been transfered
call_direction	call_direction_type	Call direction (''incoming" : call from the outside, received by XiVO ; ''outgoing" : call to the outside, originated by an endpoint associated to XiVO ; ''internal" : call taking place entirely inside the XiVO)
src_num	VARCHAR	Calling number
transfer_direction	call_direction_type	Indicates the transfer direction, if relevant
src_agent	VARCHAR	Agent originating the call
dst_agent	VARCHAR	Agent receiving the call, if it is a direct call on an agent. Not filled when the call is destined to a queue
src_interface	VARCHAR	Interface originating the call (in the Asterisk sense, ex : SCCP/01234)

attached_data

Data attached to the call (cf. *Attached Data*)

Column	Type	Description
id	INTEGER	
id_call_data	INTEGER	Id of the associated tuple in *call_data*
key	VARCHAR	Name of the attached data
value	VARCHAR	Value of the attached data

call_element

Part of a call matching the reaching of an endpoint

Column	Type	Description
id	INTEGER	
call_data_id	INTEGER	Id of the associated tuple in *call_data*
start_time	TIMESTAMP	Time at which the endpoint was called
answer_time	TIMESTAMP	Answer time for the endpoint
end_time	TIMESTAMP	End time of this call part
interface	VARCHAR	Endpoint interface

call_on_queue

Calls on a queue

Column	Type	Description
id	INTEGER	
callid	VARCHAR	Call unique reference, generated by Asterisk
queue_time	TIMESTAMP	Time of entrance in the queue
total_ring_time	INTEGER	Total ring time, in seconds (includes ringing of non-answered calls)
answer_time	TIMESTAMP	Answer time
hangup_time	TIMESTAMP	Hangup time
status	call_exit_type	Call status (*full*: full queue; *closed*: closed queue; *joinempty*: call arrived on empty queue; *leaveempty* : exit when queue becomes empty; *divert_ca_ratio* : call redirected because the ratio waiting calls/agents was exceeded ; *divert_waittime*: call redirected because estimated waiting time was exceeded; *answered*: call answered ; *abandoned*: call abandoned; *timeout* : maximum waiting time exceeded)
queue_ref	VARCHAR	Technical queue name
agent_num	VARCHAR	Number of the agent taking the call, if relevant

hold_periods

Hold periods

Column	Type	Description
id	INTEGER	
linkedid	VARCHAR	Call unique reference, generated by Asterisk
start	TIMESTAMP	Hold start time
end	TIMESTAMP	Hold end time

stat_queue_periodic

Statistics aggregated by queue and time interval (15 minutes)

Column	Type	Description
id	INTEGER	
time	TIMESTAMP	Start time of the considered interval
queue	VARCHAR	Queue technical name
answered	INTEGER	Number of answered calls
abandoned	INTEGER	Number of abandoned calls
total	INTEGER	Total number of calls received on the queue (which excludes the calls dissuaded before entering the queue)
full	INTEGER	Number of calls arrived on a full queue (diversion before entering the queue)
closed	INTEGER	Number of calls arrived on a closed queue, outside of the configured schedules (diversion before entering the queue)
joinempty	INTEGER	Number of calls arrived on an empty queue (diversion before entering the queue)
leaveempty	INTEGER	Number of calls redirected because of a queue becoming empty
divert_ca_ratio	INTEGER	Number of calls arrived when the calls / available agents ratio is exceeded (diversion before entering the queue)
divert_waittime	INTEGER	Number of calls arrived when the estimated waiting time is exceeded (diversion before entering the queue)
timeout	INTEGER	Nombre of calls redirecting because maximum waiting time is exceeded

stat_agent_periodic

Statistics aggregated by agent and time interval (15 minutes)

Column	Type	Description
id	INTEGER	
time	TIMESTAMP	Start time of the considered interval
agent	VARCHAR	Agent number
login_time	INTERVAL	Login time
pause_time	INTERVAL	Pause time
wrapup_time	INTERVAL	Wrap-up time

stat_queue_specific

Statistics aggregated by queue, called number and time interval (15 minutes)

Column	Type	Description
time	TIMESTAMP	Start time of the considered interval
queue_ref	VARCHAR	Technical name of the queue
dst_num	VARCHAR	Called number
nb_offered	INTEGER	Number of presented calls
nb_abandoned	INTEGER	Number of abandoned calls
sum_resp_delay	INTEGER	Wait time, in seconds
answer_less_t1	INTEGER	Number of calls answered in less than t1 seconds
abandoned_btw_t1_t2	INTEGER	Number of calls abandoned between t1 and t2 seconds
answer_btw_t1_t2	INTEGER	Number of calls answered between t1 and t2 seconds
abandoned_more_t2	INTEGER	Number of calls answered in more than t2 seconds
communication_time	INTEGER	Total communication time in seconds
hold_time	INTEGER	Total hold time in seconds
wrapup_time	INTEGER	Total wrap-up time in seconds

The thresholds t1 and t2 are configurable:

- in the table queue_specific_time_period for the default values in seconds. Installation values are t1=15 seconds and t2=20 seconds. Data is saved in the form of *(name, seconds)* pairs, for example : ('t1', 15).
- in the table queue_threshold_time for values specific to a queue. Data is saved in the form of a tuple (queue name, t1, t2).

stat_agent_specific

Statistics aggregated by agent and time interval (15 minutes)

Important: Hold times are considered as conversation time and so are included.

Column	Type	Description
time	TIMESTAMP	Start time of the considered interval
agent_num	VARCHAR	Agent number
nb_offered	INTEGER	Number of calls presented from a queue
nb_answered	INTEGER	Number of calls answered from a queue
conversation_time	INTEGER	Conversation time on incoming calls from a queue (ACD), in seconds
ringing_time	INTEGER	Ringing time on incoming cals from a queue (ACD), in seconds
nb_outgoing_calls	INTEGER	Number of calls emitted to the outside
conversation_time_outgoing_calls	INTEGER	Conversation time in calls emitted to the outside, in seconds
hold_time	INTEGER	Hold time for any kind of calls (ACD, internal and outgoing) in seconds
nb_received_internal_calls	INTEGER	Number of received internal calls
conversation_time_received_internal_calls	INTEGER	Conversation time on received internal calls, in seconds
nb_transfered_intern	INTEGER	Number of calls coming from a queue and transfered to an internal destination
nb_transfered_extern	INTEGER	Number of calls coming from a queue and transfered to an external destination
nb_emitted_internal_calls	INTEGER	Number of emitted internal calls
conversation_time_emitted_internal_calls	INTEGER	Conversation time on emitted internal calls, in seconds
nb_incoming_calls	INTEGER	Number of received incoming calls
conversation_time_incoming_calls	INTEGER	Conversation time on received incoming calls, in seconds

stat_agent_queue_specific

Statistics aggregated by queue, called number, agent and time interval (15 minutes)

Column	Type	Description
time	TIMESTAMP	Start time of the considered interval
agent_num	VARCHAR	Agent number
queue_ref	VARCHAR	Technical name of the queue
dst_num	VARCHAR	Called number
nb_answered_calls	INTEGER	Number of answered calls
communication_time	INTEGER	Communication time, in seconds
hold_time	INTEGER	Hold time, in seconds
wrapup_time	INTEGER	Wrap-up time, in seconds

agentfeatures

Gather information about agent profile

Column	Type	Description
id	INTEGER	
numgroup	INTEGER	Agent group number
number	VARCHAR	Agent number (line number)
firstname	VARCHAR	Agent first name
lastname	VARCHAR	Agent last name

agent_position

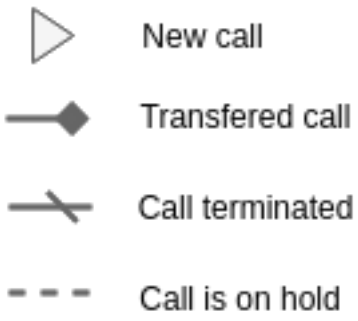
Line number used by agents

Column	Type	Description
agent_num	VARCHAR	Agent number
line_num	VARCHAR	Line number of the device used
start_time	TIMESTAMP	Begin date of line use
end_time	TIMESTAMP	End date of line use
sda	VARCHAR	Line direct inward dial

6.10.8 Sample of statistic data from various call flow

This section gives an overview of what kind of data can be found in the different statistics tables when performing usual contact center call flow.

For the folowing diagrams, here the legend that would apply to understand interactions:



Single call put on hold

Let's take the assumption that one Agent (*A1* with id *8000* and phone number *1000*) calls an internal User (*U1* with phone number *1001*).

This simple call flow generates in *call_data* 1 line (simplified table to keep only important fields):

id	uniqueId	dst_num	answer_time	end_time	src_agent
1	12345678.9	1001	2018-12-01 15:02:00	2018-12-01 15:22:00	8000

also get in *hold_period* 1 line:

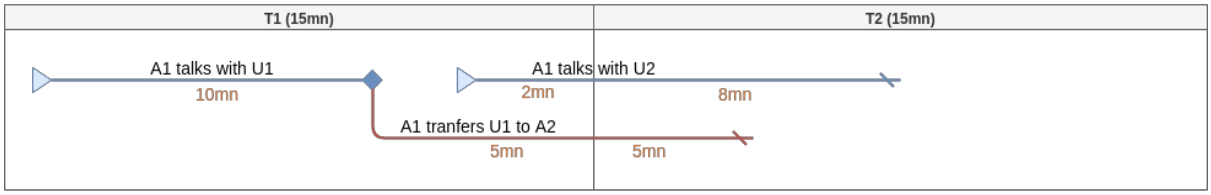
id	linkedId	start	end
1	12345678.9	2018-12-01 15:13:00	2018-12-01 15:17:00

Statistics (done by pack-reporting for spago reports) generates then *stat_agent_specific*:

time	agent_num	nb_emitted_internal_calls	conversation_time_emitted_internal_calls	hold_time
2018-11-29 15:15:00	8000	1	720	120
2018-11-29 15:30:00	8000	1	480	120

Single call, direct transfer then new call

Let's take the assumption that one Agent (*A1* with id *8000* and phone number *1000*) calls an internal User (*U1* with phone number *1001*). This call is transfered to Agent *A2* (with id *8002*) blindly. Then *A1* calls internal another User (*U2* with phone number *1002*)



This call flow generates in *call_data* 2 lines (simplified table to keep only important fields) one for each call (*U1* and *U2*):

id	uniqueId	dst_num	answer_time	end_time	src_agent	dst_agent	transfered	transfer_direction
1	12345678.9	1001	2018-12-01 15:02:00	2018-12-01 15:12:00	8000	8002	true	internal
2	12345679.0	1002	2018-12-01 15:13:00	2018-12-01 15:23:00	8000			

and get in *transfers* 1 line (where *callidto* id the *uniqueId* of the leg of Agent *A2* with user U1):

id	callidfrom	callidto
1	12345678.9	98765432.1

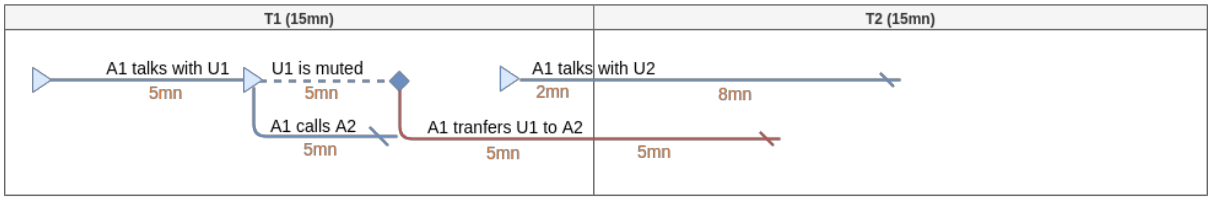
Statistics (done by pack-reporting for spago reports) generates then *stat_agent_specific*:

time	agent_num	nb_emitted_internal_calls	conversation_time_emitted_internal_calls	hold_time
2018-11-29 15:15:00	8000	1	720	0
2018-11-29 15:30:00	8000	1	480	0

Important: We don't consider the time of the transfer besides initial call and only time of established calls.

Single call, attented transfer then new call

Let's take the assumption that one Agent (*A1* with id *8000* and phone number *1000*) calls an internal User (*U1* with phone number *1001*). This call is transfered to Agent *A2* (with id *8002*) after talking with him. Finally *A1* calls internal another User (*U2* with phone number *1002*)



This call flow generates in *call_data* 3 lines (simplified table to keep only important fields) one for each call between *U1* and *U2* and one for *A2*:

id	uniqueId	dst_num	answer_time	end_time	src_agent	dst_agent	transfered	transfer_direction
1	12345678.9	1001	2018-12-01 15:02:00	2018-12-01 15:12:00	8000	8002	true	internal
2	12345679.0	1002	2018-12-01 15:13:00	2018-12-01 15:23:00	8000			

and get in *transfers* 1 line (where *callidto* id the *uniqueId* of the leg of Agent *A2* with user U1):

id	callidfrom	callidto
1	12345678.9	98765432.1

also get in *hold_period* 1 line:

id	linkedId	start	end
1	12345678.9	2018-12-01 15:07:00	2018-12-01 15:12:00

Statistics (done by pack-reporting for spago reports) generates then *stat_agent_specific*:

time	agent_num	nb_emitted_internal_calls	conversation_time_emitted_internal_calls	hold_time
2018-11-29 15:15:00	8000	1	1020	300
2018-11-29 15:30:00	8000	1	480	0

Important: Conversation time can be over 900s (15mn) as hold time is included in this specific call flow.

6.10.9 Sample SQL statistic queries

This section describes some SQL query achievements done based on *Database schema*.

List all received agent calls

This query get the phone set number on which the agent took the call. It lists all calls answered by agent with line number on which he was logged in. The query here is limiting to all calls answered the first day of August, but it can be easily customized to your needs.

```
SELECT cq.answer_time,
       cq.hangup_time,
       COALESCE(af.firstname, '') || ' ' || COALESCE(af.lastname, '') AS agent_
↪name,
```

(continues on next page)

(continued from previous page)

```
       cd.src_num AS caller,
       ap.line_number AS line_number
FROM call_on_queue cq
LEFT JOIN call_data cd ON cq.callid = cd.uniqueid
INNER JOIN agentfeatures af ON cq.agent_num = af.number
INNER JOIN agent_position ap ON cq.agent_num = ap.agent_num AND cq.answer_time␣
↪between ap.start_time and ap.end_time
AND to_char(cq.answer_time,'YYYY') = '2016'
AND to_char(cq.answer_time,'MM') = '08'
AND to_char(cq.answer_time,'DD') = '01'
AND cq.agent_num IS NOT NULL;
```

This query will result to something like:

Answer time	Hangup time	Agent name	Caller number	Line number used
2016-08-01 09:01:36.803	2016-08-01 09:02:38.916	Agent A	xxxxxxxxxx	101
2016-08-01 09:08:52.8	2016-08-01 09:09:31.97	Agent B	xxxxxxxxxx	102
2016-08-01 09:03:43.797	2016-08-01 09:07:18.452	Agent A	xxxxxxxxxx	101
2016-08-01 09:09:06.895	2016-08-01 09:09:56.549	Agent C	xxxxxxxxxx	103

Distribution of received call by month and DID

This query aggregates all received call by month and by direct inward dial number.

```
SELECT extract(year from cq.queue_time) as Year,
       to_char(cq.queue_time,'Mon') as Month,
       dst_num AS DID,
       COUNT(CASE WHEN cq.status IN ('answered', 'abandoned', 'leaveempty',
↪'timeout', 'exit_with_key') OR cq.status IS NULL THEN 1 END) AS Presented,
       COUNT(CASE WHEN cq.answer_time IS NOT NULL THEN 1 END) as Answered,
       to_char(AVG(CASE WHEN cq.answer_time IS NOT NULL THEN cq.hangup_time - cq.
↪answer_time END), 'HH24:MI:SS') as ACT,
       COUNT(CASE WHEN cq.status = 'timeout' THEN 1 END) as Dissuaded,
       COUNT(CASE WHEN cq.status = 'abandoned' THEN 1 END) as Hungup,
       COUNT(CASE WHEN cq.status = 'closed' THEN 1 END) as Refused,
       COUNT(CASE WHEN cq.status = 'abandoned' AND (cd.end_time - cq.queue_time) <
↪'15 seconds'::interval THEN 1 END) as Abandoned_T1,
       to_char(SUM(CASE WHEN cq.status = 'answered' THEN
            EXTRACT(epoch FROM (cq.answer_time - cq.queue_time)) ELSE 0 END) /
            NULLIF(COUNT(CASE WHEN cq.status IN ('answered', 'abandoned',
↪'leaveempty', 'timeout', 'exit_with_key')
            OR cq.status IS NULL THEN 1 END),0) * INTERVAL '1 second', 'HH24:MI:SS
↪') as AWT,
       SUM(CASE WHEN cd.transfered THEN 1 ELSE 0 END) AS Transfered,
       ROUND(COUNT(CASE WHEN cq.answer_time IS NOT NULL THEN 1 END)::numeric /
       NULLIF(COUNT(CASE WHEN cq.status IN ('answered', 'abandoned', 'leaveempty',
↪'timeout', 'exit_with_key')
       OR cq.status IS NULL THEN 1 END),0)::numeric * 100,2)  as Accepted_ratio
FROM call_on_queue cq
LEFT JOIN call_data cd ON cq.callid = cd.uniqueid
GROUP BY 1,2,3;
```

This query will result to something like:

Year	Month	DID	Presented	Answered	Average Call Time	Dissuaded	Hangup	Refused	Abandonned	Average Waiting Time	Transfered	Answered Rate
2016	Aug	1101	2	2	00:04:49	0	0	0	0	00:00:03	0	100
2016	Aug	1105	1	1	00:03:53	0	0	0	0	00:00:06	0	100
2016	Aug	1106	331	306	00:03:11	10	15	38	3	00:00:17	5	92.45
2016	Aug	1107	8	8	00:01:55	0	0	12	0	00:00:18	0	100
2016	Aug	1114	1	1	00:04:20	0	0	0	0	00:00:06	0	100
2016	Aug	1115	2	2	00:01:30	0	0	0	0	00:00:09	0	100
2016	Aug	1118	53	49	00:01:20	1	3	2	3	00:00:17	1	92.45
2016	Aug	1119	3	0		2	1	0	0	00:00:00	0	0
2016	Aug	1120	1	1	00:00:51	0	0	0	0	00:00:42	0	100

Tables join

Tables **call_data**, **call_on_queue** and **hold_periods** can be linked together by doing a join on a column holding the call reference. The columns are the following:

Table	Reference column
call_data	uniqueid
call_on_queue	callid
hold_periods	linkedid

On the other hand, tables **attached_data** and **call_element** contains foreign key referencing the **id** column of **call_data**.

—

Tables **call_on_queue**, **agentfeatures** and **agent_position** can be linked together by doing a join on a column holding the agent number reference. The columns are the following:

Table	Reference column
call_on_queue	agent_num
agentfeatures	number
agent_position	agent_num

6.11 Reporting

You may use your own reporting tools to be able to produce your own reports provided **you do not use the XiVO server original tables**, but copy the tables to your own data server. You may use the following procedure as a template :

- Allow remote database access on XiVO
- Create a postgresql account read only on asterisk database
- Create target tables in your database located on the data server
- Copy the statistic table content to your data server

6.11.1 General Architecture

1. The *queue_log* table of the *asterisk* database is filled by events from Asterisk and by custom dialplan events

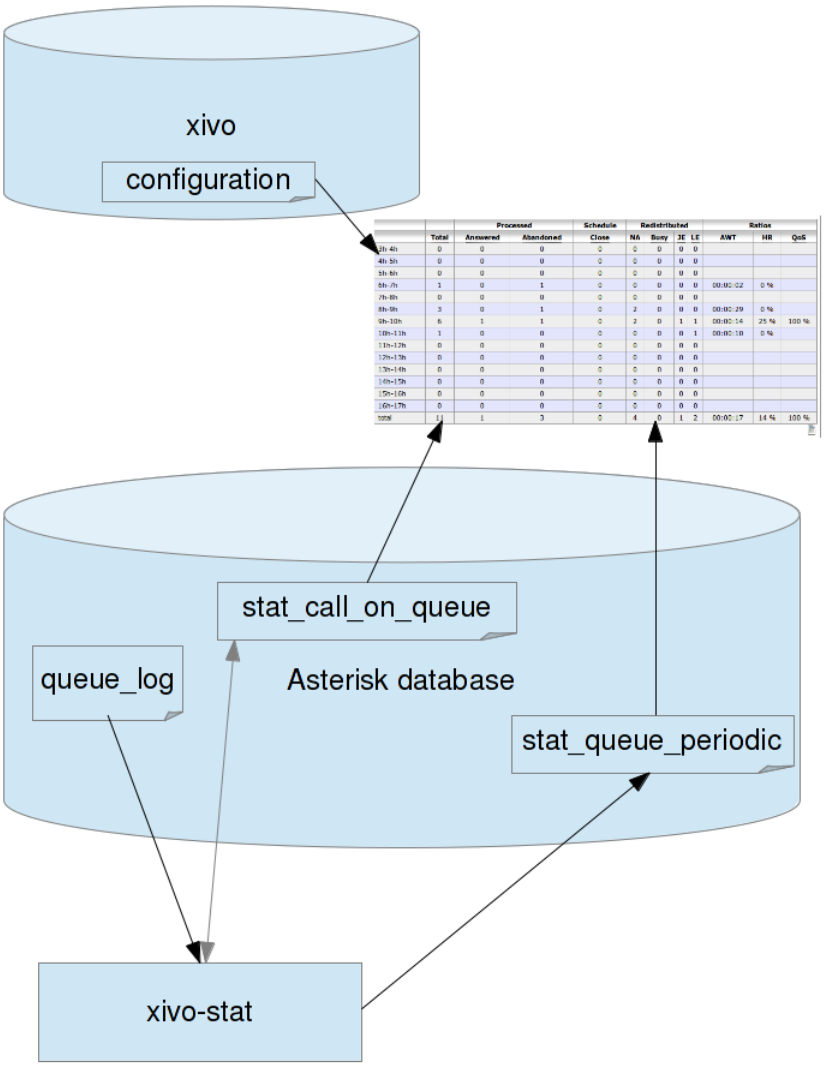


Fig. 8: Statistics Architecture

2. *xivo-stat fill_db* is then used to read data from the *queue_log* table and generate the tables *stat_call_on_queue* and *stat_queue_periodic*
3. The web interface generate tables and graphics from the *stat_call_on_queue* and *stat_queue_periodic* tables depending on the selected configuration

6.11.2 Statistic Data Table Content

stat_call_on_queue

This table is used to store each call individually. Each call received on a queue generates a single entry in this table containing time related fields and a foreign key to the agent who answered the call and another on the queue on which the call was received.

It also contains the status of the call ie. answered, abandoned, full, etc.

Field	Values	Description
id	generated	
callid	numeric value	This call id is also used in the CEL table and can be used to get call detail information
time	Call time	
ringtime		Ringing duration time in seconds
talktime		Talk time duration in seconds
waittime		Wait time duration in seconds
status		See status description below
queue_id		Id of the queue, the name of the queue can be found in table `stat_queue`, using this name queue details can be found in table `queuefeatures`
agent_id		Id of the agent, the agent name can be found in table `stat_agent`, using this name agent details can be found in table `agentfeatures` using the number in the second part of the name Exemple : Agent/1002 is agent with number 1002 in table `agentfeatures`

Queue Call Status

Status	Description
full	Call was not queued because queue was full, happens when the number of calls is greater than the maximum number of calls allowed to wait
closed	Closed due to the schedule applied to the queue
joinempty	No agents were available in the queue to take the call (follows the join empty parameter of the queue)
leaveempty	No agents available while the call was waiting in the qeuue
divert_ca_ratio	Call diverted because the ratio number of agent number of calls waiting configured was exceeded
divert_waittime	Call diverted because the maximum expected waiting time configured was exceeded
answered	Call was answered
abandoned	Call hangup by the caller
timeout	Call stayed longer than the maximum time allowed in queue parameter

stat_queue_periodic Table

This table is an aggregation of the queue_log table.

This table contains counters on each queue for each given period. The granularity at the time of this writing is an hour and is not configurable. This table is then used to compute statistics for a given range of hours, days, week, month or year.

Field	Description
id	Generated id
time	time period, all counters are aggregated for an hour
answered	Number of answered calls during the period
abandoned	Number of abandoned calls during the period
total	Total calls received during the period
full	Number of calls received when queue was full
closed	Number of calls received on close
joinempty	Number of calls received no agents available
leaveempty	Number of calls diverted agents not available during the wait
divert_ca_ratio	Number of calls diverted due to the number of agent number versus calls waiting configured was exceeded
divert_waittime	Number of calls diverted because the maximum expected waiting time configured was exceeded
timeout	Number of calls diverted because the maximum time allowed in queue parameter was exceeded
queue_id	

stat_agent

This table is used to match agents to an id that is different from the id in the agent configuration table. This is necessary to avoid loosing statistics on a deleted agent. This also means that if an agent changes number ie. Agent/1001 to Agent/1202, the supervisor will have to take this information into account when viewing the statistics. Affecting an old number to a another agent also means that the supervisor will have to ignore entries for this given agent for the period before the number assignment to the new agent.

stat_queue

This table is used to store queues in a table that is different from the queue configuration table. This is necessary to avoid losing statistics on a deleted queue. Renaming a queue is also not handled at this time.

CHAPTER 7

XiVO Centralized User Management

The XiVO Centralized User Management allows to manage several XiVO servers through a unique web interface. Thanks to this interface, it becomes possible to quickly add users that are automatically routed across servers. This documentation will describe the installation process of the interface, how to use the web interface and the REST API it exposes.

7.1 Intended usage and features

The XiVO Centralized User Management (XCU) is intended for multi-Xivo systems with centralized routing and user management.

Warning: Using XCU for other use-cases than the one described bellow is neither supported nor recommended. If XCU routing schema or centralized user management does not fit your use case, XCU is not good solution for your telephony system. Using XCU for user management only is not recommended. Non-standard installations may be broken by update to future version without warning.

7.1.1 Routing

Routing supports heterogeneous numbering plan administration:

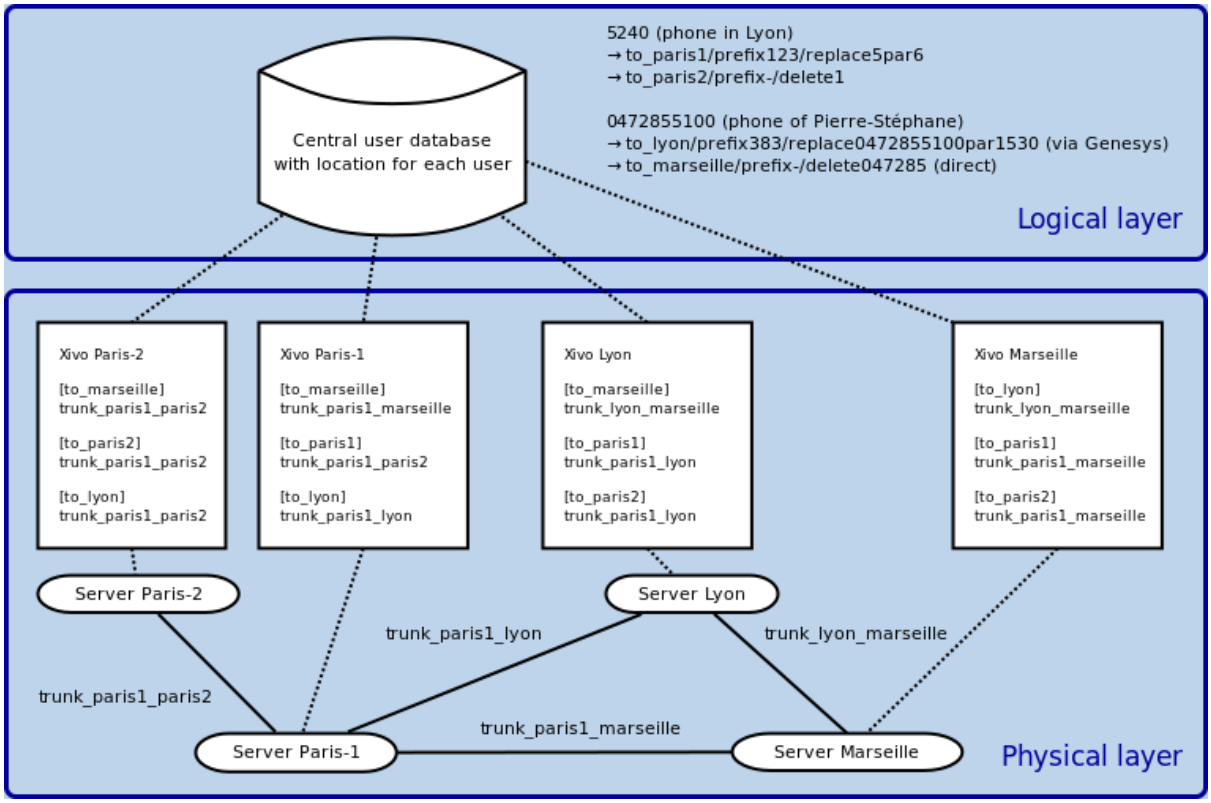
- Centralized dialplan management.
- Route incoming and outgoing calls, independently of the entry point to the target telephony subsystem hosting this number.
- Simple configuration of dialplan richness (prefix, short numbers, numbers of different length, emergency calls, live destination modification).
- Easily configurable protection against routing loops.

Routing is using concept of two layers:

1. “Logical” layer - based on contexts, users are routed using centralized database returning the context to be used to reach the user, with extensions correctly routed irrespective to point of entry of call.

2. "Physical" layer - arbitrary connection (direct or indirect) via trunks is supported. Trunks are associated to contexts to reflect the real network topology, there's no need of full-mesh topology, a call can pass by multiple Xivos before reaching the user.

Mapping between "Logical" and "Physical" layer is done by routing contexts. Every Xivo has routing context for every other Xivo (attached to trunk it will use). Routing context with the same name on different Xivo will be configured accordingly to the position of the Xivo in the network. Intervals can overlap between Xivos, therefore logical structure of dialplan can be independent on physical location of users.



See *Configuration of Xivos for Centralized Routing* for more configuration details.

Implementation of routing

When there is dial request on some Xivo and destination number is not found locally, AGI script in the dialplan requests a route from the routing database and gets:

- A context used to process the call.
- Rules to update the destination number.
- Requests can be chained.

Note: Routing systems provides also the conversion of the direct incoming number to the user's short number, this conversion is done when the call enters the system, between Xivos only the short number is used. This condition needs to be respected when creating a new system or integrating an existing one. When integrating an existing one, you may need to create some routes manually.

Fault tolerance:

- Local calls work, even when other network links and/or centralized user database are unreachable.
- Routing server can be setup in High Availability master-slave mode. When master routing server is down, routing requests are automatically processed by the slave.

7.1.2 Centralized user management

User management allows centralized administration without knowledge of low-level telephony details:

- Configuration is done from point of view of organization administrator, not telephony technician.
- User creation is simplified by line templates.
- Only relevant choices and options are presented.
- Extension number for new user is checked to be unique among all Xivos.
- Numbers proposed when creating a user are based on their availability, the number not used since the longest time will be reassigned first.
- XCU accounts can be restricted to manage only part of the system.
- All configuration changes are logged for auditability.

Compatibility with configuration via Xivo WebUI:

- Users added by Xivo WebUI before adding Xivo to XCU are imported, but they not reachable by centralized routing.
- Combining user management by XCU and by Xivo WebUI is bad idea, which usually leads to misconfiguration.
- Managing users by XCU and call-center configuration (Queues, Agents) via Xivo WebUI is possible, but queue numbers are not reachable by centralized routing automatically, you need to add manual routes if needed.
- Configuring conference rooms via Xivo WebUI is possible, but conference room numbers are not reachable by centralized routing automatically, you need to add manual routes if needed.

7.1.3 XiVO integration

Both freshly installed XiVO or an already configured XiVO can be added to the user management system. Steps to be followed are:

- When adding a freshly installed XiVO, you need to pass the XiVO Wizard and then follow steps described in *Create XiVO*.
- When adding an already configured XiVO (a XiVO used since a while with users etc.), you follow the same procedure as for a freshly installed XiVO, but you must pay attention to following restrictions:
  - Existing users will be imported in the centralized management, but currently the system doesn't create any route and these users are not reachable automatically, you need to add manually required routes.
  - Existing internal contexts are converted to Entities and created in the management system.
  - Ensure to have considered the interval overlapping option described in the *Configuration*.

7.2 Installation

7.2.1 Requirements & Limitations

The XiVO Centralized User Management requires :

- A server with:
  - Debian 8
  - PostgreSQL >= 9.5 (see Debian backports or Postgresql Wiki for installing instructions)
  - Docker > 1.12 and corresponding Docker-Compose. Since version 2018.04 XCU requires Docker-CE instead of Docker Engine.

  - git installed
  - sudo installed
- Some XiVOs to manage !
  - see the next section for limitations on managed XiVOs.

7.2.2 XiVO(s) Requirements & Limitations

Warning: Please double-check these requirements to prevent unexpected behavior.

For each Xivo which will be added to XCU ensure:

1. Create an Incoming calls interval in the *from-exten* context with a did length equal to the internal number length for each interval managed by XCU.
2. SCCP devices are not supported an may trigger error in the Centralized User Management. You must remove them on your XiVO before using this application.
3. On any context, Users interval *Number range start* and *Number range end* from must be 1-6 six digits (no other characters are allowed).
4. If you are making circular inclusions of asterisk context the XCU can potentially load users for a while, you should be **very** careful with such deployment.

Centralized routing will require further configuration - see *Configuration of Xivos for Centralized Routing*.

7.2.3 Installation by installer package

Install the *gcu-installer* package via *apt*:

1. Create the xivo sources list file `/etc/apt/sources.list.d/xivo-dist.list` and add the following line (replace **VERSION** with the current version, e.g. *2017.11*):

```
deb http://mirror.xivo.solutions/archive/ xivo-VERSION-latest main
```

2. Add GPG key of XiVO repository:

```
wget http://mirror.xivo.solutions/xivo_current.key -O - | apt-key add -
```

3. Update your source list and install the package:

```
apt-get update
apt-get install gcu-installer
```

The configuration files are located in `/etc/docker`.

7.2.4 Configuration

The XCU configuration files are installed by the installer package to the */etc/docker/* directory.

Authentication

Authentication is configured in `/etc/docker/interface-centralisee/application.conf`, section `authentication`:

- in `authentication.login` you can change initial user credentials (default admin / superpass)
- in `authentication.ldap` you can add configuration to use authentication via LDAP

Interval overlapping

A parameter called *allowIntervalOverlap* with default value *false* is available in `/etc/docker/interface-centralisee/application.conf`. When set to *false*, the XCU does not allow use overlapping intervals, when an interval is created or edited the XCU checks whether the interval overlaps with other intervals on all XiVOs and if it does the action is rejected. This default setting helps you to preserve a coherent numbering plan.

If for some reason you need to allow interval overlapping, you just need to change the value in the configuration file to *true* and restart the XCU. It can be useful when some existing XiVO servers with overlapping intervals were imported or when you want to be able to migrate some user to another XiVO without changing its number.

7.2.5 Run the application

Star XCU by following command:

```
docker-compose -p icdu -f /etc/docker/compose/icdu.yml up -d
```

Alternatively, you can set a bash alias for conveniently run XCU:

```
alias dcomp='docker-compose -p icdu -f /etc/docker/compose/icdu.yml'
```

In that case you can use simpler command :

```
dcomp up -d
```

XCU should now be accessible through http://my-server-ip or http://my-server-ip:9001

7.2.6 Application logs

1. General application log is in `/var/log/interface-centralisee/application.log` with daily rotation, historic logs retained for 5 days.
2. User actions are logged to `/var/log/interface-centralisee/user_actions.log` with daily rotation, historic logs retained for 366 days.

By default `user_actions.log` contains only brief information about which authorize XCU user did what action. To log with more detail (including data of create and update actions), change in `/etc/docker/interface-centralisee/logback.xml` line:

```
<logger name="UserActions"  level="INFO">
```

into:

```
<logger name="UserActions"  level="DEBUG">
```

7.3 Upgrade

The XiVO Centralized User Management (XCU) upgrade.

7.3.1 Prepare sources list

Before upgrading you have to create or change your sources list. It should be located in the file `/etc/apt/sources.list.d/xivo-dist.list`.

Set sources list for upgrade to latest version

To upgrade to the latest version the sources list must point towards *debian* URI and *xivo-polaris* suite:

```
deb http://mirror.xivo.solutions/debian/ xivo-polaris main
```

Set sources list for upgrade to specific version

To upgrade to a **specific** version the sources list must point towards *archive* URI and *xivo-VERSION-latest* suite.

For example if you want to upgrade to **2017.11** version you should have:

```
deb http://mirror.xivo.solutions/archive/ xivo-2017.11-latest main
```

Note the `/archive/` and `-2017.11-latest` above.

7.3.2 Upgrade steps

Preparing the upgrade

Read *Release Notes* starting from your version to the version you target.

Read *Version specific XCU upgrade procedures* starting from your version to the version you target.

Upgrade

When you have checked the sources.list you can upgrade:

1. Execute with the following commands:

   ```
   apt-get update
   apt-get install gcu-installer
   ```

2. If are using postgresql installed using sources from Postgresql Wiki, upgrading `gcu-installer` may trigger installing new version of PostgreSQL (in parallel to curent one). Suggested action is to disable this new version. If for example GCU use postgresql 9.5 and new version 9.6 was added:
   - stop postgresql 9.6 manually by: `service postgresql stop 9.6`
   - disable postgresql 9.6 autostart: in `/etc/postgresql/9.6/main/start.conf` replace `auto` with `disabled`
   - ensure postgresql 9.5 is running by: `service postgresql start 9.5`
3. Download the new images:

   ```
   docker-compose -p icdu -f /etc/docker/compose/icdu.yml pull
   ```

4. And run the new container (**All XCU services will be restarted**):

   ```
   docker-compose -p icdu -f /etc/docker/compose/icdu.yml up -d
   ```

Note: Please, ensure your server date is correct before starting. If system date differs too much from correct date, you may get an authentication error preventing download of the docker images.

7.3.3 Version specific XCU upgrade procedures

Upgrade from versions before 2018.04

Docker-CE must be installed instead of Docker Engine.

Upgrade from versions before 2017.06

If you installed GCU version older than 2017.06, you can upgrade it by following *Installation* instructions. Further instructions and notices:

1. It is strongly recommended you backup your PostgreSQL `icx` database and SSH keys before proceeding.
2. Installation will install new version of configuration files (like `/etc/docker/compose/icdu.yml`). If some file already exists with different content, you be prompted to choose correctly version or merge differences. Unless you are sure you need special version, use choice: *install the package maintainer's version*.
3. Installation will reuse your SSH certificate stored in `/etc/docker/interface-centralisee/ssh_key` if exists.
4. Installation will reuse PostgreSQL user `icx` if exists - make sure it has password `icx` and access to database `icx`.
5. Installation will reuse PostgreSQL database `icx` if exists.

Check carefully output of `apt-get install gcu-installer`, you will be informed about each component reused and about any manual checks or actions needed, if necessary.

7.4 Configuration of Xivos for Centralized Routing

When you add Xivo with configuration, basic Xivo configuration is done automatically. However there are still some steps which have to be done manually on each Xivo using Xivo WebUI.

7.4.1 Xivo configuration for centralized routing

Things to verify on Xivo before manual configuration

Verify that:

1. Xivo was added to XCU with configuration.
2. Xivo was restarted (automatically when added do XCU or manually later).
3. You have know routing context name for each xivo in form `to_xxx` - it is configured in XCU when adding Xivo.
4. Routing script `/usr/share/asterisk/agi-bin/xivo_routage_agi.py` is there, executable by asterisk user.
5. Dialplan routing configuration `/etc/asterisk/extensions_extra.d/routage.conf` is there, readable by asterisk user.

Manually update address of routing server

There must be a configuration file `/etc/xivo_routage.conf` with `hosts = IP:9000` where IP is the IP address of XCU server. If you have a backup routing server, there can be multiple IP:PORT couples separated by comma. Example:

```
[general]
hosts = 192.168.111.222:9000,192.168.111.333:9000
```

Connect Xivos by trunks

Ensure there is (direct or indirect) trunk connection between every two Xivos - see *Interconnect two XiVO directly*. This trunks must have Context set to `Incall`.

Add routing contexts

In each Xivo create routing contexts for every other Xivo. In Xivo WebUI go to: Services / IPBX / IPBX configuration / Contexts and click Plus icon. For example: if you have three Xivos A, B, C with routing contexts `to_xivo_a`, `to_xivo_b`, `to_xivo_c`, then in Xivo B you create routing contexts `to_xivo_a` and `to_xivo_c`. Type of routing context must be Outcall and it must not include any sub-contexts.

Add outgoing calls

For each routing context in every Xivo you need to add Outgoing call. In Xivo WebUI go to: Services / IPBX / Call management / Outgoing calls and click Plus icon. This outgoing call has:

- usually the same name as respective routing context
- Context set to respective routing context
- Trunk set to trunk connecting (directly or indirectly) to target Xivo
- in Exten tab single line with Exten = `X.` and Stripnum = `0`

7.4.2 Debug centralized routing

On each Xivo you can check log requests and result for routing requests by command:

```
tail -F /var/log/asterisk/xivo-routage-agi.log
```

On XCU you can check log requests and result for routing requests by command:

```
docker logs -f icdu_routing_server_1
```

To test manually query on routing server (replace IP by XCU's):

```
curl -v 'http://192.168.32.68:9000/route?digits=1234'
```

Response is either `200` with body containing route in JSON or `404` with body containing `{"Result":"No such element"}` if not found.

7.5 Manual installation

Warning: Manual installation steps are provided for debugging purposes and for special cases - for common cases please follow *Installation*.

The configuration files and the Docker-Compose files are available in a specific Git repository.

7.5.1 Database setup

XCU stores some data in a PostgreSQL database. By default, `application.conf` is configured to connect to a local database named `icx` with the username `icx` and password `icx`. You can change these parameters if you wish. We will use the default parameters in this documentation.

First, we need to install PostgresSQL extensions to use UUID functions :

```
sudo apt-get install postgresql-contrib
```

We can now create the user and the database associated :

```
sudo -u postgres psql -c "CREATE USER icx WITH PASSWORD 'icx'"
```

```
sudo -u postgres psql -c "CREATE DATABASE icx WITH OWNER icx"
```

We then have to enable UUID extension on the `icx` database. Connect as `root` on the `icx` database :

```
sudo -u postgres psql icx -c 'CREATE EXTENSION IF NOT EXISTS "uuid-ossp";'
```

I can't connect to PostgreSQL

It is possible that PostgreSQL complains when you're trying to connect. The solution is to modify the `pg_hba.conf` (in Debian, located in `/etc/postgresql/X.X/main`) and add the following line at the end :

```
local   all     all     trust
```

7.5.2 Generate SSH key

In order to let XCU communicate with the various XiVOs, an SSH key is used. Generate one using the following command :

```
ssh-keygen -t rsa -f /etc/docker/interface-centralisee/ssh_key
```

7.6 Web interface

The XiVO Centralized User Management (XCU) is managed through a web interface. In the following sections, we will highlight the main features of the system.

7.6.1 Definitions

XCU uses a few concepts that are important to understand in order to use the interface correctly.

XiVO The XiVOs servers that are managed by XCU. XCU will automatically retrieve the entities and the users from them and apply the configuration to them.

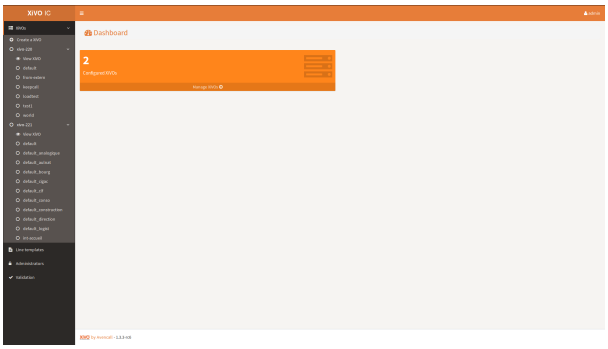
Entity Entities, also called Contexts, are the parts of the dialplan. Users are attached to them.

Line template Line templates are used to quickly create users : they define a few default options (ringing time, voice mail, etc.) that will be applied to the new user. **A line template is required to create a user**.

User Actual users that are associated with a phone number

Administrators Users that are able to connect to the XCU and manage the XiVOs.

7.6.2 Dashboard

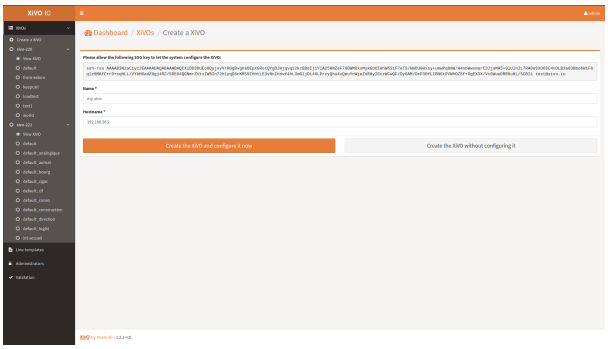


The dashboard provides you some insights about your XiVO systems.

The left sidebar, displayed in every page of the application, gives you access to the various actions you can perform. The list of the configured XiVOs and their entities is shown to give a quick access to the one you want to manage.

7.6.3 XiVO

Create XiVO



This page allows you to add a new XiVO that will be managed by XCU.

Warning: Before adding XiVO, please make sure it fulfills requirements - see *XiVO(s) Requirements & Limitations*.

The first step is to add the displayed SSH key to the authorized keys of your XiVO server. This will allow XCU to connect and configure the XiVO server. You could do this kind of command :

```
echo 'ssh-rsa TheVeryLongSSHKeyYouCopied toto@someserver' | ssh root@xivoIp 'cat >>
↪ .ssh/authorized_keys'
```

Then, you have to provide the following informations :

- **Name** : name of the XiVO server that will be displayed in XCU
- **Hostname** : hostname or IP address of the XiVO server

You then have two options :

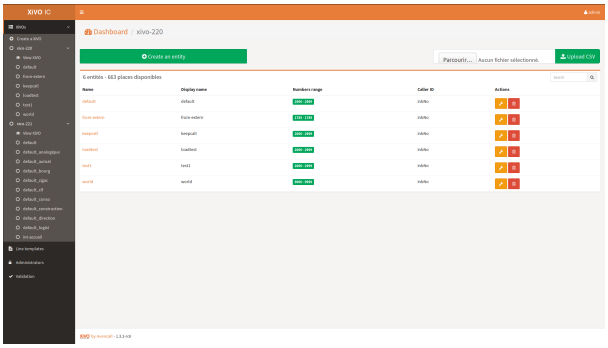
- **Create the XiVO and configure it now** : XCU will save the information, try to connect to the XiVO server and perform the configuration. XiVO services will be unavailable during the operation.

Warning: The configuration takes a while. Relax, go drink a coffee, XCU is doing the legwork for you :)

Note: If you want Xivo to be configured for centralized routing between multiple Xivos, please follow steps described in *Configuration of Xivos for Centralized Routing*.

- **Create the XiVO without configuring it** : XCU will only save the informations.

View XiVO



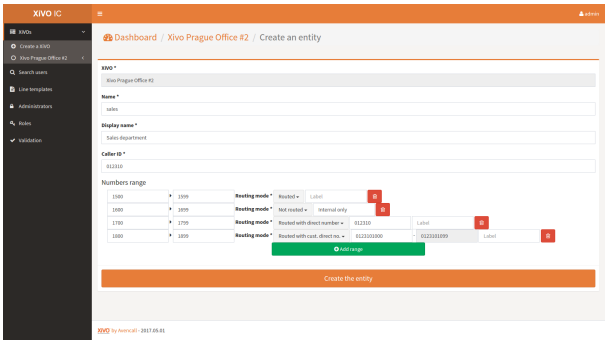
On the sidebar, each XiVO has its own **View XiVO** link. This page allows you to :

- **Add a new entity** to this XiVO by clicking on the green button
- **See the entities associated to this XiVO and perform some operations to them :**
  - **Edit one** by clicking on the yellow button with the wrench icon
  - **Delete one** by clicking on the red button with the trash icon

7.6.4 Entity

Note: Be sure to check the *Interval overlapping* configuration option before working with entities.

Create entity



This page allows you to add a new entity to a XiVO. You have to provide the following informations :

- **Name** : name that will be used by the XiVO server
- **Display name** : name that will be displayed on XCU
- **Caller ID** : phone number that will be displayed on outgoing call from this entity

- **Intervals** : ranges of phone numbers that will be available to this entity. For each one, provide :
  - **Start**
  - **End**
  - **Routing mode** (see bellow)
  - **Direct number prefix** (only for interval Routed with direct number)
  - **First direct number** (only for interval Routed with custom direct number)

The system will return an error if the intervals overlap with other entities

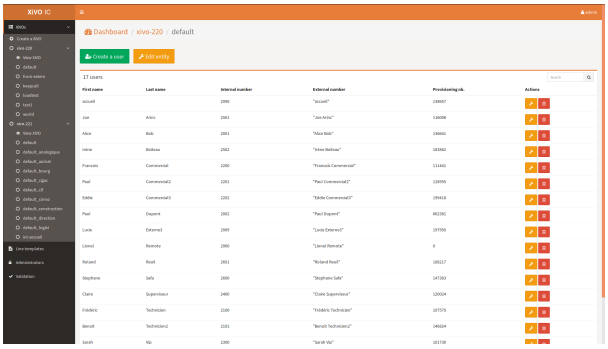
Routing mode affects how numbers from given interval are routed via centralized routing:

Routing mode	centralized routing
Routed	Internal number is used for centralized routing
Not routed	Numbers are not routed via centralized routing
Routed with direct number	*Direct number prefix* + internal number is used for centralized routing
Routed with custom direct number	Users has custom routed numbers in range from *First direct number* up to the width of the interval

If you have user in interval *Routed with (custom) direct number* on XiVO-A and call him from XiVO-B using his long (external) number:

1. target user's long (external) number is translated to short (internal) number by the routing mecanism on the first XiVO (XiVO B in this case)
2. on XiVO-A there is an incoming call with short (internal) number of target user

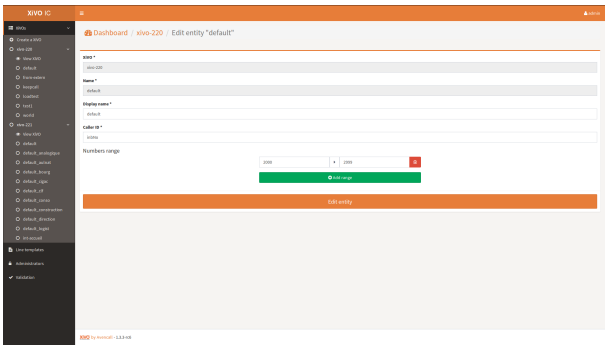
View entity



On the sidebar, each entity has its own link. This page allows you to :

- **Add a new user** to this entity by clicking on the green button
- **Edit the entity** by clicking on the yellow button with the wrench icon
- **See the users associated to this entity and perform some operations to them :**
  - **Edit one** by clicking on the yellow button with the wrench icon
  - **Delete one** by clicking on the red button with the trash icon. *At first click, the icon turns into a question mark. You have 5 seconds to click again to launch user deletion. This process prevents you from accidentally delete users.*

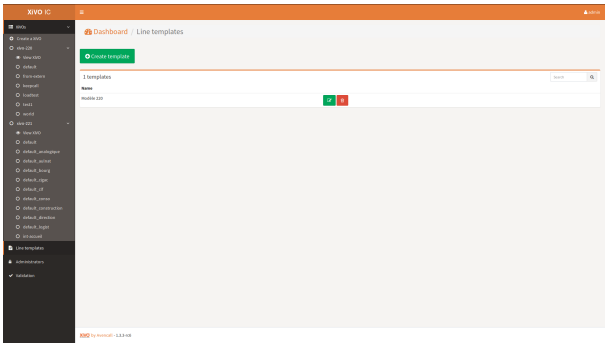
Edit entity



This page allow you to modify an entity. Please refer to the *Create entity* section for fields details.

7.6.5 Line templates

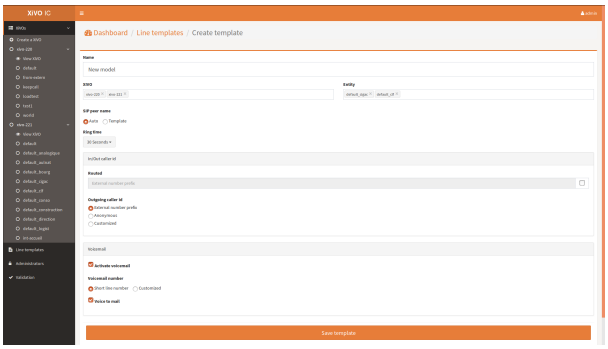
List templates



On the sidebar, **Line template** has its own link. This page allows you to :

- **Add a new line template** by clicking on the green button
- **See all the line templates and perform some operations to them :**
  - **Edit one** by clicking on the yellow button with the wrench icon
  - **Delete one** by clicking on the red button with the trash icon

Create template

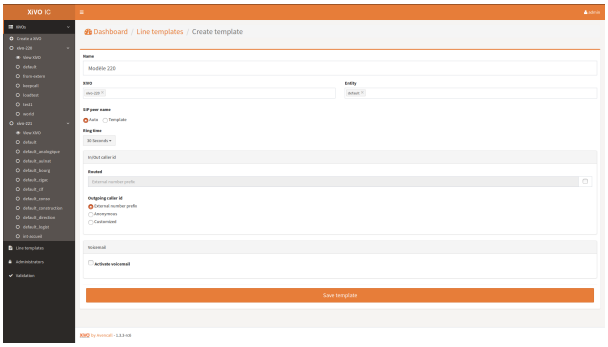


This page allows you to add a new line template. You have to provide the following informations :

- **Name** : name that will be be displayed on XCU
- **XiVO** : select the XiVOs for which this template will be available

- **Entity** : select the entities for which this template will be available. *Only entities of the selected XiVOs are displayed*
- **SIP peer name** : *Auto* or *Model*
- **Ringing time** : number of seconds before incoming call is rejected
- **Routed** :
  - The text field allows you to provide the SDA prefix to call the phone
  - Uncheck the checkbox if you don't want the phone to be called from the outside
- **Outgoing caller id** : specify what number is displayed on outgoing call. Possible values are :
  - External number prefix
  - Anonymous
  - Customized : a text field appears to provide the custom number
- **Voicemail** :
  - **Activate voicemail** : enable or not the voicemail
  - **Voicemail number** : specify what number is used to call the voice mail. Possible values are :
    - **Short line number** : use the default short number
    - **Customized** : a text field appear to provide the custom number
  - **Voice to mail** : whether or not to send an email when a new message is left, the email is the user's email for short line number box and the one configured in the customized voicemail for the customized box, see *Create user* for details.

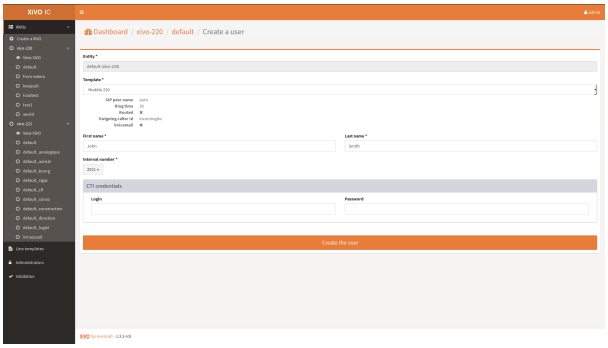
Edit template



This page allows you to modify a template. Please refer to the *Create template* section for fields details.

7.6.6 User

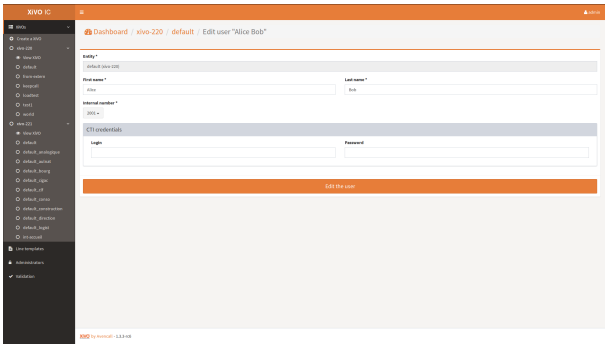
Create user



This page allows you to add a new user to an entity. You have to provide the following information :

- **Template** : line template to use as a template to create the user. *The main options of the template are displayed below*
- **First name**
- **Last name**
- **Internal number** : number that will be used to internally call the user. *Only the available numbers are displayed*
- **Email** : shown in the directory and used when voice to mail feature is activated
- **Voicemail** : Optional, activated only if present in the used template
  - When a private box is choosen (short line number), the box is created and the user's email is obligatory when Voice to mail feature is activated.
  - When a custom voicemail box is used, the interface will create it if the box doesn't exist on XiVO, it will be created with user's email. Otherwise the existing one is used and the email is not changed.
  - Currently there's a limitation due to a XiVO bug - when you update a user with associated custom voicemail, the voicemail name is replaced with the user's name.
- **CTI credentials** : provide a login and a password to allow the user to connect through CTI interfaces

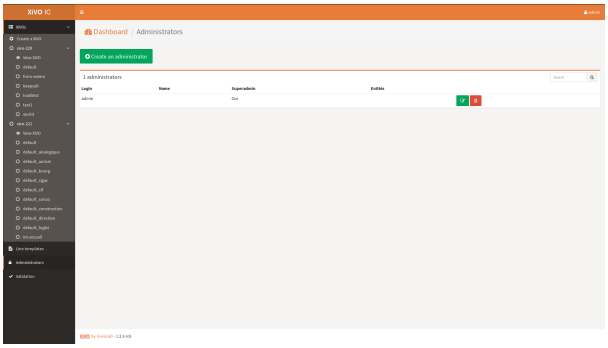
Edit user



This page allows you to modify a user. Please refer to the *Create user* section for fields details.

7.6.7 Administrators

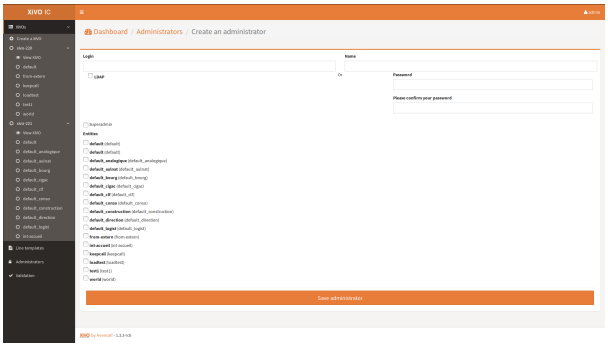
List administrators



On the sidebar, **Administrators** has its own link. This page allows you to :

- **Add a new administrator** by clicking on the green button
- **See all the administrators and perform some operations to them :**
  - **Edit one** by clicking on the yellow button with the wrench icon
  - **Delete one** by clicking on the red button with the trash icon

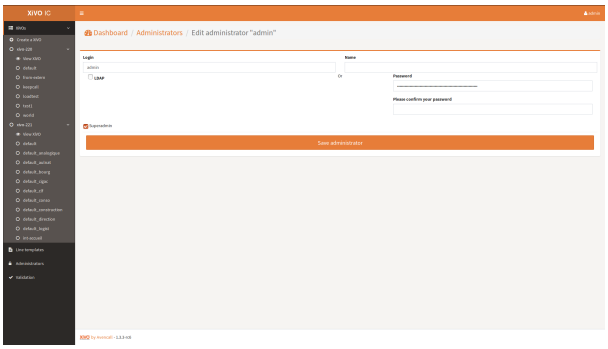
Create administrator



This page allows you to add a new administrator. You have to provide the following informations :

- **Login** : login used by the administrator to connect to XCU
- **Name** : name that will be displayed on XCU
- **LDAP** : if checked, the LDAP authentication configured in `application.conf` will be used
- **Password** : password used by the administrator to connect to XCU. *Shown only if LDAP disabled*
- **Superadmin** : whether or not this administrator is a super-administrator. Super-administrators can manage everything in XCU
- **Entities** : select the entities this administrator will be able to manage *Shown only if Superadmin disabled*

Edit administrator



This page allows you to modify an administrator. Please refer to the *Create administrator* section for fields details.

7.7 REST API

The XiVO Centralized User Management (XCU) exposes some REST API that you can use to integrate with your tools.

7.7.1 General form

> http://\protect\T1\textdollarmy-server-ip:\protect\T1\textdollarxcuport/api/1.0/\protect\T1\textdollarmethod
>
> withHeaders((“Content-Type”, “application/json”))

- $xcuport : XCU port number (default 9001)
- $method : See available methods below

7.7.2 Login

A login request is required before subsequent API calls in order to get a session cookie.

> POST /api/1.0/login

Payload parameters :

`login` (String) Login to connect with

`password` (String) Password corresponding to the login

The server will return a cookie and you will be able to do other API calls. Example with CURL :

```
curl 'http://localhost:9000/api/1.0/login' -H 'Content-Type: application/json' -c
↪'xcu-cookie' --data-binary '{"login":"admin","password":"superpass"}'
curl 'http://localhost:9000/api/1.0/xivo' -H 'Content-Type: application/json' -b
↪'xcu-cookie'
```

7.7.3 XiVO

The following methods allow you to operate on the XiVOs managed by XCU.

List

List all the XiVOs configured on XCU.

GET /api/1.0/xivo


```
{
  "items": [
    {
      "id": 1,
      "uuid": "8f159082-4b25-48b3-afec-1873491a60be",
      "name": "xivo-220",
      "host": "192.168.29.220",
      "remainingSlots": 664
    },
    {
      "id": 2,
      "uuid": "15585b75-1d75-45b1-8678-520d1210ec59",
      "name": "xivo-221",
      "host": "192.168.29.221",
      "remainingSlots": 280
    }
  ]
}
```


Get

Get a XiVO by its `id`.

GET /api/1.0/xivo/$id


```
{
  "id": 1,
  "uuid": "8f159082-4b25-48b3-afec-1873491a60be",
  "name": "xivo-220",
  "host": "192.168.29.220",
  "remainingSlots": 664
}
```


Create

Create a new XiVO.

POST /api/1.0/xivo

Payload parameters :

`name` (String) Display name of the XiVO

`host` (String) Hostname or IP address of the XiVO

`configure` (Boolean) If set to `true`, XCI will immediately make the necessary configurations on the XiVO. If set to `false`, it will only be added to XCI but not configured.

Synchronize configuration files

GET /api/1.0/xivo/synchronize_config_files

7.7.4 Entities

The following methods allow you to operate on the entities made available by the XiVOS.

List

List all the entities available.

GET /api/1.0/entities


```
{
  "items": [
    {
      "id": 17,
      "combinedId": "default@15585b75-1d75-45b1-8678-520d1210ec59",
      "name": "default",
      "displayName": "default",
      "xivo": {
        "id": 2,
        "uuid": "15585b75-1d75-45b1-8678-520d1210ec59",
        "name": "xivo-221",
        "host": "192.168.29.221",
        "remainingSlots": 280
      },
      "intervals": [
        {
          "start": "1700",
          "end": "1799"
        },
        {
          "start": "1961",
          "end": ""
        },
        {
          "start": "2600",
          "end": "2799"
        }
      ],
      "presentedNumber": "inbNo"
    },
    {
      "id": 22,
      "combinedId": "default_analogique@15585b75-1d75-45b1-8678-520d1210ec59",
      "name": "default_analogique",
      "displayName": "default_analogique",
      "xivo": {
        "id": 2,
        "uuid": "15585b75-1d75-45b1-8678-520d1210ec59",
        "name": "xivo-221",
        "host": "192.168.29.221",
        "remainingSlots": 280
      },
      "intervals": [
        {
          "start": "3990000",
          "end": "3999999"
        },
        {
          "start": "39990000",
          "end": "39999999"
        }
      ],
```


(continues on next page)

(continued from previous page)

```
      "presentedNumber": "inbNo"
    }
  ]
}
```

Get

Get an entity by its `combinedId`.

GET /api/1.0/entities/$combinedId

```
{
  "id": 22,
  "combinedId": "default_analogique@15585b75-1d75-45b1-8678-520d1210ec59",
  "name": "default_analogique",
  "displayName": "default_analogique",
  "xivo": {
    "id": 2,
    "uuid": "15585b75-1d75-45b1-8678-520d1210ec59",
    "name": "xivo-221",
    "host": "192.168.29.221",
    "remainingSlots": 280
  },
  "intervals": [
    {
      "start": "3990000",
      "end": "3999999"
    },
    {
      "start": "39990000",
      "end": "39999999"
    }
  ],
  "presentedNumber": "inbNo"
}
```

Create

Create a new entity.

POST /api/1.0/entities

Payload parameters :

`name` (String) Name of the entity

`displayName` (String) Displayed name of the entity

`xivoId` (Integer) Id of the XiVO the entity will be attached to

`intervals` (Array) Intervals of numbers this entity will support

`start` (String) Starting number of the interval

`end` (String) Ending number of the interval

`presentedNumber` (String) Number to show on outgoing calls

Delete

Delete an entity.

DELETE /api/1.0/entities/$combinedId

Edit

Edit an entity. See *Create entity* for fields details.

PUT /api/1.0/entities/$combinedId

List users

List users attached to an entity.

GET /api/1.0/entities/$combinedId/users


```
{
  "items": [
    {
      "id": 559,
      "entity": {
        "id": 22,
        "combinedId": "default_analogique@15585b75-1d75-45b1-8678-520d1210ec59",
        "name": "default_analogique",
        "displayName": "default_analogique",
        "xivo": {
          "id": 2,
          "uuid": "15585b75-1d75-45b1-8678-520d1210ec59",
          "name": "xivo-221",
          "host": "192.168.29.221",
          "remainingSlots": 280
        },
        "intervals": [
          {
            "start": "3990000",
            "end": "3999999"
          },
          {
            "start": "39990000",
            "end": "39999999"
          }
        ],
        "presentedNumber": "inbNo"
      },
      "firstName": "Sous sol Logistique",
      "lastName": "CLF 88:40 P3",
      "internalNumber": "6260",
      "externalNumber": "\"Sous sol Logistique CLF 88:40 P3\"",
      "mail": null,
      "ctiLogin": null,
      "ctiPassword": null,
      "provisioningNumber": "114133"
    }
  ]
}
```


List available numbers

List available numbers for an entity

GET /api/1.0/entities/$combinedId/available_numbers

```
{
  "items": [
    "3990000",
    "3990001",
    "3990002",
    "3990003",
    "3990004"
  ]
}
```


7.7.5 Users

The following methods allow you to operate on the users made available by the XiVOS.

Get

Get a user by its `id`.

GET /api/1.0/users/$id


```
{
  "id": 559,
  "entity": {
    "id": 22,
    "combinedId": "default_analogique@15585b75-1d75-45b1-8678-520d1210ec59",
    "name": "default_analogique",
    "displayName": "default_analogique",
    "xivo": {
      "id": 2,
      "uuid": "15585b75-1d75-45b1-8678-520d1210ec59",
      "name": "xivo-221",
      "host": "192.168.29.221",
      "remainingSlots": 280
    },
    "intervals": [
      {
        "start": "3990000",
        "end": "3999999"
      },
      {
        "start": "39990000",
        "end": "39999999"
      }
    ],
    "presentedNumber": "inbNo"
  },
  "firstName": "Sous sol Logistique",
  "lastName": "CLF 88:40 P3",
  "internalNumber": "6260",
  "externalNumber": null,
  "mail": null,
  "ctiLogin": null,
  "ctiPassword": null,
  "provisioningNumber": "114133"
}
```


Create

Create a new user.

POST /api/1.0/users

Payload parameters :

entityCId (String) Entity combinedId the user will be attached to

templateId (Integer) Line template to apply to the user

firstName (String) First name of the user

lastName (String) Last name of the user

internalNumber (String) Internal phone number of the user

ctiLogin (String) ***Optional*** CTI login of the user

ctiPassword (String) ***Optional*** CTI password of the user

Delete

Delete a user.

DELETE /api/1.0/users/$id

Edit

Edit a user. See *Create user* for fields details.

PUT /api/1.0/users/$id

7.7.6 Templates

The following methods allow you to operate on the line templates used to create users.

List

List all the templates available.

GET /api/1.0/templates


```
[
  {
    "id": 1,
    "name": "Modèle 220",
    "peerSipName": "auto",
    "routedInbound": false,
    "callerIdMode": "incomingNo",
    "ringingTime": 30,
    "voiceMailEnabled": false,
    "voiceMailNumberMode": "short_number",
    "xivos": [
      1
    ],
    "entities": [
      "default@8f159082-4b25-48b3-afec-1873491a60be"
    ]
  }
]
```

Get

Get a template by its `id`.

GET /api/1.0/templates/$id

```
{
  "id": 1,
  "name": "Modèle 220",
  "peerSipName": "auto",
  "routedInbound": false,
  "callerIdMode": "incomingNo",
  "ringingTime": 30,
  "voiceMailEnabled": false,
  "voiceMailNumberMode": "short_number",
  "xivos": [
    1
  ],
  "entities": [
    "default@8f159082-4b25-48b3-afec-1873491a60be"
  ]
}
```

Create

Create a new template.

POST /api/1.0/templates

Payload parameters :

`name` (String) Name of the template

`xivos` (Array of Integer) List of XiVOs ids the template will be available to

`entities` (Array of String) List of entities combinedIds the template will be available to

`peerSipName` (String) Possible values are `auto` or `model`

`ringingTime` (Integer) Number of seconds before incoming call is rejected

`routedInbound` (Boolean) Whether or not the phone can be called from the outside

`routedInboundPrefix` (String) ***Compulsory if*** **`routedInbound` *is* `true`** SDA prefix to call the phone

`callerIdMode` (String)

Option specifying what number is displayed on outgoing call. Possible values are :

- `incomingNo` : use the SDA prefix
- `anonymous` : masked call
- `custom` : a custom number

`customCallerId` (String) ***Compulsory if*** **`callerIdMode` *is* `custom`** Custom number to display on outgoing call

`voiceMailEnabled` (Boolean) Whether or not to enable the voice mail

`voiceMailNumberMode` (Boolean)

Option specifying what number is used to call the voice mail. Possible values are :

- `short_number` : use the default short number
- `custom` : a custom number

voiceMailCustomNumber (String) *Compulsory if* **voiceMailNumberMode** *is* **custom** Custom number to call the voice mail

voiceMailSendEmail (Boolean) Whether or not to send an email when a new message is left

Delete

Delete a template.

DELETE /api/1.0/templates/$id

Edit

Edit a template. See *Create template* for fields details.

PUT /api/1.0/templates/$id

7.7.7 Administrators

The following methods allow you to operate on the administrators of the XCI.

List

List all the administrators present.

GET /api/1.0/administrators


```
{
  "items": [
    {
      "id": 1,
      "login": "admin",
      "name": "",
      "password": "+\/\/rIncoyp\/Ai\/
↪8l3xSEeSY+P+x4uNle7cHkL6rpPS3ucgr2EAJIqnQbsIpSGwHj",
      "superAdmin": true,
      "ldap": false,
      "entities": [

      ]
    }
  ]
}
```


Get

Get an administrator by its `id`.

GET /api/1.0/administrators/$id


```
{
  "id": 1,
  "login": "admin",
  "name": "",
  "password": "+\/\/rIncoyp\/Ai\/8l3xSEeSY+P+x4uNle7cHkL6rpPS3ucgr2EAJIqnQbsIpSGwHj
↪",
  "superAdmin": true,
  "ldap": false,
```


(continues on next page)

(continued from previous page)

```
  "entities": [

  ]
}
```

Create

Create a new administrator.

POST /api/1.0/administrators

Payload parameters :

`login` (String) Login of the administrator

`name` (String) Displayed name of the administrator

`ldap` (Boolean) Whether or not to use the LDAP authentication configured in `application.conf`

`password` (String) ***Compulsory if*** **`ldap` *is* `false`** Password used by the administrator to login

`superAdmin` (Boolean) Whether or not this administrator is a super-administrator. Super-administrators can manage everything in XCI.

`entityIds` (Array of Integer) List of entities this administrator has the rights to manage

Delete

Delete an administrator.

DELETE /api/1.0/administrators/$id

Edit

Edit an administrator. See *Create administrator* for fields details.

PUT /api/1.0/administrators/$id

7.7.8 Example (Python 3)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from urllib.parse import urlencode
from urllib.request import Request, urlopen
import json, sys

class XCIApiExample:
        base_url = None
        cookie = None

        def __init__(self, base_url, login, password):
                self.base_url = base_url
                self.make_login(login, password)

        def make_login(self, login, password):
                data = {"login": login, "password": password}
                response = self.make_post_request("/login", data)
                self.cookie = response.info()["Set-Cookie"]
```

(continues on next page)

(continued from previous page)

```
        def get_entities(self):
                response = self.make_get_request("/entities")
                return self.handle_json_response(response)

        def get_available_numbers(self, entity):
                response = self.make_get_request("/entities/" + entity["combinedId
↪"] + "/available_numbers")
                return self.handle_json_response(response)

        def create_line_template(self, data):
                self.make_post_request("/templates", data)

        def get_line_templates(self):
                response = self.make_get_request("/templates")
                return self.handle_json_response(response)

        def create_user(self, data):
                self.make_post_request("/users", data)

        def make_get_request(self, method):
                request = Request(self.base_url + method, headers = {"Cookie":␣
↪self.cookie})
                response = urlopen(request)
                return response

        def make_post_request(self, method, data):
                header = {"Content-Type": "application/json", "Cookie": self.
↪cookie if self.cookie else ""}
                request = Request(self.base_url + method, json.dumps(data).
↪encode(), header)
                response = urlopen(request)
                return response

        def handle_json_response(self, response):
                return json.loads(response.read().decode())


# Initialize API
api_example = XCIApiExample("http://192.168.29.103:9001/api/1.0", "admin",
↪"superpass")

# Get an entity and its XiVO
entities = api_example.get_entities()["items"]
if (len(entities) == 0):
        sys.exit("There isn't any XiVO configured yet or they don't have any␣
↪entity !")
else:
        entity = entities[1]
        xivo = entity["xivo"]
        print("Selected entity \"%s\" in XiVO \"%s\""%(entity["name"], xivo["name
↪"]))

# Create a line template
template_data = {
        "name": "My line template",
        "xivos": [xivo["id"]],
        "entities": [entity["combinedId"]],
        "peerSipName": "auto",
        "ringingTime": 30,
        "routedInbound": False,
```

(continues on next page)

(continued from previous page)

```
        "callerIdMode": "anonymous",
        "voiceMailEnabled": False
}
api_example.create_line_template(template_data)
line_template = api_example.get_line_templates()[0]
print("New line template created")

# Create a user
user_data = {
        "entityCId": entity["combinedId"],
        "templateId": line_template["id"],
        "firstName": "Alice",
        "lastName": "In Wonderland",
        "internalNumber": api_example.get_available_numbers(entity)["items"][0]
}
api_example.create_user(user_data)
print("New user created")
```

CHAPTER 8

User's Guide

End user help and documentation.

8.1 UC Assistant

Note: This section describes the feature of the UC Assistant application. It is available as a web application from your Web Browser. It is also available as a *desktop application* with these additionnal features:

- show integrated popup when receiving call
- get keyboard shortcut to answer/hangup and make call using *Select2Call feature*
- *handle callto: and tel: links*

To install the *desktop application*, see *the desktop application installation* page.

What is the XiVO UC Assistant ?

The *XiVO UC Assistant* is a Web application that enables a user to:

- search contacts and show their presence, phone status
- make calls through physical phone or using WebRTC
- transfer incoming or outgoing calls
- access voicemail
- enable call forwarding and *Do Not Disturb* (aka DND)
- show history of calls

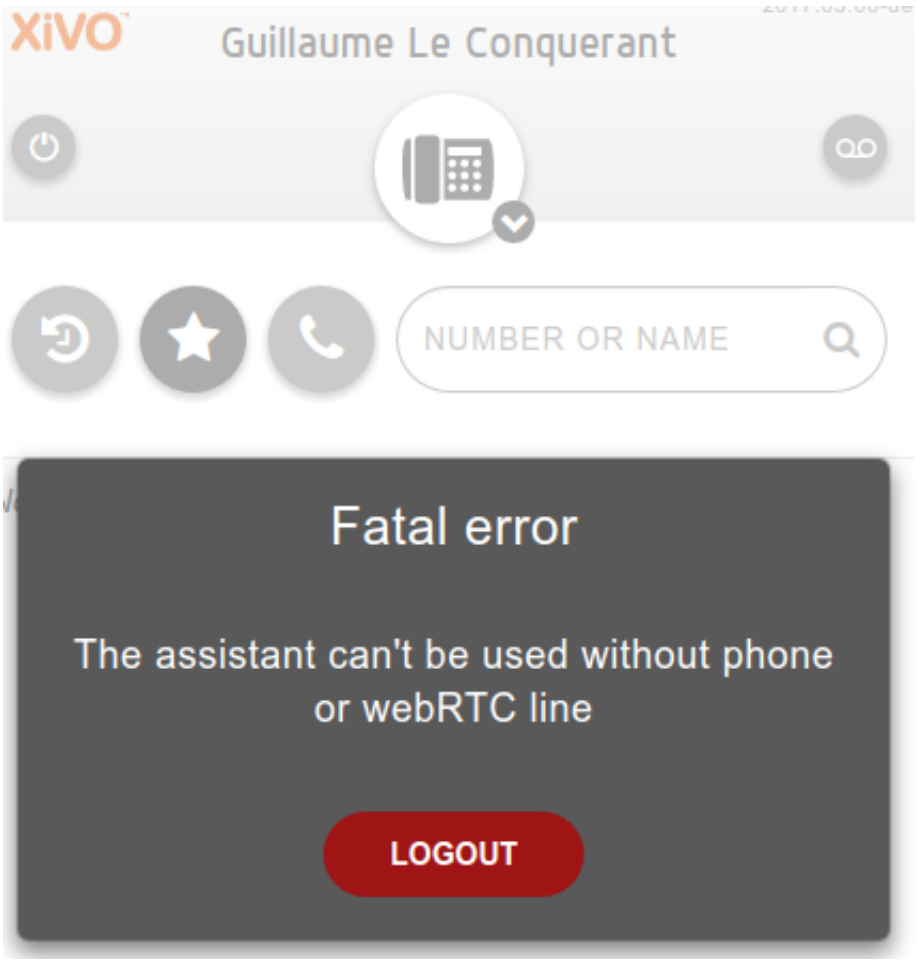
8.1.1 Login

To login, you must have a user configured on the *XiVO PBX* with:

- XiVO Client enabled,
- Login, password and profile configured

- A configured line with a number

Warning: If a user tries to login without a line, an error message is displayed and user is redirected to the login page (this applies also to *Desktop Applications*)



Note: A **Remember me** option is available at prompt page to keep you signed in, when you want automatic login.

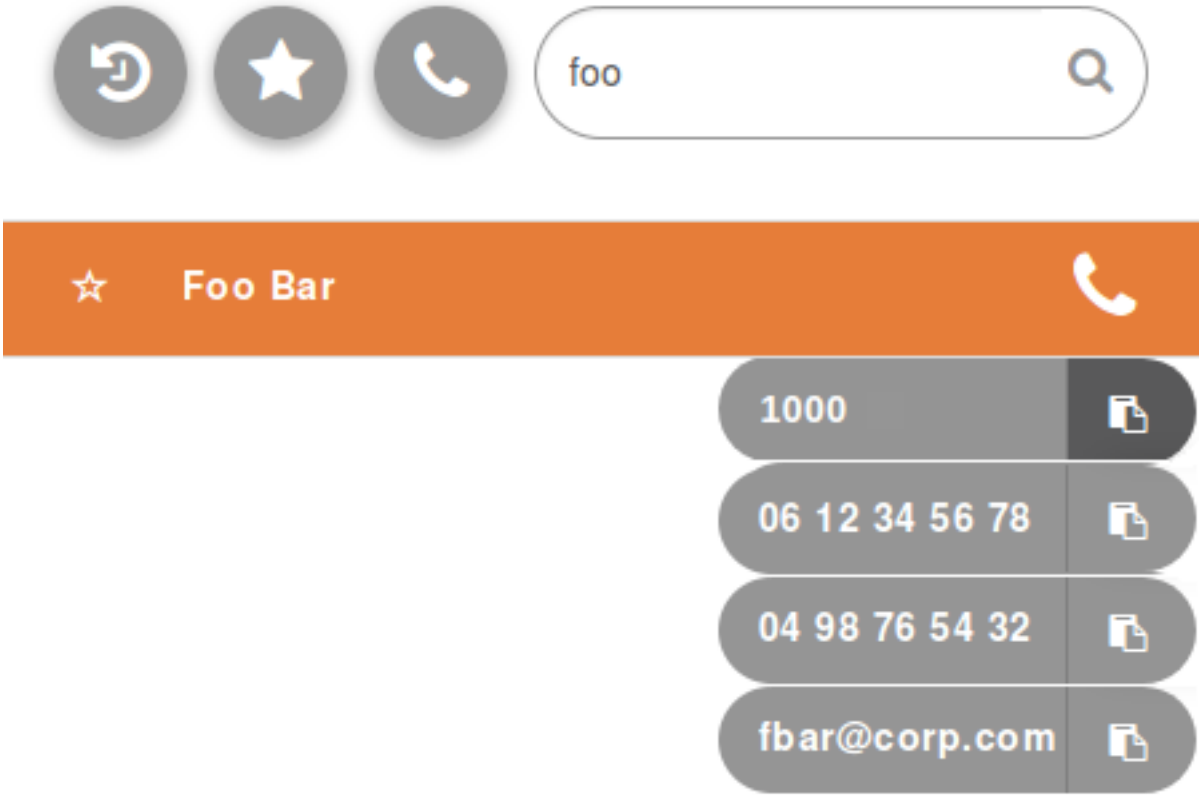
8.1.2 Search

You can use the search section to lookup for people in the company, results will display all information known for the user (phone numbers and email). You can either click on number to call, or click on copy button to put the number in your clipboard to paste it elsewhere.

To enable this feature, you must configure the directories in the *XiVO PBX* as described in *Directories* and *Views*.

Note: Integration note: the *UC Assistant* support only the display of

- 1 field for name (the one of type *name* in the directory display)
- 3 numbers (the one of type *number* and the first two of type *callable*)
- and 1 email



8.1.3 Forwarding Calls and DND

From UC Assistant you can activate *Do Not Disturb* to block all incoming calls or forward call to any another number just by clicking on action button as seen on following screenshot:

Once you choose an action, you just need to **apply** on clicking on associated button.

Action possibles are :

- Enable DND
- Disable DND
- Edit call forwarding (for both unconditional or on missed call only)

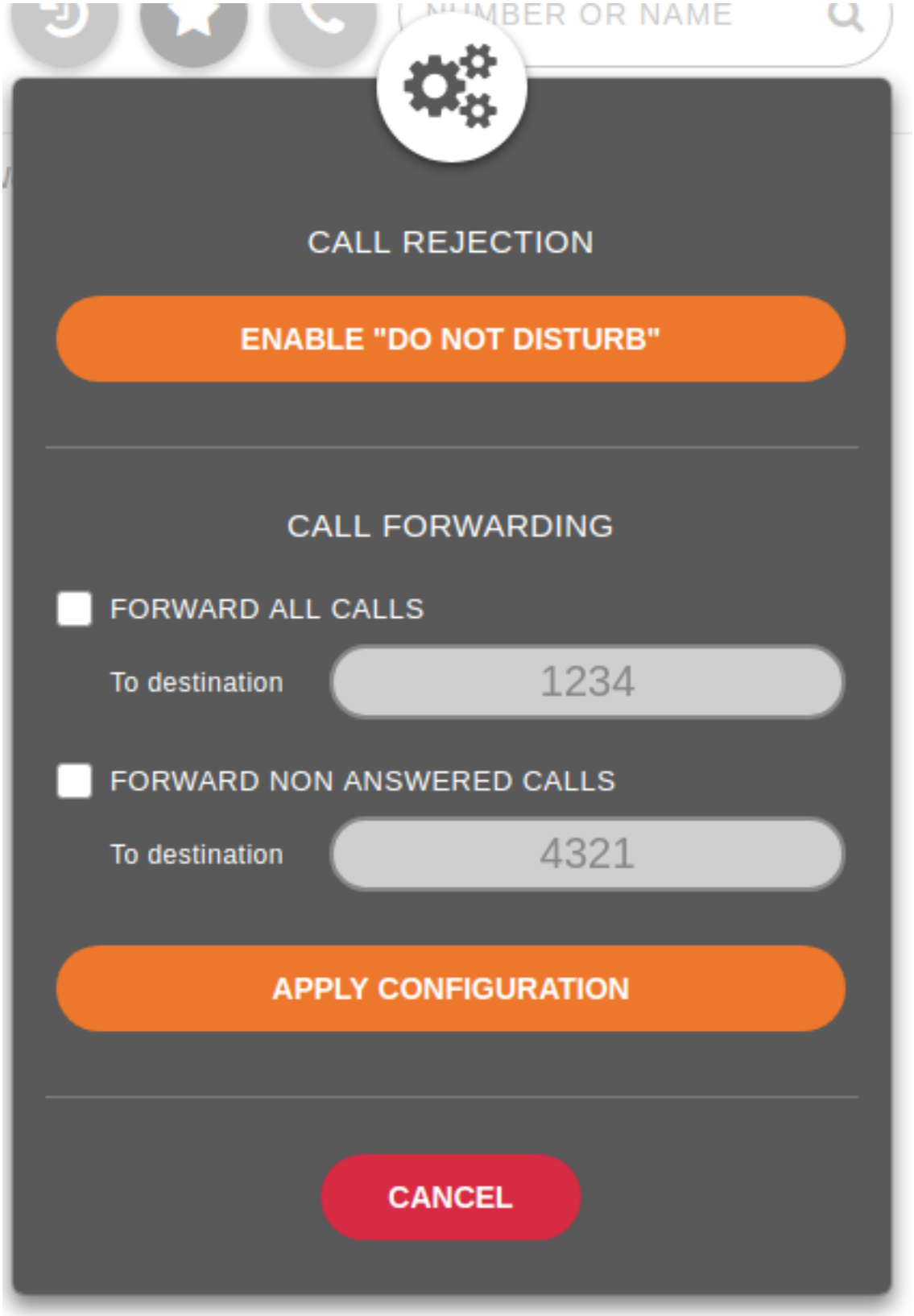
You know that all incoming calls will be rejected once you see the following logo in the header bar :

All calls are forwarded once you see this following one :

Finally, calls are forwarded only if you missed it when you see this one :

Note: If calls are redirected, the forward number will be shown under your name.

Nevertheless, there is a precedence, if DND mode is enabled and also call forwarding, calls will be rejected. If forward on miss call and all call forward are enabled, all calls will be forwarded to number configured for all call


NUMBER OR NAME
CALL REJECTION
ENABLE "DO NOT DISTURB"
CALL FORWARDING
FORWARD ALL CALLS
To destination
1234
FORWARD NON ANSWERED CALLS
To destination
4321
APPLY CONFIGURATION
CANCEL

forwarding.

8.1.4 Favorites

Click on the star to put a contact in its list of favorites. Favorites must be configured in the *XiVO PBX* as described in *Favorites*.

8.1.5 Personal contacts

From top-right hamburger menu, it is possible to display additional actions to handle you personal contacts. You will be able to **create**, **delete all**, **import** and **export** personal contact that you will be either able to search from toolbar or find them in *favorites* panel if starred.

Create a personal contact

Just fill wanted fields (such as name and number), click on star if you want this contact to be displayed in *favorites* panel.

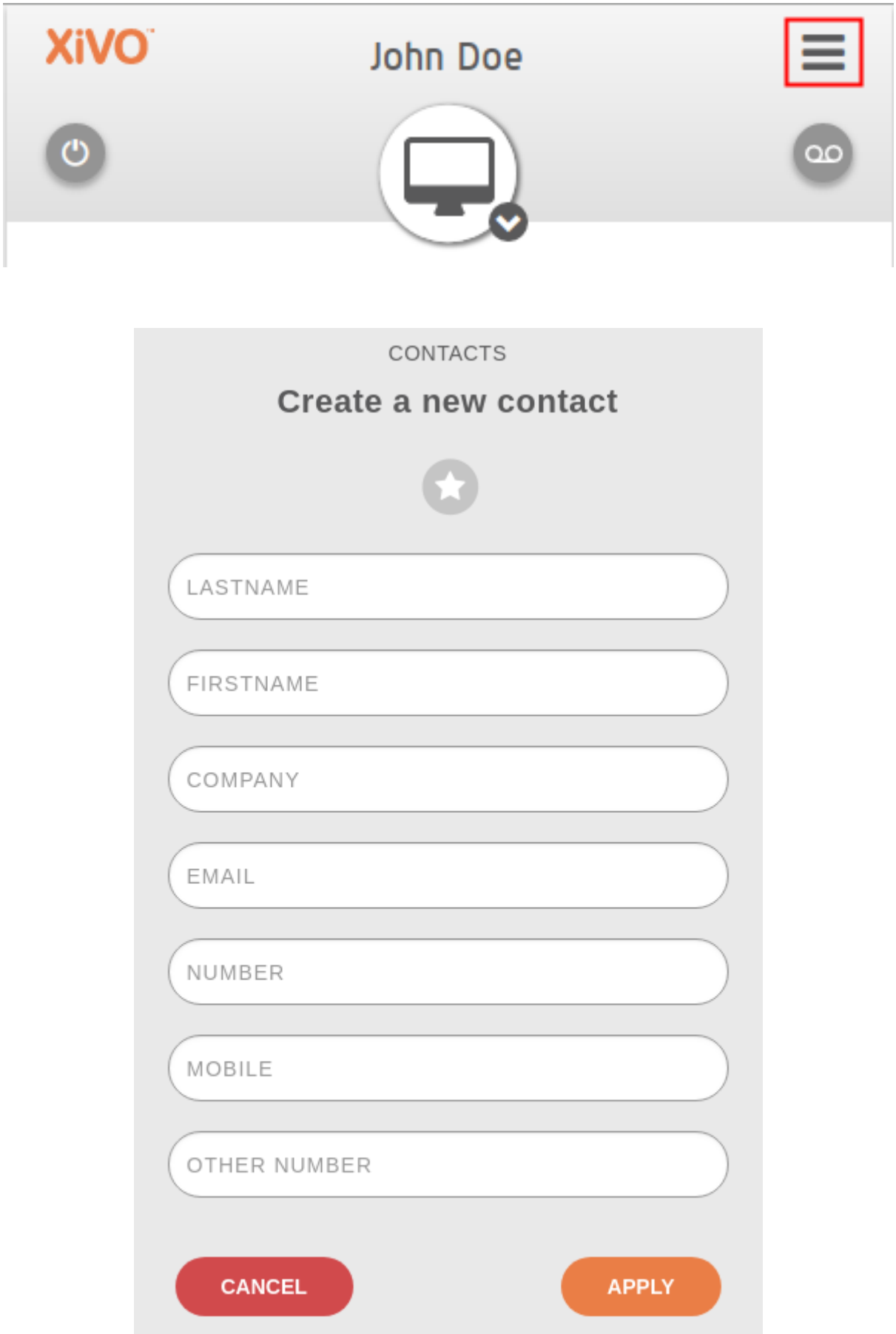
> **Warning:** It is not possible to have twice the same personal contact, at least one field must differ.

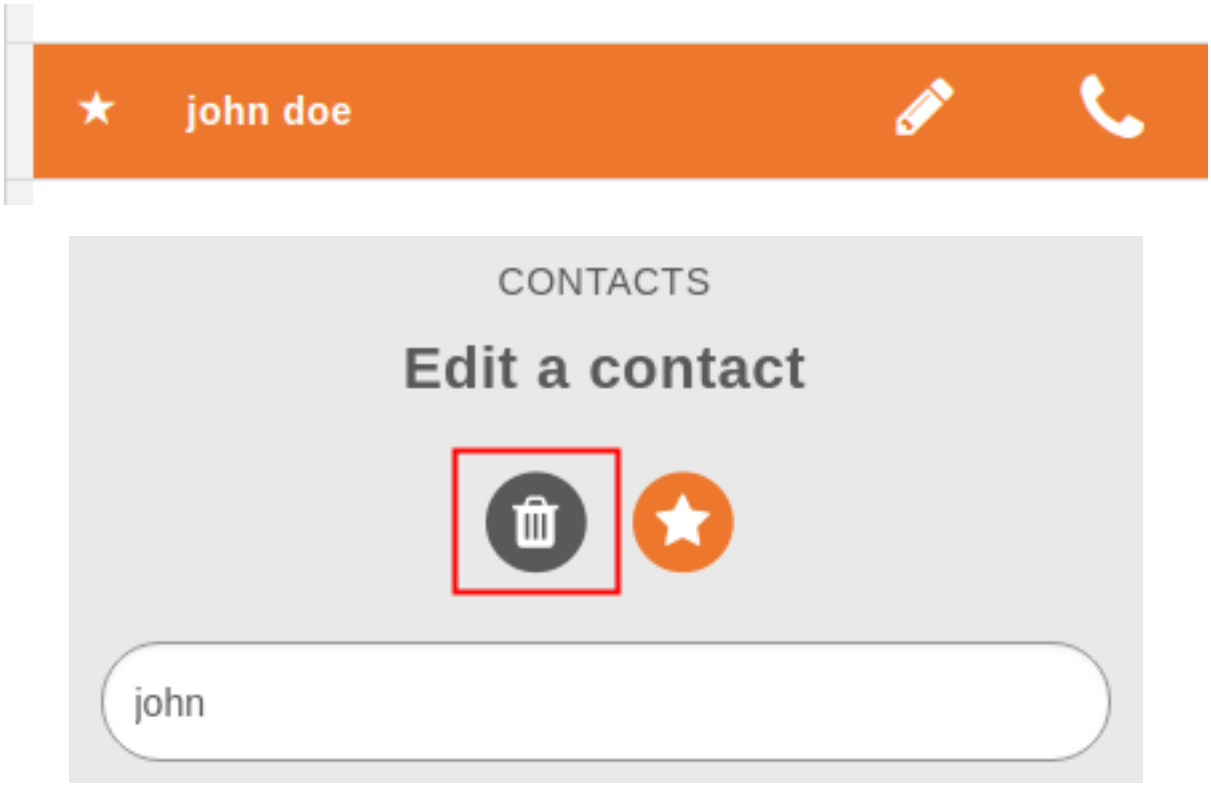
It's also possible to create a personal contact from call history by hovering a call item and so have pre-filled fields.

Edit a personal contact

To edit a personal contact, you should search it first, then you just hover it, and a pencil icon should appear as in following screen:

Once clicked, you are redirected to edition pane where you just fill wanted fields.


XiVO
John Doe
CONTACTS
Create a new contact
LASTNAME
FIRSTNAME
COMPANY
EMAIL
NUMBER
MOBILE
OTHER NUMBER
CANCEL
APPLY



Delete a personal contact

To delete a personal contact, you should edit it first, then you just need to click on trashcan icon :

Once clicked, you are invited to confirm or not the deletion of this contact.

Import personal contacts

From menu, you can upload a **.csv** file that contains all the data of your personal contacts. You can either use a file exported from this same interface or create yours.

Here are the list of available attributes of a personal contact:

- `company`
- `email`
- `fax`
- `firstname`
- `lastname`
- `mobile`
- `number`

As an example here a csv file that can be imported

```
company,email,fax,firstname,lastname,mobile,number
corp,jdoe@company.corp,3333,John,Doe,2222,1111
```

Note: File exported from previous *xivo-client* is also compatible with *UC assistant*.

Reverse lookup

By default, *reverse lookup* is enabled for personal contact display on incoming calls. Configuration is set to display `firstname` and `lastname` if `number` or `mobile` matches an existing personal contact.

8.1.6 Phone integration

The *UC Assistant* can integrate with the phone to :

- Call / Hangup
- Put on hold
- Do direct or indirect transfers *
- Make a conference *

*) See *Known limitations*

As these features are closely linked to the phone to work, you must check supported phones for UC Assistant and follow the *Required configuration* page.

Once, you're phone is properly configured and you are connected as a user, you know that your using SIP phone once you see the following logo in the header bar :

8.1.7 On hold notifications

You can be notified if you forget a call in hold for a long time, see *configuration section*.

8.1.8 Conferences

When joining a conference, either as an attendee or an organizer, the *UC Assistant* will display specific informations about the conference you are joining.

Conference information:

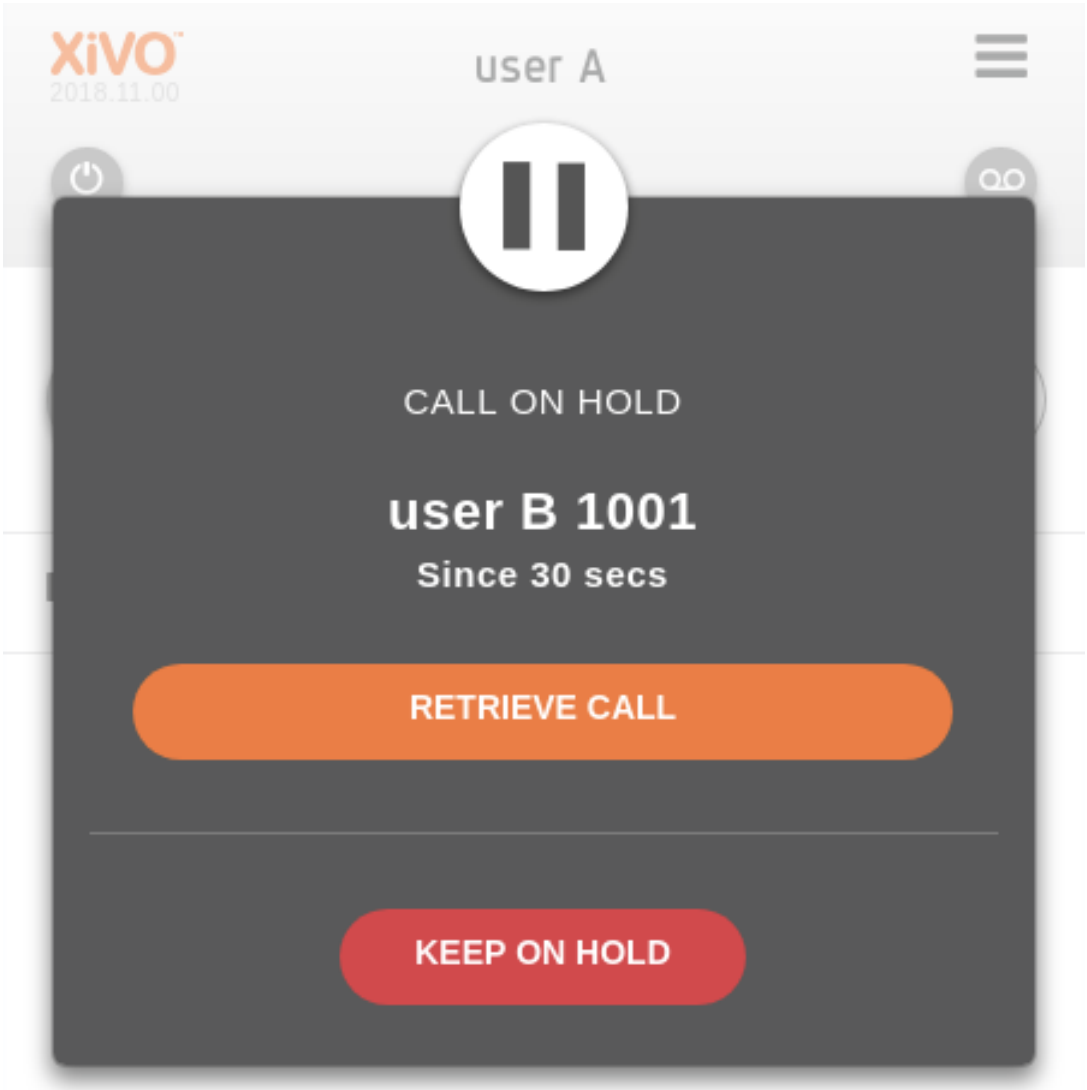
- The timer next to the conference name displays how long the conference has been running
- When hovering, the number of attendee will be displayed

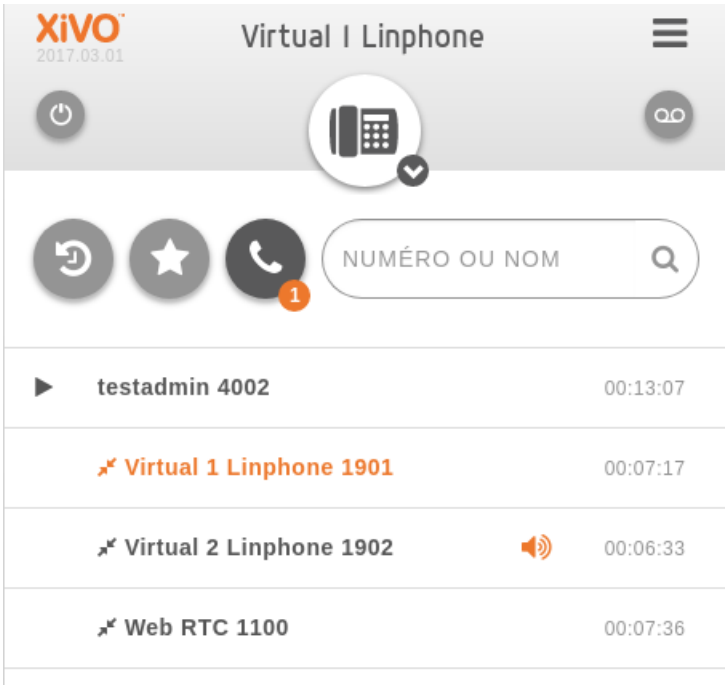
Conference actions:

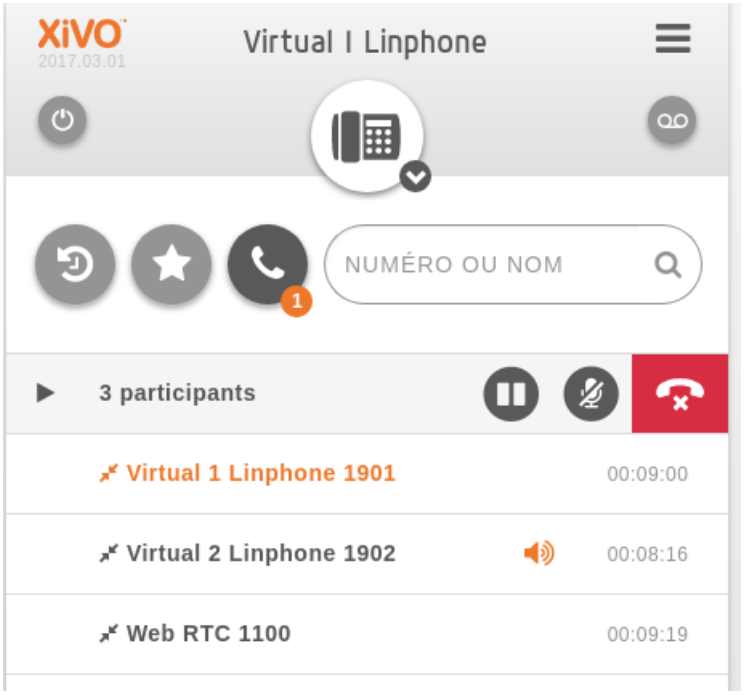
As an attendee, you can only:

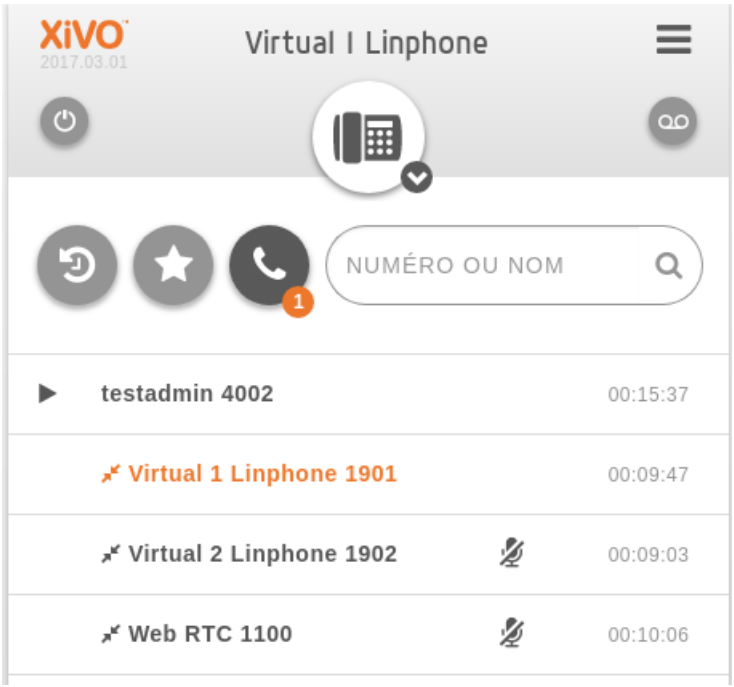
- Exit a conference room by clicking the hangup button
- Put the conference on hold. Other attendees will not hear any hold music but will not be able to hear you neither you will be able to hear the conference room.

As an organizer, you will be able to mute and unmute all other attendee in the conference room.


XiVO
2018.11.00
user A
CALL ON HOLD
user B 1001
Since 30 secs
RETRIEVE CALL
KEEP ON HOLD


XiVO
2017.03.01
Virtual I Linphone
NUMÉRO OU NOM
1
testadmin 4002
00:13:07
Virtual 1 Linphone 1901
00:07:17
Virtual 2 Linphone 1902
00:06:33
Web RTC 1100
00:07:36

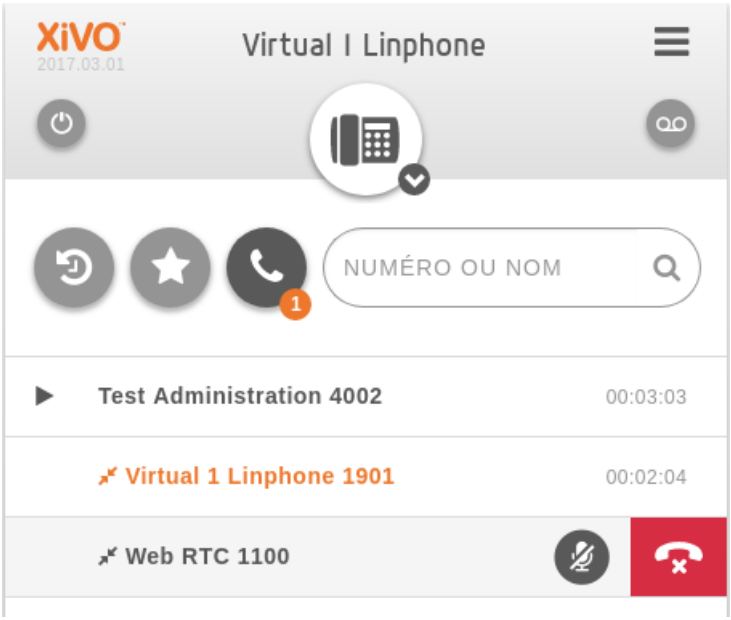

XiVO
2017.03.01
Virtual I Linphone
NUMÉRO OU NOM
1
3 participants
Virtual 1 Linphone 1901
00:09:00
Virtual 2 Linphone 1902
00:08:16
Web RTC 1100
00:09:19


XiVO
2017.03.01
Virtual I Linphone
NUMÉRO OU NOM
1
testadmin 4002
00:15:37
Virtual 1 Linphone 1901
00:09:47
Virtual 2 Linphone 1902
00:09:03
Web RTC 1100
00:10:06

Attendees information:

- Attendees name, number and timer are displayed below the conference name.
- Attendees are ordered by name with the exception of the first one which always reflect the current user
- Conference organizer are displayed in orange
- When an attendee is muted, an slashed microphone icon will be displayed next to its name
- When an attendee is talking, an orange speaker will be displayed next to its name

Attendee action:



When hovering your own user, you can mute and unmute yourself. Organizer can also:

- Mute and unmute any attendee
- Kick out an attendee. A message will be played to the kicked out attendee before leaving the conference.

8.1.9 WebRTC integration

The *UC Assistant* can be used by users with WebRTC configuration, without physical phone.

For configuration and requirements, see *WebRTC Environment*.

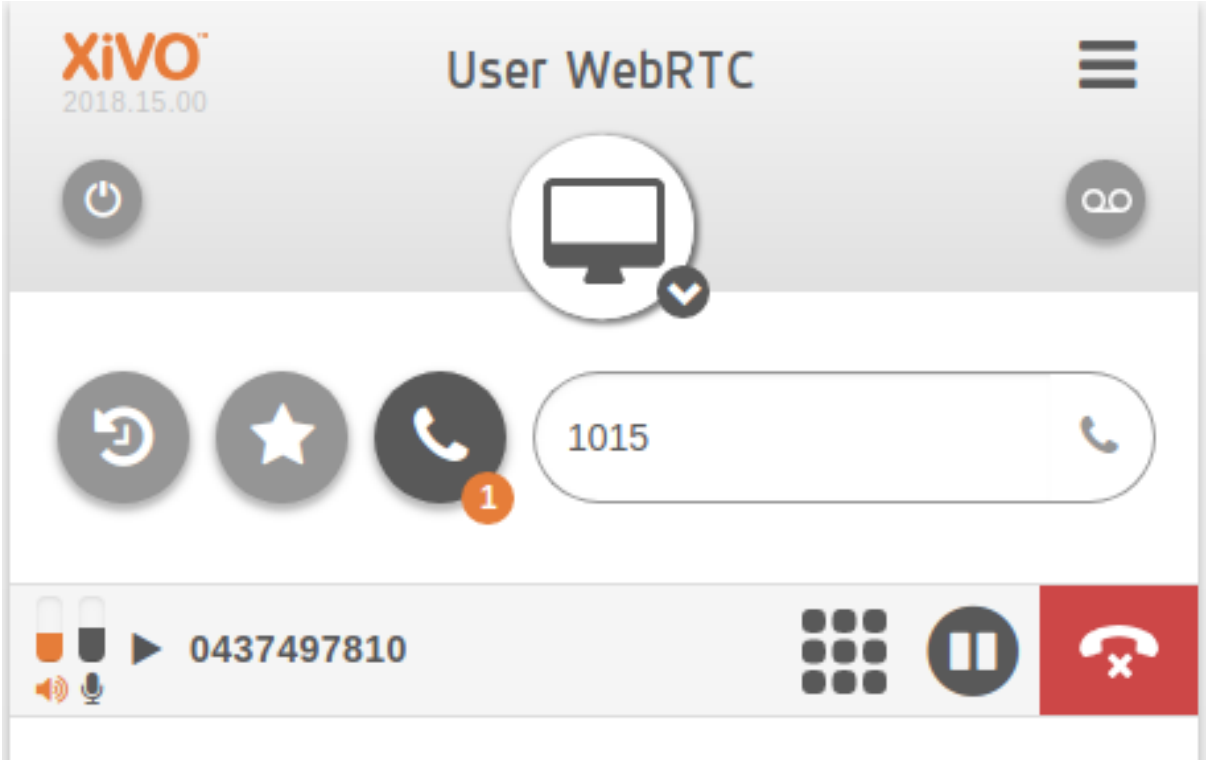
You know that your using WebRTC once you see the following logo in the header bar :

**55* (echo test)

To test your microphone and speaker, you may call the echo test application. After a welcome message, the application will echo everything what you speak.

1. Dial the **55* number from your *Desktop Assistant*.
2. You should hear the "Echo" announcement.
3. **After the announcement, you will hear back everything you say.** If you hear what you are saying it means that your microphone and speakers are working.
4. Press # or hangup to exit the echo test application.

Volume indication



Two progress bars show the volume level of the speaker and the microphone. It certifies that the audio flow has been sent.

8.1.10 Experimental video call feature

Note: This experimental feature needs to be enabled in the configuration (see *Experimental video call feature*).

When logged in as a user with a WebRTC line you can initiate and receive an audio/video call. The video call is possible only from your favorite contacts or the directory search result through the video camera icon. When the called user doesn't support video calls, call is established as audio only.

In the current implementation you can put the call on hold, go fullscreen, but for the moment you can have only one video call at a time and you can't transfer the video call. Conference is not supported yet.

8.2 Desktop Applications

The *XiVO Desktop Application* is a standalone executable for either *UC Assistant* or *CC Agent Environment*. It is available for Windows (64bits) and Linux Debian based distributions (64bits). It offers some additional features compares to the existing web version that can be run in a browser.

8.2.1 Installation

The **UC Assistant** and **CC Agent** are available as desktop application through Electron packaging, to be able to use these applications in a standalone executable you need to deploy this container first on client machine.

Windows (64bits)

To download the latest version available on your environment, just open the following url from your computer:

```
http://<xucmgt_host>:<xucmgt_port>/install/win64
```

and then start the downloaded program.

Note: If you have a secured installation (using https/wss) the port can be omitted as the default port is already 443, generally speaking use the uc-assistant URL followed by */install/win64*.

Linux (Debian 64bits)

To install the latest version, you need to add a repository linked to the xucmgt host. Edit your /etc/apt/sources.list and add the following line:

```
deb http://<xucmgt_host>:<xucmgt_port>/updates/debian jessie contrib
```

Then run

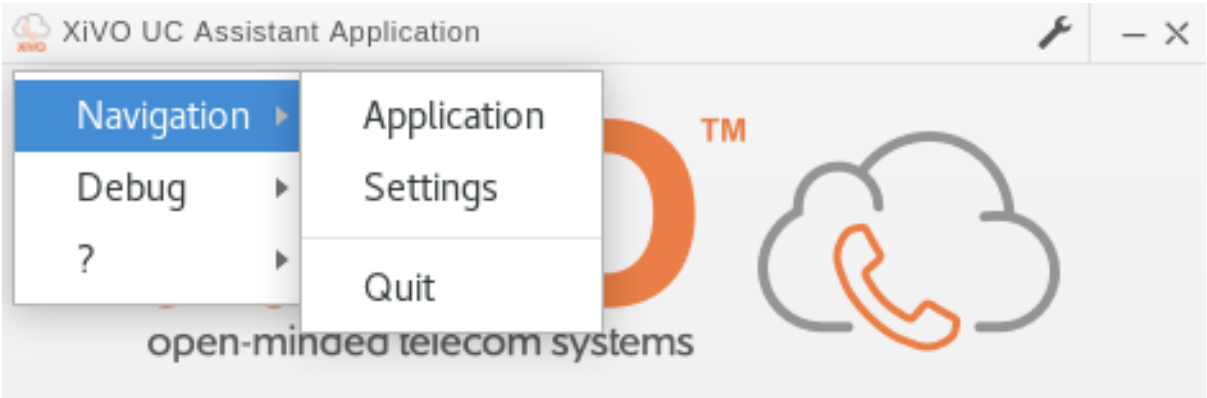
```
sudo apt-get update
sudo apt-get install xivo-desktop-assistant
```

Note: This repository is currently not signed at all.

8.2.2 Configuration

On first launch the application will display the settings page and ask for the application address. Basically it should be the IP address of your XiVO CC/UC applications. If you don't know it, you need to ask your system administrator or refer to *Update* paragraph.

Navigation



The top menu (accessible through tray icon or right click on the logo) allows you to navigate either to the application or to the settings page. If you did not enter any setting, the application will redirect you to the settings page.

About

By clicking the **?** menu you will open a popup that show you technical information about the application that can be used to report bugs.

Settings

Settings page is either accessible from top menu or by clicking directly on the wrench icon in topmost right bar.

Application Options

- **Launch at startup** if enabled, the app starts automatically when you log in to your machine.
- **Close in tray** if enabled, the app stays running in the notification area after app window is closed.

Global keyboard shortcut and Select2Call

This field allow you to define one shortcut in application to handle all basic actions that can be done for managing calls. With one combination keypress you should be able to:

- **Call** the phone number contained in your clipboard (a.k.a **Select2Call**)
- **Answer** a call if phone is ringing
- **Hangup** if you are already on call

Note:

To be able to **call** someone, you **must** beforehand have copied in your clipboard a phone number from any source (web page, e-mail, text document...)

- **Linux**: select phone number then trigger *shortcut*
- **Windows**: select phone number, type `Ctrl+C` then trigger *shortcut*

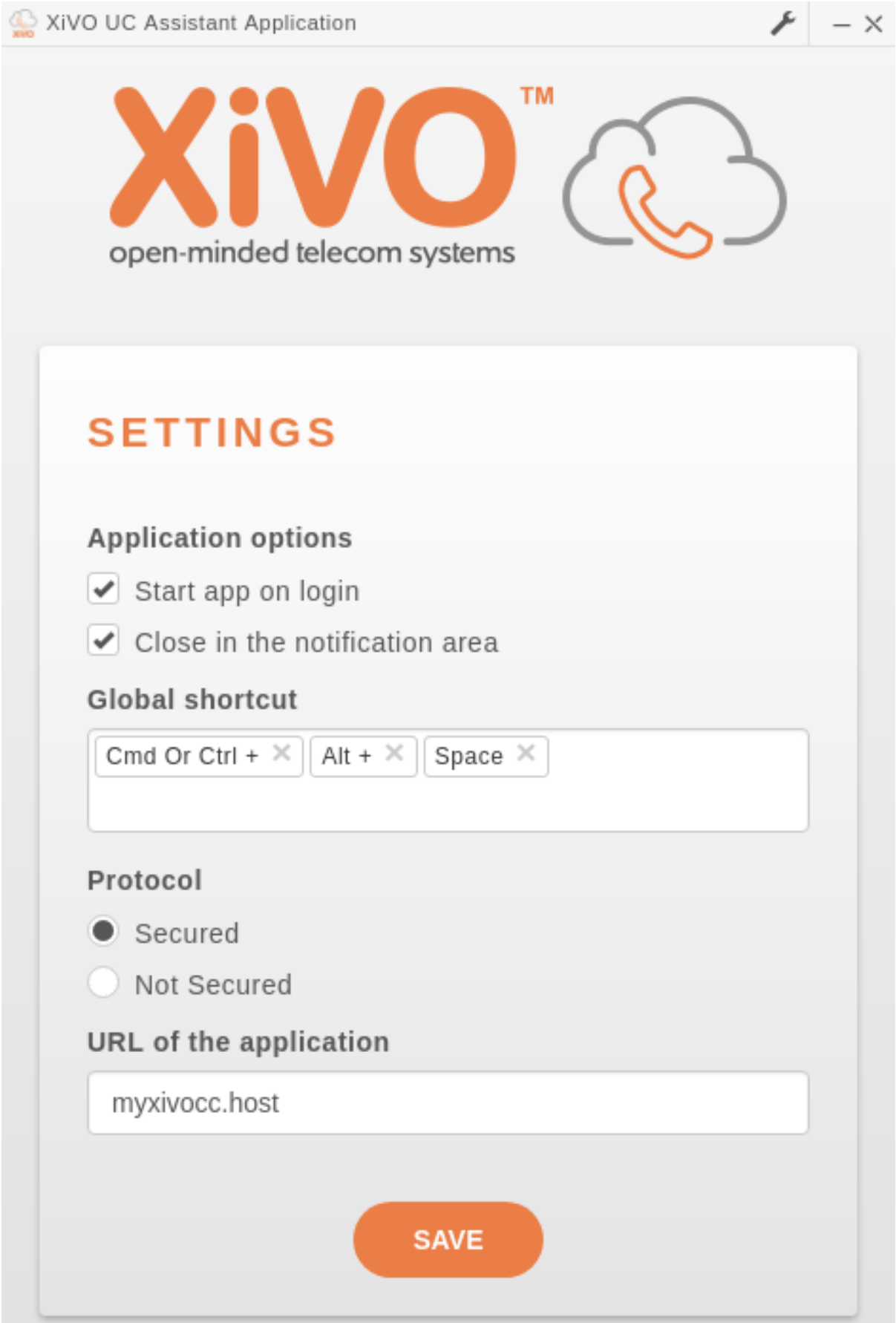
Default **Select2Call** shortcut is `Ctrl+Space` for **Windows** and **Linux**, you can either change it or disable it by leaving the field blank.

> **Warning:** You must be logged in for using global shortcut and automatic dialing to work.

Protocol and connection URL

In these two fields you need to specify the protocol and address to reach the *XiVO CC* application. The values depend on the Protocol used and the Application you want to reach. The table below list the possible value:

		Connection URL	
Protocol		**Secure (recommended)**	Non Secure (should not be used)
Application	*UC Assistant*	`XiVOCC_IP`	`XiVOCC_IP:8070`
	CC Agent	`XiVOCC_IP/ccagent`	`XiVOCC_IP:8070/ccagent`


XiVO UC Assistant Application
XiVO™
open-minded telecom systems
SETTINGS
Application options
Start app on login
Close in the notification area
Global shortcut
Cmd Or Ctrl +
Alt +
Space
Protocol
Secured
Not Secured
URL of the application
myxivocc.host
SAVE

Note that `XiVOCC_IP` or `XiVOCC_IP:8070` can be replaced by a FQDN if your administrator has set one and must **not** be prefixed by a protocol (e.g. https)

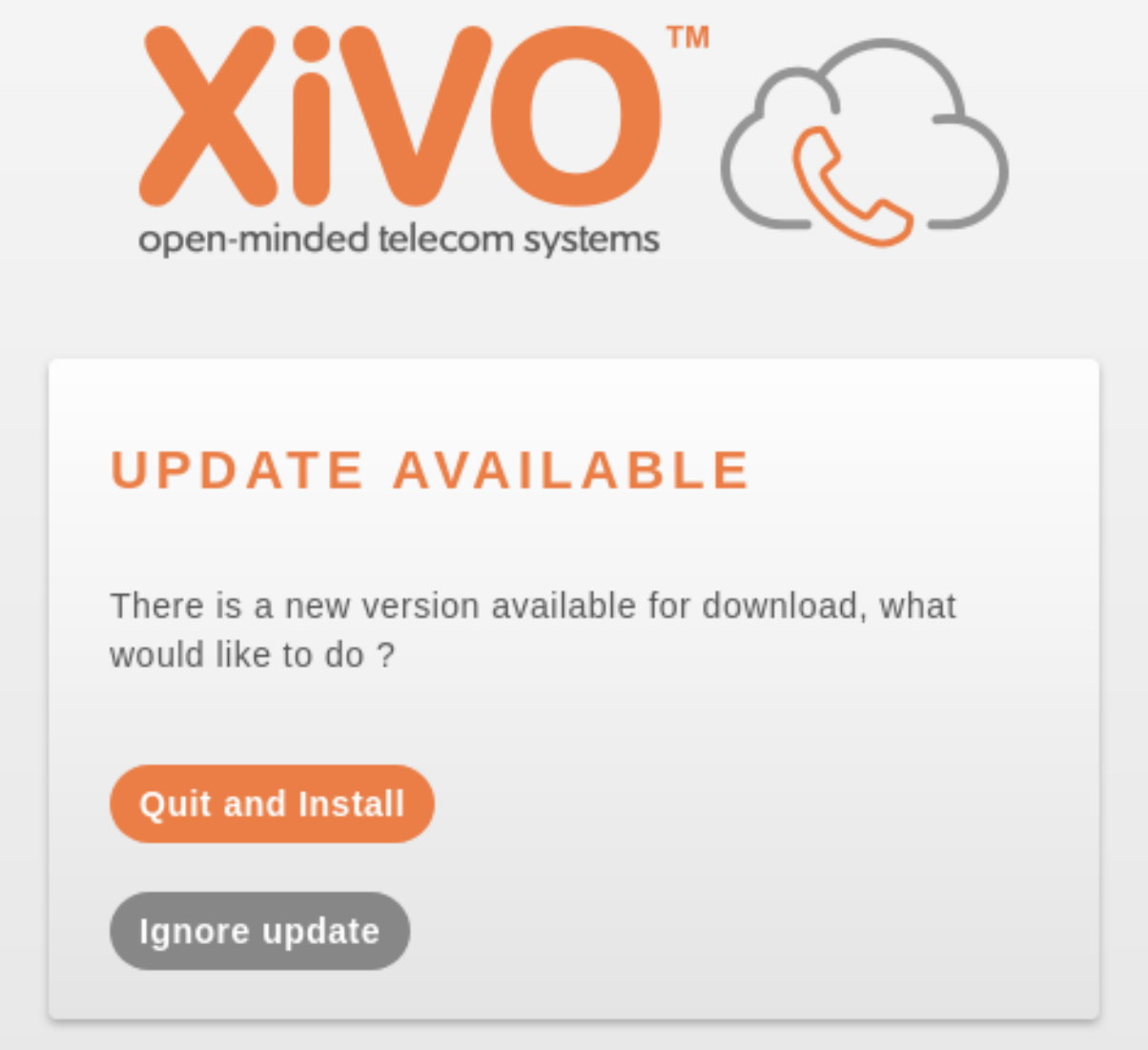
Handling callto: and tel: URLs

The *Desktop Application* can handle telephone number links that appear in web pages. The *Desktop Application* will automatically dial the number when you click on a link.

It is supported on both Windows and Linux Debian based distributions (with a desktop environment compatible with Freedesktop).

Update

On Windows, the application will check at startup for a new version of the application and offer to upgrade if one is available.



On Debian, the update relies on the package manager behavior. However you can check for any update by issuing the following commands:

```
sudo apt-get update
apt-cache policy xivo-desktop-assistant
```

Startup options

The Desktop Application can be started with following options:

- `--ignore-certificate-errors` to disable certificate verification, this option is meant **only** for test purposes. You can use it with self-signed certificates.
- `-d` to enable debug menu items

Note: On **Windows** both options must be set to the shortcut `xivo-desktop-assistant.exe` pointing to application located in `C:\Users\<USER>\AppData\Local\xivo\xivo-desktop-assistant.exe` so that **Target** of shortcut looks like for example to: `C:\Users\IEUser\AppData\Local\xivo\xivo-desktop-assistant.exe --ignore-certificate-errors -d`

Custom user data directory

It is possible to set an environment variable named `CUSTOM_USER_DATA` where will be stored application configuration. This is usefull if you want for example two shortcuts with two distincts configuration (one for *UC Assistant* and one for *CC Agent*) on the same desktop application installed.

On **Windows** here an example of two shortcuts that set the environment variable (**uc** or **cc**) and launch the application. Following lines must be updated with your **Windows** user name and correct *XiVO* version and must be set in `Target` field of the shortcut).

UC assistant:

```
C:\Windows\System32\cmd.exe /V /C "SET CUSTOM_USER_DATA=C:/Users/IEUser/AppData/
↪Local/xivo-desktop-assistant/uc&& START /D ^"C:\Users\IEUser\AppData\Local\xivo-
↪desktop-assistant\app-2018.7.0^" xivo-desktop-assistant.exe"
```

CC ccagent:

```
C:\Windows\System32\cmd.exe /V /C "SET CUSTOM_USER_DATA=C:/Users/IEUser/AppData/
↪Local/xivo-desktop-assistant/cc&& START /D ^"C:\Users\IEUser\AppData\Local\xivo-
↪desktop-assistant\app-2018.7.0^" xivo-desktop-assistant.exe"
```

Note: You can set *Run* mode to *Minimized* in shortcut *General* tab to avoid **cmd.exe** blinking at startup.

Known limitations

- Click on links using protocol *tel:* on **Windows** may not work if any version of *Skype* / *Lync* is installed on the PC.

Troubleshoot Application

If you don't succeed to reach login page of desired application (i.e. just give you the possibility to retry or to change parameters) and if you observe some errors about certificate in debug mode, you should

- Check that you installed correctly the certificate under */etc/docker/nginx/ssl* (*Signed SSL/TLS certificate for WebRTC*).
- Take care if your move a **.cer* to **.crt*. You must concatenate a key file to your **.cer* (`cat certifcate.cer certificate.key > certificate.crt`). Just rename it will not work
- Check that XUC_HOST in */etc/docker/compose/custom.env* is also configured with the same FQDN as in the certificate, not the IP address.

8.2.3 Specific Features

This section lists the specific features per application available with the desktop application.

CC Agent

Resize Window

Once *CC Agent* is launched through standalone application, a new button appears to be able to switch between a vertical minimalist view and default one.

8.3 WebRTC Environment

One can use WebRTC with *XiVO PBX* and *XiVO CC* in the following environment:

- LAN network (currently no support for WAN environment),
- with the:
  - *UC Assistant* or *CC Agent* with Chrome browser version **57** or later
  - or *Desktop Application*

8.3.1 Requirements

The **requirements** are:

- to have a microphone and headphones for your PC,
- to configure your *XiVO PBX*:
  - open WebRTC access (see:*Configuration of XiVO PBX for WebRTC*) taking *great care of the security notice*,
  - and then create users with a WebRTC line (see: *Configuration of user with WebRTC line*),
- have a SSL/TLS certificate signed by a certification authority installed on the nginx of *XiVO CC* (see: *Signed SSL/TLS certificate for WebRTC*),
- and use *https*:
  - *UC Assistant*: you must connect to the *UC Assistant* via *https* protocol,
  - *Desktop Application*: you must check *Protocol -> Secure* in the application parameters.

Note: Currently you can not have a user configured for both WebRTC and a phone set at the same time.

8.3.2 Limitations

Known limitation are :

- Voice may not be able to hear if your computer have more than 4 network interfaces up at the same time (this can happen if you use virtualization)

Note:

To check if you have more than 4 network interfaces you can type following command:

```
ls /sys/class/net
```

00:00:00
00:00:00
00:00:00

Then just use:
`ifdown <ifname>`

This will switch off network interface not required to make your call.

CHAPTER 9

API and SDK

9.1 Unified Communication Framework

This framework is mainly provided by the xuc server. It provides

- Javascript API
- Rest Web services
- Sample application
- Real Time Statistics

9.1.1 Web Socket API

The xivo solutions web socket API enables you to integrate enterprise communication functions to your business application. It exposes Cti functions using javascript methods calls and web socket events.

You may add your own handlers for your application to react to telephony / contact center events.

This API is using websockets, and therefore needs a modern browser supporting them (firefox, chrome . . .)

Developers Guide

Integration Principles

- Include the Cti and Callback javascript API from the Xuc Server

```
<script src="http://<xucserver>:<xucport>/assets/javascripts/shotgun.js" type=
↪"text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/cti.js" type="text/
↪javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/callback.js" type=
↪"text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/membership.js" type=
↪"text/javascript"></script>
```

- Include also the xc_webrtc and SIPml5 javascript APIs for the webRTC support:

```
<script src="http://<xucserver>:<xucport>/assets/javascripts/xc_webrtc.js" type=
↪"text/javascript"></script>
<script src="http://<xucserver>:<xucport>/assets/javascripts/SIPml-api.js" type=
↪"text/javascript"></script>
```

- Connect to the Xuc server using new Authentication token (see *Obtain authentication token*)

```
var wsurl = "ws://"+server+"/xuc/api/2.0/cti?token="+token;
Cti.WebSocket.init(wsurl,username,phoneNumber);
```

- Connect to the Xuc server using XiVO client username and password (**DEPRECATED**)

```
var wsurl = "ws://"+server+"/ctichannel?username="+username+"&amp;agentNumber=
↪"+phoneNumber+"&amp;password="+password;
Cti.WebSocket.init(wsurl,username,phoneNumber);
```

- **Setup event handlers to be notified on**
  - Phone state changes
  - Agent state changes
  - Statistics
  - ...
- **Eventually also webRTC handlers**
  - general
  - register
  - incoming
  - outgoing
- **Once web socket communication is established you are able to call XuC Cti javascript methods.**
  - Place a call, log an agent ....

```
...
$('#login_btn').click(function(event){
   Cti.loginAgent($('#agentPhoneNumber').val());
});
$('#logout_btn').click(function(event){
   Cti.logoutAgent();
});
$('#xuc_dial_btn').click(function(event){
   Cti.dial($("#xuc_destination").val());
});
...
```

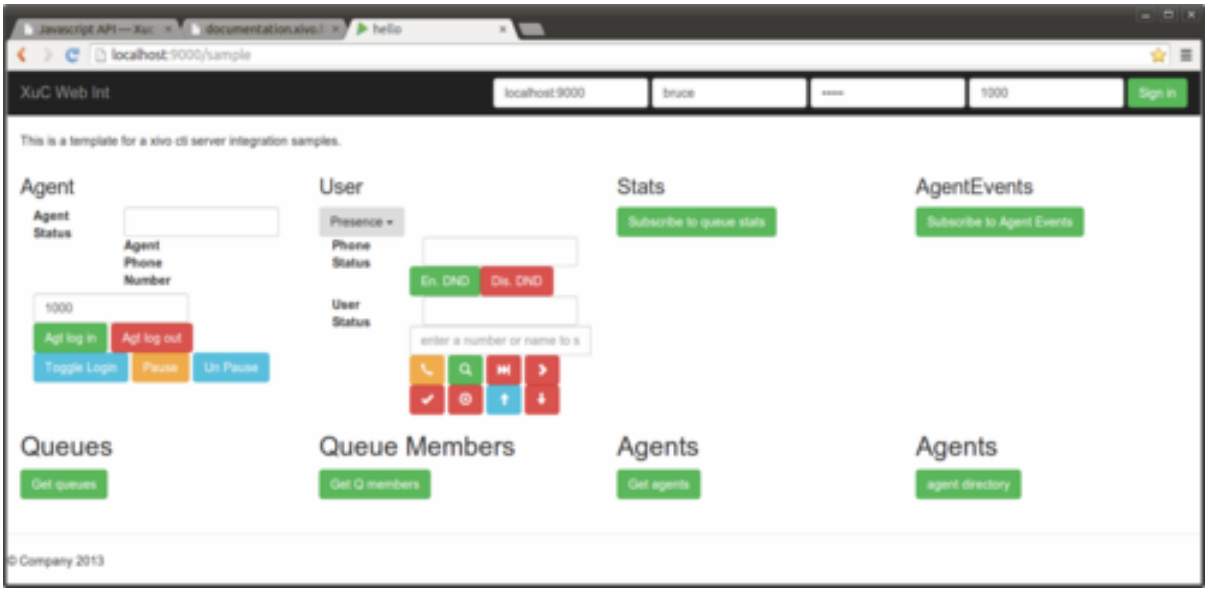
Sample Application

A sample application is provided by the XuC server. This application allows to display events and using different methods exposed by the XuC

```
http://<sucserver>:<xucport>/sample
```

You may browse and use the `sample.js` javascript file as an example

- Calling Cti methods :




```
.$('#xuc_login_btn').click(function(event) {
     Cti.loginAgent($('#xuc_agentPhoneNumber').val());
 });

 $('#xuc_logout_btn').click(function(event) {
     Cti.logoutAgent();
 });
 $('#xuc_pause_btn').click(function(event) {
     Cti.pauseAgent();
 });
 $('#xuc_unpause_btn').click(function(event) {
     Cti.unpauseAgent();
 });
 $('#xuc_subscribe_to_queue_stats_btn').click(function(event) {
     Cti.subscribeToQueueStats();
 });
 $('#xuc_answer_btn').click(function(event) {
     Cti.answer();
 });
 $('#xuc_hangup_btn').click(function(event) {
     Cti.hangup();
 });
 $('#xuc_login_btn').click(function(event) {
     Cti.loginAgent($('#xuc_agentPhoneNumber').val());
 });
 $('#xuc_logout_btn').click(function(event) {
     Cti.logoutAgent();
 });
 $('#xuc_togglelogin_btn').click(function(event) {
     Cti.toggleAgentLogin();
 });
 $('#xuc_pause_btn').click(function(event) {
     Cti.pauseAgent();
 });
 $('#xuc_unpause_btn').click(function(event) {
     Cti.unpauseAgent();
 });
 $('#xuc_subscribe_to_queue_stats_btn').click(function(event) {
     Cti.subscribeToQueueStats();
 });
```

(continues on next page)

(continued from previous page)

```
 $('#xuc_answer_btn').click(function(event) {
     Cti.answer();
 });
 $('#xuc_hangup_btn').click(function(event) {
     Cti.hangup();
 });
 $('#xuc_get_agent_call_history').click(function() {
     Cti.getAgentCallHistory(7);
 });
 $('#xuc_get_user_call_history').click(function() {
     Cti.getUserCallHistory(7);
 });

..............
```

- Declaring events handlers :

```
Cti.setHandler(Cti.MessageType.USERSTATUSES, usersStatusesHandler);
Cti.setHandler(Cti.MessageType.USERSTATUSUPDATE, userStatusHandler);
Cti.setHandler(Cti.MessageType.USERCONFIGUPDATE, userConfigHandler);
Cti.setHandler(Cti.MessageType.LOGGEDON, loggedOnHandler);
Cti.setHandler(Cti.MessageType.PHONESTATUSUPDATE, phoneStatusHandler);
Cti.setHandler(Cti.MessageType.VOICEMAILSTATUSUPDATE, voiceMailStatusHandler);
Cti.setHandler(Cti.MessageType.LINKSTATUSUPDATE, linkStatusHandler);
Cti.setHandler(Cti.MessageType.QUEUESTATISTICS, queueStatisticsHandler);
Cti.setHandler(Cti.MessageType.QUEUECONFIG, queueConfigHandler);
Cti.setHandler(Cti.MessageType.QUEUELIST, queueConfigHandler);
Cti.setHandler(Cti.MessageType.QUEUEMEMBER, queueMemberHandler);
Cti.setHandler(Cti.MessageType.QUEUEMEMBERLIST, queueMemberHandler);
Cti.setHandler(Cti.MessageType.DIRECTORYRESULT, directoryResultHandler);

Cti.setHandler(Cti.MessageType.AGENTCONFIG, agentConfigHandler);
Cti.setHandler(Cti.MessageType.AGENTLIST, agentConfigHandler);
Cti.setHandler(Cti.MessageType.AGENTGROUPLIST, agentGroupConfigHandler);
Cti.setHandler(Cti.MessageType.AGENTSTATEEVENT, agentStateEventHandler);
Cti.setHandler(Cti.MessageType.AGENTERROR, agentErrorHandler);
Cti.setHandler(Cti.MessageType.ERROR, errorHandler);
Cti.setHandler(Cti.MessageType.AGENTDIRECTORY, agentDirectoryHandler);

Cti.setHandler(Cti.MessageType.CONFERENCES, conferencesHandler);
Cti.setHandler(Cti.MessageType.CALLHISTORY, callHistoryHandler);

xc_webrtc.setHandler(xc_webrtc.MessageType.GENERAL, webRtcGeneralEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.REGISTRATION,␣
↪webRtcRegistrationEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.INCOMING, webRtcIncomingEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.OUTGOING, webRtcOutgoingEventHandler);
```

Debugging

Cti features

Cti events can be logged in the console if the `Cti.debugMsg` variable is set to `true`, you can do it directly in the developer tools console:

```
Cti.debugMsg=true;
```

You'll then get send and received events in the console log (prefixed by `S>>>` and `R<<<` respectively):

```
2016-11-23 14:48:59.180 S>>> {"claz":"web","command":"dial","destination":"111",
↪"variables":{}}
2016-11-23 14:48:59.557 R<<< {"msgType":"PhoneStatusUpdate","ctiMessage":{"status":
↪"CALLING"}}
```

Description

Login and Authentication

User login

Users can connect using login, password and phone number:

```
var wsurl = "ws://"+server+"/xuc/api/2.0/cti?token="+token;
Cti.WebSocket.init(wsurl,username,phoneNumber);
```

Agent login

An agent can be logged in using *Cti.loginAgent(agentPhoneNumber, agentId)*. For the moment, the phone number used for agent login should be the same as the one used for user login, otherwise you will get many error messages "LoggedInOnAnotherPhone".

Following cases are handled:

- agent is not logged and requests a login to a known line: the agent is logged in
- agent is not logged and requests a login to an unknown line: an error is raised:

```
{"Error":"PhoneNumberUnknown"}
```

- agent is already logged on the requested line: the agent stays logged
- agent is already logged on another line: an error is raised and the agent stays logged (on the number where he was logged before the new request). It's up to the implementation to handle this case.

```
{"Error":"LoggedInOnAnotherPhone","phoneNb":"1002","RequestedNb":"1001"}
```

- agent is not logged and requests a login to a line already used by another agent: the agent takes over the line and the agent previously logged on the line is unlogged

Generic CTI Events

Error

- Cti.MessageType.ERROR

LoggedOn

- Cti.MessageType.LOGGEDON

Directory And Favorites

Cti.directoryLookUp: function(term)

This command deprecates previously used *Cti.searchDirectory(pattern)*. This command deprecates previously used *Cti.searchDirectory(pattern)* removed in xuc xivo16 versions.

Associated Handler

- Cti.MessageType.DIRECTORYRESULT

Triggered by command *Cti.directoryLookUp(pattern)*.

```
{ "msgType": "DirectoryResult",
    "ctiMessage": {
        "entries": [
            { "status": 0, "entry": [ "hawkeye", "pierce", "1002", "0761187406",
↪"false"]},
            { "status": -2, "entry": [ "peter", "pan", "1004", "", "false"]}],
        "headers":
            [""Firstname", "Lastname", "Number", "Mobile", "Favorite"]}}
```

Cti.getFavorites: function()

Cti.addFavorite: function(contactId, source)

Cti.removeFavorite: function(contactId, source)

Cti Events

Sheet

- Cti.MessageType.SHEET

```
{
 "msgType": "Sheet",
 "ctiMessage": {
   "timenow": 1425055334,
   "compressed": true,
   "serial": "xml",
   "payload": {
     "profile": {
       "user": {
         "sheetInfo": [
           {
             "value": "http://www.google.fr/",
             "name": "popupUrl",
             "order": 10,
             "type": "url"
           },
           {
             "value": "&folder=1234",
             "name": "folderNumber",
             "order": 30,
             "type": "text"
           },
```

(continues on next page)

(continued from previous page)

```
            {
              "value": "http://www.google.fr/",
              "name": "popupUrl1",
              "order": 20,
              "type": "url"
            }
          ]
        }
      }
    },
    "channel": "SIP/1k4yj2-00000013"
  }
}
```

User Statuses

- Cti.MessageType.USERSTATUSES : “UsersStatuses”

User Status Update

- Cti.MessageType.USERSTATUSUPDATE : “UserStatusUpdate”,

User Config Update

- Cti.MessageType.USERCONFIGUPDATE : “UserConfigUpdate”,

```
{"msgType":"UserConfigUpdate",
    "ctiMessage":{"userId":9,"dndEnabled":false,"naFwdEnabled":false,
↪"naFwdDestination":"","uncFwdEnabled":false,"uncFwdDestination":"",
↪"busyFwdEnabled":false,"busyFwdDestination":"",
    "firstName":"Alice","lastName":"Johnson","fullName":"Alice Johnson",
↪"mobileNumber":"064574512","agentId":22,"lineIds":[5],"voiceMailId":58,
↪"voiceMailEnabled":true}}
```

Phone Status Update

- Cti.MessageType.PHONESTATUSUPDATE

Phone Events

- Cti.MessageType.PHONEEVENT
- Cti.MessageType.CURRENTCALLSPHONEEVENTS

Phone events are automatically sent when application is connected.

Format

```
{
    "msgType":"PhoneEvent",
    "ctiMessage":{
        "eventType":"EventRinging",
        "DN":"1118",
        "otherDN":"1058",
```

(continues on next page)

(continued from previous page)

```
        "otherDName":"Jane Black",
        "linkedId":"1447670380.34",
        "uniqueId":"1447670382.37",
        "queueName":"blue",
        "callDirection": "Incoming",
        "userData":{
            "XIVO_CONTEXT":"default","XIVO_USERID":"9","XIVO_SRCNUM":"1058","XIVO_
↪DSTNUM":"3000"
        }
    }
}
```

fields	Description
Event types	• EventReleased • EventDialing • EventRinging • EventEstablished • EventOnHold
DN	The directory number of the event
otherDN	Can be calling number of called number
otherDName	Can be name of caller of called number
queueName	Optional, the queue name for inbound acd calls
callDirection	Can be Incoming or Outgoing
UserData	Contains a list of attached data, system data XIVO_ or data attached to the call key beginning by USR_

If you use the following preprocess subroutine

```
[user_data_test]
exten = s,1,Log(DEBUG,**** set user data ****)
same  =       n,SET(USR_DATA1=hello)
same  =       n,SET(USR_DATA2=24)
same  =       n,SET(USR_DATA3=with space)
same  =       n,Return()
```

you will get these data in the events. Data can also be attached using the *Cti.dial* command.

You can also request a message with a concatenation of PhoneEvents for current calls by *Cti.getCurrentCallsPhoneEvents* command. The response to this command is formatted as follows:

```
{
    "msgType":"CurrentCallsPhoneEvents",
    "ctiMessage":{
        "events": [
        {
          "eventType":"EventRinging",
          "DN":"1118",
          "otherDN":"1058",
          "otherDName":"Jane Black",
          "linkedId":"1447670380.34",
          "uniqueId":"1447670382.37",
          "queueName":"blue",
          "callDirection": "Incoming",
          "userData":{
              "XIVO_CONTEXT":"default","XIVO_USERID":"9","XIVO_SRCNUM":"1058",
↪"XIVO_DSTNUM":"3000"
```

(continues on next page)

(continued from previous page)

```
            }
        },
        {
          "eventType":"EventEstablished",
          ...
        },
        ...
    }
}
```

Voice Mail Status Update

- VOICEMAILSTATUSUPDATE : "VoiceMailStatusUpdate",

```
{"msgType":"VoiceMailStatusUpdate","ctiMessage":{"voiceMailId":58,"newMessages":2,
↪"waitingMessages":1,"oldMessages":1}}
```

Link Status Update

- Cti.MessageType.LINKSTATUSUPDATE

User Right Profile

- Cti.MessageType.RIGHTPROFILE: "RightProfile"

```
{"msgType":"RightProfile","ctiMessage":{"profile":"Supervisor"}}
```

This message is sent upon connection to the xuc websocket. The profile can be one of: "Supervisor", "Admin", "NoRight".

Queue Statistics

- Handler on : Cti.MessageType.QUEUESTATISTICS

The handler is executed when a notification of new statistic values is received. Each message contains one or more counters for one queue. The queue is identified by its queueId. See example below for reference. The queue's id can be used to retrieve queue's configuration, see *Queue Configuration*.

Following counters are available:

- TotalNumberCallsEntered
- TotalNumberCallsAnswered
- PercentageAnsweredBefore15
- TotalNumberCallsAbandonned
- TotalNumberCallsAbandonnedAfter15
- PercentageAbandonnedAfter15
- WaitingCalls
- LongestWaitingTime
- EWT
- AvailableAgents

- TalkingAgents


```
{
    "msgType":"QueueStatistics",
    "ctiMessage":{
        "queueId":11,"counters":[{"statName":"AvailableAgents","value":0},{
↪"statName":"LoggedAgents","value":0},{"statName":"TalkingAgents","value":0},{
↪"statName":"EWT","value":0}]
    }
}
```


Some events contain a queueRef with a queue's name instead of the queueId. This issue should be eliminated in future versions.


```
{"queueRef":"travels","counters":[{"statName":"TotalNumberCallsAbandonned","value
↪":19}]}
```


Queue Calls

- Handler on: Cti.MessageType.QUEUECALLS

Awaiting calls in a queue. Subscription to the events with : *Cti.subscribeToQueueCalls(9)* (9 being the queueId). Unsubscription with: *Cti.unSubscribeToQueueCalls(9)*.


```
{"queueId":9,"calls":[{"position":1,"name":"John Doe","number":"33356782212",
↪"queueTime":"2015-07-16T10:40:16.626+02:00"}]}
```


Queue Configuration

- QUEUECONFIG : "QueueConfig",


```
{"id":8,"context":"default","name":"blue","displayName":"blue sky","number":"3506",
↪"recording_mode":"recorded","recording_activated":1}
```


Queue List

- QUEUELIST : "QueueList",


```
{
    "msgType":"QueueList",
    "ctiMessage":[
        {"id":170,"context":"default","name":"bluesky","displayName":"Bl Record",
↪"number":"3012"},
        {"id":5,"context":"default","name":"noagent","displayName":"noagent",
↪"number":"3050"},
        {"id":6,"context":"default","name":"__switchboard_hold","displayName":
↪"Switchboard hold","number":"3005"},
        {"id":173,"context":"default","name":"outbound","displayName":"outbound",
↪"number":"3099"},
        {"id":2,"context":"default","name":"yellow","displayName":"yellow stone",
↪"number":"3001"},
        {"id":7,"context":"default","name":"green","displayName":"green openerp",
↪"number":"3006"},
        {"id":3,"context":"default","name":"red","displayName":"red auto polycom",
↪"number":"3002"},
        {"id":11,"context":"default","name":"pool","displayName":"Ugips Pool",
↪"number":"3100"},
```


(continues on next page)

(continued from previous page)


```
        {"id":4,"context":"default","name":"__switchboard","displayName":
↪"Switchboard","number":"3004"}
        ]
}
```


Queue Member

- Handler on : Cti.MessageType.QUEUEMEMBER

Received when an agent is associated to a queue or a penalty is updated. Penalty is -1 when agent is removed from a queue


```
{"agentId":19,"queueId":3,"penalty":12}
```


Queue Member List

- Handler on : Cti.MessageType.QUEUEMEMBERLIST


```
{
    "msgType":"QueueMemberList",
    "ctiMessage":[
        {"agentId":129,"queueId":8,"penalty":2},
        {"agentId":139,"queueId":168,"penalty":2},
        {"agentId":129,"queueId":10,"penalty":0},
        {"agentId":129,"queueId":11,"penalty":0}
        ]
}
```


Agent State Event

- Cti.MessageType.AGENTSTATEEVENT
  - AgentLogin (DEPRECATED: Agent are now going directly from AgentLoggedOut to AgentReady)


```
{"name":"AgentLogin","agentId":19,"phoneNb":"1000","since
↪":1423839787,"queues":[8,14,170,4,1],"cause":""}
```


  - AgentReady


```
{"name":"AgentReady","agentId":19,"phoneNb":"1000","since":0,"queues
↪":[8,14,170,4,1],"cause":"available"}
```


  - AgentOnPause


```
{"name":"AgentOnPause","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1],"cause":"available"}
```


  - AgentOnWrapup


```
{"name":"AgentOnWrapup","agentId":19,"phoneNb":"1000","since":2,
↪"queues":[8,14,170,4,1],"cause":"available"}
```


  - AgentRinging

```
{"name":"AgentRinging","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1],"cause":"available"}
```

- AgentDialing

```
{"name":"AgentDialing","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1],"cause":"available"}
```

- AgentOnCall

```
{"msgType":"AgentStateEvent","ctiMessage":
    {"name":"AgentOnCall","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1],
        "cause":"available","acd":false,"direction":"Incoming",
↪"callType":"External","monitorState":"ACTIVE"}}
```

- AgentLoggedOut

```
{"name":"AgentLoggedOut","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1],"cause":"available"}
```

Agent Error

- Cti.MessageType.AGENTERROR

Agent Directory

- Cti.MessageType.AGENTDIRECTORY

Triggered by command *Cti.getAgentDirectory*

```
{"directory": [
    { "agent":
        {"context": "default", "firstName": "bj", "groupId": 1, "id": 8, "lastName
↪": "agent", "number": "2000"},
        "agentState": {"agentId": 8, "cause": "", "name": "AgentReady", "phoneNb":
↪"1001", "queues": [1, 2], "since": 2 }}]}
```

Agent Configuration

- AGENTCONFIG: “AgentConfig”

Triggered when agent configuration changes

```
{"id":23,"firstname":"Jack","lastname":"Flash","number":"2501","context":"default",
↪"member": [
    {"queue_name":"queue1","queue_id":1,"interface":"Agent/2501","penalty":1,
↪"commented":0,"usertype":"Agent","userid":1,"channel":"Agent","category":"Queue",
↪"position":1},
    {"queue_name":"queue2","queue_id":2,"interface":"Agent/2501","penalty":2,
↪"commented":0,"usertype":"Agent","userid":1,"channel":"Agent","category":"Queue",
↪"position":1}],
"numgroup":1,"userid":1}
```

Agent List

- Cti.MessageType.AGENTLIST

Receives agent configuration list in a javascript Array : Command *Cti.getList("agent");*


```
[
    {"id":24,"firstName":"John","lastName":"Waynes","number":"2601","context":
↪"default","groupId":1},
    {"id":20,"firstName":"Maricé","lastName":"Saprïtchà","number":"2602","context":
↪"default","groupId":1},
    {"id":147,"firstName":"Etienne","lastName":"Burgad","number":"30000","context":
↪"default","groupId":1},
    {"id":148,"firstName":"Caroline","lastName":"HERONDE","number":"29000","context
↪":"default","groupId":2},
    {"id":149,"firstName":"Eude","lastName":"GARTEL","number":"75000","context":
↪"default","groupId":3},
    {"id":22,"firstName":"Alice","lastName":"Johnson","number":"2058","context":
↪"default","groupId":5}
]
```


Agent Listen

- AGENTLISTEN: "AgentListen",

Receives agent listen stop / start event, received automatically if user is an agent, no needs to subscribe.


```
{"started":false,"phoneNumber":"1058","agentId":22}
```


Agent Group List

- AGENTGROUPLIST : "AgentGroupList"

Agent group list triggered by command : *Cti.getList("agentgroup")*


```
[
    {"id":1,"name":"default"},
    {"id":2,"name":"boats"},
    {"id":3,"name":"broum"},
    {"id":4,"name":"bingba3nguh"},
    {"id":5,"name":"salesexpert"},
    {"id":6,"name":"a_very_long_group_name"}
]
```


Agent Statistics

Received on subscribe to agent statistics with method *Cti.subscribeToAgentStats()*, current statistics are received automatically on subscribe.

- AGENTSTATISTICS : "AgentStatistics"


```
{"id":22,
    "statistics":[
        {"name":"AgentPausedTotalTime","value":0},
        {"name":"AgentWrapupTotalTime","value":0},
        {"name":"AgentReadyTotalTime","value":434},
        {"name":"LoginDateTime","value":"2015-04-27T08:15:01.081+02:00"},
        {"name":"LogoutDateTime","value":"2015-04-27T08:14:49.427+02:00"}
```


(continues on next page)

(continued from previous page)

```
        ]
}
```

Call History

Cti.getUserCallHistory(size)

Get the call history of the logged in user, limited to the last *size* calls.

Cti.getAgentCallHistory(size)

Get the call history of the logged in agent, limited to the last *size* calls.

Cti.getQueueCallHistory(queue, size)

Get a call history for a queue or a set of queues. You may pass part of a queue name (not display name).

i.e. pass bl if you want to match queue name blue, black and blow

Associated Handler CALLHISTORY

Received when calling the above methods *Cti.getAgentCallHistory(size)* or *Cti.getUserCallHistory(size)* .

- CALLHISTORY : "CallHistory"

```
{
    "start":"2014-01-01 08:00:00",
    "duration":"00:21:35",
    "srcNum":"0115878",
    "dstNum":"2547892",
    "status":"answered"
}
```

For queue calls status can be :

- full - full queue
- closed - closed queue
- joinempty - call arrived on empty queue
- leaveempty - exit when queue becomes empty
- divert_ca_ratio -call redirected because the ratio waiting calls/agents was exceeded
- divert_waittime - call redirected because estimated waiting time was exceeded;
- answered - call answered
- abandoned - call abandoned
- timeout - maximum waiting time exceeded

For other calls

- emitted
- missed
- ongoing

Callback Events

Callback lists

Received when calling *Callback.getCallbackLists()*.

- CALLBACKLISTS : "CallbackLists"

```
{"uuid":"b0849ac0-4f4a-4ed0-9386-53ab2afd94b1",
 "name":"Liste de test",
 "queueId":1,
 "callbacks":[
   {"uuid":"a967da84-bc41-4bf4-a4fc-2bcc54e11606",
   "listUuid":"b0849ac0-4f4a-4ed0-9386-53ab2afd94b1",
   "phoneNumber":"0230210082",
   "mobilePhoneNumber":"0789654123",
   "firstName":"Alice",
   "lastName":"O'Neill",
   "company":"YourSociety",
   "description":null,
   "agentId":null,
   "dueDate": "2016-08-01",
   "preferredPeriod": {
      "default": false,
      "name": "Afternoon",
      "periodStart": "14:00:00",
      "periodEnd": "17:00:00",
      "uuid": "d3270038-e20e-498a-af71-3cf69b5cc792"
   }}
 ]}
```

Callback Taken

Received after taking a callback with *Callback.takeCallback(uuid)*.

- CALLBACKTAKEN : "CallbackTaken"

```
{"uuid":"a967da84-bc41-4bf4-a4fc-2bcc54e11606",
 "agentId":2}
```

Callback Started

Received after starting a callback with *Callback.startCallback(uuid, phoneNumber)*.

- CALLBACKSTARTED : "CallbackStarted"

```
{"requestUuid":"a967da84-bc41-4bf4-a4fc-2bcc54e11606",
 "ticketUuid":"8e82de0f-847a-4606-97bf-bef5a18ea8b0"}
```

Callback Clotured

Received after giving to a callback a status different of `Callback`.

- CALLBACKCLOTURED : "CallbackClotured"

```
{"uuid":"a967da84-bc41-4bf4-a4fc-2bcc54e11606"}
```

Callback Released

Received after releasing a callback with *Callback.releaseCallback(uuid)*.

- CALLBACKRELEASED : "CallbackReleased"

```
{"uuid":"a967da84-bc41-4bf4-a4fc-2bcc54e11606"}
```

Callback Updated

Received when calling *Callback.updateCallbackTicket(uuid, status, description, dueDate, periodUuid)* with a new due date or period.

- CALLBACKREQUESTUPDATED : "CallbackRequestUpdated"

```
{"request":{
   "uuid":"a967da84-bc41-4bf4-a4fc-2bcc54e11606",
   "listUuid":"b0849ac0-4f4a-4ed0-9386-53ab2afd94b1",
   "phoneNumber":"0230210082",
   "mobilePhoneNumber":"0789654123",
   "firstName":"Alice",
   "lastName":"O'Neill",
   "company":"YourSociety",
   "description":null,
   "agentId":null,
   "dueDate": "2016-08-01",
   "preferredPeriod": {
      "default": false,
      "name": "Afternoon",
      "periodStart": "14:00:00",
      "periodEnd": "17:00:00",
      "uuid": "d3270038-e20e-498a-af71-3cf69b5cc792"
   }
}}
```

Membership Events

User default membership

Received when calling *Membership.getUserDefaultMembership(userId)*.

- USERQUEUEDEFAULTMEMBERSHIP: "UserQueueDefaultMembership"

```
{
  "userId":186,
  "membership": [
    {"queueId":8,"penalty":1},
    {"queueId":17,"penalty":0},
    {"queueId":18,"penalty":0},
    {"queueId":23,"penalty":0}
  ]
}
```

Agent Methods

Cti.loginAgent(agentPhoneNumber, agentId)

Log an agent

Cti.logoutAgent(agentId)

Un log an agent

Cti.pauseAgent(agentId)

Change agent state to pause

Cti.unpauseAgent(agentId)

Change agent state to ready

Cti.listenAgent(agentId)

Listen to an agent

Cti Methods

Cti.changeUserStatus()

Update user status using a cti server configured status name

Cti.dnd(state)

Set or unset do not disturb, state true or false

Phone Methods

Cti.setData(variables)

Attach data to the device current calls. When there is a call connected to a device, Data can be attached by passing key values as a json object *Cti.setData("{'var1':'val1','USR_var2':'val2'}");*

The folowing json message is then sent to the server :

```
{"claz":"web","command":"setData","variables":{"var1":"val1","USR_var2":"val2"}}
```

When the call is transfered i.e. (*Cti.directTransfer(destination)*), data is sent in the event ringing see *Phone Events*, and in subsequent events. Data is not propagated in the reporting database.

```
{"eventType":"EventRinging","DN":"1000","otherDN":"0427466347","linkedId":
↪"1469709757.74","uniqueId":"1469709960.78","queueName":"bluesky",
   "userData":{     "XIVO_CONTEXT":"from-extern",
                    "XIVO_USERID":"1",
                    "USR_var1":"val1",
                    "USR_var2":"val2",
                    "XIVO_EXTENPATTERN":"_012305XXXX",
```

(continues on next page)

(continued from previous page)

```
                    "XIVO_SRCNUM":"0427466347",
                    "XIVO_DST_FIRSTNAME":"Brucé",
                    "XIVO_DSTNUM":"0123053012","XIVO_DST_LASTNAME":"Wail"}}
```

Note that *USR_* prefix is added to the key, if the key does not start with it. Only attached data beginning with *USR_* are sent back to the client API.

> **Warning:** When transfering a call, these variables are attached to the new channel however to prevent propagation on all trunk channels, your trunk name must contain 'trunk' so they can be distinguished from sip devices.

Cti.dial(destination, variables)

Place a call to destination with the provided variables. Variables must take the following form:

```
{
    var1: "value 1",
    var2: "value 2"
}
```

USR_var1 and USR_var2 will be attached to the call and propagated to *Phone Events*

Cti.dialFromMobile(destination, variables)

Place a call from logged user's mobile number to destination with the provided variables. Variables must take the following form:

```
{
    var1: "value 1",
    var2: "value 2"
}
```

USR_var1 and USR_var2 will be attached to the call and propagated to *Phone Events*

Cti.dialFromQueue(destination, queueId, callerId, variables)

Creates outgoing call to `destination` from some free Agent attached to `queueId`. Caller id on both sides is set to `callerId`.

Variables must take the following form:

```
{
    var1: "value 1",
    var2: "value 2"
}
```

USR_var1 and USR_var2 will be attached to the call and propagated to *Phone Events*

Limitations: Queue No Answer settings does not work - see *No Answer*. Except: when there is no free Agent to queue (none attached, all Agents on pause or busy), then No answer settings work (but Fail does not).

Note: Line should be configured with enabled "Ring instead of On-Hold Music" enabled (on "Application: tab in queue configuration - see *Queues*). Otherwise the queue will answers the call and the destination rings even if there are no agents available.

Cti.originate(destination)

Originate a call

Cti.hangup()

Hangup a call

Cti.answer()

Answer a call

Cti.hold()

Put current call on hold

Cti.directTransfer(destination)

Tranfert to destination

Cti.attendedTransfer(destination)

Start a transfer to a destination

Cti.completeTransfer()

Complete previously started transfer

Cti.cancelTransfer()

Cancel a transfer

Cti.monitorPause(agentId)

Pause call recording

Note: You can only pause the recording of a call answered by an agent (i.e. a call sent via a Queue towards an Agent).

Cti.monitorUnpause(agentId)

Unpause call recording

Note: You can only pause the recording of a call answered by an agent (i.e. a call sent via a Queue towards an Agent).

Cti.getList(objectType)

Request a list of configuration objects, objectType can be :

- queue
- agent
- queuemember

Triggers handlers QUEUELIST, AGENTLIST, QUEUEMEMBERLIST. Subscribes to configuration modification changes, handlers QUEUECONFIG, AGENTCONFIG, QUEUEMEMBER can also be called

Cti.getConferenceRooms()

Request the list of conference rooms. Also receive event when the list is updated.

> **Warning:** The xuc user must have a line.


```
[
  {
    "number": "4000",
    "name": "public",
    "pinRequired": false,
    "startTime": 1519659524032,
    "members": [
      {
        "joinOrder": 1,
        "joinTime": 1519659524032,
        "muted": false,
        "name": "James Bond",
        "number": "1002"
      }
    ]
  },
  {
    "number": "4001",
    "name": "conference_support",
    "pinRequired": true,
    "startTime": 0,
    "members": []
  }
]
```


Cti.getCurrentCallsPhoneEvents()

Request PhoneEvents for current device calls. See *Phone Events* for answer description.

Cti.setAgentQueue(agentId, queueId, penalty)

- agentId (Integer) : id of agent, returned in message *Agent Configuration*
- queueId (Integer) : id of queue, returned in message *Queue Configuration*
- penaly (Integer) : positive integer

If agent is not associated to the queue, associates it, otherwise changes the penalty

On success triggers a *Queue Member* event, does not send anything in case of failure :

```
{"agentId":<agentId>,"queueId":<queueId>,"penalty":<penalty>}
```

Cti.removeAgentFromQueue(agentId, queueId)

- agentId (Integer) : id of agent, returned in message *Agent Configuration*
- queueId (Integer) : id of queue, returned in message *Queue Configuration*

On success triggers a queue member event with penalty equals to -1, does not send anything in case of failure :

```
{"agentId":<agentId>,"queueId":<queueId>,"penalty":-1}
```

Cti.subscribeToAgentStats()

Subscribe to agent statistics notification. When called all current statistics are receive, and a notification is received for each updates. Both initial values and updates are transmitted by the *Agent Statistics* events.

Cti.subscribeToQueueStats()

This command subscribes to the queue statistics notifications. First, all actual statistics values are sent for initialisation and then a notification is sent on each update. Both initial values and updates are transmitted by the QUEUESTATISTICS events.

Cti.naFwd(destination,state)

Forward on non answer

Cti.uncFwd(destination,state)

Unconditionnal forward

Cti.busyFwd(destination,state)

Forward on busy

Callback Commands

Callback.getCallbackLists()

Retrieve the lists of callbacks with teir associated callback requests, and subscribe to callback events.

Callback.takeCallback(uuid)

Take the callback with the given uuid with the logged-in agent.

Callback.releaseCallback(uuid)

Release the callback which was previously taken

Callback.startCallback(uuid, phoneNumber)

Launch the previously taken callback with the provided phone number.

Callback.updateCallbackTicket(uuid, status, description, dueDate, periodUuid)

Update a callback ticket wih the provided description and status. Allowaed values for status are:

- NoAnswer
- Answered
- Fax
- Callback

dueDate is an optional parameter specifying the new due date using ISO format ("YYYY-MM-DD").

periodUuid is an optional parameter specifying the new preferred period for the callback.

Membership Commands

Membership.init(cti)

Initialize the Membership library using the given Cti object.

Membership.getUserDefaultMembership(userId)

Request the default membership for the given user id. Warning, the userId is not the same as the agentId.

Membership.setUserDefaultMembership(userId, membership)

Set the default membership for the given user id. Warning, the userId is not the same as the agentId. 'membership' should be an array of Queue membership like:

```
[
  {"queueId":8,"penalty":1},
  {"queueId":17,"penalty":0},
  {"queueId":18,"penalty":0},
  {"queueId":23,"penalty":0}
]
```

Membership.setUsersDefaultMembership(userIds, membership)

Set the default membership for the given array of user id. Warning, the userId is not the same as the agentId. 'userIds' should be an array of user id like :

```
[1, 2, 3]
```

'membership' should be an array of Queue membership like:

```
[
  {"queueId":8,"penalty":1},
  {"queueId":17,"penalty":0},
  {"queueId":18,"penalty":0},
  {"queueId":23,"penalty":0}
]
```

Membership.applyUsersDefaultMembership(userIds)

Apply the existing default configuration to a set of users. Warning, the userId is not the same as the agentId. 'usersIds' should be an array of userId like:

:: [1, 2, 7, 9]

Security considerations

Defining a user profile in the ConfigMGT impact the behavior of this api.

No Profile

If no profile is found, the behavior falls back on the Admin Profile behavior.

Admin Profile

An admin profile will be allowed to receive all events and send all commands.

Supervisor Profile

A supervisor profile has the some properties impacting the events he can receive:

- A list of queue which will filter the following events based on the queues in this list (send event only for queues defined in the list):
  - QueueList
  - QueueMemberList
  - QueueStatistics
- A list of groups which will filter the following events based on the groups in this list (send event only if matching agent group is in the list):
  - AgentStateEvent
  - AgentStatistics
  - AgentGroupList
  - AgentList

Conferences API

This api is to manipulate voice conference room

You can refer to *Conference Room* configuration for organizer feature

Conference Events

See associated *Conference Methods*

Conference Event

- Cti.MessageType.CONFERENCEEVENT

Received when you enter or leave a conference room.

```
{
  "eventType": "Join",
  "uniqueId": "1519658665.8",
  "conferenceNumber": "4001",
  "conferenceName": "conference_support",
  "participants": [
    {
      "index": 1,
      "callerIdName": "James Bond",
      "callerIdNum": "1002",
      "since": 0,
      "isTalking": false,
      "role": "User",
      "isMuted": false,
      "isMe": true,
    }
  ],
  "since": 0
}
```

Fields description :

- eventType: “Join” or “Leave”
- uniqueId: channel / call unique id entering this conference (related to Phone events)
- conferenceNumber: DN Number of the joined/left conference
- conferenceName: Name of the joined/left conference
- participants: array of participant
- since: delay in seconds since the beginning of the conference

Conference Participant Event

- Cti.MessageType.CONFERENCEPARTICIPANTEVENT

Received when a participant enter, leave, or be updated in your conference room.

```
{
  "eventType": "Update",
  "uniqueId": "1519658665.8",
  "conferenceNumber": "4001",
  "index": 1,
  "callerIdName": "James Bond",
  "callerIdNum": "1002",
  "since": 0,
  "isTalking": true,
  "role": "Organizer",
  "isMuted": false,
  "isMe": false
}
```

Fields description :

- eventType: “Join”, “Leave” or “Update”
- uniqueId: channel / call unique id entering this conference (related to Phone events)

- conferenceNumber: DN Number of the joined/left conference
- conferenceName: Name of the joined/left conference
- index: position of the participant in the conference
- callerIdName: Name of the participant
- callerIdNum: DN Number of the participant
- since: delay in seconds since the beginning of the conference
- isTalking: true or false if participant is talking
- role: participant role, either “User” or “Organizer”
- isMuted: indicate if participant is muted
- isMe: indicate if participant is the current user

Conference Command Error Event

- Cti.MessageType.CONFERENCECOMMANDERROR

Received after a conference command (mute/unmute, muteme/unmuteme, muteall/unmuteall, kick) if an error is encountered while processing the command

```
{
  "error": "NotAMember"
}
```

The error code can be one of the following:

- NotAMember: The current user is not a member of the given conference number.
- NotOrganizer: The current user is not an organizer in the given conference number and cannot perform the command.
- CannotKickOrganizer: You cannot kick an organizer out of a conference.
- ParticipantNotFound: The targeted participant wasn’t found.

Conference Methods

Cti.conference()

Start a conference using phone set capabilities

Cti.conferenceMuteMe(conferenceNumber)

Mute the current user in the given conference.

See also *conference_command_error*.

Cti.conferenceUnmuteMe(conferenceNumber)

Un-mute the current user in the given conference

See also *conference_command_error*.

Cti.conferenceMuteAll(conferenceNumber)

Mute all attendees except current user in the given conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

Cti.conferenceUnmuteAll(conferenceNumber)

Un-Mute all attendees in the given conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

Cti.conferenceMute(conferenceNumber, index)

Mute an attendee by its index in the given conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

Cti.conferenceUnmute(conferenceNumber, index)

Un-Mute an attendee by its index in the given conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

Cti.conferenceKick(conferenceNumber, index)

Kick an attendee out of the given conference by its index. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

WebRTC API

WebRTC features

The WebRTC debug can be activated separately by the following method:

```
xc_webrtc.setDebug(sipml5level, event, handler)
```

Where:

- sipml5level refers to the SIPml5 library log level string as described on SIPml5 log level documentation,
- event is a boolean value activating event logging (each event is prefixed by `RE<<<`),
- handler is a boolean value activating logging of message handler subscription/unsubscription.

WebRTC on sample page

Once logged on the sample page, you can init the webRTC through the init button, follow events shown in the webRTC section and send and receive calls. You can terminate a call by the terminate button in the phone section. Attended transfer can be performed using xfer buttons. Hold and DTMF features are available via the webRTC API. You can receive or make up to 2 calls, answer and hold methods do their best to put other calls on hold when answering or hold the call which is not held (or resume the one which is held), but currently the SIPml5 session ids are not exposed, so you have to avoid getting in a situation where it's not clear from the context what needs to be done, for example putting on hold two calls in the same time.

Current browsers doesn't allow media sharing without secure connections - https and wss. The xivoxc_nginx docker image contains the configuration required for loading the sample page over a secure connection using an auto-signed certificate. This certificate is automatically generated by the installation script. It is meant to be used only for test purposes, you should replace it by a signed certificate before switching to production. The sample page is available on the following address: https://MACHINE_IP:8443/sample

webRTC Methods

Once the cti login done, you can init the webRTC component by calling the *xc_webrtc.init* method.

xc_webrtc.init(name, ssl, websocketPort, remoteAudio, ip)

Init the webRTC connection and register the user's line.

- name - user's login to get the line details,
- ssl - if set to true the wss is used,
- websocketPort, ip - port and address for the webRTC websocket connection, when ip is not passed the xivo ip is used,
- remoteAudio - id of the HTML5 audio element for remote audio player, if not passed 'audio_remote' is used. The element should look like:

```
<audio id="audio_remote" autoplay="autoplay"></audio>
```

xc_webrtc.dial(destination, video)

Start a webRTC call to the *destination*.

Warning: The video call support is currently an experimental feature and the API can change any time.

The 'video' parameter is optional, if omitted it's considered as false and allows you to start an audio/video call. Currently, multiple concurrent video calls are not supported, all video calls reuse the same video elements. The local video element has to be created by the user of the API with id `video-local` and a second video element for remote video stream with id `video-remote`.

xc_webrtc.answer()

Answer an incoming webRTC call.

Note: The library accepts as an optional parameter the session id, but these ids are not currently reported in events. This parameter is currently reserved for internal use.

xc_webrtc.hold()

Toggle hold on a webrtc call.

Note: The library accepts as an optional parameter the session id, but these ids are not currently reported in events. This parameter is currently reserved for internal use.

xc_webrtc.dtmf(key)

Send a DTMF.

xc_webrt.attendedTransfer(destination)

Start an attendedTransfer to the destination. This method designed to work with the AMI based transfer implemented by the XUC server, so it first puts current calls on hold and then starts a new call in an auto-answer mode.

xc_webrtc.compeleteTransfer()

Complete an attended transfer. It's a simple wrapper of the Cti method introduced to complete the API of the xc_webrtc library.

xc_webrtc.setHandler(eventName, handler)

Set a handler for eventName from xc_webrtc.MessageType.

xc_webrtc.disableICE()

Disable ICE server use, only LAN addresses will be used in the SDP.

xc_webrtc.setIceUrls(urls)

Set a list of STUN/TURN servers, for example:

```
[{ url: 'stun:stun.l.google.com:19302'}, { url:'turn:turn.server.org', username:␣
↪'user', credential:'myPassword'}]
```

webRTC Events

There are four groups of events:

- general,
- register,
- incoming,
- outgoing.

List of associated events is defined in the *xc_webrtc.General*, *xc_webrtc.Registration*, *xc_webrtc.Incoming*, *xc_webrtc.Outgoing*. See the xc_webrtc.js on https://gitlab.com/xivo.solutions/xucserver/blob/master/app/assets/javascripts/xc_webrtc.js. The error state events contains a description in the reason field. Call establishment event contains *caller* or *callee* detail. Use the sample page to see some examples.

9.1.2 Rest API

General form

http://localhost:\protect\T1\textdollarxucport/xuc/api/\protect\T1\textdollarversion/\protect\T1\textdollarmethod/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

withHeaders((“Content-Type”, “application/json”))

- $xucport : Xuc port number (default 8090)
- $version : 1.0 or 2.0
- $method : See available methods below
- $domain : Represents a connection site, can be anything
- $username : XiVO client user username

Events

Xuc post JSON formated events on URL `eventUrl = "http://localhost:8090/xivo/1.0/event/avencall.com/dropbox/"` configured in /usr/share/xuc/application.conf

Phone Event Notification

Related to a username, phone event is in message payload same structure as javascript *Phone Events*

```
{
  "username":"alicej",
  "message":{
     "msgType":"PhoneEvent",
     "ctiMessage":{"eventType":"EventDialing","DN":"1058","otherDN":"3000",
↪"linkedId":"1447670380.34","uniqueId":"1447670380.34","userData":{"XIVO_USERID":
↪"9"}}}}
```

Obtain authentication token

Authentication is needed to use version 2.0 of the API

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/auth/login

```
curl -XPOST -d '{"login": "<login>", "password": "<password>"}' -H "Content-Type:␣
↪application/json"  http://localhost:8090/xuc/api/2.0/auth/login
```

Will retrieve an object

```
{login: "<login>", token: "<token>"}
```

or an error

```
{error:"<error_code>", message:"<error_message>"}
```

where error_code is one of:

- UserNotFound
- InvalidPassword
- InvalidJson
- UnhandledError

This token can then be used with the *CTI Authentication* and *Check authentication token*.

Obtain authentication token (SSO/Kerberos)

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/auth/sso

```
curl -XGET http://localhost:8090/xuc/api/2.0/auth/sso
```

Will retrieve an object

```
{login: "<login>", token: "<token>"}
```

or an error

```
{error:"<error_code>", message:"<error_message>"}
```

where error_code is one of:

- UserNotFound
- SsoAuthenticationFailed
- UnhandledError

This token can then be used with the *CTI Authentication* and *Check authentication token*.

Obtain authentication token (SSO/CAS)

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/auth/cas?service=https://xucmgt.example.org&ticket=ST-11-Qsicgrh1mZ3dgoeOx7m6-af27d9025e0c

```
curl -XGET http://localhost:8090/xuc/api/2.0/auth/cas?service=https://xucmgt.
↪example.org&ticket=ST-11-Qsicgrh1mZ3dgoeOx7m6-af27d9025e0c
```

Will retrieve an object

```
{login: "<login>", token: "<token>"}
```

or an error

```
{error:"<error_code>", message:"<error_message>"}
```

where error_code is one of:

- UserNotFound: User was authenticated using SSO but the corresponding user does not exist on XiVO
- CasServerUrlNotSet: XiVOCC containers are not configured (see *CAS SSO Authentication*)
- CasServerInvalidResponse: The CAS server returned an invalid response
- CasServerInvalidParameter: The Parameters sent to the CAS Server are invalid
- CasServerInvalidRequest: The Request to the CAS server is invalid
- CasServerInvalidTicketSpec: The ticket specification is invalid
- CasServerUnauthorizedServiceProxy: The CAS service proxy is not authorized

- CasServerInvalidProxyCallback: The CAS service proxy callback is invalid
- CasServerInvalidTicket: The ticket is invalid (probably expired or defined for another service)
- CasServerInvalidService: The service is invalid
- CasServerInternalError: CAS Server internal error
- UnhandledError

This token can then be used with the *CTI Authentication* and *Check authentication token*.

Check authentication token

You can check the validity of a token using the following web service with an Authorization header set to 'Bearer <token>'

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/auth/check

```
curl -X GET -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" "http://localhost:8090/xuc/api/2.0/
↪auth/check"
```

Will retrieve an object with a refreshed token

```
{login: "<login>", token: "<token>"}
```

or an error

```
{error:"<error_code>", message:"<error_message>"}
```

where error_code is one of:

- InvalidToken
- InvalidJson
- BearerNotFound
- AuthorizationHeaderNotFound
- TokenExpired
- UnhandledError

Connection

This api is deprecated all the method are now able to autoconnect the user.

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/connect/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
{"password" : "password"}

 curl -XPOST -d '{"password":"<password>"}' -H "Content-Type: application/json" ␣
↪http://localhost:8090/xuc/api/1.0/connect/avencall.com/<username>/
```

DND

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/dnd/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
{"state" : [false|true]}


curl -XPOST -d '{"state":false}' -H "Content-Type: application/json"  http://
↪localhost:8090/xuc/api/1.0/dnd/avencall.com/<username>/
```

Dial

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/dial/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
{"number" : "1101"}

curl -XPOST -d '{"number":"<number>"}' -H "Content-Type: application/json"  http://
↪localhost:8090/xuc/api/1.0/dial/avencall.com/<username>/
```

DialByUsername

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/dialByUsername/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
{"username" : "john"}
```

curl -XPOST -d '{"username":"<username>"}' -H "Content-Type: application/json" http://localhost:8090/xuc/api/1.0/dialByUsername/avencall.com/<username>/

DialFromQueue

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/dialFromQueue/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
{"destination":"1002","queueId":5,"callerIdName":"Thomas","callerIdNumber":"999999
↪","variables":{"foo":"bar"}}

curl -XPOST -d '{"destination":"1002","queueId":5,"callerIdName":"Thomas",
↪"callerIdNumber":"999999","variables":{"foo":"bar"}}' -H "Content-Type:␣
↪application/json"  http://localhost:8090/xuc/api/1.0/dialFromQueue/avencall.com/
↪<username>/
```

Limitations: Queue No Answer settings does not work - see *No Answer*. Except: when there is no free Agent to queue (none attached, all Agents on pause or busy), then No answer settings work (but Fail does not).

Note: Line should be configured with enabled "Ring instead of On-Hold Music" enabled (on "Application: tab in queue configuration - see *Queues*). Otherwise the queue will answers the call and the destination rings even if there are no agents available.

Phone number sanitization

Dial command automatically applies filters to the phone number provided to make it valid for Xivo. Especially, it removes invalid characters and handles properly different notations of international country code.

Some countries don't follow the international standard and actually keep the leading zero after the country code (e.g. Italy). Because of this, if the zero isn't surrounded by parenthesis, the filter keeps it[1].

[1] See Redmine ticket #150

Forward

All forward commands use the above payload

```
{"state" : [true|false],
  "destination" : "1102")
```

Unconditionnal

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/uncForward/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
curl -XPOST -d '{"state":true,"destination":"<destnb>"}' -H "Content-Type:␣
↪application/json"  http://localhost:8090/xuc/api/1.0/uncForward/avencall.com/
↪<username>/
```

On No Answer

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/naForward/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
curl -XPOST -d '{"state":true,"destination":"<destnb>"}' -H "Content-Type:␣
↪application/json"  http://localhost:8090/xuc/api/1.0/naForward/avencall.com/
↪<username>/
```

On Busy

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/busyForward/\protect\T1\textdollardomain/\protect\T1\textdollarusername/

```
curl -XPOST -d '{"state":true,"destination":"<destnb>"}' -H "Content-Type:␣
↪application/json"  http://localhost:8090/xuc/api/1.0/busyForward/avencall.com/
↪<username>/
```

Personal Contacts

To handle personal contacts of a user

Error management

If an error occurs while using API actions, error will always be raised with proper HTTP return code and will be wrapped in JSON object with following format

```
{
  "error": <error_code>
  "message": <cause>
}
```

Error codes

Error code	HTTP header code	Possible cause
InvalidContact	400	Personal contact has bad format
ContactNotFound	404	Personal contact does not exist
DuplicateContact	409	Personal contact already exists
Unreachable	503	Directory server is not reachable
JsonParsingError	500	Personal Contact sent to API is not JSON valid
NotHandledError	500	Error not covered in current implementation

List

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/personal

```
curl -XGET -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/personal
```

Will retrieve an object containing all personal contacts *entries* with associated values defined in *headers* (HTTP code 200)

```
{
  "entries": [
      { "status": 0, "entry": [ "hawkeye", "pierce", "1002", "0761187406", "false
↪"]},
      { "status": -2, "entry": [ "peter", "pan", "1004", "", "false"]}],
  "headers":
      ["Firstname", "Lastname", "Number", "Mobile", "Favorite"]
 }
```

Get

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/personal/\protect\T1\textdollarcontactId

```
curl -XGET -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/personal/28079dc0-2c6b-4332-9075-61da9229389f
```

Will retrieve an object containing a single personal contact (HTTP code 200)

```
{
  "id":"28079dc0-2c6b-4332-9075-61da9229389f",
  "firstname":"doe",
  "lastname":"john",
  "number":"1111",
  "mobile":"2222",
  "fax":"3333",
  "email":"j.doe@my.corp",
  "company":"corp"
}
```

Create

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/personal

```
 {
   "firstname":"doe",
   "lastname":"john",
   "number":"1111",
   "mobile":"2222",
   "fax":"3333",
   "email":"j.doe@my.corp",
   "company":"corp"
 }

curl -XPOST -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/personal -d '{"lastname":"john","firstname":"doe","company":"corp","email
↪":"j.doe@my.corp","number":"1111","mobile":"2222","fax":"3333"}'
```

Will create and return object containing a single personal contact with id (HTTP code 201)

```
{
  "id":"28079dc0-2c6b-4332-9075-61da9229389f",
  "firstname":"doe",
  "lastname":"john",
  "number":"1111",
  "mobile":"2222",
  "fax":"3333",
  "email":"j.doe@my.corp",
  "company":"corp"
}
```

Update

PUT http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/personal/\protect\T1\textdollarcontactId

```
 {
   "firstname":"doe",
   "lastname":"john",
   "number":"1111",
   "mobile":"2222",
   "fax":"3333",
   "email":"j.doe@my.corp",
   "company":"corp"
 }

curl -XPUT -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/personal/28079dc0-2c6b-4332-9075-61da9229389f -d '{"lastname":"john",
↪"firstname":"doe","company":"corp","email":"j.doe@my.corp","number":"1111",
↪"mobile":"2222","fax":"3333"}'
```

Will update and return object containing single personal contact with id (HTTP code 200)

```
{
  "id":"28079dc0-2c6b-4332-9075-61da9229389f",
  "firstname":"doe",
  "lastname":"john",
  "number":"1111",
  "mobile":"2222",
  "fax":"3333",
  "email":"j.doe@my.corp",
  "company":"corp"
}
```

Delete

DELETE http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/personal/\protect\T1\textdollarcontactId

```
curl -XGET -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/personal/28079dc0-2c6b-4332-9075-61da9229389f
```

Will return empty content (HTTP code 204)

Delete All

DELETE http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/personal

```
curl -XGET -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/personal
```

Will return empty content (HTTP code 204)

Export

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/export/personal

```
curl -XGET -H "X-Auth-Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/export/personal
```

Will return csv text content with UTF-8 encoding (HTTP code 200) containing all personal contacts of a user

```
company,email,fax,firstname,lastname,mobile,number
corp,jdoe@company.corp,3333,John,Doe,2222,1111
```

Import

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/contact/import/personal

Note: request needs to have plain csv as content with utf-8 charset

```
company,email,fax,firstname,lastname,mobile,number
corp,jdoe@company.corp,3333,John,Doe,2222,1111

curl -XPOST -H "Content-Type: text/csv;charset=UTF-8" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪contact/import/personal -d 'company,email,fax,firstname,lastname,mobile,
↪number\ncorp,jdoe@company.corp,3333,John,Doe,2222,1111'
```

Will create and return object containing result of success/failure for each personal contact inserted (HTTP code 201)

```
{
  "created": [{
    "id": "e62ee672-f74a-498c-b138-86ac1b0ae429",
    "lastname": "Wayne"
  }, {
    "id": "d3a67f8e-3d8a-465a-9633-bde65a41f1bb",
    "lastname": "Hawking"
  }],
  "failed": [{
    "line": 3,
    "errors": ["too many fields"]
  }]
}
```

Handshake

Will repost all events on the configured URL

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/1.0/handshake/\protect\T1\textdollardomain/

Agent

AgentLogin

Log an agent on an extension

POST http://\protect\T1\textdollarxuchost:\protect\T1\textdollarxucport/xuc/api/1.0/agentLogin/

```
curl -XPOST -d '{"agentphonenumber":"<phone number>", "agentnumber":"<agent number>
↪"}' -H "Content-Type: application/json"  http://localhost:8090/xuc/api/1.0/
↪agentLogin/
```

AgentLogout

Logout un agent

POST http://\protect\T1\textdollarxuchost:\protect\T1\textdollarxucport/xuc/api/1.0/agentLogout/

```
curl -XPOST -d '{"phoneNumber":"<phoneNumber>"}' -H "Content-Type: application/json
↪"  http://localhost:8090/xuc/api/1.0/agentLogout/
```

TogglePause

Change state of an agent, pause if ready, ready if on pause or on wrapup

POST http://\protect\T1\textdollarxuchost:\protect\T1\textdollarxucport/xuc/api/1.0/togglePause/

```
curl -XPOST -d '{"phoneNumber":"<phoneNumber>"}' -H "Content-Type: application/json
↪"  http://localhost:8090/xuc/api/1.0/togglePause/
```

User Call History

Get user call history

```
curl -XGET -H "Content-Type: application/json"  http://localhost:8090/xuc/api/1.0/
↪historyByUsername/<domain>/<username>?size=X
```

Answer

```
[
  {"start":"2018-09-20 17:38:41","duration":"00:00:02","srcUsername":"bruce",
↪"dstUsername":"alicej","status":"emitted"},
  {"start":"2018-09-20 17:19:40","duration":"00:00:01","srcUsername":"bruce",
↪"dstUsername":"cquefia","status":"emitted"},
  {"start":"2018-09-20 17:15:00","duration":"00:00:18","srcUsername":"cquefia",
↪"dstUsername":"bruce","status":"missed"},
  {"start":"2018-09-20 17:14:16","duration":"00:00:11","srcUsername":"cquefia",
↪"dstUsername":"bruce","status":"missed"}
]
```

Qualifications

To retrieve call qualification options and create call qualification answer.

Error management

If an error occurs while using API actions, error will always be raised with proper HTTP return code and will be wrapped in JSON object with following format

```
{
  "error": <error_code>
  "message": <cause>
}
```

Error codes

Error code	HTTP header code	Possible cause
JsonBodyNotFound	400	Qualification answer sent to API is not found
JsonErrorDecoding	400	Qualification answer sent to API is not JSON valid
Unreachable	503	Config management server is not available
NotHandledError	500	Error not covered in current implementation

Get

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/call_qualification/queue/\protect\T1\textdollarqueueId

```
curl -XGET -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪call_qualification/queue/$queueId
```

Will retrieve a list of objects containing all qualifications and sub qualifications for a single queue (HTTP code 200)

```
[
  {
    "id": 6,
        "name": "General Questions",
        "subQualifications": [
          { "id": 14, "name": "Common"},
          { "id": 15, "name": "Technical" }
        ]
  },
  {
    "id": 5,
        "name": "Support",
        "subQualifications": [
          { "id": 12, "name": "Technical" },
          { "id": 13, "name": "General" }
        ]
  }
]
```

Create

POST http://localhost:\protect\T1\textdollarxucport/xuc/api/2.0/call_qualification

```
 {
   "sub_qualification_id": 1,
   "time": "2018-03-21 17:00:00",
   "callid": "callid1",
   "agent": 1,
   "queue": 1,
   "first_name": "john",
   "last_name": "doe",
   "comment": "some comment",
   "custom_data": "some custom data"
 }

curl -XPOST -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪call_qualification -d -d '{"sub_qualification_id":1,"time":"2018-03-21 17:00:00",
↪"callid":"callid1","agent":1,"queue":1,"first_name":"john","last_name":"doe",
↪"comment":"some comment","custom_data":"some custom data"}'
```

Will create a call qualification answer and return id of the created call qualification answer (HTTP code 201)

Export

GET http://localhost:\protect\T1\textdollarxucport/xuc/api/api/2.0/call_qualification/csv/\protect\T1\textdollarqueueId/\protect\T1\textdollarfrom/\protect\T1\textdollarto

```
curl -XGET -H "Content-Type: application/json" -H "X-Auth-Token:␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6InZtYWNoaW4iLCJleHBpcmVzIjoxNDg5OTk4Mzg2fQ.
↪R6bzc5fBRs74l7ZBdLVjSIbNx3j3e07mV6mAX4_sklI" http://localhost:8090/xuc/api/2.0/
↪call_qualification/csv/$queueId/$from/$to
```

Will return csv text content with UTF-8 encoding (HTTP code 200) containing all call qualification answer of a queue

```
sub_qualification_id,time,callid,agent,queue,first_name,last_name,comment,custom_
↪data
1,2018-03-21 17:00:00,callid1,1,1,john,doe,some comment,some custom data
```

9.1.3 Real Time Statistics

Exposed by xuc

Queue statistics

These real time statistics are calculated nearly in real time from the queue_log table Statistic are reset to 0 at midnight (24h00) can be changed by configuration

Real time calculated Queue statistic

name	Description
TotalNumberCallsEntered	Total number of calls entered in a queue
TotalNumberCallsAbandonned	Total number of calls abandoned in a queue (not answered)
TotalNumberCallsAbandonnedAfter15	Total number of calls abandoned after 15 seconds
TotalNumberCallsAnswered	Total number of calls answered
TotalNumberCallsAnsweredBefore15	Total number of calls answered before 15 seconds
PercentageAnsweredBefore15	Percentage of calls answered before 15 seconds over total number of calls entered
PercentageAbandonnedAfter15	Percentage of calls abandoned after 15 seconds over total number of calls entered
TotalNumberCallsClosed	Total number or calls received when queue is closed
TotalNumberCallsTimeout	Total number or calls diverted on queue timeout

All queue statistics counters (except percentage) are also available for the sliding last hour by adding LastHour to the name .i.e. TotalNumberCallsAbandonnedLastHour.

For percentage, there is no historical value (LastHour). If you need historical percentage, you should compute it using the historical total numbers.

Additional Thresholds

You can configure xuc to add statistics for thresholds other than 15 seconds. In this case the xuc server will automatically publish TotalNumberCallsAbandonnedAfterXX, TotalNumberCallsAnsweredBeforeXX, PercentageAnsweredBeforeXX, PercentageAbandonnedAfterXX. XX will be replace by all the defined thresholds values.

Configuration

You need to include in the compose.yml file a link to a specific configuration file by adding in xuc section a specific volume and an environment variable to specify the alternate config file location

```
xuc:

....

environment:
....
- CONFIG_FILE=/conf/xuc.conf

volumes:
- /etc/docker/xuc:/conf
```

Edit in /etc/docker/xuc/ a configuration file named xuc.conf to add new thresholds configuration (empty by default)

```
 include "application.conf"

 xucstats {
  queues {
    statTresholdsInSec = [10,30] # 15 sec treshold is automatically added
  }
}
```

Recreate and restart the container : *xivocc-dcomp up -d xuc*

Note: In this example, xuc will publish counters for 10, 15 and 30 seconds periods.

Other queue statistics

Other queue statistics are calculated by xivo cti server

- AvailableAgents
- TalkingAgents
- LongestWaitTime
- WaitingCalls
- EWT

Definition in xivo documentation xivo documentation

Calculated Agent statistics

name	Description
PausedTime	Total time agent in pause
WrapupTime	Total time agent in wraup
ReadyTime	Total time agent ready
InbCalls	Total number of inbound calls received internal and external
InbAcdCalls	Total number of inbound ACD calls received internal and external
InbCallTime	Total time for inbound calls received internal and external
InbAcdCallTime	Total time for inbound ACD calls received internal and external
InbAcdCallTime	Total time for inbound ACD calls received internal and external
InbAnsCalls	Answered inbound calls received internal and external
InbAnsAcdCalls	Answered inbound ACD calls received internal and external
InbUnansCalls	Unanswered inbound calls received internal and external
InbUnansAcdCalls	Unanswered inbound ACD calls received internal and external
InbPercUnansCalls	Percentage of unanswered inbound calls received internal and external
InbPercUnansAcdCalls	Percentage of unanswered inbound ACD calls received internal and external
InbAverCallTime	Average time for inbound calls received internal and external
InbAverAcdCallTime	Average time for inbound ACD calls received internal and external
OutCalls	Total number of outbound calls received internal and external
OutCallTime	Total time for outbound calls received internal and external
LoginDateTime	Last login date time
LogoutDateTime	Last logout date time

Terms:

inbound ACD calls all calls received by an agent via ACD.

inbound calls all calls received by an agent, internal, external or ACD calls.

oubound calls all calls dialed by an agent, internal or external calls.

Agent statistics are calculated internaly on a daily basis and reset to 0 at midnight (default configuration). see javascript api

If some status are configured in xivo cti server with activate pause to all queue = true, additionnal statistics computing the total time in not ready with this status are calculated. This statistics name is equal to the presence name configuration in XiVO.

9.1.4 Technical structure of XiVO-CC

Reporting

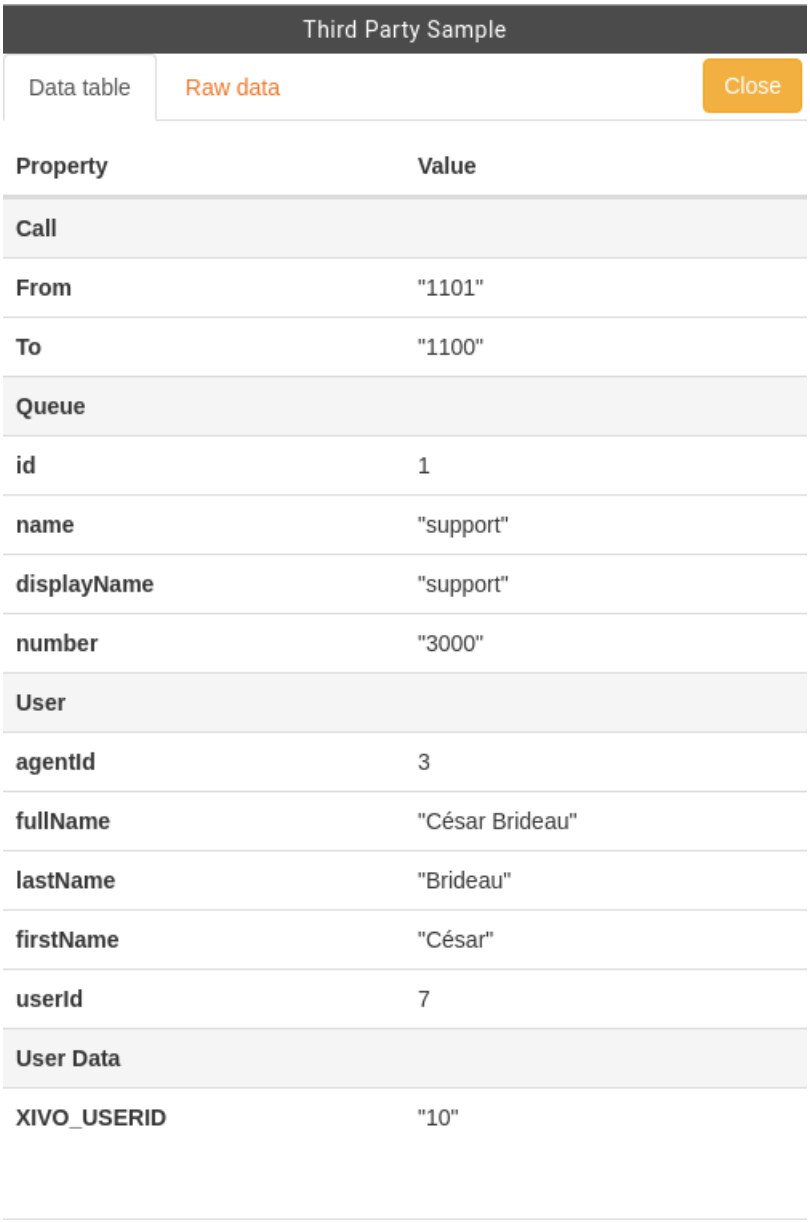
The reporting is composed of four packages: pack-reporting, xivo-full-stats, xivo-reporting-db and xivo-db replication.

These packages will feed the tables of the xivo_stats database:

- xivo-db-replication feeds the tables cel and queue_log in real time, and the configuration tables (dialaction, linefeatures, etc. . .) every 5 minutes
- xivo-full-stats feeds in real time tha tables call_on_queue, call_data, stat_queue_periodic, stat_agent_periodic and agent_position
- xivo-reporting-db and pack-reporting work together to feed the tables stat_queue_specific, stat_agent_queue_specific and stat_agent_specific every 15 minutes

9.2 Third Party Integration

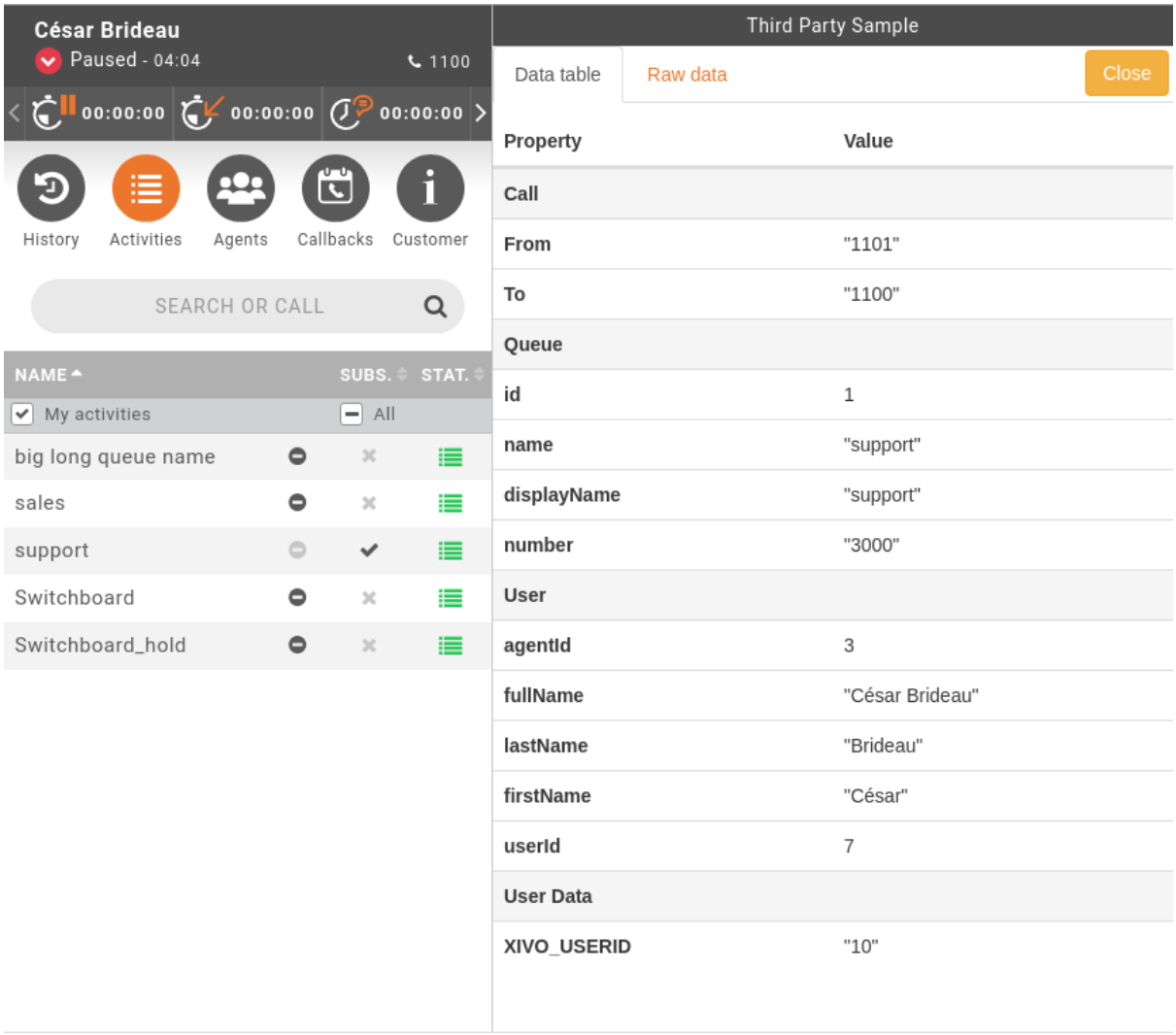
Third party web application integration is possible inside the XucMgt Agent application. Upon each call, you can display a custom panel next to the agent interface:



9.2.1 Workflow

When a call is ringing on the agent phone, the Application will call the external web service (see *Configuration* below). The web service response will dictate the behaviour of the integration. For example, if the speficied action is to open the application when the call is hung up, a new panel will be created and opened inside the agent interface, showing the content specified by the web service response. (see *Web Service API* for available options).

When the work is complete in the integrated application, the application must post a message to terminate the third party application pane inside the agent application (see *Completion*).


César Brideau
Paused - 04:04
1100
00:00:00
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
NAME
SUBS.
STAT.
My activities
All
big long queue name
sales
support
Switchboard
Switchboard_hold
Third Party Sample
Data table
Raw data
Close
Property
Value
Call
From
"1101"
To
"1100"
Queue
id
1
name
"support"
displayName
"support"
number
"3000"
User
agentId
3
fullName
"César Brideau"
lastName
"Brideau"
firstName
"César"
userId
7
User Data
XIVO_USERID
"10"

9.2.2 Configuration

You need to specify the third party application web service url to integrate this application inside the XucMgt Agent interface. This can be done by giving a value to the `THIRD_PARTY_URL` environment variable in the `/etc/docker/compose/custom.env` file

```
...
THIRD_PARTY_URL=http://some.url.com/ws/endpoint
```

The speficied URL must be accessible from the client browser (i.e. the end user of the Agent application). The call wil be made from his browser.

9.2.3 Web Service API

The Web Service url specified in the :*Configuration* must conforms to the following behaviour.

The service will receive a POST request with a payload as `application/json`, for example:

```
{
    "user":{
        "userId":4,
        "agentId":1,
        "firstName":"James",
        "lastName":"Bond",
        "fullName":"James Bond"
    },
    "callee":"1000",
    "caller":"1001",
    "queue":{
        "id":2,
        "name":"trucks",
        "displayName":"Trucks",
        "number":"3001"
    },
    "userData":{
        "XIVO_CONTEXT":"default",
        "XIVO_USERID":"2",
        "XIVO_SRCNUM":"1001",
        "XIVO_DSTNUM":"3001"
    }
}
```

- `user` contains the connected user information
- `callee` contains the number called
- `queue` queue properties
- `userData` call data presented by Xivo

The Web service must answer with an `application/json` content. For example:

```
{
    "action":"open",
    "event":"EventReleased",
    "url":"/thirdparty/open/6bd37819-b4a6-43d3-8fa3-6eb6489bb705",
    "autopause":true,
    "autopauseReason:"backoffice",
    "title":"Third Party Sample"
}
```

or:

```
{
    "action":"none"
}
```

- `action` is one of
  - `open`: Will open the given `url` inside the integration pane
  - `popup`: Will open the given `url` in a popup
  - `none`: No action will be performed
- `event` is one of `EventRinging`, `EventEstablished`, `EventReleased`. The third party application will be opened when one the specified event occurs
- `url` should be the url to open inside the application. This url should point to a valid web application that can be specific for each call.
- `autopause` if set to true, the agent will be put on pause when the application pane is opened and back to ready when the application is completed.
- `autopauseReason` an optional field allowing to set the type of the pause used while setting the agent on pause (see the line above).
- `multitab` if set to true and `action` is set to `popup`, then the integration will be opened a in new popup window (or tab) each time instead of reusing the same window (or tab).
- `title` will set the title of the tabs that will be opened.

Warning, when the XucMgt application and the integrated application are on different server, domain, url,... (which should be common case), You may get CORS errors. To workaround this issue, you should implement the OPTIONS request on your web service. This method will be called by the browser before issuing the POST request to ensure the target web server allows calls from the original application. You application must set at least the following headers in order to overcome the CORS errors:

- `Access-Control-Allow-Origin`: * or the domain hosting the XucMgt application
- `Access-Control-Allow-Methods`: POST, OPTIONS (at least)
- `Access-Control-Allow-Headers`: Origin, X-Requested-With, Content-Type, Accept (at least)

9.2.4 Completion

Once the work is complete inside the third party application, it should post a completion message (`closeThirdParty`) to the application using the Web Messaging API.

For example, here is how to define a close method in javascript to send the message to the hosting application and bind it to a simple button:

```
(function () {
      function close() {
          parent.window.postMessage("closeThirdParty", "*");
      }

      document.getElementById("close").addEventListener("click", close, false);
})();
```

9.3 Recording server REST Api

This section describes the Recording server API.

In the following, all url are relative to the recording server base url and port. For example a relative URL `/records/search` is meant to be replaced by `http://192.168.29.101:9400/records/search` assuming the recording server is available on port `9400` at `192.168.29.101`

9.3.1 Authentication

To use the recording api you need to add an additional header in the HTTP Request. The header name is `X-Auth-Token` and its value must be the same as the `authentication.token` value in the `application.conf` of the recording server.

Example:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json"
         -H 'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k'  http://192.168.29.
↪101:9400/records/search?pageSize=3  -d '{"agent": "1573"}'
```

9.3.2 Records

Search

This api allows to search for recorded calls in the database using criteria.

Description:

URL `/records/search`

Method `POST`

Url parameters

page The page number to return, counting from 1

pageSize The number of elements per page

Request body Json object with field & value pair.

Allowed field names:

agent The agent number to filter on

queue The queue number to filter on

start Return only calls starting or ending after the given value

end Return only calls starting before the given value

callee Return only calls with the given destination number

caller Return only calls with the give source number

direction Filter calls based on the call direction, one of `incoming`, `outgoing` or `all`

key filter based on the given key name in the attached data along with the value of the `value` field

value value of the key defined in the `key` field

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json"
          -H 'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k'  http://192.168.29.
↪101:9400/records/search?pageSize=3  -d '{"agent": "1573"}'
```

Response:


```
{
    "hasNext": true,
    "records": [
        {
            "agent": "Joe Dalton (1573)",
            "attached_data": {
                "recording": "xivocc_gateway-1459433866.13971"
            },
            "dst_num": "73555",
            "duration": "00:00:20",
            "id": "xivocc_gateway-1459433866.13971",
            "queue": "oneforone (3555)",
            "src_num": "loadtester",
            "start": "2016-03-31 16:17:46",
            "status": "answered"
        },
        {
            "agent": "Joe Dalton (1573)",
            "attached_data": {
                "recording": "xivocc_gateway-1459433330.13665"
            },
            "dst_num": "73555",
            "duration": "00:01:01",
            "id": "xivocc_gateway-1459433330.13665",
            "queue": "oneforone (3555)",
            "src_num": "loadtester",
            "start": "2016-03-31 16:08:51",
            "status": "answered"
        }
    ]
}
```


Search by call id

This api allows to search for a recorded call based on call id.

Description:

URL `/records/callid_search`

Method `POST`

Url parameters

callid The call id to retrieve

Example Query:

```
curl -XPOST -H "Accept: application/json"
          -H 'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k'  http://192.168.29.
↪101:9400/records/callid_search?callid=1459435466.286075
```

Response:


```
{
    "hasNext": false,
    "records": [
        {
            "agent": "Dino Falconetti (1564)",
```


(continues on next page)

(continued from previous page)

```
            "attached_data": {
                "recording": "xivocc_gateway-1459435465.15089"
            },
            "dst_num": "73556",
            "duration": "",
            "id": "xivocc_gateway-1459435465.15089",
            "queue": "hotline (3556)",
            "src_num": "loadtester",
            "start": "2016-03-31 16:44:26",
            "status": "answered"
        }
    ]
}
```

Retrieve audio file

This api allows to get metadata or retrieve audio file of a given call. Each action performmed will be logged in access log file. See *Access logs*.

Description:

URL `/records/records/<file-id>/audio/<action>`

Method `GET`

Url parameters

file-ud The file to retrieve

action Action done on the audio file

Allowed actions:

result To retrieve metadata from a search

listen To notify that we listened to the file

download To get the file locally

Example:

```
curl -XGET -H "Accept: application/json" -H 'X-Auth-Token: u@pf#41[gYHJm
↪<]9N[a0iWDQQ7`e9k'
          http://192.168.29.101:9400/records/xivocc_gateway-1459435465.15089/audio/
↪download
```

Attach call data

This api allows to attach data to a given call

Description:

URL `/call_data/<call-data-id>/attached_data`

Method `POST`

Url parameters

call-data-id The id of the call-data, not to be confused by the call id or unique id.

Request Body An array of key value

Example:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k'
          http://192.168.29.101:9400/call_data/761054/attached_data  -d '[{"key":
↪"color", "value": "green"}]'
```

9.3.3 History

Search

This api gives the call history of a given interface.

Description:

URL `/history`

Method `POST`

Url parameters

size The maximum number of result to return

Request Body A json object with a field named `interface` containing the interface to search for.

Example Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' http://localhost:9400/history  -d
↪'{"interface":"SIP/az9kf7"}'
```

Response:

```
[
 {
  "start": "2017-01-27 15:37:54",
  "duration": "00:00:18",
  "src_num": "1001",
  "dst_num": "3000",
  "status": "emitted",
  "src_firstname": "Poste",
  "src_lastname": "Poste 1001",
  "dst_firstname": null,
  "dst_lastname": "Cars"
 }
]
```

Search by customer

This api helps to find call history of a customer thanks to a list of predefined filters.

Description:

URL `/history/customer`

Method `POST`

Request Body A json object with filters (optional) named `filters` containing the customer to search for and `size` to limit the results returned.

Response `total` the number of call received by this customer, `list` the call details reduced to the `size` set in query.

A filter is composed of a `field` as key (basically column name), an `operator` (=, <, >) and a `value`.

Allowed filter field:

src_num The customer phone number

key Call Attached data key

value Call Attached data value

Example Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k'
     http://localhost:9400/history/customer -d '{"filters": [{"field":"src_num",
↪"operator": "=", "value": "1456"}], "size":2}'
```

Response:

```
{
  "total":11,
  "list":[
    {"start":"2017-06-13 17:32:45","duration":"00:00:08",
    "wait_time":"00:00:06","agent_name":"Brucé Waill",
    "agent_num":"2500","queue_name":"Bl Record",
    "queue_num":"3012","status":"answered"},
    {"start":"2017-06-13 17:26:54","duration":"00:00:06",
    "wait_time":"00:00:05","agent_name":"Brucé Waill",
    "agent_num":"2500","queue_name":"Blue Ocean",
    "queue_num":"3000","status":"answered"
    }
  ]
}
```

Last agent for number

This api retrieves the last agent id who answered a given caller number.

Description:

URL `/last_agent`

Method `GET`

Url parameters

callerNo The calling number

since The number of days to search in the history

Example Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" -H 'X-
↪Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9400/last_agent?
↪callerNo=1002&since=100'
```

Response:

```
{"agentNumber":"2000"}
```

9.4 XiVO REST Api

The XiVO REST APIs are the privileged way to programmatically interact with XiVO.

9.4.1 Reference

xivo-agentd REST API

You can view the API documentation at http://<youxivo>.api.

Changelog

15.19

- Token authentication is now required for all routes, i.e. it is not possible to interact with xivo-agentd without a xivo-auth authentication token.

15.18

- xivo-agentd now uses HTTPS instead of HTTP.

15.15

- The resources returning agent statuses, i.e.:
  - GET /agents
  - GET /agents/by-id/{agent_id}
  - GET /agents/by-number/{agent_number}

  are now returning an additional argument named "state_interface", which is "the interface (e.g. SIP/alice) that is used to determine if an agent is in use or not".

xivo-confd REST API

Note: REST API 1.1 for confd is currently evolving. New features and small fixes are regularly being added over time. We invite the reader to periodically check the *changelog* for an update on new features and changes.

xivo-confd REST API changelog

16.06

- A new API for initializing a XiVO (passing the wizard):
  - GET `/1.1/wizard`
  - POST `/1.1/wizard`

  - GET `/1.1/wizard/discover`
- A new API for associating a user with an entity has been added:
  - GET `/1.1/users/<user_id>/entities`
  - PUT `/1.1/users/<user_id>/entities/<entity_id>`

16.05

- A new API for associating a user with a call permission has been added:
  - GET `/1.1/users/<user_id>/callpermissions`
  - PUT `/1.1/users/<user_id>/callpermissions/<call_permission_id>`
  - DELETE `/1.1/users/<user_id>/callpermissions/<call_permission_id>`
  - GET `/1.1/callpermissions/<call_permission_id>/users`
- Two new parameters have been added to the users resource:
  - `call_permission_password`
  - `enabled`
- A new API for user's forwards has been added:
  - PUT `/1.1/users/<user_id>/forwards`
- SIP endpoint: `allow` and `disallow` options are not split into multiple options anymore.
- SCCP endpoint: `allow` and `disallow` options are not split into multiple options anymore.

16.04

- The `summary` view has been added to `/users` (GET `/users?view=summary`)
- A new API for user's services has been added:
  - GET `/1.1/users/<user_id>/services`
  - GET `/1.1/users/<user_id>/services/<service_name>`
  - PUT `/1.1/users/<user_id>/services/<service_name>`
- A new API for user's forwards has been added:
  - GET `/1.1/users/<user_id>/forwards`
  - GET `/1.1/users/<user_id>/forwards/<forward_name>`
  - PUT `/1.1/users/<user_id>/forwards/<forward_name>`
- GET `/1.1/users/export` now requires the following header for CSV output:

```
Accept: text/csv; charset=utf-8
```

- Added call permissions endpoints:
  - GET `/1.1/callpermissions`
  - POST `/1.1/callpermissions`
  - GET `/1.1/callpermissions/<callpermission_id>`
  - PUT `/1.1/callpermissions/<callpermission_id>`
  - DELETE `/1.1/callpermissions/<callpermission_id>`

16.03

- Added switchboard endpoints:
  - GET `/1.1/switchboards`
  - GET `/1.1/switchboards/<switchboard_id>/stats`
- A new API for associating a line with a device has been added:
  - PUT `/1.1/lines/<line_id>/devices/<device_id>`
  - DELETE `/1.1/lines/<line_id>/devices/<device_id>`
- The following URLs have been deleted. Please use the new API instead:
  - GET `/1.1/devices/<device_id>/associate_line/<line_id>`
  - GET `/1.1/devices/<device_id>/dissociate_line/<line_id>`

16.02

- Added users endpoints in REST API:
  - GET `/1.1/users/<user_uuid>/lines/main/associated/endpoints/sip`

16.01

- The SIP API has been improved. `options` now accepts any extra parameter. However, due to certain database limitations, parameters that appear in *Supported parameters on SIP endpoints* may only appear once in the list. This limitation will be removed in future versions.
- A new API for custom endpoints has been added: `/1.1/endpoints/custom`
- A new API for associating custom endpoints has been added: `/1.1/lines/<line_id>/endpoints/custom/<endpoint_id>`

15.20

- A new API for mass updating users has been added: PUT `/1.1/users/import`
- A new API for exporting users has been added: GET `/1.1/users/export`

15.19

- A new API for mass importing users has been added: POST `/1.1/users/import`
- The following fields have been added to the `/users` API:
  - supervision_enabled
  - call_tranfer_enabled
  - ring_seconds
  - simultaneous_calls

15.18

- Ports 50050 and 50051 have been removed. Please use 9486 and 9487 instead
- Added sccp endpoints in REST API:
  - GET `/1.1/endpoints/sccp`
  - POST `/1.1/endpoints/sccp`
  - DELETE `/1.1/endpoints/sccp/<sccp_id>`
  - GET `/1.1/endpoints/sccp/<sccp_id>`
  - PUT `/1.1/endpoints/sccp/<sccp_id>`
  - GET `/1.1/endpoints/sccp/<sccp_id>/lines`
  - GET `/1.1/lines/<line_id>/endpoints/sccp`
  - DELETE `/1.1/lines/<line_id>/endpoints/sccp/<sccp_id>`
  - PUT `/1.1/lines/<line_id>/endpoints/sccp/<sccp_id>`
- Added lines endpoints in REST API:
  - GET `/1.1/lines/<line_id>/users`

15.17

- A new API for SIP endpoints has been added. Consult the documentation on http://xivo/api/ for further details.
- The `/lines_sip` API has been deprecated. Please use `/lines` and `/endpoints/sip` instead.
- Due to certain limitations in the database, only a limited number of optional parameters can be configured. This limitation will be removed in future releases. Supported parameters are listed further down.
- Certain fields in the `/lines` API have been modified. List of fields are further down

Fields modified in the `/lines` API

Name	Replaced by	Editable ?	Required ?
id		no	
device_id		no	
name		no	
protocol		no	
device_slot	position	no	
provisioning_extension	provisioning_code	no	
context		yes	yes
provisioning_code		yes	
position		yes	
caller_id_name		yes	
caller_id_num		yes	

Supported parameters on SIP endpoints

- md5secret
- language
- accountcode

- amaflags
- allowtransfer
- fromuser
- fromdomain
- subscribemwi
- buggymwi
- call-limit
- callerid
- fullname
- cid-number
- maxcallbitrate
- insecure
- nat
- promiscredir
- usereqphone
- videosupport
- trustrpid
- sendrpid
- allowsubscribe
- allowoverlap
- dtmfmode
- rfc2833compensate
- qualify
- g726nonstandard
- disallow
- allow
- autoframing
- mohinterpret
- useclientcode
- progressinband
- t38pt-udptl
- t38pt-usertpsource
- rtptimeout
- rtpholdtimeout
- rtpkeepalive
- deny
- permit
- defaultip
- setvar

- port
- regexten
- subscribecontext
- fullcontact
- vmexten
- callingpres
- ipaddr
- regseconds
- regserver
- lastms
- parkinglot
- protocol
- outboundproxy
- transport
- remotesecret
- directmedia
- callcounter
- busylevel
- ignoresdpversion
- session-timers
- session-expires
- session-minse
- session-refresher
- callbackextension
- timert1
- timerb
- qualifyfreq
- contactpermit
- contactdeny
- unsolicited_mailbox
- use-q850-reason
- encryption
- snom-aoc-enabled
- maxforwards
- disallowed-methods
- textsupport

15.16

- The parameter `skip` is now deprecated. Use `offset` instead for:
  - `GET /1.1/devices`
  - `GET /1.1/extensions`
  - `GET /1.1/voicemails`
  - `GET /1.1/users`
- The users resource can be referred to by `uuid`
  - `GET /1.1/users/<uuid>`
  - `PUT /1.1/users/<uuid>`
  - `DELETE /1.1/users/<uuid>`

15.15

- The field `enabled` has been added to the voicemail model
- A line is no longer required when associating a voicemail with a user
- Voicemails can now be edited even when they are associated to a user

15.14

- All optional fields on a user are now always null (sometimes they were empty strings)
- The caller id is no longer automatically updated when the firstname or lastname is modified. You must update the caller id yourself if you modify the user's name.
- Caller id will be generated if and only if it does not exist when creating a user.

14.16

- Association user-voicemail, when associating a voicemail whose id does not exist:
  - before: error 404
  - after: error 400

14.14

- Association line-extension, a same extension can not be associated to multiple lines

14.13

- Resource line, field `provisioning_extension`: type changed from `int` to `string`

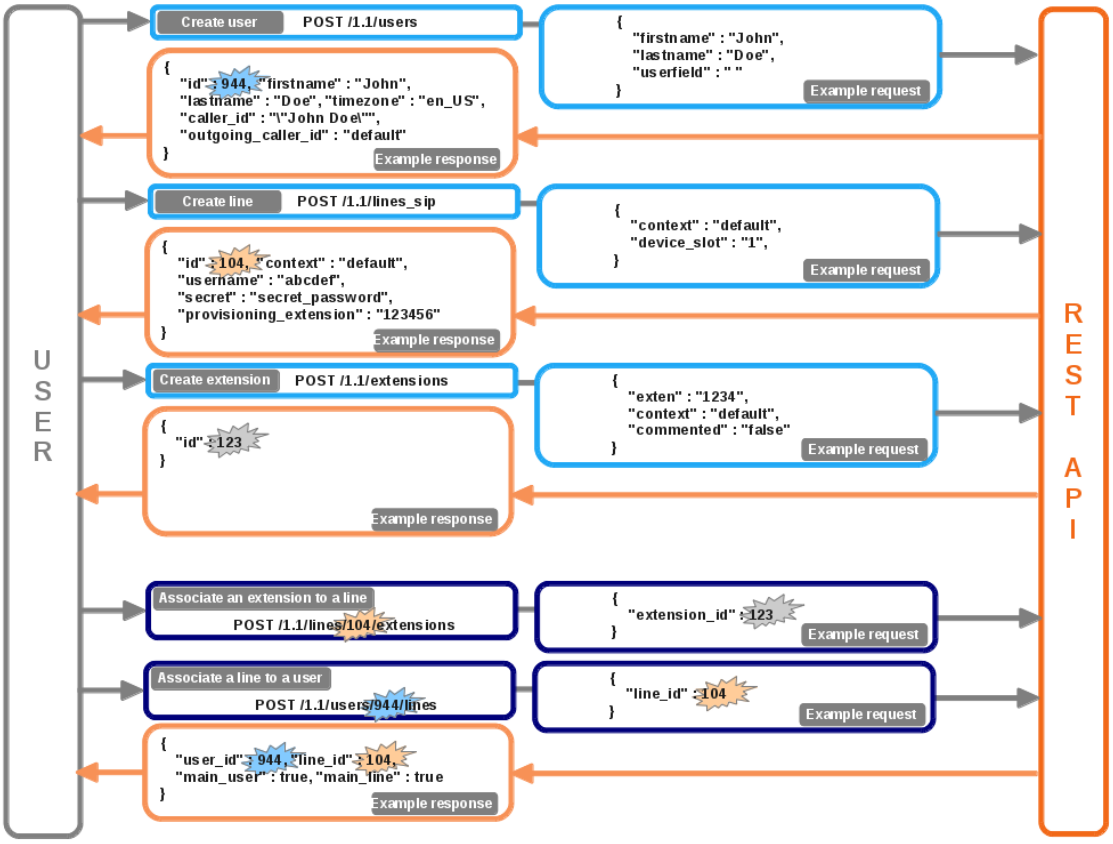


Fig. 1: Download source. (source)

REST API 1.1 examples

Create User for a line and a exten

Add user, line and exten with association

Add voicemail with association

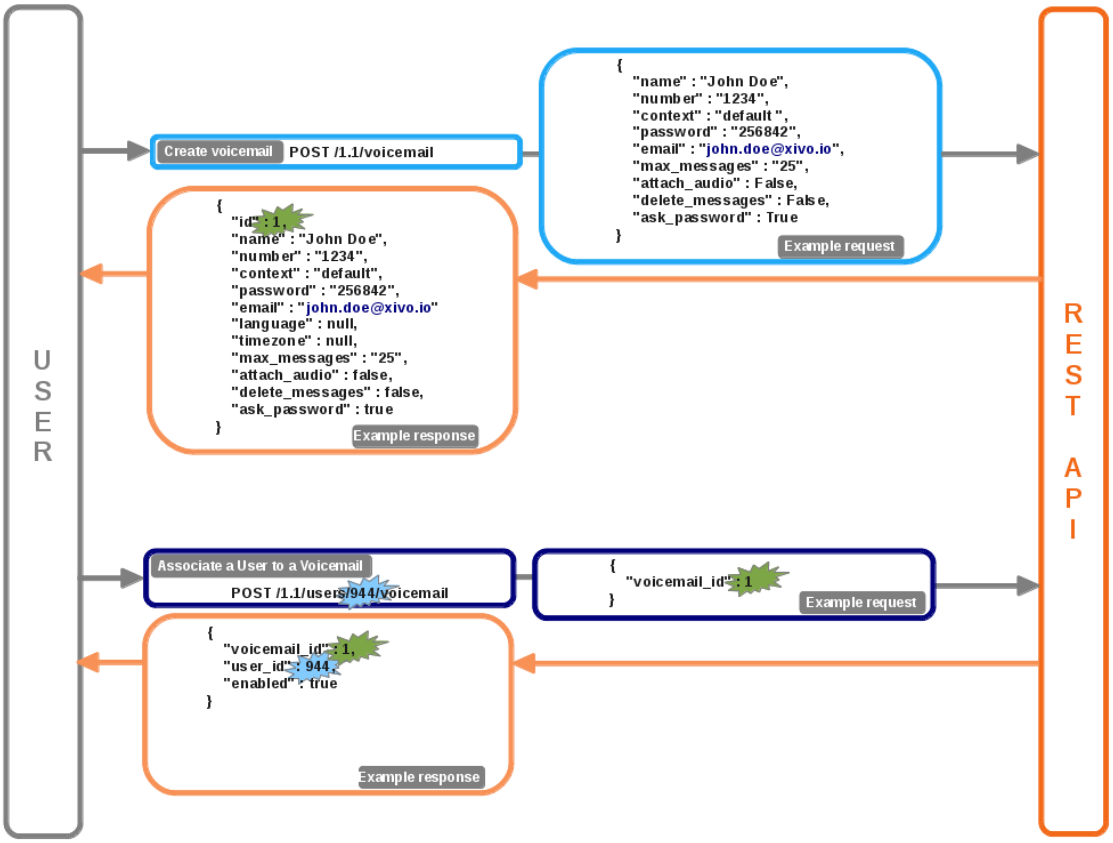


Fig. 2: Download source. (source)

Choice and add CTI profile with association

Multiple users for a line association

API reference

This section contains extended documentation for certain aspects of the API.

Function Keys

Function keys can be used as shortcuts for dialing a number, or accomplishing other menial tasks, by pushing a button on the phone. A function key's action is determined by its destination.

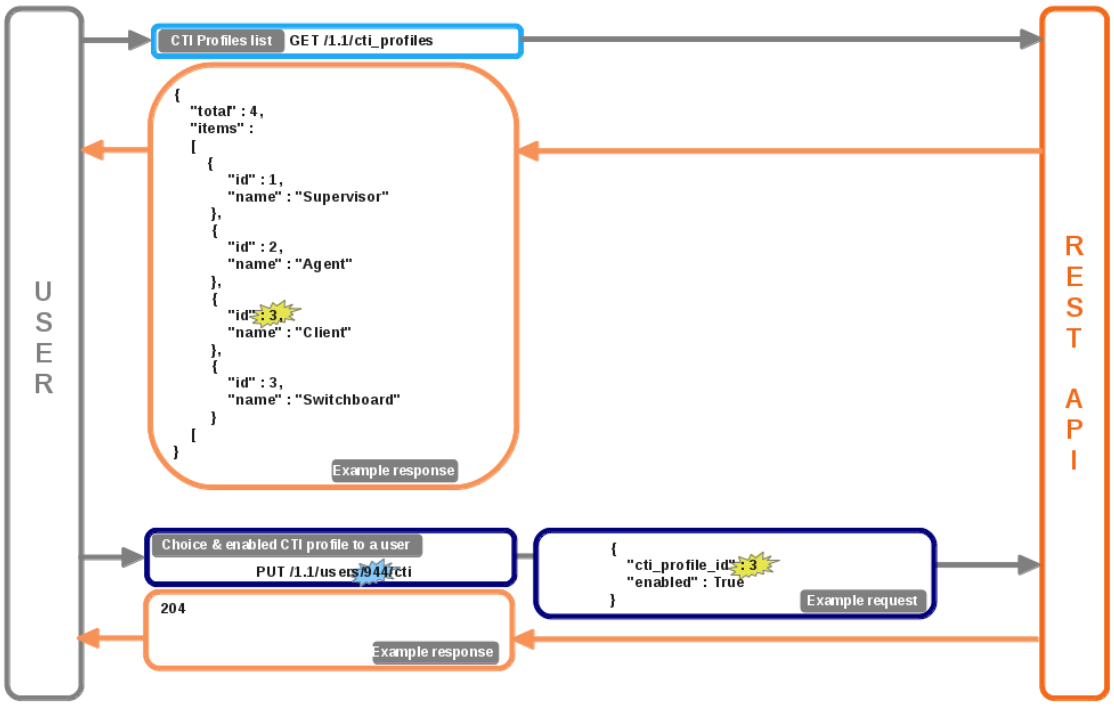


Fig. 3: Download source. (source)

Function keys can be added directly on a user, or in a template. Templates are useful for creating a set of common function keys that can be used by the same group of people.

This page only describes the data models used by the REST API. Consult the API documentation for further details on URLs.

Function Key Template

Parameters

Field	Type	Required	Description
name	string	No	A name for the template.
keys	*Function Key*	No	A collection of function keys under the form `{"position": "funckey"}`. See the example for more details.

Example

```
{
    "name": "Example template",
    "keys": {
        "1": {
            "destination": {
                "type": "user",
                "user_id": 34
            }
        },
```

(continues on next page)

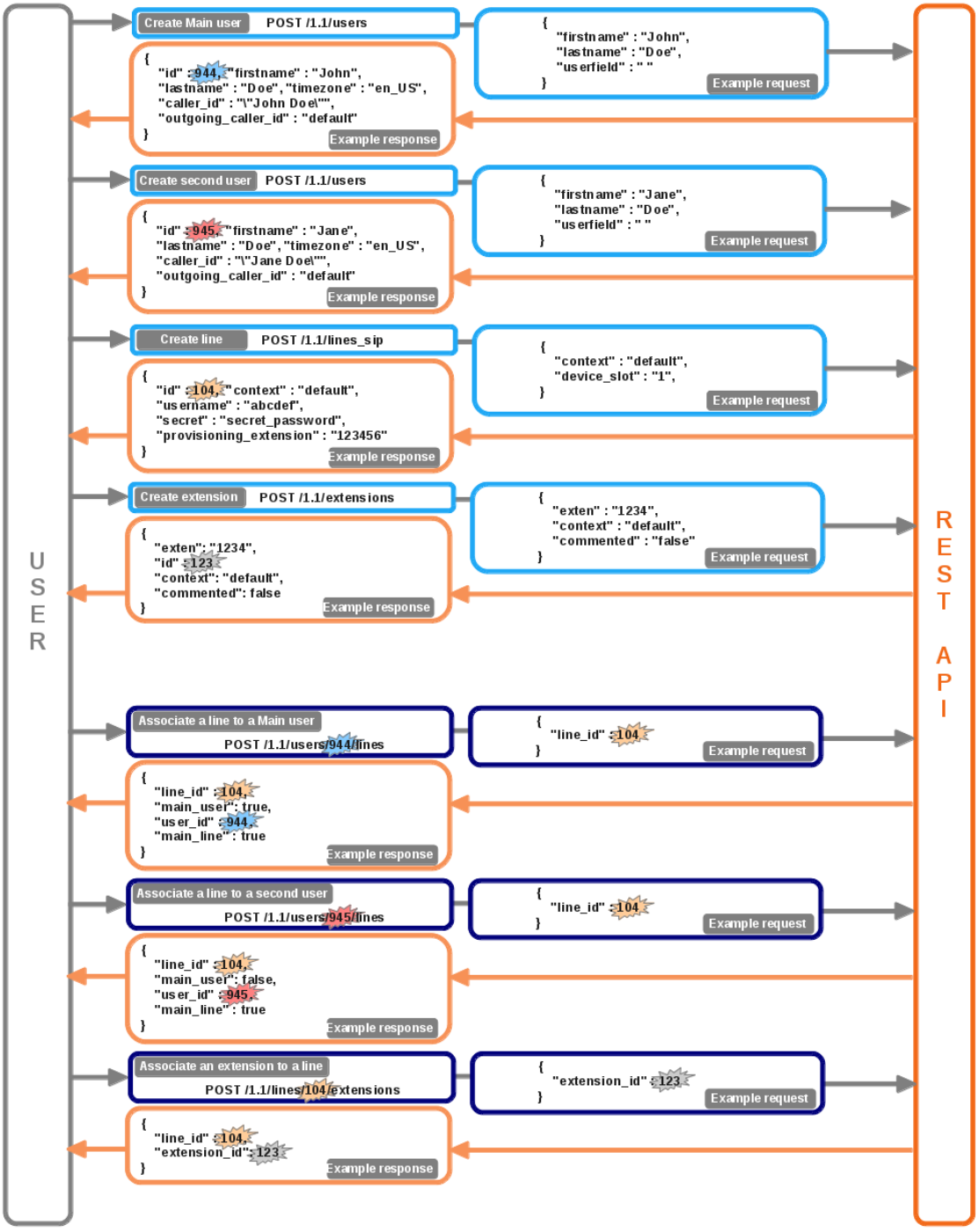


Fig. 4: Download source. (source)

(continued from previous page)

```
        "2": {
            "blf": true,
            "label": "Call mom",
            "destination": {
                "type": "custom",
                "exten": "5551234567"
            }
        }
    }
}
```

Function Key

Description

Field	Type	Required	Description
blf	boolean	No	Turn on BLF when there is activity on the destination
label	string	No	Label to display next to the function key
destination	*Destination*	Yes	Destination to call

Example

```
{
    "blf": True,
    "label": "Call john",
    "destination": {
        "type": "user",
        "user_id": 34
    }
}
```

Destination

A destination determines the number to dial when using a function key. Destinations are composed of a parameter named `type` and any additional parameters required by its type.

Available destination types:

agent An agent

bsfilter Boss/Secretary filter

conference Conference room

custom A custom number to dial

forward Forward a call towards another number

group A group

onlinerec Record a conversation during a call

paging A paging

park Park a call

park_position Pick up a parked call

queue Call queue

service A call service

transfer Transfer a call

user A User

Here are the parameters required for each destination:

Agent

Field	Type	Value
agent_id	numeric	Agents's id

BSFilter

Field	Type	Value
filter_member_id	numeric	ID of the filter member

Conference

Field	Type	Value
conference_id	numeric	Conference's id

Custom

Field	Type	Value
exten	string	Number to dial

Forward

Field	Type	Value
forward	string	Type of forward. Possible values: busy, noanswer, unconditional
exten	string	Number to dial (optional)

Group

Field	Type	Value
group_id	numeric	Group's id

Online call recording

No parameters are required for this destination

Paging

Field	Type	Value
paging_id	numeric	Pagings's id

Parking

No parameters are required for this destination

Parking Position

Field	Type	Value
position	numeric string	Position of the parking to pick up

Queue

Field	Type	Value
queue_id	numeric	User's id

Service

Field	Type	Value
service	string	Name of the service

Currently supported services:

phonestatus Phone Status

recsnd Sound Recording

callrecord Call recording

incallfilter Incoming call filtering

enablednd Enable "Do not disturb" mode

pickup Group Interception

calllistening Listen to online calls

directoryaccess Directory access

fwdundoall Disable all forwaring

enablevm Enable Voicemail

vmusermsg Consult the Voicemail

vmuserpurge Delete messages from voicemail

Transfer

Field	Type	Value
transfer	string	Type of transfer. Possible values: blind, attended

User

Field	Type	Value
user_id	numeric	User's id

CSV User Import

Users and common related resources can be imported onto a XiVO server by sending a CSV file with a predefined *set of fields*.

This page only documents additional notes useful for API users.

Uploading files

Files may be uploaded as usual through the web interface, or from a console by using HTTP utilities and the REST API. When uploading through the API, the header *Content-Type: text/csv charset=utf-8* must be set and the CSV data must be sent in the body of the request. A file may be uploaded using *curl* as follows:

```
curl -k -H "Content-Type: text/csv; charset=utf-8" -u username:password --data-
↪binary "@file.csv" https://xivo:9486/1.1/users/import
```

The response can be reindented in a more readable format by piping the output through *python -m json.tool* in the following way:

```
curl (...) | python -m json.tool
```

Migration from 1.0

The API version 1.0 is no longer supported and has been removed. In most cases, code that used the old API can be migrated to version 1.1 without much hassle by updating the URL. For example, in 1.0, the URL to list users was:

```
   /1.0/users/

In 1.1, it is::

   /1.1/users
```

Please note that there are no trailing slashes in URLs for version 1.1.

For further details consult the documentation at http://<youxivo>.api

xivo-provd REST API

This section describes the REST API provided by the xivo-provd application.

If you want to interact with the REST API of the xivo-provd daemon that is executing as part of XiVO, you should be careful on which operation you are doing as to not cause stability problem to other parts of the XiVO ecosystem. Mostly, this means being careful when editing or deleting devices and configs.

By default, the REST API of xivo-provd is accessible only from localhost on port 8666. No authentication is required.

Warning: Major changes could happen to this API.

API

The description of the API has been split into these sections:

Provd Management

Get the Provd Manager

The provd manager resource represents the main entry point to the xivo-provd REST API.

It links to the following resources:

- The `dev` relation links to a *device manager*.
- The `cfg` relation links to a *config manager*.
- The `pg` relation links to a *plugin manager*.
- The `srv.configure` relation links to the provd manager *configuration service*.

Query

```
GET /provd HTTP/1.1
```

Example request

```
GET /provd HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "links": [
        {
            "href": "/provd/dev_mgr",
            "rel": "dev"
        },
        {
            "href": "/provd/cfg_mgr",
            "rel": "cfg"
        },
        {
            "href": "/provd/pg_mgr",
            "rel": "pg"
        },
        {
            "href": "/provd/configure",
            "rel": "srv.configure"
        }
    ]
}
```

Devices Management

Get the Device Manager

The device manager links to the following resources:

- The `dev.synchronize` relation links to the *device synchronization service*.
- The `dev.reconfigure` relation links to the *device reconfiguration service*.
- The `dev.dhcpinfo` relation links to the *device DHCP information service*.
- The `dev.devices` relation links to the *list of devices*.

Query

```
GET /provd/dev_mgr HTTP/1.1
```

Example request

```
GET /provd/dev_mgr HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "links": [
        {
            "href": "/provd/dev_mgr/synchronize",
            "rel": "dev.synchronize"
        },
        {
            "href": "/provd/dev_mgr/reconfigure",
            "rel": "dev.reconfigure"
        },
        {
            "href": "/provd/dev_mgr/dhcpinfo",
            "rel": "dev.dhcpinfo"
        },
        {
            "href": "/provd/dev_mgr/devices",
            "rel": "dev.devices"
        }
    ]
}
```

List Devices

Query

```
GET /provd/dev_mgr/devices  HTTP/1.1
```

Query Parameters

Field	Description
q	A selector, encoded in JSON, describing which device should be returned. All devices are returned if not specified. Example: `q={"ip":"10.34.1.119"}`
fields	A list of fields, separated by comma. Example: `fields=mac,ip`
skip	An integer specifing the number of devices to skip. Example: `skip=10`
sort	The key on which to sort the results. Example: `sort=id`
sort_or	The order of sort; either `ASC` or `DESC`.

Example request

```
GET /provd/dev_mgr/devices HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "devices": [
        {
            "added": "auto",
            "config": "38e5e08ffe804b468f5aa53b9536bb25",
            "configured": true,
            "description": "",
            "id": "38e5e08ffe804b468f5aa53b9536bb25",
            "ip": "10.34.1.122",
            "mac": "00:08:5d:33:e5:76",
            "model": "6731i",
            "plugin": "xivo-aastra-3.3.1-SP2",
            "remote_state_sip_username": "je5qtq",
            "vendor": "Aastra",
            "version": "3.3.1.2235"
        }
    ]
}
```

Create a Device

Query

```
POST /provd/dev_mgr/devices  HTTP/1.1
```

Example request

```
POST /provd/dev_mgr/devices HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "device": {
        "ip": "192.168.1.1",
        "mac": "00:11:22:33:44:55",
        "plugin": "xivo-aastra-3.3.1-SP2"
    }
}
```

Example response

```
HTTP/1.1 201 Created
Content-Type: application/vnd.proformatique.provd+json
Location: /provd/dev_mgr/devices/68b10c99945b4fb889f22a7559fc3271

{"id": "68b10c99945b4fb889f22a7559fc3271"}
```

If the `id` field is not given, then an ID is automatically generated by the server.

Get a Device

Query

```
GET /provd/dev_mgr/devices/<device_id> HTTP/1.1
```

Example request

```
GET /provd/dev_mgr/devices/68b10c99945b4fb889f22a7559fc3271 HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "device": {
        "added": "auto",
        "config": "38e5e08ffe804b468f5aa53b9536bb25",
        "configured": true,
        "description": "",
        "id": "38e5e08ffe804b468f5aa53b9536bb25",
        "ip": "10.34.1.122",
        "mac": "00:08:5d:33:e5:76",
        "model": "6731i",
        "plugin": "xivo-aastra-3.3.1-SP2",
        "remote_state_sip_username": "je5qtq",
```

(continues on next page)

(continued from previous page)

```
        "vendor": "Aastra",
        "version": "3.3.1.2235"
    }
}
```

Update a Device

Query

```
PUT /provd/dev_mgr/devices/<device_id> HTTP/1.1
```

Example request

```
PUT /provd/dev_mgr/devices/68b10c99945b4fb889f22a7559fc3271 HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "device": {
        "added": "auto",
        "config": "38e5e08ffe804b468f5aa53b9536bb25",
        "configured": true,
        "description": "",
        "id": "38e5e08ffe804b468f5aa53b9536bb25",
        "ip": "10.34.1.122",
        "mac": "00:08:5d:33:e5:76",
        "model": "6731i",
        "plugin": "xivo-aastra-3.4",
        "remote_state_sip_username": "je5qtq",
        "vendor": "Aastra",
        "version": "3.3.1.2235"
    }
}
```

Example response

```
HTTP/1.1 204 No Content
```

Delete a Device

Query

```
DELETE /provd/dev_mgr/devices/<device_id> HTTP/1.1
```

Example request

```
DELETE /provd/dev_mgr/devices/68b10c99945b4fb889f22a7559fc3271 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Synchronize a Device

Query

```
POST /provd/dev_mgr/synchronize HTTP/1.1
```

Example request

```
POST /provd/dev_mgr/synchronize HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "id": "d035bccaf0dd4a8396fc57a3329ca0a4"
}
```

Example response

```
HTTP/1.1 201 Created
Location: /provd/dev_mgr/synchronize/42
```

The URI returned in the `Location` header points to an *operation in progress* resource.

Reconfigure a Device

Query

```
POST /provd/dev_mgr/reconfigure HTTP/1.1
```

Errors

Error code	Error message	Description
400	invalid device ID	

Example request

```
POST /provd/dev_mgr/reconfigure HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "id": "d035bccaf0dd4a8396fc57a3329ca0a4"
}
```

Example response

```
HTTP/1.1 204 No Content
```

Push DHCP Request Information

Query

```
POST /provd/dev_mgr/dhcpinfo HTTP/1.1
```

Example request

```
POST /provd/dev_mgr/dhcpinfo HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "dhcp_info": {
        "ip": "192.168.1.100",
        "mac": "00:11:22:33:44:55",
        "op": "commit",
        "options": [
            "06066.6f.6f.62.61.72.a"
        ]
    }
}
```

Example response

```
HTTP/1.1 204 No Content
```

Configs Management

Get the Config Manager

The config manager links to the following resources:

- The `cfg.configs` relation links to the *list of configs*.
- The `cfg.autocreate` relation links to the *config autocreate service*.

Query

```
GET /provd/cfg_mgr HTTP/1.1
```

Example request

```
GET /provd/cfg_mgr HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json


{
    "links": [
        {
            "href": "/provd/cfg_mgr/configs",
            "rel": "cfg.configs"
        },
        {
            "href": "/provd/cfg_mgr/autocreate",
            "rel": "cfg.autocreate"
        }
    ]
}
```

List Configs

Query

```
GET /provd/cfg_mgr/configs HTTP/1.1
```

Query Parameters

These are the *same parameters as for the list devices* action.

Example request

```
GET /provd/cfg_mgr/configs HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "configs": [
        {
            "configdevice": "defaultconfigdevice",
            "deletable": true,
            "id": "38e5e08ffe804b468f5aa53b9536bb25",
            "parent_ids": [
                "base",
                "defaultconfigdevice"
            ],
            "raw_config": {
                "X_key": "",
                "exten_dnd": "*25",
```

(continues on next page)

(continued from previous page)

```
                "exten_fwd_busy": "*23",
                "exten_fwd_disable_all": "*20",
                "exten_fwd_no_answer": "*22",
                "exten_fwd_unconditional": "*21",
                "exten_park": null,
                "exten_pickup_call": "*8",
                "exten_pickup_group": null,
                "exten_voicemail": "*98",
                "funckeys": {
                    "1": {
                        "label": "",
                        "line": 1,
                        "type": "speeddial",
                        "value": "1005"
                    }
                },
                "protocol": "SIP",
                "sip_dtmf_mode": "SIP-INFO",
                "sip_lines": {
                    "1": {
                        "auth_username": "je5qtq",
                        "display_name": "El\u00e8s 01",
                        "number": "1001",
                        "password": "T2S7C0",
                        "proxy_ip": "10.34.1.11",
                        "registrar_ip": "10.34.1.11",
                        "username": "je5qtq"
                    }
                }
            }
        }
    ]
}
```

Create a Config

Query

```
POST /provd/cfg_mgr/configs HTTP/1.1
```

Example request

```
POST /provd/cfg_mgr/configs HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "config": {
        "parent_ids": [
            "base"
        ],
        "raw_config": {
            "sip": {
                "lines": {
                    "1": {
                        "auth_username": "100",
```

(continues on next page)

(continued from previous page)

```
                        "display_name": "Foo",
                        "password": "100",
                        "username": "100"
                    }
                }
            }
        }
    }
}
```

Example response

```
HTTP/1.1 201 Created
Content-Type: application/vnd.proformatique.provd+json
Location: /provd/cfg_mgr/configs/77839d0f05c84662864b0ae5c27b33e4

{"id": "77839d0f05c84662864b0ae5c27b33e4"}
```

If the `id` field is not given, then an ID id automatically generated by the server.

Get a Config

Query

```
GET /provd/cfg_mgr/configs/<config_id> HTTP/1.1
```

Example request

```
GET /provd/cfg_mgr/configs/77839d0f05c84662864b0ae5c27b33e4 HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "config": {
        "id": "77839d0f05c84662864b0ae5c27b33e4",
        "parent_ids": [
            "base"
        ],
        "raw_config": {
            "sip": {
                "lines": {
                    "1": {
                        "auth_username": "100",
                        "display_name": "Foo",
                        "password": "100",
                        "username": "100"
                    }
```

(continues on next page)

(continued from previous page)

```
                    }
                }
            }
        }
}
```

Get a Raw Config

Query

```
GET /provd/cfg_mgr/configs/<config_id>/raw HTTP/1.1
```

Example request

```
GET /provd/cfg_mgr/configs/77839d0f05c84662864b0ae5c27b33e4/raw HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "raw_config": {
        "X_xivo_phonebook_ip": "10.34.1.11",
        "http_port": 8667,
        "ip": "10.34.1.11",
        "ntp_enabled": true,
        "ntp_ip": "10.34.1.11",
        "sip": {
            "lines": {
                "1": {
                    "auth_username": "100",
                    "display_name": "John",
                    "password": "100",
                    "username": "100"
                }
            }
        },
        "tftp_port": 69
    }
}
```

Update a Config

Query

```
PUT /provd/cfg_mgr/configs/<config_id> HTTP/1.1
```

Example request

```
PUT /provd/cfg_mgr/configs/77839d0f05c84662864b0ae5c27b33e4 HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "config": {
        "id": "77839d0f05c84662864b0ae5c27b33e4",
        "parent_ids": [
            "base"
        ],
        "raw_config": {
            "sip": {
                "lines": {
                    "1": {
                        "auth_username": "100",
                        "display_name": "John",
                        "password": "100",
                        "username": "100"
                    }
                }
            }
        }
    }
}
```

Example response

```
HTTP/1.1 204 No Content
```

Delete a Config

Query

```
DELETE /provd/cfg_mgr/configs/<config_id> HTTP/1.1
```

Example request

```
DELETE /provd/cfg_mgr/configs/77839d0f05c84662864b0ae5c27b33e4 HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Autocreate a Config

This service is used to create a new config from the config that has the `autocreate` role.

Query

```
POST /provd/cfg_mgr/autocreate HTTP/1.1
```

Example request

```
POST /provd/cfg_mgr/autocreate HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{}
```

Example response

```
HTTP/1.1 201 Created
Content-Type: application/vnd.proformatique.provd+json
Location: /provd/cfg_mgr/configs/autoprov1411400365

{"id":"autoprov1411400365"}
```

Plugins Management

Get the Plugin Manager

The plugin manager links to the following resources:

- The `srv.install` relation links to the plugin manager *installation service*. This installation service permits installing/uninstalling plugins.
- The `pg.plugins` relation links to the *list of plugins*.
- The `pg.reload` relation links to the *plugin reload service*.

Query

```
GET /provd/pg_mgr HTTP/1.1
```

Example request

```
GET /provd/pg_mgr HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
```

(continues on next page)

(continued from previous page)

```
    "links": [
        {
            "href": "/provd/pg_mgr/install",
            "rel": "srv.install"
        },
        {
            "href": "/provd/pg_mgr/plugins",
            "rel": "pg.plugins"
        },
        {
            "href": "/provd/pg_mgr/reload",
            "rel": "pg.reload"
        }
    ]
}
```

List Plugins

List the installed plugins.

If you want to install/uninstall plugins, you need to go trough the plugin installation service.

Query

```
GET /provd/pg_mgr/plugins HTTP/1.1
```

Example request

```
GET /provd/pg_mgr/plugins HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "plugins": {
        "xivo-aastra-3.3.1-SP2": {
            "links": [
                {
                    "href": "/provd/pg_mgr/plugins/xivo-aastra-3.3.1-SP2",
                    "rel": "pg.plugin"
                }
            ]
        },
        "xivo-cisco-sccp-9.0.3": {
            "links": [
                {
                    "href": "/provd/pg_mgr/plugins/xivo-cisco-sccp-9.0.3",
                    "rel": "pg.plugin"
                }
```

(continues on next page)

(continued from previous page)

```
            ]
        }
    }
}
```

Get a Plugin

The plugin links to the following resources:

- The `pg.info` relation links to the *plugin information*.
- The `srv.install` relation links to the plugin *installation service*. Plugins usually provided this service to install/uninstall firmware and language files.

Query

```
GET /provd/pg_mgr/plugins/<plugin_id> HTTP/1.1
```

Example request

```
GET /provd/pg_mgr/plugins/xivo-aastra-3.3.1-SP2 HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "links": [
        {
            "href": "/provd/pg_mgr/plugins/xivo-aastra-3.3.1-SP2/info",
            "rel": "pg.info"
        },
        {
            "href": "/provd/pg_mgr/plugins/xivo-aastra-3.3.1-SP2/install",
            "rel": "srv.install"
        }
    ]
}
```

Get Information of a Plugin

Query

```
GET /provd/pg_mgr/plugins/<plugin_id>/info HTTP/1.1
```

Example request

```
GET /provd/pg_mgr/plugins/xivo-aastra-3.3.1-SP2/info HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "plugin_info": {
        "capabilities": {
            "Aastra, 6730i, 3.3.1.5089": {
                "sip.lines": 6
            },
            "Aastra, 6731i, 3.3.1.2235": {
                "sip.lines": 6,
                "switchboard": true
            },
            "Aastra, 6735i, 3.3.1.5089": {
                "sip.lines": 9
            },
            "Aastra, 6737i, 3.3.1.5089": {
                "sip.lines": 9
            },
            "Aastra, 6739i, 3.3.1.2235": {
                "sip.lines": 9
            },
            "Aastra, 6753i, 3.3.1.2235": {
                "sip.lines": 9
            },
            "Aastra, 6755i, 3.3.1.2235": {
                "sip.lines": 9,
                "switchboard": true
            },
            "Aastra, 6757i, 3.3.1.2235": {
                "sip.lines": 9,
                "switchboard": true
            },
            "Aastra, 9143i, 3.3.1.2235": {
                "sip.lines": 9
            },
            "Aastra, 9480i, 3.3.1.2235": {
                "sip.lines": 9
            }
        },
        "description": "Plugin for Aastra 6730i, 6731i, 6735i, 6737i, 6739i, 6753i,
↪ 6755i, 6757i, 6757i CT, 9143i, 9480i, 9480i CT in version 3.3.1 SP2.",
        "version": "1.1"
    }
}
```

Reload a Plugin

Reload the given plugin. This is mostly useful during plugin development, after changing the code of the plugin, instead of restarting the xivo-provd application.

Query

```
POST /provd/pg_mgr/reload HTTP/1.1
```

Example request

```
POST /provd/pg_mgr/reload HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "id": "xivo-aastra-3.3.1-SP2"
}
```

Example response

```
HTTP/1.1 204 No Content
```

General Resources

This section describes the resources that are available from more than one URI or are generic enough to not fit in a more specific section.

Operation In Progress

This resource represents an operation in progress and is used to follow the progress of an underlying operation. Said differently, it is a monitor on an operation that can change over time.

Get Current Status

Query

```
GET <uri> HTTP/1.1
```

Example request

```
GET <uri> HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "status": "progress"
}
```

The `status` field describe the current status of the operation. The format is `[label|]state[;current[/end]](\(sub_oips\))*`. Here's some examples:

- progress
- download|progress
- download|progress;10
- download|progress;10/100
- download|progress(file_1|progress;20/100)(file_2|waiting;0/50)
- download|progress;20/150(file_1|progress)(file_2|waiting)
- op|progress(op1|progress(op11|progress)(op12|waiting))(op2|progress)

The state of an operation is either `waiting`, `progress`, `success` or `fail`.

Delete

Delete the "operation in progress" resource.

This does not cancel the underlying operation; it only deletes the monitor. Every monitor that is created should be deleted, else they won't be freed by the process and they will accumulate, taking memory.

Query

```
DELETE <uri> HTTP/1.1
```

Example request

```
DELETE <uri> HTTP/1.1
Host: xivoserver
```

Example response

```
HTTP/1.1 204 No Content
```

Configuration Service

Get the Configuration

Query

```
GET <uri> HTTP/1.1
```

Example request

Example request for the configuration service of the *provd manager*.

```
GET /provd/configure HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "params": [
        {
            "description": "The plugins repository URL",
            "id": "plugin_server",
            "links": [
                {
                    "href": "/provd/configure/plugin_server",
                    "rel": "srv.configure.param"
                }
            ],
            "value": "http://provd.xivo.solutions/plugins/1/stable"
        },
        {
            "description": "The proxy for HTTP requests. Format is \"http://
↪[user:password@]host:port\"",
            "id": "http_proxy",
            "links": [
                {
                    "href": "/provd/configure/http_proxy",
                    "rel": "srv.configure.param"
                }
            ],
            "value": null
        },
        {
            "description": "The proxy for FTP requests. Format is \"http://
↪[user:password@]host:port\"",
            "id": "ftp_proxy",
            "links": [
                {
                    "href": "/provd/configure/ftp_proxy",
                    "rel": "srv.configure.param"
                }
            ],
            "value": null
        },
        {
            "description": "The proxy for HTTPS requests. Format is \"host:port\"",
            "id": "https_proxy",
            "links": [
                {
                    "href": "/provd/configure/https_proxy",
                    "rel": "srv.configure.param"
                }
            ],
            "value": null
        },
        {
            "description": "The current locale. Example: fr_FR",
            "id": "locale",
            "links": [
                {
                    "href": "/provd/configure/locale",
                    "rel": "srv.configure.param"
                }
            ],
```

(continues on next page)

(continued from previous page)

```
            "value": null
        },
        {
            "description": "Set to 1 if all the devices are behind a NAT.",
            "id": "NAT",
            "links": [
                {
                    "href": "/provd/configure/NAT",
                    "rel": "srv.configure.param"
                }
            ],
            "value": 0
        }
    ]
}
```

Get the Value of a Parameter

Query

```
GET <uri> HTTP/1.1
```

Example request

Example request for the NAT option of the configuration service of the provd entry point.

```
GET /provd/configure/NAT HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "param": {
        "value": 0
    }
}
```

Set the Value of a Parameter

Query

```
PUT <uri> HTTP/1.1
```

Example request

Example request for the NAT option of the configuration service of the *provd manager*.

```
PUT /provd/configure/NAT HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "param": {
        "value": 1
    }
}
```

Example response

```
HTTP/1.1 204 No Content
Content-Type: application/vnd.proformatique.provd+json
```

Installation Service

Get the Installation Service

Query

```
GET <uri> HTTP/1.1
```

Example request

Example request for the installation service of the *plugin manager*.

```
GET /provd/pg_mgr/install HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "links": [
        {
            "href": "/provd/pg_mgr/install/install",
            "rel": "srv.install.install"
        },
        {
            "href": "/provd/pg_mgr/install/uninstall",
            "rel": "srv.install.uninstall"
        },
        {
            "href": "/provd/pg_mgr/install/installed",
            "rel": "srv.install.installed"
        },
        {
            "href": "/provd/pg_mgr/install/installable",
```

(continues on next page)

(continued from previous page)

```
            "rel": "srv.install.installable"
        },
        {
            "href": "/provd/pg_mgr/install/upgrade",
            "rel": "srv.install.upgrade"
        },
        {
            "href": "/provd/pg_mgr/install/update",
            "rel": "srv.install.update"
        }
    ]
}
```

The upgrade and update services are optional and not all installation service provide them.

Install a Package

Query

```
POST <uri> HTTP/1.1
```

Example request

Example request for the installation service of the plugin manager.

```
POST /provd/pg_mgr/install/install HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "id": "xivo-polycom-4.0.4"
}
```

Example response

```
HTTP/1.1 201 Created
Location: /provd/pg_mgr/install/install/1
Content-Type: application/vnd.proformatique.provd+json
```

The URI returned in the `Location` header points to an *operation in progress* resource.

Uninstall a Package

Query

```
POST <uri> HTTP/1.1
```

Example request

Example request for the installation service of the plugin manager.

```
POST /provd/pg_mgr/install/uninstall HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "id": "xivo-polycom-4.0.4"
}
```

Example response

```
HTTP/1.1 204 No Content
Content-Type: application/vnd.proformatique.provd+json
```

Upgrade a Package

Query

```
POST <uri> HTTP/1.1
```

Example request

Example request for the installation service of the plugin manager.

```
POST /provd/pg_mgr/install/upgrade HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{
    "id": "xivo-polycom-4.0.4"
}
```

Example response

```
HTTP/1.1 201 Created
Location: /provd/pg_mgr/install/upgrade/1
Content-Type: application/vnd.proformatique.provd+json
```

The URI returned in the `Location` header points to an *operation in progress* resource.

Update the List of Installable Packages

Query

```
POST <uri> HTTP/1.1
```

Example request

Example request for the installation service of the plugin manager.

```
POST /provd/pg_mgr/install/update HTTP/1.1
Host: xivoserver
Content-Type: application/vnd.proformatique.provd+json

{}
```

Example response

```
HTTP/1.1 201 Created
Location: /provd/pg_mgr/install/update/1
Content-Type: application/vnd.proformatique.provd+json
```

The URI returned in the `Location` header points to an *operation in progress* resource.

List Installable Packages

Query

```
GET <uri> HTTP/1.1
```

Example request

Example request for the installation service of the plugin manager.

```
GET /provd/pg_mgr/install/installable HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "pkgs": {
        "null": {
            "capabilities": {
                "*, *, *": {
                    "sip.lines": 0
                }
            },
            "description": "Plugin that offers no configuration service and␣
↪rejects TFTP/HTTP requests.",
            "dsize": 1073,
            "sha1sum": "90b2fb6c2b135a9d539488b6a85779dd95e0e876",
            "version": "1.0"
        },
        "xivo-aastra-3.3.1-SP2": {
            "capabilities": {
                "Aastra, 6730i, 3.3.1.5089": {
                    "sip.lines": 6
                },
                "Aastra, 6731i, 3.3.1.2235": {
```

(continues on next page)

(continued from previous page)


```
                    "sip.lines": 6,
                    "switchboard": true
                },
                "Aastra, 6735i, 3.3.1.5089": {
                    "sip.lines": 9
                },
                "Aastra, 6737i, 3.3.1.5089": {
                    "sip.lines": 9
                },
                "Aastra, 6739i, 3.3.1.2235": {
                    "sip.lines": 9
                },
                "Aastra, 6753i, 3.3.1.2235": {
                    "sip.lines": 9
                },
                "Aastra, 6755i, 3.3.1.2235": {
                    "sip.lines": 9,
                    "switchboard": true
                },
                "Aastra, 6757i, 3.3.1.2235": {
                    "sip.lines": 9,
                    "switchboard": true
                },
                "Aastra, 9143i, 3.3.1.2235": {
                    "sip.lines": 9
                },
                "Aastra, 9480i, 3.3.1.2235": {
                    "sip.lines": 9
                }
            },
            "description": "Plugin for Aastra 6730i, 6731i, 6735i, 6737i, 6739i,␣
↪6753i, 6755i, 6757i, 6757i CT, 9143i, 9480i, 9480i CT in version 3.3.1 SP2.",
            "dsize": 9397,
            "sha1sum": "68dbed6afa87cf624a89166bdc6bdf7413cb84df",
            "version": "1.1"
        }
    }
}
```


List Installed Packages

Query

```
GET <uri> HTTP/1.1
```

Example request

Example request for the installation service of the plugin manager.

```
GET /provd/pg_mgr/install/installed HTTP/1.1
Host: xivoserver
Accept: application/vnd.proformatique.provd+json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.proformatique.provd+json

{
    "pkgs": {
        "xivo-aastra-3.3.1-SP2": {
            "capabilities": {
                "Aastra, 6730i, 3.3.1.5089": {
                    "sip.lines": 6
                },
                "Aastra, 6731i, 3.3.1.2235": {
                    "sip.lines": 6,
                    "switchboard": true
                },
                "Aastra, 6735i, 3.3.1.5089": {
                    "sip.lines": 9
                },
                "Aastra, 6737i, 3.3.1.5089": {
                    "sip.lines": 9
                },
                "Aastra, 6739i, 3.3.1.2235": {
                    "sip.lines": 9
                },
                "Aastra, 6753i, 3.3.1.2235": {
                    "sip.lines": 9
                },
                "Aastra, 6755i, 3.3.1.2235": {
                    "sip.lines": 9,
                    "switchboard": true
                },
                "Aastra, 6757i, 3.3.1.2235": {
                    "sip.lines": 9,
                    "switchboard": true
                },
                "Aastra, 9143i, 3.3.1.2235": {
                    "sip.lines": 9
                },
                "Aastra, 9480i, 3.3.1.2235": {
                    "sip.lines": 9
                }
            },
            "description": "Plugin for Aastra 6730i, 6731i, 6735i, 6737i, 6739i,␣
↪6753i, 6755i, 6757i, 6757i CT, 9143i, 9480i, 9480i CT in version 3.3.1 SP2.",
            "version": "1.1"
        }
    }
}
```

xivo-sysconfd REST API

This service provides a public API that can be used to change the configuration that are on a XiVO.

> **Warning:** The 0.1 API is currently in development. Major changes could still happen and new resources will be added over time.

API reference

Asterisk Voicemail

Delete voicemail

Query

```
GET /delete_voicemail
```

Parameters

Mandatory

name the voicemail name

Optional

context the voicemail context (default is 'default')

Errors

Error code	Error message	Description
404	Not found	The voicemail does not exist

Example requests

```
GET /delete_voicemail HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    nothing
}
```

Common configuration

Apply configuration

Query

```
GET /commonconf_apply
```

Generate configuration

Query

```
POST /commonconf_generate
```

Dhcpd configuration

Update configuration

Query

```
GET /dhcpd_update
```

Ethernet configuration

Discover interfaces

Query

```
GET /discover_netifaces
```

Example request

```
GET /discover_netifaces HTTP/1.1
Host: xivoserver
Accept: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
   "lo":
   {
        "hwaddress": "00:00:00:00:00:00",
        "typeid": 24,
        "alias-raw-device": null,
        "network": "127.0.0.0",
        "family": "inet",
        "physicalif": false,
        "vlan-raw-device": null,
        "vlanif": false,
        "dummyif": false,
```

(continues on next page)

(continued from previous page)


```
        "mtu": 65536,
        "broadcast": "127.255.255.255",
        "hwtypeid": 772,
        "netmask": "255.0.0.0",
        "carrier": true,
        "flags": 9,
        "address": "127.0.0.1",
        "vlan-id": null,
        "type": "loopback",
        "options": null,
        "aliasif": false,
        "name": "lo"
    },
    "eth0":
    {
        "alias-raw-device": null,
        "family": "inet",
        "hwaddress": "36:76:70:29:69:c2",
        "vlan-id": null,
        "network": "172.17.0.0",
        "physicalif": false,
        "vlan-raw-device": null,
        "vlanif": false,
        "type": "eth",
        "aliasif": false,
        "broadcast": "172.17.255.255",
        "netmask": "255.255.0.0",
        "address": "172.17.0.101",
        "typeid": 6,
        "name": "eth0",
        "hwtypeid": 1,
        "dummyif": false,
        "mtu": 1500,
        "carrier": true,
        "flags": 3,
        "options": null
    }
}
```


Get interface

Query

```
GET /netiface/<interface>
```

Example request

```
GET /netiface/eth0 HTTP/1.1
Host: xivoserver
Content-Type: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
   "eth0":
   {
        "alias-raw-device": null,
        "family": "inet",
        "hwaddress": "36:76:70:29:69:c2",
        "vlan-id": null,
        "network": "172.17.0.0",
        "physicalif": false,
        "vlan-raw-device": null,
        "vlanif": false,
        "type": "eth",
        "aliasif": false,
        "broadcast": "172.17.255.255",
        "netmask": "255.255.0.0",
        "address": "172.17.0.101",
        "typeid": 6,
        "name": "eth0",
        "hwtypeid": 1,
        "dummyif": false,
        "mtu": 1500,
        "carrier": true,
        "flags": 3,
        "options": null
   }
}
```

Modify interface

Description

Field	Values	Description
iface	string	Interface name like eth0
method	list	static or dhcp
address	string	
netmask	string	
broadcast	string	
gateway	string	
mtu	int	
auto	boolean	
up	boolean	
options	list	dns-search and dns-nameservers

Query

```
PUT /modify_physical_eth_ipv4
```

Example request

```
PUT /modify_physical_eth_ipv4 HTTP/1.1
Host: xivoserver
```

(continues on next page)

(continued from previous page)

```
Content-Type: application/json
{
    "ifname": "eth0",
    "method": "dhcp",
    "auto": "True"
}
```

Replace virtual interface

Query

```
PUT /replace_virtual_eth_ipv4
```

Example request

```
PUT /replace_virtual_eth_ipv4 HTTP/1.1
Host: xivoserver
Content-Type: application/json
{
    "ifname": "eth0:0",
    "new_ifname": "eth0:1",
    "method": "dhcp",
    "auto": "True"
}
```

Modify interface

Query

```
PUT /modify_eth_ipv4
```

Example request

```
PUT /modify_eth_ipv4 HTTP/1.1
Host: xivoserver
Content-Type: application/json
{
    'ifname' : 'eth0'
    'address': '192.168.0.1',
    'netmask': '255.255.255.0',
    'broadcast': '192.168.0.255',
    'gateway': '192.168.0.254',
    'mtu': 1500,
    'auto': True,
    'up': True,
    'options': [['dns-search', 'toto.tld tutu.tld'],
                ['dns-nameservers', '127.0.0.1 192.168.0.254']]
}
```

Change state

Query

```
PUT /change_state_eth_ipv4
```

Example request

```
PUT /change_state_eth_ipv4 HTTP/1.1
Host: xivoserver
Content-Type: application/json
{
    'ifname' : 'eth0',
    'state': True
}
```

Delete interface ipv4

Query

```
GET /delete_eth_ipv4/<interface>
```

Example request

```
GET /delete_eth_ipv4/eth0 HTTP/1.1
Host: xivoserver
Content-Type: application/json
```

HA configuration

Get HA configuration

Query

```
GET /get_ha_config
```

Update HA configuration

Query

```
POST /update_ha_config
```

network configuration

Get network configuration

Query

```
GET /network_config
```

Rename ethernet interface

Query

```
POST /rename_ethernet_interface
```

swap ethernet interface

Query

```
POST /swap_ethernet_interfaces
```

Routes

Query

```
POST /routes
```

OpenSSL configuration

List certificates

Query

```
GET /openssl_listcertificates
```

Get certificate infos

Query

```
GET /openssl_certificateinfos
```

Export public key

Query

```
GET /openssl_exportpubkey
```

Export SSL certificate

Query

```
GET /openssl_export
```

Create CA certificate

Query

```
POST /openssl_createcacertificate
```

Create certificate

Query

```
POST / openssl_createcertificate
```

Delete certificate

Query

```
GET /openssl_deletecertificate
```

Import SSL certificate

Query

```
POST /openssl_import
```

DNS configuration

Host configuration

Query

```
POST /hosts
```

Resolv.conf configuration

Query

```
POST /resolv_conf
```

Services daemon

Reload services

Query

```
POST /services
```

Xivo Services

Reload XiVO services

Query

```
POST /xivoctl
```

Handlers

Execute handlers

Query

```
POST /exec_request_handlers
```

Status check

Status

Query

```
GET /status-check
```

Example request

```
GET /status-check HTTP/1.1
Host: xivoserver
Content-Type: application/json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "status": "up"
}
```

For other services, see http://xivo/api/. This public instance does not allow you to directly test the requests (i.e. the "Try it out!" button will not work), but you may use the *embedded version of your XiVO*, where this button will work.

How to use the embedded REST API web interface (Swagger UI)

Every XiVO server embeds its own copy of the Swagger UI. The instance embedded in the XiVO allows you to directly try the requests with the in-page buttons.

For the rest of this article, we will consider that your XiVO is accessible under the hostname `MY_XIVO`.

The instance is available at: http://MY_XIVO/api

Before using the Swagger UI, there are a few prerequisites:

- Accept the HTTPS certificate for each service of the XiVO
- Add the permissions to use the REST API to a Web Services Access user
- Obtain an authentication token

HTTPS certificates

For each service on the left menu that you want to try, you need to accept the HTTPS certificate for this service. To that end:

1. click on the service in the menu on the left
2. copy the URL you see in the text box at the top of the page, something like: `https://MY_XIVO:9497/0.1/api/api.json` and paste it in your browser
3. accept the HTTPS certificate validation exception
4. go back to http://MY_XIVO/api and select the service again (or click on the top-right "Explore" button)

You should now be able to see the different sections for the REST API of that service.

REST API permissions

You must create a Web Services Access with the right permissions before using the REST API. See *Web Services Access*.

Each endpoint has its own ACL, but you may add wildcard ACLs, like:

- `auth.#` to gain access to all `xivo-auth` REST API endpoints
- `confd.#` to gain access to all `xivo-confd` REST API endpoints
- `#` to gain access to every endpoint of every service.

> **Warning:** Only use wildcards when doing tests, not with a production REST API access. You should always restrict the permissions to the bare minimum.

Obtain an authentication token

You can get a token via Swagger UI (what else?). Choose the `xivo-auth` service in the list of REST API. Under `tokens`, choose `POST /tokens`.

1. In the top-right text box of the page (left to the "Explore" button), fill "token" with the string `username:password` where those credentials come from the Web Services Access you created earlier.

2. Go back to the `POST /tokens` section and click on the yellow box to the right of the `body` parameter. This will pre-fill the `body` parameter.
3. In the `body` parameter, set:
    - `backend` to `xivo_service`
    - `expiration` to the number of seconds for the token to be valid (e.g. 3600 for one hour). After the expiration time, you will need to re-authenticate to get a new token.
4. Click "Try it out" at the end of the section
5. In the response, you should see a `token` attribute.

For more informations about the backends of xivo-auth, see *xivo-auth plugins*.

Use the authentication token

To use the authentication token, choose the service for which you want to try the REST API, then paste the token in the top-right text box. You do not need to click "Explore" to apply the token change, the new token will be used automatically at the next request you send.

You can now choose a REST API endpoint and "Try it out".

9.4.2 Access

Each REST API is available via HTTPS on *different ports*.

9.4.3 Examples (xivo-confd)

```
# Get the list of users
curl --insecure \
-H 'Accept: application/json' \
-H 'X-Auth-Token: 17496bfa-4653-9d9d-92aa-17def0fa9826' \
https://xivo:9486/1.1/users

# Create a user
# When sending data, you need the Content-Type header.
curl --insecure \
-X POST \
-d '{"firstname": "hello-world"} \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-H 'X-Auth-Token: 17496bfa-4653-9d9d-92aa-17def0fa9826' \
https://xivo:9486/1.1/users
```

9.4.4 Authentication

For all REST APIs, the main way to authenticate is to use an access token obtained from *xivo-auth*. This token should be given in the `X-Auth-Token` header in your request. For example:

```
curl <options...> -H 'X-Auth-Token: 17496bfa-4653-9d9d-92aa-17def0fa9826' https://
↪<xivo_address>:9486/1.1/users
```

Also, your token needs to have the right ACLs to give you access to the resource you want. See *REST API Permissions*.

REST API Permissions

The tokens delivered by *xivo-auth* have a list of permissions associated (ACL), that determine which REST resources are authorized for this token. Each REST resource has an associated required ACL. When you try to access to a REST resource, this resource requests xivo-auth with your token and the required ACL to validate the access.

Syntax

An ACL contains 3 parts separated by dot (`.`)

- *service*: name of service, without prefix `xivo-` (e.g. `xivo-confd` -> `confd`).
- *resource*: name of resource separated by dot (`.`) (e.g. `/users/17/lines` -> `users.17.lines`).
- *action*: action performed on resource. Generally, this is the following schema:
  - `get` -> `read`
  - `put` -> `update`
  - `post` -> `create`
  - `delete` -> `delete`

Substitutions

There are 3 substitution values for an ACL.

- `*`: replace only one word between dot.
- `#`: replace one or multiple words.
- `me`: replace the `user_uuid` from sent token.

Example

The ACL `confd.users.me.#.read` will have access to the following REST resources:

```
GET /users/{user_id}/cti
GET /users/{user_id}/funckeys
GET /users/{user_id}/funckeys/{position}
GET /users/{user_id}/funckeys/templates
GET /users/{user_id}/lines
GET /users/{user_id}/lines/{line_id}
GET /users/{user_id}/voicemail
```

- *service*: `confd`
- *resource*: `users.me.#`
- *action*: `read`

The ACL `confd.users.me.funckeys.*.*` will have access to the following REST resources:

```
DELETE /users/{user_id}funckeys/{position}
GET /users/{user_id}funckeys/{position}
PUT /users/{user_id}funckeys/{position}
GET /users/{user_id}funckeys/templates
```

- *service*: `confd`
- *resource*: `users.me.funckeys.*`

- *action*: `*`

Where `{user_id}` is the user uuid from the token.

Available ACLs

The ACL corresponding to each resource is documented. Some resources may not have any associated ACL yet, so you must use `{service}.#` instead.

See also *Service Authentication* for details about the token-based authentication process.

Other methods (xivo-confd)

Warning: DEPRECATED

For compatibility reason, xivo-confd may accept requests without an access token. For this, you must create a webservices user in the web interface (*Configuration* → *Management* → *Web Services Access*):

- if an IP address is specified for the user, no authentication is needed
- if you choose not to specify an IP address for the user, you can connect to the REST API with a HTTP Digest authentication, using the user name and password you provided. For instance, the following command line allows to retrieve XiVO users through the REST API, using the login **admin** and the password **passadmin**:

```
curl <options...> --digest --cookie '' -u admin:passadmin https://<xivo_
↪address>:9486/1.1/users
```

9.4.5 HTTP status codes

Standard HTTP status codes are used. For the full definition see IANA definition.

- 200: Success
- 201: Created
- 400: Incorrect syntax
- 404: Resource not found
- 406: Not acceptable
- 412: Precondition failed
- 415: Unsupported media type
- 500: Internal server error

See also *Errors* for general explanations about error codes.

9.4.6 General URL parameters

Example usage of general parameters:

```
GET http://<xivo_address>:9486/1.1/voicemails?limit=X&offset=Y
```

Parameters

order Sort the list using a column (e.g. "number"). See specific resource documentation for columns allowed.

direction 'asc' or 'desc'. Sort list in ascending (asc) or descending (desc) order

limit total number of resources to show in the list. Must be a positive integer

offset number of resources to skip over before starting the list. Must be a positive integer

search Search resources. Only resources with a field containing the search term will be listed.

9.4.7 Data representation

Data retrieved from the REST server

JSON is used to encode returned or sent data. Therefore, the following headers are needed:

- **when the request is supposed to return JSON:** Accept = application/json
- **when the request's body contains JSON:** Content-Type = application/json

---

Note: Optional properties can be added without changing the protocol version in the main list or in the object list itself. Properties will not be removed, type and name will not be modified.

Getting object lists

`GET /1.1/objects`

When returning lists the format is as follows:

- total - number of items in total in the system configuration (optional)
- items - returned data as an array of object properties list.

Other optional properties can be added later.

`Response data format`

```
{
    "total": 2,

    "items":
    [
        {
            "id": "1",
            "prop1": "test"
        },
        {
            "id": "2",
            "prop1": "ssd"
        }
    ]
}
```

Getting An Object

Format returned is a list of properties. The object should always have the same attributes set, the default value being the equivalent to NULL in the content-type format.

```
GET /1.1/objects/<id>
```

Response data format

```
{
   "id": "1",
   "prop1": "test"
}
```

Data sent to the REST server

The XiVO REST server implements POST and PUT methods for item creation and update respectively. Data is created using the POST method via a root URL and is updated using the PUT method via a root URL suffixed by /<id. The server expects to receive JSON encoded data. Only one item can be processed per request. The data format and required data fields are illustrated in the following example:

Request data format

```
{
   "id": "1",
   "prop1": "test"
}
```

When updating, only the id and updated properties are needed, omitted properties are not updated. Some properties can also be optional when creating an object.

Errors

A request to the web services may return an error. An error will always be associated to an HTTP error code, and eventually to one or more error messages. The following errors are common to all web services:

Error code	Error message	Description
406	empty	Accept header missing or contains an unsupported content type
415	empty	Content-Type header missing or contains an unsupported content type
500	list of errors	An error occured on the server side; the content of the message depends of the type of errors which occured

The 400, 404 and 412 errors depend on the web service you are requesting. They are separately described for each of them.

The error messages are contained in a JSON list, even if there is only one error message:

```
[ message_1, message_2, ... ]
```

9.5 XiVO Configuration MGT REST Api

This section describes the Config-MGT API.

In the following, all url are relative to the config-mgt base url and port. For example a relative URL `/callback_lists` is meant to be replaced by

```
http://192.168.29.101:9100/configmgt/api/1.0/callback_lists
```

assuming that XiVO is available at `192.168.29.101`

9.5.1 Authentication

To use the config-mgt api you need to add an additional header in the HTTP Request. The header name is `X-Auth-Token` and its value must be the same as the `authentication.token` value in the `application.conf` of the config-mgt.

Example:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json"
     -H 'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k'  http://192.168.29.101:9100/
↪configmgt/api/1.0/callback_lists
```

9.5.2 Profiles

Users can be given a certain *profile*. The profiles are:

- Administrator
- Supervisor
- Teacher

A profile defines:

1. an access right to some applications (CC Manager, Recording Server . . .)
2. and also some rights inside the application.

These profiles are described in *Profile Management* page.

Get all user's profiles

This api retrieves the profiles of all XiVO users having login.

Description:

URL `/users`

Method `GET`

Example Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" -H 'X-
↪Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/api/
↪1.0/users'
```

Response:

```
[{"name":"Ménage","firstname":"Jean","login":"jmenage","profile":"teacher"},{"name
↪":"Urbain","firstname":"Jocelyn","login":"jurbain","profile":"admin"}]
```

Get a user's profile and rights

This api retrieves the profile and the associated right of a XiVO user having login.

Description:

URL `/rights/user`

Method `GET`

Url parameters

login user's login name

Example Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" -H 'X-
↪Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/api/
↪1.0/rights/user/jbond'
```

Response:

```
{"type":"supervisor","data":{"queueIds":[3,2,1,7],"groupIds":[3,1,2],"incallIds
↪":[],"recordingAccess":true}}
```

Update user's profile or rights

Updates the profile or the associated right of a XiVO user having login.

Description:

URL `/rights/user`

Method `POST`

Url parameters

login user's login name

Request body Json object with field & value pair.

Allowed field names:

type The profile to set

data The rights to update

Example Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/
↪api/1.0/rights/user/jbond' -d '{"type":"supervisor"}'
```

Delete user's profile

Delete the profile associated to a XiVO user having login.

Description:

URL `/rights/user`

Method `DELETE`

Url parameters

login user's login name

Example Query:

```
curl -XDELETE -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/
↪api/1.0/rights/user/jbond'
```

9.5.3 Callbacks

The following API allow to define callbacks, a callback, is a shared note for agents in the same queue to know that he must call his customer before a deadline. It is used in *CC Agent feature* existing in Xivo solutions.

A what so called *Callback* is in fact splitted in four distinct entities:

- **Callback list**: Container object used to define a set of callback requests
- **Callback request**: Core object that contains firstname, lastname, phone number... of the customer to call back
- **Callback period**: The preferred interval of time in which the call should be performed
- **Callback ticket**: Once callback request is taken by an agent, a ticket is created to sum up actions made on the request, like the status of the call and if the request is now closed or not.

More information on how to *Process Callbacks with CCAgent*

Create Callbacks list

Create a Callback list container for a queue

Description:

URL `/callback_lists`

Method `POST`

Request body Json object with field & value pair.

Allowed field names:

name The name of the list

queueId The queue to affect the callback requests

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/
↪api/1.0/callback_lists' -d '{"name":"newlist", "queueId":1}'
```

Response:

```
{
    "callbacks": [],
    "name": "newlist",
    "queueId": 1,
    "uuid": "9d28d8fe-0548-4d45-aa08-9623ef69a04b"
}
```

Get Callbacks list

List all the Callback list containers

Description:

URL `/callback_lists`

Method `GET`

Url parameters

withRequest boolean to retrieve list if and only if it contains ongoing callback requests

---

Example

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" -H 'X-
↪Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/api/
↪1.0/callback_lists'
```

Response:

```
[{"uuid":"fea963f5-1920-468c-b52a-93dc88791ba8","name":"Mine","queueId":2,
↪"callbacks":
  [{"uuid":"edb734e7-9d8f-403d-8cf6-d42ecf9e48d7",
    "listUuid":"fea963f5-1920-468c-b52a-93dc88791ba8",
    "phoneNumber":"0230210092",
    "mobilePhoneNumber":"0689746321",
    "firstName":"John",
    "lastName":"Doe",
    "company":"MyCompany",
    "description":"Call back quickly",
    "preferredPeriodUuid":"31f91ef6-ebda-4e0d-a9fa-5ebd3da30951",
    "dueDate":"2017-09-27",
    "queueId":2,
    "clotured":false,
    "preferredPeriod":{"uuid":"31f91ef6-ebda-4e0d-a9fa-5ebd3da30951","name":"Toute␣
↪la journée","periodStart":"09:00:00","periodEnd":"17:00:00","default":true}}
  ]},{"uuid":"75509ad3-3f81-40be-ad90-2c36d7a2c809","name":"another","queueId":2,
↪"callbacks":[]}]
```

Delete Callbacks list

Delete a Callback list container

Description:

URL `/callback_lists`

Method `DELETE`

Example

Query:

```
curl -XDELETE -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/
↪api/1.0/callback_lists'
```

Response:

```
{
    "callbacks": [],
    "name": "newlist",
    "queueId": 1,
    "uuid": "9d28d8fe-0548-4d45-aa08-9623ef69a04b"
}
```

Import Callbacks requests

Import callback request in a Callback list container

Description:

URL `/callback_lists/<listUuid>/callback_requests/csv`

Method `POST`

Url parameters

listUuid id of the callback list

Request body should be compliant with *following format*

Example

Query:

```
curl -XPOST -H "Content-Type: text/plain" -H "Accept: text/plain" -H 'X-Auth-
↪Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/api/1.0//
↪callback_lists/9d28d8fe-0548-4d45-aa08-9623ef69a04b/callback_requests/csv'-d '
 ␣
↪phoneNumber|mobilePhoneNumber|firstName|lastName|company|description|dueDate|period
   0230210092|0689746321|John|Doe|MyCompany|Call back quickly||
   0587963214|0789654123|Alice|O'Neill|YourSociety||2016-08-01|Afternoon'
```

Create Callback request

Create a single callback request in a callback list

Description:

URL `/callback_lists/<listUuid>/callback_requests`

Method `POST`

Url parameters

listUuid id of the callback list

Request body Json object with field & value pair.

Allowed field names:

phoneNumber The number to call

mobilePhoneNumber Alternate number to call

firstName Contact first name (optional)

lastName Contact last name (optional)

company Contact company name (optional)

description Note displayed inside the callback for the agent (optional)

dueDate Deadline of the callback, using ISO format: YYYY-MM-DD

period Name of the period as defined in *callback list*. (optional)

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json"  -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/
↪api/1.0//callback_lists/9d28d8fe-0548-4d45-aa08-9623ef69a04b/callback_requests' -
↪d '
{
    "company":"Cie",
    "phoneNumber":"0298765432",
    "mobilePhoneNumber":"0654321234",
    "firstName":"Jack"
}'
```

Dynamic filters

A **dynamic filter** is just a JSON representation to create lite look-a-like SQL assertions. It contains:

- **field**: Field to make your query on
- **operator**: Can be one of = , *!=* , > , >= , < , <= , *like* , *ilike* , *is null* , *is not null*
- **value**: value to filter on
- **order**: can be *ASC* or *DESC*

Find Callback request

Finds a callback request using dynamic filters.

Description:

URL `callback_requests/find`

Method `POST`

Request body Json object with field & value pair.

Allowed field names:

filters List of dynamic filters

offset Distance between the beginning and the first result to retrieve, used for pagination (optional)

limit Max number of result to return (optional)

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" -H
↪'X-Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/
↪api/1.0/callback_requests/find' -d '
{
  "filters":[
    {"field":"phoneNumber", "operator":"=", "value":"1000"}
  ],
  "offset":0,
  "limit":100}'
```

Response:

```
{
  "total":1,
  "list":[{
    "uuid":"7691e6c8-6ebc-4d8f-a41c-45c049ac0dd4",
    "listUuid":"fea963f5-1920-468c-b52a-93dc88791ba8",
    "phoneNumber":"1000",
    "mobilePhoneNumber":"2000",
    "preferredPeriodUuid":"a6119323-a793-4264-987b-c565ceac342b",
    "dueDate":"2018-07-05",
    "queueId":2,
    "clotured":false,
    "preferredPeriod":{
      "uuid":"a6119323-a793-4264-987b-c565ceac342b",
      "name":"Toute la journée",
      "periodStart":"09:00:00",
      "periodEnd":"17:00:00",
      "default":true
    }
  }]
}
```


9.5.4 Agents

Get agent configuration

This api retrieves the agent configuration together with associated queues

Description:

URL `/agent_config`

Method `GET`

Url parameters

id agent's id

Example

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" -H 'X-
↪Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/api/
↪1.0/agent_config/1'
```

Response:


```
{"id":1,"firstname":"John","lastname":"Doe","number":"1001","context":"default",
↪"member": [
    {"queue_name":"queue1","queue_id":1,"interface":"Agent/1001","penalty":1,
↪"commented":0,"usertype":"Agent","userid":1,"channel":"Agent","category":"Queue",
↪"position":1},
    {"queue_name":"queue2","queue_id":2,"interface":"Agent/1001","penalty":2,
↪"commented":0,"usertype":"Agent","userid":1,"channel":"Agent","category":"Queue",
↪"position":1}],
"numgroup":1,"userid":1}
```


Get all agent configurations list

List all the agents together with associated queues

Description:

URL `/agent_config`

Method `GET`

Url parameters

Example

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" -H 'X-
↪Auth-Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9100/configmgt/api/
↪1.0/agent_config'
```

Response:


```
  [{
  "id": 1,
  "firstname": "Agent",
  "lastname": "One",
  "number": "1001",
  "context": "default",
  "member": [{
      "queue_name": "queue1",
      "queue_id": 1,
      "interface": "Agent/1001",
      "penalty": 1,
      "commented": 0,
      "usertype": "Agent",
      "userid": 1,
      "channel": "Agent",
      "category": "Queue",
      "position": 1
    }],
  "numgroup": 1,"userid": 1
},
{
  "id": 2,
  "firstname": "Agent",
  "lastname": "Two",
  "number": "1002",
  "context": "default",
  "member": [{
      "queue_name": "queue2",
      "queue_id": 2,
      "interface": "Agent/1002",
      "penalty": 1,
      "commented": 0,
      "usertype": "Agent",
      "userid": 2,
      "channel": "Agent",
      "category": "Queue",
      "position": 1
    }],
  "numgroup": 1,"userid": 2 }]
```

9.6 Subroutine

9.6.1 What is it ?

The preprocess subroutine allows you to enhance XiVO features through the Asterisk dialplan. Features that can be enhanced are :

- User
- Group
- Queue
- Meetme
- Incoming call
- Outgoing call

There are three possible categories :

- Subroutine for one feature
- Subroutine for global forwarding
- Subroutine for global incoming call to an object

Subroutines are called at the latest possible moment in the dialplan, so that the maximum of variables have already been set: this way, the variables can be read and modified at will before they are used.

Here is an example of the dialplan execution flow when an external incoming call to a user being forwarded to another external number (like a forward to a mobile phone):

9.6.2 Adding new subroutine

If you want to add a new subroutine, we propose to edit a new configuration file in the directory `/etc/asterisk/extensions_extra.d`. You can also add this file by the web interface.

An example:

```
[myexample]
exten = s,1,NoOp(This is an example)
same  =   n,Return()
```

Subroutines should always end with a `Return()`. You may replace `Return()` by a `Goto()` if you want to completely bypass the XiVO dialplan, but this is not recommended.

To plug your subroutine into the XiVO dialplan, you must add `myexample` in the subroutine field in the web interface, e.g. *Services → IPBX → PBX Settings → Users → Edit → tab General → Preprocess subroutine*.

9.6.3 Global subroutine

There is predefined subroutine for this feature, you can find the name and the activation in the `/etc/xivo/asterisk/xivo_globals.conf`. The variables are:

```
; Global Preprocess subroutine
XIVO_PRESUBR_GLOBAL_ENABLE = 1
XIVO_PRESUBR_GLOBAL_USER = xivo-subrgbl-user
XIVO_PRESUBR_GLOBAL_AGENT = xivo-subrgbl-agent
XIVO_PRESUBR_GLOBAL_GROUP = xivo-subrgbl-group
XIVO_PRESUBR_GLOBAL_QUEUE = xivo-subrgbl-queue
XIVO_PRESUBR_GLOBAL_MEETME = xivo-subrgbl-meetme
XIVO_PRESUBR_GLOBAL_DID = xivo-subrgbl-did
```

(continues on next page)

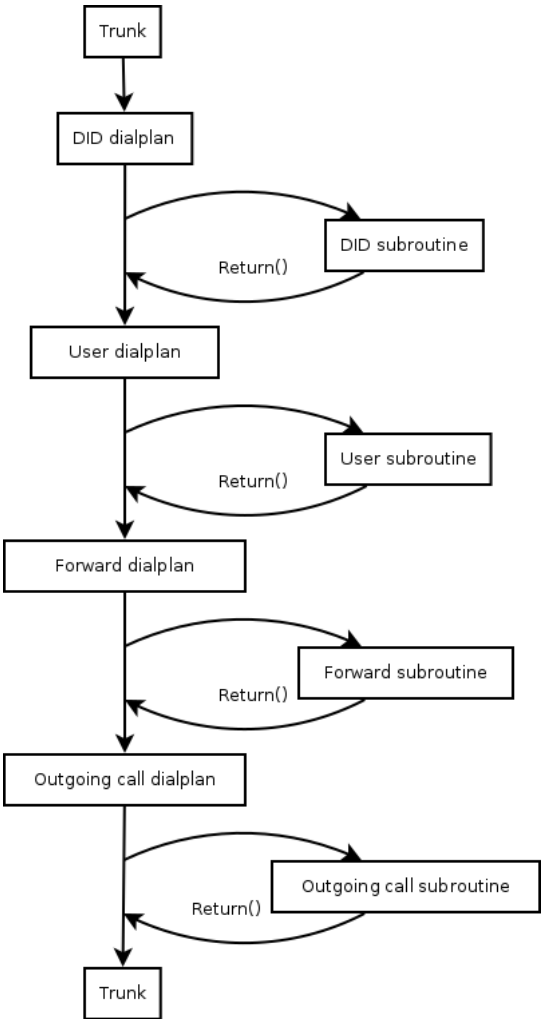


Fig. 5: Where subroutines are called in dialplan

(continued from previous page)

```
XIVO_PRESUBR_GLOBAL_OUTCALL = xivo-subrgbl-outcall
XIVO_PRESUBR_GLOBAL_PAGING = xivo-subrgbl-paging
```

So if you want to add a subroutine for all of your XiVO users you can do this:

```
[xivo-subrgbl-user]
exten = s,1,NoOp(This is an example for all my users)
same  =   n,Return()
```

9.6.4 Forward subroutine

You can also use a global subroutine for call forward.

```
; Preprocess subroutine for forwards
XIVO_PRESUBR_FWD_ENABLE = 1
XIVO_PRESUBR_FWD_USER = xivo-subrfwd-user
XIVO_PRESUBR_FWD_GROUP = xivo-subrfwd-group
XIVO_PRESUBR_FWD_QUEUE = xivo-subrfwd-queue
XIVO_PRESUBR_FWD_MEETME = xivo-subrfwd-meetme
XIVO_PRESUBR_FWD_VOICEMAIL = xivo-subrfwd-voicemail
XIVO_PRESUBR_FWD_SCHEDULE = xivo-subrfwd-schedule
XIVO_PRESUBR_FWD_VOICEMENU = xivo-subrfwd-voicemenu
XIVO_PRESUBR_FWD_SOUND = xivo-subrfwd-sound
XIVO_PRESUBR_FWD_CUSTOM = xivo-subrfwd-custom
XIVO_PRESUBR_FWD_EXTENSION = xivo-subrfwd-extension
```

9.6.5 Dialplan variables

Some of the XiVO variables can be used and modified in subroutines (non exhaustive list):

- `XIVO_CALLOPTIONS`: the value is a list of options to be passed to the Dial application, e.g. `hHtT`. This variable is available in agent, user and outgoing call subroutines. Please note that it may not be set earlier, because it will be overwritten.
- `XIVO_CALLORIGIN`: the value is:
  - `extern` for calls coming from a DID
  - `intern` for all other calls

  This variable is used by xivo-agid when *selecting the ringtone* for ringing a user. This variable is available only in user subroutines.
- `XIVO_DSTNUM`: the value is the extension dialed, as received by XiVO (e.g. an internal extension, a DID, or an outgoing extension including the local prefix). This variable is available in all subroutines.
- `XIVO_GROUPNAME`: the value is the name of the group being called. This variable is only available in group subroutines.
- `XIVO_GROUPOPTIONS`: the value is a list of options to be passed to the Queue application, e.g. `hHtT`. This variable is only available in group subroutines.
- `XIVO_INTERFACE`: the value is the *Technology/Resource* pairs that are used as the first argument of the Dial application. This variable is only available in the user subroutines.
- `XIVO_MOBILEPHONENUMBER`: the value is the phone number of a user, as set in the web interface. This variable is only available in user subroutines.
- `XIVO_QUEUENAME`: the value is the name of the queue being called. This variable is only available in queue subroutines.

- XIVO_QUEUEOPTIONS: the value is a list of options to be passed to the Queue application, e.g. hHtT. This variable is only available in queue subroutines.
- XIVO_SRCNUM: the value is the callerid number of the originator of the call: the internal extension of a user (outgoing callerid is ignored), or the public extension of an external incoming call. This variable is available in all subroutines.

9.7 Queue logs

Queue logs are events logged by Asterisk in the queue_log table of the asterisk database. Queue logs are used to generate XiVO call center statistics.

9.7.1 Queue log sample

Agent callback login

```
          time            |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |␣
↪data5
---------------------------+-----------------+-----------+------------+------------
↪---------+---------------------+-----------------+---------------+-------+-------
2012-07-03 15:27:23.896208 | 1341343640.4    | NONE      | Agent/3001 |␣
↪AGENTCALLBACKLOGIN  | 1002@pcm-dev        |                 |               |    ␣
↪    |
```

Agent callback logoff

Agent/3001 is logged in queues q1 and q2.

```
          time            |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |␣
↪data5
---------------------------+-----------------+-----------+------------+------------
↪---------+---------------------+-----------------+---------------+-------+-------
2012-07-03 15:28:07.348244 | NONE            | q2        | Agent/3001 | UNPAUSE    ␣
↪          |                     |                 |               |       |
2012-07-03 15:28:07.346320 | NONE            | q1        | Agent/3001 | UNPAUSE    ␣
↪          |                     |                 |               |       |
2012-07-03 15:28:07.327425 | NONE            | NONE      | Agent/3001 | UNPAUSEALL␣
↪          |                     |                 |               |       |
2012-07-03 15:28:06.249357 | NONE            | NONE      | Agent/3001 |␣
↪AGENTCALLBACKLOGOFF | 1002@pcm-dev        | 43              | CommandLogoff |    ␣
↪    |
```

Call on a Queue with join empty conditions met

```
          time            |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |␣
↪data5
---------------------------+-----------------+-----------+------------+------------
↪---------+---------------------+-----------------+---------------+-------+-------
2012-07-04 07:27:55.640421 | 1341401275.9    | q1        | NONE       | JOINEMPTY  ␣
↪          |                     |                 |               |       |
```

Enter the queue and get answered by an agent

```
           time             |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |
↪data5
----------------------------+-----------------+-----------+------------+------------
↪---------+--------------------+-----------------+---------------+-------+-------
2012-07-04 07:33:23.085718 | 1341401601.24   | q1        | Agent/3001 | CONNECT
↪          | 2                  | 1341401601.27   | 1             |       |
2012-07-04 07:33:21.165823 | 1341401601.24   | q1        | NONE       | ENTERQUEUE
↪          |                    | 1000            | 1             |       |
```

Agent or caller ends the call after 12 seconds

```
           time             |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |
↪data5
----------------------------+-----------------+-----------+------------+------------
↪---------+--------------------+-----------------+---------------+-------+-------
2012-07-04 07:37:46.601754 | 1341401851.34   | q1        | Agent/3001 |
↪COMPLETEAGENT       | 2                  | 12              | 1             |
↪    |
```

Call on a full queue

```
           time             |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |
↪data5
----------------------------+-----------------+-----------+------------+------------
↪---------+--------------------+-----------------+---------------+-------+-------
2012-07-04 07:40:17.339945 | 1341402016.44   | q1        | NONE       | FULL
↪          |                    |                 |               |       |
```

Call on a closed queue

```
           time             |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |
↪data5
----------------------------+-----------------+-----------+------------+------------
↪---------+--------------------+-----------------+---------------+-------+-------
2012-07-04 07:48:03.455999 | 1341402482.49   | q1        | NONE       | CLOSED
↪          |                    |                 |               |       |
```

Caller abandon before an answer

```
            time             |     callid      | queuename |   agent    |
↪event        |        data1        |      data2      |     data3     | data4 |
↪data5
----------------------------+-----------------+-----------+------------+------------
↪---------+--------------------+-----------------+---------------+-------+------
↪-
 2012-07-04 07:49:52.939802 | 1341402586.51   | q1        | NONE       | ABANDON
↪          | 1                  | 1               | 6             |       |
```

CHAPTER 10

Contributing

General information:

10.1 Contributing to the Documentation

XiVO documentation is generated with Sphinx. The source code is available on GitLab at https://gitlab.com/xivo.solutions/xivo-solutions-doc

Provided you already have Python installed on your system. You need first to install Sphinx : `easy_install -U Sphinx`[1].

Quick Reference

- http://docutils.sourceforge.net/docs/user/rst/cheatsheet.txt
- http://docutils.sourceforge.net/docs/user/rst/quickref.html
- http://openalea.gforge.inria.fr/doc/openalea/doc/_build/html/source/sphinx/rest_syntax.html

10.1.1 Documentation guideline

Here's the guideline/conventions to follow for the XiVO documentation.

Language

The documentation must be written in english, and only in english.

Sections

The top section of each file must be capitalized using the following rule: capitalization of all words, except for articles, prepositions, conjunctions, and forms of to be.

Correct:

[1] `easy_install` can be found in the debian package python-setuptools : `sudo apt-get install python-setuptools`

```
The Vitamins are in My Fresh California Raisins
```

Incorrect:

```
The Vitamins Are In My Fresh California Raisins
```

Use the following punctuation characters:

- * with overline, for "file title"
- =, for sections
- -, for subsections
- ^, for subsubsections

Punctuation characters should be exactly as long as the section text.

Correct:

```
Section1
========
```

Incorrect:

```
Section2
==========
```

There should be 2 empty lines between sections, except when an empty section is followed by another section.

Correct:

```
Section1
========

Foo.


Section2
========

Bar.
```

Correct:

```
Section1
========

Foo.


.. _target:

Section2
========

Bar.
```

Correct:

```
Section1
========

Subsection1
```

(continues on next page)

(continued from previous page)

```
-----------

Foo.
```

Incorrect:

```
Section1
========

Foo.

Section2
========

Bar.
```

Lists

Bullet lists:

```
* First item
* Second item
```

Autonumbered lists:

```
#. First item
#. Second item
```

Literal blocks

Use `::` on the same line as the line containing text when possible.

The literal blocks must be indented with three spaces.

Correct:

```
Bla bla bla::

   apt-get update
```

Incorrect:

```
Bla bla bla:

::

   apt-get update
```

Inline markup

Use the following roles when applicable:

- `:file:` for file, i.e.:

  ```
  The :file:`/dev/null` file.
  ```

- `:menuselection:` for the web interface menu:

```
The :menuselection:`Configuration --> Management --> Certificates` page.
```

- `:guilabel:` for designating a specific GUI element:

```
The :guilabel:`Action` column.
```

Others

- There must be no warning nor error messages when building the documentation with `make html`.
- There should be one and only one newline character at the end of each file
- There should be no trailing whitespace at the end of lines
- Paragraphs must be wrapped and lines should be at most 100 characters long

10.2 Debugging Asterisk

10.2.1 Precondition

To debug asterisk crashes or freezes, you need the following debug packages on your XiVO:

General rule	XiVO < 14.18	XiVO >= 14.18
Example version	14.12	14.18

Commands	`apt-get install xivo-fai-`
	`↪14.12`
	`apt-get update`
	`apt-get install gdb`
	`apt-get install -t xivo-`
	`↪14.12 asterisk-dbg`
	`↪xivo-libsccp-dbg`
		`xivo-dist xivo-14.18`
		`apt-get update`
		`apt-get install gdb`
		`apt-get install -t xivo-`
		`↪14.18 asterisk-dbg`
		`↪xivo-libsccp-dbg`

10.2.2 So There is a Problem with Asterisk. Now What ?

1. Find out the time of the incident from the people most likely to know
2. Determine if there was a segfault
    1. The command `grep segfault /var/log/syslog` should return a line such as the following:

    ```
    Oct 16 16:12:43 xivo-1 kernel: [10295061.047120] asterisk[1255]: segfault
    ↪at e ip b751aa6b sp b5ef14d4 error 4 in libc-2.11.3.so[b74ad000+140000]
    ```

    2. Note the exact time of the incident from the segfault line.
    3. Follow the *Debugging Asterisk Crash* procedure.
3. If you observe some of the following common symptoms, follow the *Debugging Asterisk Freeze* procedure.
    - The output of command `service asterisk status` says Asterisk PBX is running.
    - No more calls are distributed and phones go to `No Service`.
    - Command `core show channels` returns only headers (no data) right before returning
4. Fetch Asterisk logs for the day of the crash (make sure file was not already logrotated):

```
cp -a /var/log/asterisk/full /var/local/`date +"%Y%m%d"`-`hostname`-asterisk-
↪full.log
```

5. Fetch xivo-ctid logs for the day of the crash (make sure file was not already logrotated):

```
cp -a /var/log/xivo-ctid.log /var/local/`date +"%Y%m%d"`-`hostname`-xivo-ctid.
↪log
```

10.2.3 Debugging Asterisk Crash

When asterisk crashes, it usually leaves a core file in `/var/spool/asterisk/`.

You can create a backtrace from a core file named `core_file` with:

```
gdb -batch -ex "bt full" -ex "thread apply all bt" asterisk core_file > bt-threads.
↪txt
```

10.2.4 Debugging Asterisk Freeze

You can create a backtrace of a running asterisk process with:

```
gdb -batch -ex "thread apply all bt" asterisk $(pidof asterisk) > bt-threads.txt
```

If your version of asterisk has been compiled with the DEBUG_THREADS flag, you can get more information about locks with:

```
asterisk -rx "core show locks" > core-show-locks.txt
```

Note: Debugging freeze without this information is usually a lot more difficult.

Optionally, other information that can be interesting:

- the output of `asterisk -rx 'core show channels'`
- the verbose log of asterisk just before the freeze

10.2.5 Recompiling Asterisk

It's relatively straightforward to recompile the asterisk version of your XiVO with the DEBUG_THREADS and DONT_OPTIMIZE flag, which make debugging an asterisk problem easier.

The steps are:

1. Uncomment the `deb-src` line for the XiVO sources:

```
sed -i 's/^# *deb-src/deb-src/' /etc/apt/sources.list.d/xivo*
```

2. Fetch the asterisk source package:

```
mkdir -p ~/ast-rebuild
cd ~/ast-rebuild
apt-get update
apt-get install -y build-essential
apt-get source asterisk
```

3. Install the build dependencies:

```
apt-get build-dep -y asterisk
```

4. Enable the DEBUG_THREADS and DONT_OPTIMIZE flag:

```
cd <asterisk-source-folder>
vim debian/rules
```

5. Update the changelog by appending `+debug1` in the package version:

```
vim debian/changelog
```

6. Rebuild the asterisk binary packages:

```
dpkg-buildpackage -us -uc
```

This will create a couple of .deb files in the parent directory, which you can install via dpkg.

Recompiling a vanilla version of Asterisk

It is sometimes useful to produce a "vanilla" version of Asterisk, i.e. a version of Asterisk that has none of the XiVO patches applied, to make sure that the problem is present in the original upstream code. This is also sometimes necessary before opening a ticket on the Asterisk issue tracker.

The procedure is similar to the one described above. Before calling `dpkg-buildpackage`, you just need to:

1. Make sure `quilt` is installed:

```
apt-get install -y quilt
```

2. Unapply all the currently applied patches:

```
quilt pop -a
```

3. Remove all the lines in the `debian/patches/series` file:

```
truncate -s0 debian/patches/series
```

When installing a vanilla version of Asterisk on a XiVO 16.08 or earlier, you'll need to stop monit, otherwise it will restart asterisk every few minutes.

10.2.6 Running Asterisk under Valgrind

1. Install valgrind:

```
apt-get install valgrind
```

2. Recompile asterisk with the DONT_OPTIMIZE flag.
3. Edit `/etc/asterisk/modules.conf` so that asterisk doesn't load unnecessary modules. This step is optional. It makes asterisk start (noticeably) faster and often makes the output of valgrind easier to analyze, since there's less noise.
4. Edit `/etc/asterisk/asterisk.conf` and comment the `highpriority` option. This step is optional.
5. Stop monit and asterisk:

```
monit quit
service asterisk stop
```

6. Stop all unneeded XiVO services. For example, it can be useful to stop xivo-ctid, so that it won't interact with asterisk via the AMI.
7. Copy the valgrind.supp file into /tmp. The valgrind.supp file is located in the contrib directory of the asterisk source code.
8. Execute valgrind in the /tmp directory:

```
cd /tmp
valgrind --leak-check=full --log-file=valgrind.txt --suppressions=valgrind.
↪supp --vgdb=no asterisk -G asterisk -U asterisk -fnc
```

Note that when you terminate asterisk with Control-C, asterisk does not unload the modules before exiting. What this means is that you might have lots of "possibly lost" memory errors due to that. If you already know which modules is responsible for the memory leak/bug, you should explicitly unload it before terminating asterisk.

Running asterisk under valgrind takes a lots of extra memory, so make sure you have enough RAM.

10.2.7 External links

- https://wiki.asterisk.org/wiki/display/AST/Debugging
- https://gitlab.com/xivo.solutions/xivo-blog/blob/master/content/articles/visualizing-asterisk-deadlocks.md
- https://wiki.asterisk.org/wiki/display/AST/Valgrind

10.3 Debugging Daemons

To activate debug mode, add `debug: true` in the daemon *configuration file*). The output will be available in the daemon's *log file*.

It is also possible to run the XiVO daemon, in command line. This will allow to run in foreground and debug mode. To see how to use it, type:

```
xivo-{name} -h
```

Note that it's usually a good idea to stop monit before running a daemon in foreground:

```
systemctl stop monit.service
```

10.3.1 xivo-confgend

```
twistd -no -u xivo-confgend -g xivo-confgend --python=/usr/bin/xivo-confgend --
↪logger xivo_confgen.bin.daemon.twistd_logs
```

No debug mode in confgend.

10.3.2 xivo-provd

```
twistd -no -u xivo-provd -g xivo-provd -r epoll --logger provd.main.twistd_logs␣
↪xivo-provd -s -v
```

- -s for logging to stderr
- -v for verbose

10.3.3 consul

```
sudo -u consul /usr/bin/consul agent -config-dir /etc/consul/xivo -pid-file /var/
↪run/consul/consul.pid
```

There is no log file, but you can consult the output of consul with:

```
consul monitor
```

```
2015/08/03 09:48:25 [INFO] consul: cluster leadership acquired
2015/08/03 09:48:25 [INFO] consul: New leader elected: this-xivo
2015/08/03 09:48:26 [INFO] raft: Disabling EnableSingleNode (bootstrap)
2015/08/03 11:04:08 [INFO] agent.rpc: Accepted client: 127.0.0.1:41545
```

10.4 Generate your own prompts

If you want your XiVO to speak in your language that is not supported by XiVO, and you don't want to record the whole package of sounds in a studio, you may generate them yourself with some text-to-speech services.

The following procedure will generate prompts for `pt_BR` (portuguese from Brazil) based on the Google TTS service.

Note: There are two sets of prompts: the Asterisk prompts and the XiVO prompts. This procedure only covers the XiVO prompts, but it may be adapted for Asterisk prompts.

---

1. Create an account on Transifex and join the team of translation of XiVO.
2. Translate the prompts in the xivo-prompts resource.
3. Go to https://www.transifex.com/proformatique/xivo/xivo-prompt/pt_BR/download/for_use/ and download the file on your XiVO. You should have a file named like `for_use_xivo_xivo-prompt_pt_BR.ini`.
4. On your XiVO, download the tool to automate the use of Google TTS:

   ```
   wget https://github.com/zaf/asterisk-googletts/raw/master/cli/googletts-cli.pl
   chmod +x googletts-cli.pl
   ```

5. Then run the tool, and generate the sound files (set `LANGUAGE` and `COUNTRY` to your own language):

   ```
   LANGUAGE=pt
   COUNTRY=BR
   mkdir -p wav/{digits,letters}
   cat for_use_xivo_xivo-prompt_${LANGUAGE}_${COUNTRY}.ini | while IFS='=' read␣
   ↪file text ; do
     echo $file
     ./googletts-cli.pl -t "$text" -l ${LANGUAGE}-${COUNTRY} -s 1.4 -r 8000 -o␣
   ↪wav/$file.wav
   done
   ```

6. Install the prompts on your system:

   ```
   mv wav /usr/share/asterisk/sounds/${LANGUAGE}_${COUNTRY}
   ```

7. Make your language available in the web interface:

   ```
   sed -i "s/'nl_NL'/\0, '${LANGUAGE}_${COUNTRY}'/" /usr/share/xivo-web-interface/
   ↪lib/i18n.inc
   systemctl restart spawn-fcgi
   ```

Note that this last modification may be erased after running `xivo-upgrade`.

And that's it, you can configure a user to use your new language and he will hear the prompts in your language. You may also want to use the *xivo-confd HTTP API* to mass-update your users.

10.5 XiVO Guidelines

10.5.1 Inter-process communication

Our current goal is to use only two means of communication between XiVO processes:

- a REST API over HTTP for synchronous commands
- a software bus (RabbitMQ) for asynchronous events

Each component should have its own REST API and its own events and can communicate with every other component from across a network only via those means.

10.5.2 Service API

The current xivo-dao Git repository contains the basis of the future services Python API. The API is split between different resources available in XiVO, such as users, groups, schedules... For each resource, there are different modules :

- service: the public module, providing possible actions. It contains only business logic and no technical logic. There must be no file name, no SQL queries and no URLs in this module.
- dao: the private Data Access Object. It knows where to get data and how to update it, such as SQL queries, file names, URLs, but has no business logic.
- model: the public class used to represent the resource. It must be self-contained and have almost no methods, except for computed fields based on other fields in the same object.
- notifier: private, it knows to whom and in which format events must be sent.
- validator: private, it checks input parameters from the service module.

10.5.3 Definition of XiVO Daemon

The goal is to make XiVO as elastic as possible, i.e. the XiVO services need to be able to run on separate machines and still talk to each other.

To be in accordance with our goal, a XiVO daemon must (if applicable):

- Offer a REST API (with encryption, authentication and accepting cross-site requests)
- Be able to read and send events on a software bus
- Be able to run inside a container, such as Docker, and be separated from the XiVO server
- Offer a configuration file in YAML format.
- Access the XiVO database through the `xivo-dao` library
- Have a configurable level of logging
- Have its own log file
- Be extendable through the use of plugins
- Not run with system privileges
- Be installable from source
- Service discovery with consul

Currently, none of the XiVO daemons meet these expectations; it is a work in progress.

10.6 Network

10.6.1 Configuration for daemon

Network Flow table (IN) :

Daemon Name	Service	Protocol	Port	Listen	Authentication	Enabled
-	ICMP	ICMP	-	0.0.0.0	no	yes
postfix	SMTP	TCP	25	0.0.0.0	yes	yes
isc-dhcpd	DHCP	UDP	67	0.0.0.0	no	no
isc-dhcpd	DHCP	UDP	68	0.0.0.0	no	no
xivo-provd	TFTP	UDP	69	0.0.0.0	no	yes
ntpd	NTP	UDP	123	0.0.0.0	yes	yes
monit	HTTP	TCP	2812	127.0.0.1	no	yes
asterisk	SIP	UDP	5060	0.0.0.0	yes	yes
asterisk	IAX	UDP	4569	0.0.0.0	yes	yes
asterisk	SCCP	TCP	2000	0.0.0.0	yes	yes
asterisk	AMI	TCP	5038	127.0.0.1	yes	yes
asterisk	HTTP	TCP	5039	127.0.0.1	yes	yes
asterisk	HTTPS	TCP	5040	127.0.0.1	yes	yes
sshd	SSH	TCP	22	0.0.0.0	yes	yes
nginx	HTTP	TCP	80	0.0.0.0	yes	yes
nginx	HTTPS	TCP	443	0.0.0.0	yes	yes
munin	HTTP	TCP	4949	127.0.0.1	no	yes
xivo-ctid	XiVO-CTI/S	TCP	5003	0.0.0.0	yes	yes
postgresql	SQL	TCP	5432	127.0.0.1	yes	yes
rabbitMQ	AMQP	TCP	5672	0.0.0.0	yes	yes
consul	Consul RPC	TCP	8300	127.0.0.1	yes	yes
consul	Consul Serf LAN	TCP/UDP	8301	127.0.0.1	yes	yes
consul	Consul Serf WAN	TCP/UDP	8302	127.0.0.1	yes	yes
consul	Consul HTTPS	TCP	8500	127.0.0.1	both	yes
xivo-provd	HTTP	TCP	8666	127.0.0.1	no	yes
xivo-provd	HTTP	TCP	8667	0.0.0.0	no	yes
xivo-confgend	HTTP	TCP	8669	127.0.0.1	no	yes
xivo-sysconfd	HTTP	TCP	8668	127.0.0.1	no	yes
xivo-confd	HTTPS	TCP	9486	0.0.0.0	yes	yes
xivo-confd	HTTP	TCP	9487	127.0.0.1	no	yes
xivo-dird	HTTPS	TCP	9489	0.0.0.0	yes	yes
xivo-amid	HTTPS	TCP	9491	0.0.0.0	yes	yes
xivo-agentd	HTTPS	TCP	9493	0.0.0.0	yes	yes
xivo-ctid	HTTP	TCP	9495	127.0.0.1	no	yes
xivo-auth	HTTPS	TCP	9497	0.0.0.0	both	yes
xivo-dird-phoned	HTTP	TCP	9498	0.0.0.0	IP filtering	yes
xivo-dird-phoned	HTTPS	TCP	9499	0.0.0.0	IP filtering	yes

10.7 Debian packaging for XiVO

10.7.1 Adding a package from backports

1. Download the package:

```
apt-get download name-of-package/jessie-backports
```

2. Copy the .deb on to the mirror:

```
scp name-of-package.deb mirror.xivo.solutions:/tmp
```

3. Add package to distribution on mirror:

```
ssh mirror.xivo.solutions
cd /data/reprepro/xivo
reprepro includedeb xivo-dev /tmp/name-of-package.deb
```

10.8 Profiling Python Programs

10.8.1 Profiling CPU/Time Usage

Here's an example on how to profile xivo-ctid for CPU/time usage:

1. Stop the monit daemon:

```
service monit stop
```

2. Stop the process you want to profile, i.e. xivo-ctid:

```
service xivo-ctid stop
```

3. Start the service in foreground mode running with the profiler:

```
python -m cProfile -o test.profile /usr/bin/xivo-ctid -f
```

This will create a file named `test.profile` when the process terminates.

To profile xivo-confgend, you must use this command instead of the one above:

```
twistd -p test.profile --profiler=cprofile --savestats -no --python=/usr/bin/
↪xivo-confgend
```

Note that profiling multi-threaded program (xivo-agid, xivo-confd) doesn't work reliably.

The *Debugging Daemons* section documents how to launch the various XiVO services in foreground/debug mode.

4. Examine the result of the profiling:

```
$ python -m pstats test.profile
Welcome to the profile statistics browser.
% sort time
% stats 15
...
% sort cumulative
% stats 15
```

10.8.2 Measuring Code Coverage

Here's an example on how to measure the code coverage of xivo-ctid.

This can be useful when you suspect a piece of code to be unused and you want to have additional information about it.

1. Install the following packages:

```
apt-get install python-pip build-essential python-dev
```

2. Install coverage via pip:

```
pip install coverage
```

3. Run the program in foreground mode with `coverage run`:

```
service monit stop
service xivo-ctid stop
coverage erase
coverage run /usr/bin/xivo-ctid -f
```

The *Debugging Daemons* section documents how to launch the various XiVO service in foreground/debug mode.

4. After the process terminates, use `coverage html` to generate an HTML coverage report:

```
coverage html --include='*xivo_cti*'
```

This will generate an `htlmcov` directory in the current directory.

5. Browse the coverage report.

Either copy the directory onto your computer and open it with a web browser, or start a web server on the XiVO:

```
cd htmlcov
python -m SimpleHTTPServer
```

Then open the page from your computer (i.e. not on the xivo):

```
firefox http://<xivo-hostname>:8000
```

10.8.3 External Links

- Official python documentation
- PyMOTW
- coverage.py

10.9 Style Guide

10.9.1 Syntax

License

Python files start with a UTF8 encoding comment and the GPLv3 license. A blank line should separate the license from the imports

Example:

```
# -*- coding: utf-8 -*-
# Copyright 2016 Avencall
# SPDX-License-Identifier: GPL-3.0+

import argparse
```

Spacing

- Lines should not go further than 80 to 100 characters.
- In python, indentation blocks use 4 spaces
- In PHP, indentation blocks use tabs
- Imports should be ordered alphabetically
- Separate module imports and `from` imports with a blank line

Example:

```
import argparse
import datetime
import os
import re
import shutil
import tempfile

from StringIO import StringIO
from urllib import urlencode
```

PEP8

When possible, use pep8 to validate your code. Generally, the following errors are ignored :

- E501 (max 80 chars per line)

Example:

```
pep8 --ignore=E501 xivo_cti
```

When possible, avoid using backslashes to separate lines.

Bad Example:

```
user = session.query(User).filter(User.firstname == firstname)\
                          .filter(User.lastname == lastname)\
                          .filter(User.number == number)\
                          .all()
```

Good Example:

```
user = (session.query(User).filter(User.firstname == firstname)
                           .filter(User.lastname == lastname)
                           .filter(User.number == number)
                           .all())
```

Strings

Avoid using the + operator for concatenating strings. Use string interpolation instead.

Bad Example:

```
phone_interface = "SIP" + "/" + username + "-" + password
```

Good Example:

```
phone_interface = "SIP/%s-%s" % (username, password)
```

Comments

Redundant comments should be avoided. Instead, effort should be put on making the code clearer.

Bad Example:

```
#Add the meeting to the calendar only if it was created on a week day
#(monday to friday)
if meeting.day > 0 and meeting.day < 7:
    calendar.add(meeting)
```

Good Example:

```
def created_on_week_day(meeting):
    return meeting.day > 0 and meeting.day < 7

if created_on_week_day(meeting):
    calendar.add(meeting)
```

Conditions

Avoid using parenthesis around if statements, unless the statement expands on multiple lines or you need to nest your conditions.

Bad Examples:

```
if(x == 3):
    print "condition is true"

if(x == 3 and y == 4):
    print "condition is true"
```

Good Examples:

```
if x == 3:
    print "condition is true"

if x == 3 and y == 4:
    print "condition is true"

if (extremely_long_variable == 3
    and another_long_variable == 4
    and yet_another_variable == 5):

    print "condition is true"

if (2 + 3 + 4) - (1 + 1 + 1) == 6:
    print "condition is true"
```

Consider refactoring your statement into a function if it becomes too long, or the meaning isn't clear.

Bad Example:

```
if price * tax - bonus / reduction + fee < money:
    product.pay(money)
```

Good Example:

```
def calculate_price(price, tax, bonus, reduction, fee):
    return price * tax - bonus / reduction + fee

final_price = calculate_price(price, tax, bonus, reduction, fee)
```

(continues on next page)

(continued from previous page)

```
if final_price < money:
    product.pay(money)
```

10.9.2 Naming

- Class names are in `CamelCase`
- File names are in `lower_underscore_case`

Conventions for functions prefixed by *find*:

- Return None when nothing is found
- Return an object when a single entity is found
- Return the first element when multiple entities are found

Example:

```
def find_by_username(username):
    users = [user1, user2, user3]
    user_search = [user for user in users if user.username == username]

    if len(user_search) == 0:
        return None

    return user_search[0]
```

Conventions for functions prefixed by *get*:

- Raise an Exception when nothing is found
- Return an object when a single entity is found
- Return the first element when multiple entities are found

Example:

```
def get_user(userid):
    users = [user1, user2, user3]
    user_search = [user for user in users if user.userid == userid]

    if len(user_search) == 0:
        raise UserNotFoundError(userid)

    return user_search[0]
```

Conventions for functions prefixed by *find_all*:

- Return an empty list when nothing is found
- Return a list of objects when multiple entites are found

Example:

```
def find_all_users_by_username(username):
    users = [user1, user2, user3]
    user_search = [user for user in users if user.username == username]

    return user_search
```

Magic numbers

Magic numbers should be avoided. Arbitrary values should be assigned to variables with a clear name

Bad example:

```
class TestRanking(unittest.TestCase):

    def test_ranking(self):
        rank = Rank(1, 2, 3)

        self.assertEquals(rank.position, 1)
        self.assertEquals(rank.grade, 2)
        self.assertEquals(rank.session, 3)
```

Good example:

```
class TestRanking(unittest.TestCase):

    def test_ranking(self):
        position = 1
        grade = 2
        session = 3

        rank = Rank(position, grade, session)

        self.assertEquals(rank.position, position)
        self.assertEquals(rank.grade, grade)
        self.assertEquals(rank.session, session)
```

10.9.3 Tests

Tests for a package are placed in their own folder named “tests” inside the package.

Example:

```
package1/
__init__.py
mod1.py
tests/
    __init__.py
    test_mod1.py
package2/
__init__.py
mod9.py
tests/
    __init__.py
    test_mod9.py
```

Unit tests should be short, clear and concise in order to make the test easy to understand. A unit test is separated into 3 sections :

- Preconditions / Preparations
- Thing to test
- Assertions

Sections are separated by a blank line. Sections that become too big should be split into smaller functions.

Example:

```
class UserTestCase(unittest.TestCase):

    def test_fullname(self):
        user = User(firstname='Bob', lastname='Marley')
        expected = 'Bob Marley'

        fullname = user.fullname()

        self.assertEquals(expected, fullname)

    def _prepare_expected_user(self, firstname, lastname, number):
        user = User()
        user.firstname = firstname
        user.lastname = lastname
        user.number = number

        return user

    def _assert_users_are_equal(expected_user, actual_user):
        self.assertEquals(expected_user.firstname, actual_user.firstname)
        self.assertEquals(expected_user.lastname, actual_user.lastname)
        self.assertEquals(expected_user.number, actual_user.number)

    def test_create_user(self):
        expected = self._prepare_expected_user('Bob', 'Marley', '4185551234')

        user = create_user('Bob', 'Marley', '4185551234')

        self._assert_users_are_equal(expected, user)
```

10.9.4 Exceptions

Exceptions should not be used for flow control. Raise exceptions only for edge cases, or when something that isn't usually expected happens.

Bad Example:

```
def is_user_available(user):
    if user.available():
        return True
    else:
        raise Exception("User isn't available")

try:
    is_user_available(user)
except Exception:
    disable_user(user)
```

Good Example:

```
def is_user_available(user):
    if user.available():
        return True
    else:
        return False


if not is_user_available(user):
    disable_user(user)
```

Avoid throwing `Exception`. Use one of Python's built-in Exceptions, or create your own custom Exception. A

list of exceptions is available on the Python documentation website.

Bad Example:

```
def get_user(userid):
    user = session.query(User).get(userid)

    if not user:
        raise Exception("User not found")
```

Good Example:

```
class UserNotFoundError(LookupError):

    def __init__(self, userid):
        message = "user with id %s not found" % userid
        LookupError.__init__(self, message)

def get_user(userid):
    user = session.query(User).get(userid)

    if not user:
        raise UserNotFoundError(userid)
```

Never use `except:` without specifying any exception type. The reason is that it will also catch important exceptions, such as `KeyboardInterrupt` and `OutOfMemory` exceptions, making your program unstoppable or continuously failing, instead of stopping when wanted.

Bad Example:

```
try:
    get_user(user_id)
except:
    logger.exception("There was an error")
```

Good Example:

```
try:
    get_user(user_id)
except UserNotFoundError as e:
    logger.error(e.message)
    raise
```

10.10 Translating XiVO

French and English are maintained by Avencall. Other languages are provided by the community.

10.10.1 Asterisk and XiVO Prompts

Avencall is in contact with several studios for different languages and prompts. The information for those languages are :

- French : Super Sonic productions (supersonicprod@wanadoo.fr)
- English : Asterisk voice (allison@theasteriskvoice.com)
- German : ATS studio
- Italian : ATS studio

Prompts transcripts are listed in Transifex (*-prompts). You may translate them there.

The prompts used in XiVO are stored in xivo-sounds git repository. You may also want to *generate your own sound files*.

10.10.2 XiVO Client

All translations are in Transifex (xivo-client). The source language is English. Translations are synchronised with the code before every release.

10.10.3 Web Interface

Translations are currently available in French and English. There are no plans to translate the Web interface in other languages.

10.11 XiVO Package File Structure

10.11.1 Package naming

Let's assume we want to organise the files for xivo-confd.

- Git repo name: `xivo-confd`
- Binary file name: `xivo-confd`
- Python package name: `xivo_confd`

```
xivo-confd
|-- bin
|   `-- xivo-confd
|-- contribs
|   `-- docker
|       |-- ...
|       `-- prod
|           `-- ...
|-- debian
|   `-- ...
|-- Dockerfile
|-- docs
|   `-- ...
|-- etc
|   `-- ...
|-- integration-tests
|   `-- ...
|-- LICENSE
|-- README.md
|-- requirements.txt
|-- setup.cfg
|-- setup.py
|-- test-requirements.txt
|-- .travis.yml
`-- xivo_confd
    `-- ...
```

Sources

`etc/` Contains default configuration files.

docs/ Contains technical documentation for this package: API doc, architecture doc, diagrams, ... Should be in RST format using Sphinx.

bin/ Contains the binaries. Not applicable for pure libraries.

integration-tests/ Contains the tests bigger than unit-tests. Tests should be runnable simply, e.g. `nosetests integration-tests`.

README.md Read me in markdown (Github flavor).

LICENSE License (GPLv3)

.travis.yml Travis CI configuration file

Python

Standard files:

- setup.py
- setup.cfg
- requirements.txt
- test-requirements.txt
- xivo_confd/ (the main sources)

Debian

debian/ Contains the Debian packaging files (`control`, `rules`, ...)

Docker

Dockerfile Used to build a docker image for a working production version

contribs/docker/prod/ Contains the files necessary for running xivo-confd inside a production Docker image

contribs/docker/other/ Contains the Dockerfile and other files to run xivo-confd inside Docker with specific configuration

10.11.2 File naming

- PID file: `/var/run/xivo-confd/xivo-confd.pid`
- WSGI socket file: `/var/run/xivo-confd/xivo-confd.sock`
- Config file: `/etc/xivo-confd/config.yml`
- Log file: `/var/log/xivo-confd.log`
- Static data files: `/usr/share/xivo-confd`
- Storage data files: `/var/lib/xivo-confd`

Component specific information:

10.12 CTI Server

This section describes the informations and tools for CTI Server.

10.12.1 CTI Proxy

Here's how to run the various CTI client-server development/debugging tools. These tools can be found on GitLab, in the XiVO project.

You can get the scripts by using Git:

```
$ git clone https://gitlab.com/xivo.solutions/xivo-tools.git
```

General Information

Both the ctispy, ctisave and ctistat tools work in a similar way. They both are proxies that need to be inserted between the CTI client and the CTI server message flow.

To do this, you first start the given tool on your development machine, giving it the CTI server hostname as the first argument. You then configure your CTI client to connect to the tool on port 50030 (notice the trailing 0). The tool should then accept the connection from the client, and once this is done, will make a connection to the server, thereby being able to process all the information sent between the client and the server.

In the following examples, we suppose that the CTI server is located on the host named xivo-new.

Tools

ctispy

`ctispy` can be used to see the message flow between the client and the server in "real-time".

The simplest invocation is:

```
$ cti-proxy/ctispy xivo-new
```

You can pretty-print the messages if you want by using the `--pretty-print` option:

```
$ cti-proxy/ctispy xivo-new --pretty-print
```

By default, each message is displayed separately even though more than one message can be in a single TCP packet. You can also use the `--raw` option if you want to see the raw traffic between the client and the server:

```
$ cti-proxy/ctispy xivo-new --raw
```

Note that when using the `--raw` option, some other option doesn't work because the messages are not decoded/analyzed.

If you want to remove some fields from the messages, you can use the `--strip` option:

```
$ cti-proxy/ctispy xivo-new --strip timenow --strip commandid --strip replyid
```

If you want to see only messages matching a certain key and value, use the `--include` option:

```
$ cti-proxy/ctispy xivo-new --include class=getlist
```

Finally, you can ignore all the messages from the client or the server by using the `--no-client` or `--no-server` option respectively.

By default, ctispy will exit after the connection with the client is closed. You can bypass this behavior with the `--loop` option, that will make the CTI proxy continue, whether the client is connected or not.

ctisave

`ctisave` save the messages from the client and the server in two separate files. This is useful to do more careful post-analysis.

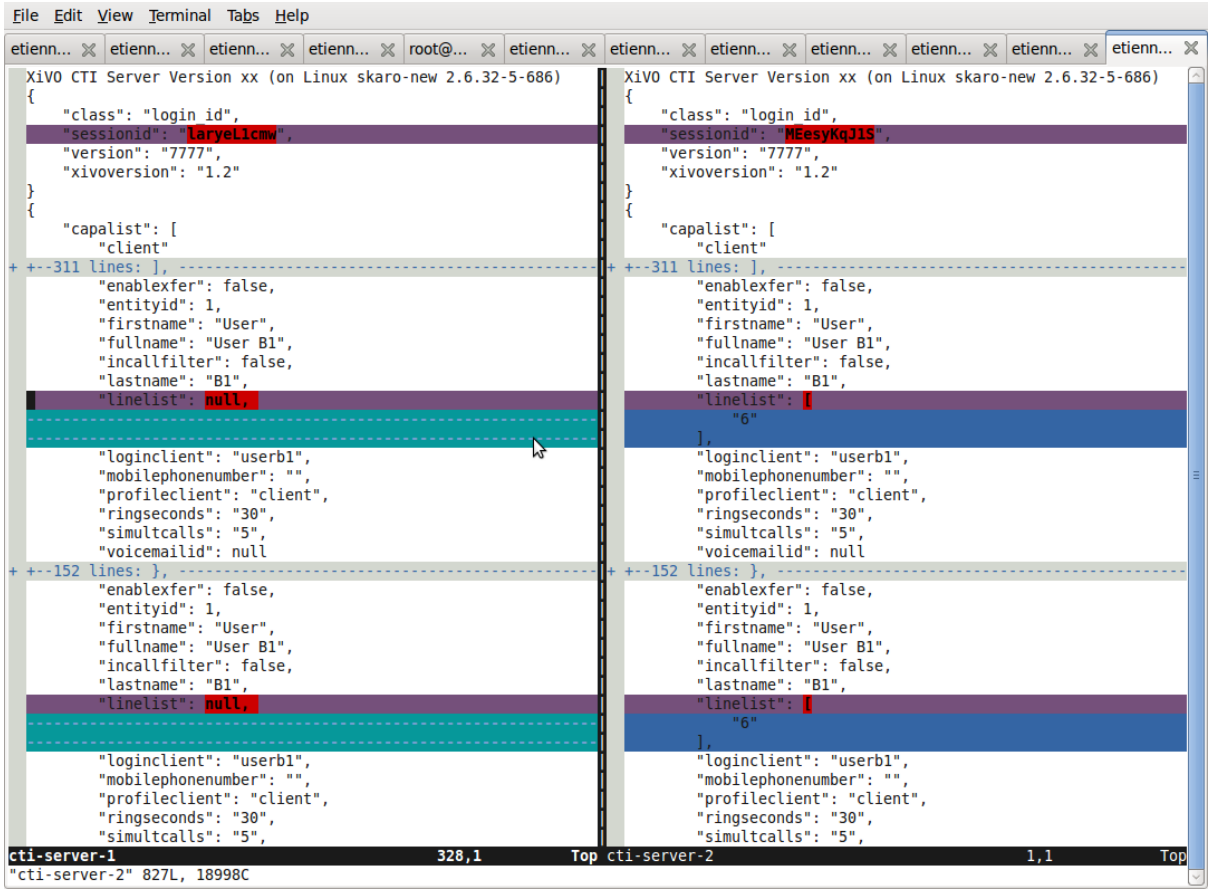
The simplest invocation is:

```
$ cti-proxy/ctisave xivo-new /tmp/cti-client /tmp/cti-server
```

To do comparison, it's often useful to strip some fields:

```
$ cti-proxy/ctisave xivo-new /tmp/cti-client /tmp/cti-server --strip timenow
--strip commandid --strip replyid
```

One useful thing to do with files generated from different ctisave invocation is to compare them with a tool like vimdiff, for example:



ctistat

`ctistat` display various statistic about a CTI "session" when it ends.

The simplest invocation is:

```
$ cti-proxy/ctistat xivo-new
```

10.12.2 CTI Protocol

Protocol Changelog

The versions below indicate the xivo version followed by the protocol version.

Warning: The CTI server protocol is subject to change without any prior warning. If you are using this protocol in your own tools please be sure to check that the protocol did not change before upgrading XiVO

16.11 - 2.2

- the *user_id* field has been added back to the *User status update*

16.09 - 2.2

- the *Register user status update* now uses the *user_uuid* instead of the *user_id*
- the *User status update* now uses the *user_uuid* instead of the *user_id*

16.04 - 2.1

- the *Chitchat* command *to* and *from* fields are now a list of two strings, *xivo_uuid* and *user_uuid*.

16.01 - 2.0

- the *lastconnswins* field has been removed from the *Login capas* command
- the *loginkind* field has been removed from the *Login capas* command
- the *ipbxcommands* and *regcommands* capakinds have been removed from *Login capas* command
- the *Login password* command has been modified. The *hashedpassword* has been replaced by the *password* field which is now sent verbatim.

15.20 - 1.2

- the *STARTTLS* command has been added

15.19 - 1.2

- the *Chitchat* command *to* field is now a list of two elements, *xivo_uuid* and *user_id*.
- the `getlist` command has been removed for the *channels* listname.
- many fields have been removed from the `getlist` command.
  - users list
    - enableclient
    - profileclient
  - phones
    - context
    - protocol
    - simultcalls

        * channels
    - voicemails
        * email
        * fullname
        * old
        * waiting
    - agents
        * phonenumber
- some ipbxcommands have been removed:
    - mailboxcount
    - atxfer
    - transfer
    - hangup
    - originate

15.18 - 1.2

- add the *Attended transfer to voicemail* command
- add the *Blind transfer to voicemail* command
- the *Send fax* command now include the size and data field.
- the *filetransfer* command has been removed.

15.16 - 1.2

- the *Get relations* command was added.
- the *Relations* message was added.

15.14 - 1.2

- the `people_purge_personal_contacts` message was added.
- the `people_personal_contacts_purged` message was added.
- the `people_personal_contact_raw` message was added.
- the `people_personal_contact_raw_result` message was added.
- the `people_edit_personal_contact` message was added.
- the `people_personal_contact_raw_update` message was added.
- the `people_import_personal_contacts_csv` message was added.
- the `people_import_personal_contacts_csv_result` message was added.
- the `people_export_personal_contacts_csv` message was added.
- the `people_export_personal_contacts_csv_result` message was added.
- for messages `people_personal_contact_deleted` and `people_favorite_update` there are no longer `data` sub-key.

15.13 - 1.2

- for `channel status update` message:
  - the value of `commstatus` have been changed from `linked-caller` and `linked-called` to `linked`.
  - the key `direction` have been removed.
  - the key `talkingto_kind` have been removed.
- the `people_personal_contacts` message was added.
- the `people_personal_contacts_result` message was added.
- the `people_create_personal_contact` message was added.
- the `people_personal_contact_created` message was added.
- the `people_delete_personal_contact` message was added.
- the `people_personal_contact_deleted` message was added.

15.12 - 1.2

- `people_search_result` has a new key in `relations`: `source_entry_id`
- the `people_favorites` message was added.
- the `people_favorites_result` message was added.
- the `people_set_favorite` message was added.
- the `people_favorite_update` message was added.

15.11 - 1.2

- the `fax_progress` message was added.

15.09 - 1.2

- for messages of class `history` the client cannot request by mode anymore. The server returns all calls and the mode is now metadata for each call.

14.24 - 1.2

- for messages of class `ipbxcommand`, the command `record` and `sipnotify` have been removed.
- the `logfromclient` message has been removed

14.22 - 1.2

- for messages of class `faxsend`, the steps `file_decoded` and `file_converted` have been removed.

14.06 - 1.2

- the `dial_success` message was added

14.05 - 1.2

- the `unhold_switchboard` command was renamed `resume_switchboard`.

13.22 - 1.2

- the `actionfiche` message was renamed `call_form_result`.

13.17 - 1.2

- for messages of class `login_capas` from server to client: the key `presence` has been removed.

13.14 - 1.2

- for messages of class `getlist`, list `agents` and function `updatestatus`: the key `availability` in the `status` object/dictionary has changed values:
  - deleted values: `on_call_non_acd_incoming` and `on_call_non_acd_outgoing`
  - added values: * `on_call_non_acd_incoming_internal` * `on_call_non_acd_incoming_external` * `on_call_non_acd_outgoing_internal` * `on_call_non_acd_outgoing_external`

13.12 - 1.2

- for messages of class `getlist`, list `agents` and function `updatestatus`: the key `availability` in the `status` object/dictionary has changed values:
  - deleted value: `on_call_non_acd`
  - added values: `on_call_non_acd_incoming` and `on_call_non_acd_outgoing`

13.10 - 1.2

- for messages of class `getlist` and function `updateconfig`, the `config` object/dictionary does not have a `rules_order` key anymore.

Commands

Objects have the format: "<type>:<xivoid>/<typeid>"

- <type> can take any of the following values: user, agent, queue, phone, group, meetme, ...
- <xivoid> indicates on which server the object is defined
- <typeid> is the object id, type dependant

e.g. user:xivo-test/5 I'm looking for the user that has the ID 5 on the xivo-test server.

Here is a non exaustive list of types:

- exten
- user
- vm_consult
- voicemail

Agent

Login agent

```
Client -> Server
```

```
{"agentphonenumber": "1000", "class": "ipbxcommand", "command": "agentlogin",
↪"commandid": 733366597}
```

agentphonenumber is the physical phone set where the agent is going to log on.

```
Server > Client
```

- Login successfull :

```
{"function": "updateconfig",
 "listname": "queuemembers",
 "tipbxid": "xivo",
 "timenow": 1362664323.94,
 "tid": "Agent/2002,blue",
 "config": {"paused": "0",
            "penalty": "0",
            "membership": "static",
            "status": "1",
            "lastcall": "",
            "interface": "Agent/2002",
            "queue_name": "blue",
            "callstaken": "0"},
 "class": "getlist"}

{"function": "updatestatus",
 "listname": "agents",
 "tipbxid": "xivo",
 "timenow": 1362664323.94,
 "status": {"availability_since": 1362664323.94,
            "queues": [],
            "on_call": false,
            "availability": "available",
            "channel": null},
   "tid": 7,
   "class": "getlist"}
```

- The phone number is already used by an other agent :

```
{"class": "ipbxcommand", "error_string": "agent_login_exten_in_use", "timenow":␣
↪1362664158.14}
```

Logout agent

```
Client -> Server
```

```
{"class": "ipbxcommand", "command": "agentlogout", "commandid": 552759274}
```

Pause

On all queues

```
Client -> Server
```

```
{"class": "ipbxcommand", "command": "queuepause", "commandid": 859140432, "member
↪": "agent:xivo/1", "queue": "queue:xivo/all"}
```

Un pause agent

On all queues

```
Client -> Server
```

```
{"class": "ipbxcommand", "command": "queueunpause", "commandid": 822604987, "member
↪": "agent:xivo/1", "queue": "queue:xivo/all"}
```

Add an agent in a queue

```
Client -> Server
```

```
{"class": "ipbxcommand", "command": "queueadd", "commandid": 542766213, "member":
↪"agent:xivo/3", "queue": "queue:xivo/2"}
```

Remove an agent from a queue

```
Client -> Server
```

```
{"class": "ipbxcommand", "command": "queueremove", "commandid": 742480296, "member
↪": "agent:xivo/3", "queue": "queue:xivo/2"}
```

Listen to an agent

```
Client -> Server
```

```
{"class": "ipbxcommand", "command": "listen", "commandid": 1423579492, "destination
↪": "xivo/1", "subcommand": "start"}
```

Configuration

The following messages are used to retrieve XiVO configuration.

Common fields

- class : getlist
- function : listid
- commandid
- tipbxid
- listname : Name of the list to be retreived : users, phones, agents, queues, voicemails, queuemembers

```
{
   "class": "getlist",
   "commandid": 489035169,
   "function": "listid",
```

(continues on next page)

(continued from previous page)

```
   "tipbxid": "xivo",
   "listname": "........."
}
```

Users configuration

Return a list of configured user id's

`Client -> Server`

```
{"class": "getlist", "commandid": 489035169, "function": "listid", "listname":
↪"users", "tipbxid": "xivo"}
```

`Server -> Client`

```
{
   "class": "getlist",
   "function": "listid", "listname": "users",
   "list": ["11", "12", "14", "17", "1", "3", "2", "4", "9"],
   "tipbxid": "xivo","timenow": 1362735061.17
   }
```

User configuration

Return a user configuration

- tid is the userid returned by *Users configuration* message

`Client -> Server`

```
{
  "class": "getlist",
  "function": "updateconfig",
  "listname": "users",
  "tid": "17",
  "tpbxid": "xivo",  "commandid": 5}
```

`Server -> Client`

```
{
   "class": "getlist",
   "function": "updateconfig",
   "listname": "users",
   "tid": "17",
   "tipbxid": "xivo",
   "timenow": 1362741166.4,
   "config": {
         "enablednd": 0, "destrna": "", "enablerna": 0,  "enableunc": 0, "destunc
↪": "", "destbusy": "", "enablebusy": 0, "enablexfer": 1,
         "firstname": "Alice",  "lastname": "Bouzat", "fullname": "Alice Bouzat",
         "voicemailid": null, "incallfilter": 0,  "enablevoicemail": 0,   "agentid
↪": 2, "linelist": ["7"], "mobilephonenumber": ""}
    }
```

Phones configuration

`Client -> Server`

```
{"class": "getlist", "commandid": 495252308, "function": "listid", "listname":
↪"phones", "tipbxid": "xivo"}
```

Server > Client

```
{"class": "getlist", "function": "listid", "list": ["1", "3", "2", "5", "14", "7",
↪"6", "9", "8"],
   "listname": "phones", "timenow": 1364994093.38, "tipbxid": "xivo"}
```

Individual phone configuration request:

```
{"class": "getlist", "commandid": 704096693, "function": "updateconfig", "listname
↪": "phones", "tid": "3", "tipbxid": "xivo"}
```

Server > Client

```
{"class": "getlist",
   "config": {"allowtransfer": null, "identity": "SIP/ihvbur", "iduserfeatures": 1,
                  "initialized": null, "number": "1000"},
   "function": "updateconfig", "listname": "phones", "tid": "3", "timenow":␣
↪1364994093.43, "tipbxid": "xivo"}
```

Agents configuration

Client -> Server

```
{"class": "getlist", "commandid": 1431355191, "function": "listid", "listname":
↪"agents", "tipbxid": "xivo"}
```

Queues configuration

Client -> Server

```
{"class": "getlist", "commandid": 719950939, "function": "listid", "listname":
↪"queues", "tipbxid": "xivo"}
```

Server -> Client

```
{"function": "listid", "listname": "queues", "tipbxid": "xivo",
      "list": ["1", "10", "3", "2", "5", "4", "7", "6", "9", "8"], "timenow":␣
↪1382704649.64, "class": "getlist"}
```

Queue configuration

tid is the id returned in the list field of the getlist response message

Client -> Server

```
{"commandid":7,"class":"getlist","tid":"3","tipbxid":"xivo","function":
↪"updateconfig","listname":"queues"}
```

Server -> Client

```
{
 "function": "updateconfig", "listname": "queues", "tipbxid": "xivo", "timenow":␣
↪1382704649.69, "tid": "3",
   "config":
```

(continues on next page)

(continued from previous page)

```
      {"displayname": "red", "name": "red", "context": "default", "number": "3002"}
↪,
 "class": "getlist"}
```

Voicemails configuration

```
Client -> Server
```

```
{"class": "getlist", "commandid": 1034160761, "function": "listid", "listname":
↪"voicemails", "tipbxid": "xivo"}
```

Queue members configuration

```
Client -> Server
```

```
{"class": "getlist", "commandid": 964899043, "function": "listid", "listname":
↪"queuemembers", "tipbxid": "xivo"}
```

```
Server -> Client
```

```
{"function": "listid", "listname": "queuemembers", "tipbxid": "xivo",
   "list": ["Agent/2501,blue", "Agent/2500,yellow", "Agent/2002,yellow", "Agent/
↪2003,__switchboard",
            "Agent/2003,blue", "Agent/108,blue", "Agent/2002,blue"],
   "timenow": 1382717016.23,
   "class": "getlist"}
```

Fax

Send fax

```
Client -> Server
```

```
{"class": "faxsend",
 "filename": "contract.pdf",
 "destination", 41400,
 "size": 100000,
 "data": "<base64 of the fax content>"}
```

Fax status

```
Server -> Client
```

- pages: number of pages sent (`NULL` if FAILED)
- status
  - FAILED: Failed to send fax.
  - PRESENDFAX: Fax number exist and converting pdf->tiff has been done.
  - SUCCESS: Fax sent with success.

```
{"class": "fax_progress", "status": "SUCCESS", "pages": 2 }
```

Call control commands

Dial

- destination can be any number
- destination can be a pseudo URL of the form "type:ibpx/id"

```
Client -> Server
```

```
{
   "class": "ipbxcommand",
   "command": "dial",
   "commandid": <commandid>,
   "destination": "exten:xivo/<extension>"
}
```

For example :

```
{
    "class": "ipbxcommand",
    "command": "dial",
    "commandid": 1683305913,
    "destination": "exten:xivo/1202"
}
```

The server will answer with either an error or a success:

```
{
    "class": "ipbxcommand",
    "error_string": "unreachable_extension:1202",
}

{
    "class": "dial_success",
    "exten": "1202"
}
```

Attended transfer to voicemail

Transfer the current call to a given voicemail and listen to the message before completing the transfer.

```
Client -> Server
```

```
{
    "class": "attended_transfer_voicemail",
    "voicemail": "<voicemail number>"
}
```

Blind transfer to voicemail

Transfer the current call to a given voicemail.

```
Client -> Server
```

```
{
    "class": "blind_transfer_voicemail",
    "voicemail": "<voicemail number>"
}
```

Login

Once the network is connected at the socket level, the login process requires three steps. If one of these steps is omitted, the connection is reset by the cti server.

- login_id, the username is sent as a login to the cti server, cti server answers by giving a sessionid
- login_pass, the password is sent to the cti server, cti server answers by giving a capaid
- login_capas, the capaid is returned to the server with the user's availability, cti server answers with a list of info relevant to the user

```
{
"commandid": <commandid>,
"class": "login_id",
}
```

- class: defined what class of command use.
- commandid : a unique integer number.

Login ID

```
Client -> Server
```

```
{
"class": "login_id",
"commandid": 1092130023,
"company": "default",
"ident": "X11-LE-24079",
"lastlogout-datetime": "2013-02-19T11:13:36",
"lastlogout-stopper": "disconnect",
"userlogin": <userlogin>,
"xivoversion": "<cti protocol version>"
}
```

```
Server -> Client
```

```
{
    "class": "login_id",
    "sessionid": "21UaGDfst7",
    "timenow": 1361268824.64,
    "xivoversion": "<cti protocol version>"
}
```

Note: sessionid is used to calculate the hashed password in next step

Login password

```
Client -> Server
```

```
{
    "class": "login_pass",
    "password": "secret",
    "commandid": <commandid>
}
```

```
Server -> Client
```

```
{
    "capalist": [
        2
    ],
    "class": "login_pass",
    "replyid": 1646064863,
    "timenow": 1361268824.68
}
```

If no CTI profile is defined on XiVO for this user, the following message will be sent:

```
{
    "error_string": "capaid_undefined",
    "class": "login_pass",
    "replyid": 1646064863,
    "timenow": 1361268824.68
}
```

Note: the first element of the capalist is used in the next step login_capas

Login capas

`Client -> Server`

```
{
"capaid": 3,
"commandid": <commandid>,
"state": "available",
"class": "login_capas"
}
```

`Server -> Client`

First message, describes all the capabilities of the client, configured at the server level

- presence : actual presence of the user
- userid : the user id, can be used as a reference
- **capas**
  - **userstatus** [a list of available statuses]
    - status name
    - color
    - selectionnable status from this status
    - default action to be done when this status is selected
    - long name
  - services : list of availble services
  - phonestatus : list of available phonestatuses with default colors and descriptive names
  - capaxlets : List of xlets configured for this profile
  - appliname

```
{
   "class": "login_capas"
   "presence": "available",
   "userid": "3",
   "ipbxid": "xivo",
   "timenow": 1361440830.99,
   "replyid": 3,
   "capas": {
            "preferences": false,
            "userstatus": {
                        "available": { "color": "#08FD20",
                                       "allowed": ["available", "away", "outtolunch
↪", "donotdisturb", "berightback"],
                                       "actions": {"enablednd": "false"}, "longname
↪": "Disponible"
                                     },
                        "berightback": {  "color": "#FFB545",
                                          "allowed": ["available", "away",
↪"outtolunch", "donotdisturb", "berightback"],
                                          "actions": {"enablednd": "false"},
↪"longname": "Bient\u00f4t de retour"
                                       },
                        "disconnected": { "color": "#202020",
                                          "actions": {"agentlogoff": ""}, "longname
↪": "D\u00e9connect\u00e9"
                                       },
                       /* a list of other status depends on the cti server␣
↪configuration */
            },
         "services": ["fwdrna", "fwdbusy", "fwdunc", "enablednd"],
         "phonestatus": {
                           "16": {"color": "#F7FF05", "longname": "En Attente"},
                           "1":  {"color": "#FF032D", "longname": "En ligne OU␣
↪appelle"},
                           "0":  {"color": "#0DFF25", "longname": "Disponible"},
                           "2":  {"color": "#FF0008", "longname": "Occup\u00e9"},
                           "-1": {"color": "#000000", "longname":
↪"D\u00e9sactiv\u00e9"},
                           "4":  {"color": "#FFFFFF", "longname": "Indisponible"},
                           "-2": {"color": "#030303", "longname": "Inexistant"},
                           "9":  {"color": "#FF0526", "longname": "(En Ligne OU␣
↪Appelle) ET Sonne"},
                           "8":  {"color": "#1B0AFF", "longname": "Sonne"}
                        }
      },
   "capaxlets": [["identity", "grid"], ["search", "tab"], ["customerinfo", "tab",
↪"1"], ["fax", "tab", "2"], ["dial", "grid", "2"], ["tabber", "grid", "3"], [
↪"history", "tab", "3"], ["remotedirectory", "tab", "4"], ["features", "tab", "5
↪"], ["people", "tab", "6"], ["conference", "tab", "7"]],
   "appliname": "Client",
}
```

Second message describes the current user configuration

```
{
   "function": "updateconfig",
   "listname": "users",
   "tipbxid": "xivo",
   "timenow": 1361440830.99,
   "tid": "3",
   "config": {"enablednd": false},
   "class": "getlist"
```

(continues on next page)

(continued from previous page)

```
}
```

Third message describes the current user status

```
{
   "function": "updatestatus",
   "listname": "users",
   "status": {"availstate": "available"},
   "tipbxid": "xivo",
   "tid": "3",
   "class": "getlist",
   "timenow": 1361440830.99
}
```

Others

call_form_result

This message is received when a *call form* is submitted from a client to the XiVO.

`Client -> Server`

```
{
    "class": "call_form_result",
    "commandid": <commandid>,
    "infos": {"buttonname": "saveandclose",
              "variables": {"XIVOFORM_varname1": "value1",
                            "XIVOFORM_varname2": "value2"}}
}
```

History

- size : Size of the list to be sent by the server

`Client -> Server`

```
{
   "class": "history",
   "commandid": <commandid>
   "size": "8",
   "xuserid": "<xivoid>/<userfeaturesid>",
}
```

`Server > Client`

Send back a table of calls :

- duration in seconds
- extension: caller/destination extension
- fullname: caller ID name
- mode
  - 0 : sent calls
  - 1 : received calls
  - 2 : missed calls

```
{
   "class": "history",
   "history": [
      {"calldate": "2013-03-29T08:44:35.273998",
       "duration": 30.148765,
       "extension": "*844201",
       "fullname": "Alice Wonderland",
       "mode": 0},
      {"calldate": "2013-03-28T16:56:48.071213",
       "duration": 58.134744,
       "extension": "41400",
       "fullname": "41400"}
       "mode": 1},
   ],
   "replyid": 529422441,
   "timenow": 1364571477.33
}
```

Chitchat

`Client > Server`

```
{
   "class": "chitchat",
   "alias": "Alice",
   "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse␣
↪venenatis velit nibh, ac condimentum felis rutrum id.",
   "to": [<xivo_uuid>, <user_uuid>],
   "commandid": <commandid>
}
```

`Server > Client`

The following message is received by the remote XiVO client

```
{
    "class": "chitchat",
    "from": [<xivo_uuid>, <user_uuid>],
    "to": [<xivo_uuid>, <user_uuid>]
    "alias": "Alice",
    "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse␣
↪venenatis velit nibh, ac condimentum felis rutrum id.",
}
```

Directory

Request directory information, names matching pattern ignore case.

`Client -> Server`

```
{
   "class": "directory",
   "commandid": 1079140548,
   "pattern": "pau"
}
```

`Server > Client`

```
{
   "class": "directory",
   "headers": ["Nom", "Num\u00e9ro", "Mobile", "Autre num\u00e9ro", "E-mail",
↪"Fonction", "Site", "Source"],
   "replyid": 1079140548,
   "resultlist": ["Claire Mapaurtal;;+33644558899;31256;cmapaurtal@societe.com;;;",
                  "Paul Salvadier;+33445236988;+33678521430;31406;
↪psalvadier@societe.com;;;"],
   "status": "ok",
   "timenow": 1378798928.26
}
```

parking

keepalive

availstate

getipbxlist

```
{
    "class": "getipbxlist",
    "commandid": <commandid>
}
```

People

Get relations

This command will trigger a *Relations* message.

```
Client -> Server
```

```
{
    "class": "get_relations"
}
```

People headers

```
Client -> Server
```

```
{
  "class": "people_headers",
}
```

```
Server -> Client
```

```
{
  "class": "people_headers_result",
  "column_headers": ["Status", "Name", "Number"],
  "column_types": [null, null, "number"],
}
```

People Search

```
Client -> Server
```

```
{
  "class": "people_search",
  "pattern": <pattern>,
}
```

Server -> Client

```
{
  "class": "people_search_result",
  "term": "Bob",
  "column_headers": ["Firstname", "Lastname", "Phone number", "Mobile", "Fax",
↪"Email", "Agent"],
  "column_types": [null, "name", "number_office", "number_mobile", "fax", "email",
↪"relation_agent"],
  "results": [
    {
      "column_values": ["Bob", "Marley", "5555555", "5556666", "5553333",
↪"mail@example.com", null],
      "relations": {
        "agent_id": null,
        "user_id": null,
        "endpoint_id": null,
        "source_entry_id": null
      },
      "source": "my_ldap_directory"
    }, {
      "column_values": ["Charlie", "Boblin", "5555556", "5554444", "5552222",
↪"mail2@example.com", null],
      "relations": {
        "agent_id": 12,
        "user_id": 34,
        "endpoint_id": 56,
        "source_entry_id": "34"
      },
      "source": "internal"
    }
  ]
}
```

Relations

This message can currently only be received as a response to the *Get relations* command.

- The *xivo_uuid* is the id of the server
- The *user_id* is the id of the current user.
- The *endpoint_id* is the id of the line of the current user or null.
- The *agent_id* is the id of the agent of the current user or null.

Server -> Client

```
{
    "class": "relations",
    "data": {"xivo_uuid": <the xivo uuid>,
             "user_id": <the user id>,
             "endpoint_id": <the endpoint id>,
             "agent_id": <the agent id>}
}
```

Favorites list

Client -> Server

```
{
  "class": "people_favorites",
}
```

Server -> Client

```
{
  "class": "people_favorites_result",
  "column_headers": ["Firstname", "Lastname", "Phone number", "Mobile", "Fax",
↪"Email", "Agent", "Favorites"],
  "column_types": [null, "name", "number_office", "number_mobile", "fax", "email",
↪"relation_agent", "favorite"],
  "results": [
    {
      "column_values": ["Bob", "Marley", "5555555", "5556666", "5553333",
↪"mail@example.com", null, true],
      "relations": {
        "agent_id": null,
        "user_id": null,
        "endpoint_id": null,
        "source_entry_id": "55"
      },
      "source": "my_ldap_directory"
    }, {
      "column_values": ["Charlie", "Boblin", "5555556", "5554444", "5552222",
↪"mail2@example.com", null, true],
      "relations": {
        "agent_id": 12,
        "user_id": 34,
        "endpoint_id": 56,
        "source_entry_id": "34"
      },
      "source": "internal"
    }
  ]
}
```

Set favorite

Client -> Server

```
{
  "class": "people_set_favorite",
  "source": "my_ldap_directory"
  "source_entry_id": "55"
  "favorite": true
}
```

Server -> Client

```
{
  "class": "people_favorite_update",
  "source": "my_ldap_directory"
  "source_entry_id": "55"
  "favorite": true
}
```

STARTTLS

The STARTTLS command is used to upgrade a connection to use SSL. Once connected, the server send a starttls offer to the client which can reply with a starttls message including the status field. The server will then send a starttls message back to the client with the same status and start the handshake if the status is true.

```
Server -> Client
```

```
{
    "class": "starttls"
}
```

```
Client -> Server -> Client
```

```
{
    "class": "starttls",
    "status": true
}
```

Note: a client which does not reply to the starttls offer will keep it's unencrypted connection.

Personal contacts list

```
Client -> Server
```

```
{
  "class": "people_personal_contacts"
}
```

```
Server -> Client
```

```
{
  "class": "people_personal_contacts_result",
  "column_headers": ["Firstname", "Lastname", "Phone number", "Mobile", "Fax",
↪"Email", "Agent", "Favorites", "Personal"],
  "column_types": [null, "name", "number_office", "number_mobile", "fax", "email",
↪"relation_agent", "favorite", "personal"],
  "results": [
    {
      "column_values": ["Bob", "Marley", "5555555", "5556666", "5553333",
↪"mail@example.com", null, false, true],
      "relations": {
        "agent_id": null,
        "user_id": null,
        "endpoint_id": null,
        "source_entry_id": "abcd-12"
      },
      "source": "personal"
    }, {
      "column_values": ["Charlie", "Boblin", "5555556", "5554444", "5552222",
↪"mail2@example.com", null, false, true],
      "relations": {
        "agent_id": null,
        "user_id": null,
        "endpoint_id": null,
        "source_entry_id": "efgh-34"
      },
      "source": "personal"
```

(continues on next page)

(continued from previous page)

```
    }
  ]
}
```

Personal contact purge

Client -> Server

```
{
  "class": "people_purge_personal_contacts",
}
```

Server -> Client

```
{
  "class": "people_personal_contacts_purged",
}
```

Personal contact raw

Client -> Server

```
{
  "class": "people_personal_contact_raw",
  "source": "personal",
  "source_entry_id": "abcd-1234"
}
```

Server -> Client

```
{
  "class": "people_personal_contact_raw_result",
  "source": "personal",
  "source_entry_id": "abcd-1234",
  "contact_infos": {
      "firstname": "Bob",
      "lastname": "Wonderland"
      ...
  }
}
```

Create personal contact

Client -> Server

```
{
  "class": "people_create_personal_contact",
  "contact_infos": {
      "firstname": "Bob",
      "lastname": "Wonderland",
      ...
  }
}
```

Server -> Client

```
{
  "class": "people_personal_contact_created"
}
```

Delete personal contact

Client -> Server

```
{
  "class": "people_delete_personal_contact",
  "source": "personal",
  "source_entry_id": "abcd-1234"
}
```

Server -> Client

```
{
  "class": "people_personal_contact_deleted",
  "source": "personal",
  "source_entry_id": "abcd-1234"
}
```

Edit personal contact

Client -> Server

```
{
  "class": "people_edit_personal_contact",
  "source": "personal",
  "source_entry_id": "abcd-1234",
  "contact_infos": {
      "firstname": "Bob",
      "lastname": "Wonderland",
      ...
  }
}
```

Server -> Client

```
{
  "class": "people_personal_contact_raw_update",
  "source": "personal",
  "source_entry_id": "abcd-1234"
}
```

Import personal contacts

Client -> Server

```
{
  "class": "people_import_personal_contacts_csv",
  "csv_contacts": "firstname,lastname\r\nBob,the Builder\r\n,Alice,Wonderland\r\n,
↪BobMissingFields\r\n"
}
```

Server -> Client

```
{
  "class": "people_import_personal_contacts_csv_result",
  "created_count": 2,
  "failed": [
      {
          "line": 3,
          "errors": [
              "missing fields"
              ]
      }

  ]
}
```

Export personal contacts

Client -> Server

```
{
  "class": "people_export_personal_contacts_csv",
}
```

Server -> Client

```
{
  "class": "people_export_personal_contacts_csv_result",
  "csv_contacts": "firstname,lastname\r\nBob,the Builder\r\n,Alice,Wonderland\r\n"
}
```

Service

- class : featuresput

Call Filtering

- function : incallfilter
- value : true, false activate deactivate filtering

Client -> Server

```
{"class": "featuresput", "commandid": 1326845972, "function": "incallfilter",
↪"value": true}
```

Server > Client

```
{
   "class": "getlist",
   "config": {"incallfilter": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456398.52, "tipbxid": "xivo"  }
```

DND

- function : enablednd

- value : true, false activate deactivate DND

Client -> Server

```
{"class": "featuresput", "commandid": 1088978942, "function": "enablednd", "value
↪": true}
```

Server > Client

```
{
   "class": "getlist",
   "config": {"enablednd": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456614.55, "tipbxid": "xivo"}
```

Recording

- function : enablerecording
- value : true, false

Activate / deactivate recording for a user, extension call recording has to be activated : *Services->IPBX->IPBX services->Extension*

Client -> Server

```
{"class": "featuresput", "commandid": 1088978942, "function": "enablerecording",
↪"value": true, "target" : "7" }
```

Server > Client

```
{
   "class": "getlist",
   "config": {"enablerecording": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "7",
   "timenow": 1361456614.55, "tipbxid": "xivo"}
```

Unconditional Forward

Forward the call at any time, call does not reach the user

- function : fwd

Client -> Server

```
{
   "class": "featuresput", "commandid": 2082138822, "function": "fwd",
   "value": {"destunc": "1002", "enableunc": true}
}
```

Server > Client

```
{
   "class": "getlist",
   "config": {"destunc": "1002", "enableunc": true},
   "function": "updateconfig",
   "listname": "users",
```

(continues on next page)

(continued from previous page)

```
   "tid": "2",
   "timenow": 1361456777.98, "tipbxid": "xivo"}
```

Forward On No Answer

Forward the call to another destination if the user does not answer

- function : fwd

```
Client -> Server
```

```
{
   "class": "featuresput", "commandid": 1705419982, "function": "fwd",
   "value": {"destrna": "1003", "enablerna": true}
   }
```

```
Server > Client
```

```
{
   "class": "getlist",
   "config": {"destrna": "1003", "enablerna": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361456966.89, "tipbxid": "xivo" }
```

Forward On Busy

Forward the call to another destination when the user is busy

- function : fwd

```
Client -> Server
```

```
{
   "class": "featuresput", "commandid": 568274890, "function": "fwd",
   "value": {"destbusy": "1009", "enablebusy": true}
   }
```

```
Server > Client
```

```
{
   "class": "getlist",
   "config": {"destbusy": "1009", "enablebusy": true},
   "function": "updateconfig",
   "listname": "users",
   "tid": "2",
   "timenow": 1361457163.77, "tipbxid": "xivo"
   }
```

Statistics

Subscribe to queues stats

This message can be sent from the client to enable statitics update on queues

```
Client -> Server
```

```
  {"commandid":36,"class":"subscribetoqueuesstats"}

``Server > Client``
```

Get queues stats

When statistic update is enable by sending message *Subscribe to queues stats*.

The first element of the message is the queue id

```
{"stats": {"10": {"Xivo-LoggedAgents": 0}},
   "class": "getqueuesstats", "timenow": 1384509582.88}
{"stats": {"1": {"Xivo-WaitingCalls": 0}},
   "class": "getqueuesstats", "timenow": 1384509582.89}
{"stats": {"1": {"Xivo-TalkingAgents": "0", "Xivo-AvailableAgents": "1", "Xivo-EWT
↪": "6"}},
   "class": "getqueuesstats", "timenow": 1384512350.25}
```

Status

These messages can also be received without any request as unsolicited messages.

User status

User status is to manage user presence

- Request user status update

```
Client -> Server
```

```
{"class": "getlist", "commandid": 107712156,
   "function": "updatestatus",
   "listname": "users",
   "tid": "14", "tipbxid": "xivo"}
```

```
Server > Client
```

```
{"class": "getlist",
   "function": "updatestatus",
   "listname": "users",
   "status": {"availstate": "outtolunch", "connection": "yes"},
         "tid": "1", "timenow": 1364994093.48, "tipbxid": "xivo"}
```

- Change User status

```
Client -> Server
```

```
{"availstate": "away",
    "class": "availstate",
    "commandid": 1946092392,
    "ipbxid": "xivo",
        "userid": "1"}
```

```
Server > Client
```

```
{"class": "getlist",
    "function": "updatestatus",
    "listname": "users",
```

(continues on next page)

(continued from previous page)

```
    "status": {"availstate": "away"},
    "tid": "1", "timenow": 1370523352.6, "tipbxid": "xivo"}
```

Phone status

- tid is the line id, found in linelist from message *User configuration*

```
Client -> Server
```

```
{"class": "getlist", "commandid": 107712156,
   "function": "updatestatus",
   "listname": "phones", "tid": "8", "tipbxid": "xivo"}
```

```
Server > Client
```

```
{"class": "getlist",
 "function": "updatestatus",
 "listname": "phones",
 "status": {"hintstatus": "0"},
 "tid": "1",
 "timenow": 1364994093.48,
 "tipbxid": "xivo"}
```

Queue status

```
Client -> Server
```

```
{"commandid":17,"class":"getlist","tid":"8","tipbxid":"xivo","function":
↪"updatestatus","listname":"queues"}
```

```
Server > Client
```

```
{"function": "updatestatus", "listname": "queues", "tipbxid": "xivo", "timenow":␣
↪1382710430.54,
   "status": {"agentmembers": ["1","5"], "phonemembers": ["8"]},
   "tid": "8", "class": "getlist"}
```

Agent status

- tid is the agent id.

```
Client -> Server
```

```
{"class": "getlist",
 "commandid": <random_integer>,
 "function": "updatestatus",
 "listname": "agents",
 "tid": "635",
 "tipbxid": "xivo"}
```

```
Server > Client
```

```
{"class": "getlist",
 "listname": "agents",
 "function": "updatestatus",
 "tipbxid": "xivo",
```

(continues on next page)

(continued from previous page)

```
"tid": 635,
"status": {
    "availability": "logged_out",
     "availability_since": 1370868774.74,
     "channel": null,
     "groups": [],
     "on_call_acd": false,
     "on_call_nonacd": false,
     "on_wrapup": false,
     "phonenumber": null,
     "queues": [
         "113"
     ]
 }}
```

- availability can take the values:
  - logged_out
  - available
  - unavailable
  - on_call_nonacd_incoming_internal
  - on_call_nonacd_incoming_external
  - on_call_nonacd_outgoing_internal
  - on_call_nonacd_outgoing_external
- availability_since is the timestamp of the last availability change
- queues is the list of queue ids from which the agent receives calls

Switchboard

Answer

This allows the switchboard operator to answer an incoming call or unhold a call on-hold.

```
{"class": "answer", "uniqueid": "12345667.89"}
```

Unsolicited Messages

These messages are received whenever one of the following corresponding event occurs: sheet message on incoming calls, or updatestatus when a phone status changes.

Sheet

This message is received to display customer information if configured at the server side

```
{
   "timenow": 1361444639.61,
   "class": "sheet",
   "compressed": true,
   "serial": "xml",
   "payload": "AAADnnicndPBToNAEAbgV1n3XgFN1AP...................",
   "channel": "SIP/e6fhff-00000007"
}
```

How to decode payload :

```
>>> b64content = base64.b64decode(<payload content>)
>>> # 4 first cars are the encoded lenght of the xml string (in Big Endian format)
>>> xmllen = struck.unpack('>I',b64content[0:4])
>>> # the rest is a compressed xml string
>>> xmlcontent = zlib.decompress(toto[4:])
>>> print xmlcontent

<?xml version="1.0" encoding="utf-8"?>
   <profile>
      <user>
         <internal name="ipbxid"><![CDATA[xivo]]></internal>
         <internal name="where"><![CDATA[dial]]></internal>
         <internal name="channel"><![CDATA[SIP/barometrix_jyldev-00000009]]></
↪internal>
         <internal name="focus"><![CDATA[no]]></internal>
         <internal name="zip"><![CDATA[1]]></internal>
         <sheet_qtui order="0010" name="qtui" type="None"><![CDATA[]]></sheet_qtui>
         <sheet_info order="0010" name="Nom" type="title"><![CDATA[0230210083]]></
↪sheet_info>
         <sheet_info order="0030" name="Origine" type="text"><![CDATA[extern]]></
↪sheet_info>
         <sheet_info order="0020" name="Num\xc3\xa9ro" type="text"><!
↪[CDATA[0230210083]]></sheet_info>
         <systray_info order="0010" name="Nom" type="title"><![CDATA[Maric\xc3\xa9␣
↪Sapr\xc3\xaftch\xc3\xa0]]></systray_info>
         <systray_info order="0030" name="Origine" type="body"><![CDATA[extern]]></
↪systray_info>
         <systray_info order="0020" name="Num\xc3\xa9ro" type="body"><!
↪[CDATA[0230210083]]></systray_info>
      </user>
   </profile>
```

The xml file content is defined by the following xsd file: `xivo-javactilib/src/main/xsd/sheet.xsd` (online version)

Phone status update

Received when a phone status change

- class : getlist
- function : updatestatus
- listname : phones

```
{
   "class": "getlist",
   "function": "updatestatus",
   "listname": "phones",
   "tipbxid": "xivo",
   "timenow": 1361447017.29,
   .........
}
```

tid is the the object identification

Example of phone messages received when a phone is ringing :

```
{.... "status": {"hintstatus": "0"}, "tid": "3"}
{.... "status": {"hintstatus": "8"}, "tid": "3"}
```

Update notification

Register agent status update

The *register_agent_status_update* command is used to register to the status updates of a list of agent. Once registered to a agent's status, the client will receive all *Agent status update* events for the registered agents.

This command should be sent when an agent is displayed in the people xlet to be able to update the agent status icon.

The *Unregister agent status update* command should be used to stop receiving updates.

```
Client -> Server
```

```
{
  "class": "register_agent_status_update",
  "agent_ids": [["<xivo-uuid>", "<agent-id1>"],
                ["<xivo-uuid>", "<agent-id2>"],
                ...,
                ["<xivo-uuid>", "<agent-idn>"]],
  "commandid": <commandid>
}
```

Unregister agent status update

The *unregister_agent_status_update* command is used to unregister from the status updates of a list of agent.

Once unregistered, the client will stop receiving the *Agent status update* events for the specified agents.

```
Client -> Server
```

```
{
  "class": "unregister_agent_status_update",
  "agent_ids": [["<xivo-uuid>", "<agent-id1>"],
                ["<xivo-uuid>", "<agent-id2>"],
                ...,
                ["<xivo-uuid>", "<agent-idn>"]],
  "commandid": <commandid>
}
```

Agent status update

The *agent_status_update* event is received when the presence of an agent changes.

To receive this event, the user must first register to the event for a specified agent using the *Register agent status update* command.

To stop receiving this event, the user must send the *Unregister agent status update* command.

- data, a dictionary containing 3 fields:
  - agent_id, is an integer containing the ID of the user affected by this status change
  - xivo_uuid: a string containing the UUID of the XiVO that sent the status update
  - status: a string containing the new status, "logged_in" or "logged_out"

```
Server -> Client
```

```
{
  "class": "agent_status_update",
  "data": {
```

(continues on next page)

(continued from previous page)

```
    "agent_id": 42,
    "xivo_uuid": "<the-xivo-uuid>",
    "status": "<status-name>"
  }
}
```

Register endpoint status update

The *register_endpoint_status_update* command is used to register to the status updates of a list of lines. Once registered to a endpoint's status, the client will receive all *Endpoint status update* events for the registered agents.

This command should be sent when a endpoint is displayed in the people xlet to be able to update the agent status icon.

The *Unregister endpoint status update* command should be used to stop receiving updates.

`Client -> Server`

```
{
  "class": "register_endpoint_status_update",
  "endpoint_ids": [["<xivo-uuid>", "<endpoint-id1>"],
                   ["<xivo-uuid>", "<endpoint-id2>"],
                   ...,
                   ["<xivo-uuid>", "<endpoint-idn>"]],
  "commandid": <commandid>
}
```

Unregister endpoint status update

The *unregister_endpoint_status_update* command is used to unregister from the status updates of a list of agent.

Once unregistered, the client will stop receiving the *Endpoint status update* events for the specified agents.

`Client -> Server`

```
{
  "class": "unregister_endpoint_status_update",
  "endpoint_ids": [["<xivo-uuid>", "<endpoint-id1>"],
                   ["<xivo-uuid>", "<endpoint-id2>"],
                   ...,
                   ["<xivo-uuid>", "<endpoint-idn>"]],
  "commandid": <commandid>
}
```

Endpoint status update

The *endpoint_status_update* event is received when the status of a line changes.

To receive this event, the user must first register to the event for a specified endpoint using the *Register endpoint status update* command.

To stop receiving this event, the user must send the *Unregister endpoint status update* command.

- data, a dictionary containing 3 fields:
  - endpoint_id, is an integer containing the ID of the line affected by this status change
  - xivo_uuid: a string containing the UUID of the XiVO that sent the status update
  - status: an integer matching an entry in the cti hint configuration

```
Server -> Client
```

```
{
  "class": "endpoint_status_update",
  "data": {
    "endpoint_id": 42,
    "xivo_uuid": "<the-xivo-uuid>",
    "status": <hint-status>
  }
}
```

Register user status update

The *register_user_status_update* command is used to register to the status updates of a list of user. Once registered to a user's status, the client will receive all *User status update* events for the registered users.

This command should be sent when a user is displayed in the people xlet to be able to update the presence status icon.

The *Unregister user status update* command should be used to stop receiving updates.

```
Client -> Server
```

```
{
  "class": "register_user_status_update",
  "user_ids": [["<xivo-uuid>", "<user-uuid1>"],
               ["<xivo-uuid>", "<user-uuid2>"],
               ...,
               ["<xivo-uuid>", "<user-uuidn>"]],
  "commandid": <commandid>
}
```

Unregister user status update

The *unregister_user_status_update* command is used to unregister from the status updates of a list of user.

Once unregistered, the client will stop receiving the *User status update* events for the specified users.

```
Client -> Server
```

```
{
  "class": "unregister_user_status_update",
  "user_ids": [["<xivo-uuid>", "<user-uuid1>"],
               ["<xivo-uuid>", "<user-uuid2>"],
               ...,
               ["<xivo-uuid>", "<user-uuidn>"]],
  "commandid": <commandid>
}
```

User status update

The *user_status_update* event is received when the presence of a user changes.

To receive this event, the user must first register to the event for a specified user using the *Register user status update* command.

To stop receiving this event, the user must send the *Unregister user status update* command.

- data, a dictionary containing the following fields:
  - user_uuid, a string containing the UUID of the user.

  - user_id, an integer containing the ID of the user.
  - xivo_uuid: a string containing the UUID of the XiVO that sent the status update
  - status: a string containing the new status of the user based on the cti profile configuration

Note: When multiple XiVO share user statuses, the cti profile configuration for presences and phone statuses should match on all XiVO to be displayed properly

`Server -> Client`

```
{
  "class": "user_status_update",
  "data": {
    "user_uuid": "<the-user-uuid>",
    "user_id": <the-user-id>,
    "xivo_uuid": "<the-xivo-uuid>",
    "status": "<status-name>"
  }
}
```

Warning: The *user_id* field is **DEPRECATED** and **should not be used**. Use the *user_uuid* field instead.

CTI server implementation

In the git repository `https://gitlab.com/xivo.solutions/xivo-ctid`

- *cti_config* handles the configuration coming from the WEBI
- *interfaces/interface_ami*, together with *asterisk_ami_definitions*, *amiinterpret* and *xivo_ami* handle the AMI connections (asterisk)
- *interfaces/interface_info* handles the CLI-like connections
- *interfaces/interface_webi* handles the requests and signals coming from the WEBI
- *interfaces/interface_cti* handles the clients' connections, with the help of *client_connection*, and it often involves *cti_command* too
- *innerdata* is meant to be the place where all statuses are computed and stored

The main loop uses *select()* syscall to dispatch the tasks according to miscellaneous incoming requests.

Requirements for *innerdata*:

- the properties fetched from the WEBI configuration shall be stored in the relevant *xod_config* structure
- the properties fetched from elsewhere shall be stored in the relevant *xod_status* structure
- at least two kinds of objects are not "predefined" (as are the phones or the queues, for instance)
  - the channels (in the asterisk SIP/345-0x12345678 meaning)
  - the group and queue members shall be handled in a special way each

The purpose of the 'relations' field, in the various structures is to keep track of relations and cross-relations between different objects (a phone logged in as an agent, itself in a queue, itself called by some channels belonging to phones . . .).

CTI server Message flow

Messages sent from the CTI clients to the server are received by the CTIServer class. The CTIServer then calls `interface_cti.CTI` class `manage_connection` method. The `interface_cti` uses his

_cti_command_handler member to parse and run the command. The CTICommandHandler get a list of classes that handle this message from the CTICommandFactory. Then the the interface_cti.CTI calls run_commands on the handler, which returns a list of all commands replies.

To implement a new message in the protocol you have to create a new class that inherits the CTICommand class. Your new class should have a static member caller required_fields which is a list of required fields for this class. Your class should also have a conditions static member which is a list of tupples of conditions to detect that an incoming message matches this class. The __init__ of your class is responsible for the initialization of it's fields and should call super(<ClassName>, self).__init__(msg). Your class should register itself to the CTICommandFactory.

```
from xivo_cti.cti.cti_command import CTICommand
from xivo_cti.cti.cti_command_factory import CTICommandFactory

class InviteConfroom(CTICommand):
    required_fields = ['class', 'invitee']
    conditions = [('class', 'invite_confroom')]
    def __init__(self):
        super(InviteConfroom, self).__init__(msg)
        self._invitee = msg['invitee']

CTICommandFactory.register_class(InviteConfroom)
```

Each CTI commands has a callback list that you can register to from anywhere. Each callback function will be called when this message is received with the command as parameter.

Refer to MeetmeList.__init__ for a callback registration example and to MeetmeList.invite for the implementation of a callback.

```
from xivo_cti.cti.commands.invite_confroom import InviteConfroom

class MySuperClass(object):
    def __init__(self):
        InviteConfroom.register_callback(self.invite_confroom_handler)

    def invite_confroom_handler(self, invite_confroom_command):
        # Do your stuff here.
        if ok:
            return invite_confroom_command.get_message('Everything is fine')
        else:
            return invite_confroom_command.get_warning('I don't know you, go away',
↪ True)
```

Note: The client's connection is injected in the command instance before calling callbacks functions. The client's connection is an interface_cti.CTI instance.

10.13 Diagrams

10.13.1 Agent states

Graphs representing states and transitions between agent states. Used in Agent status dashboard and agent list.

Download (DIA)

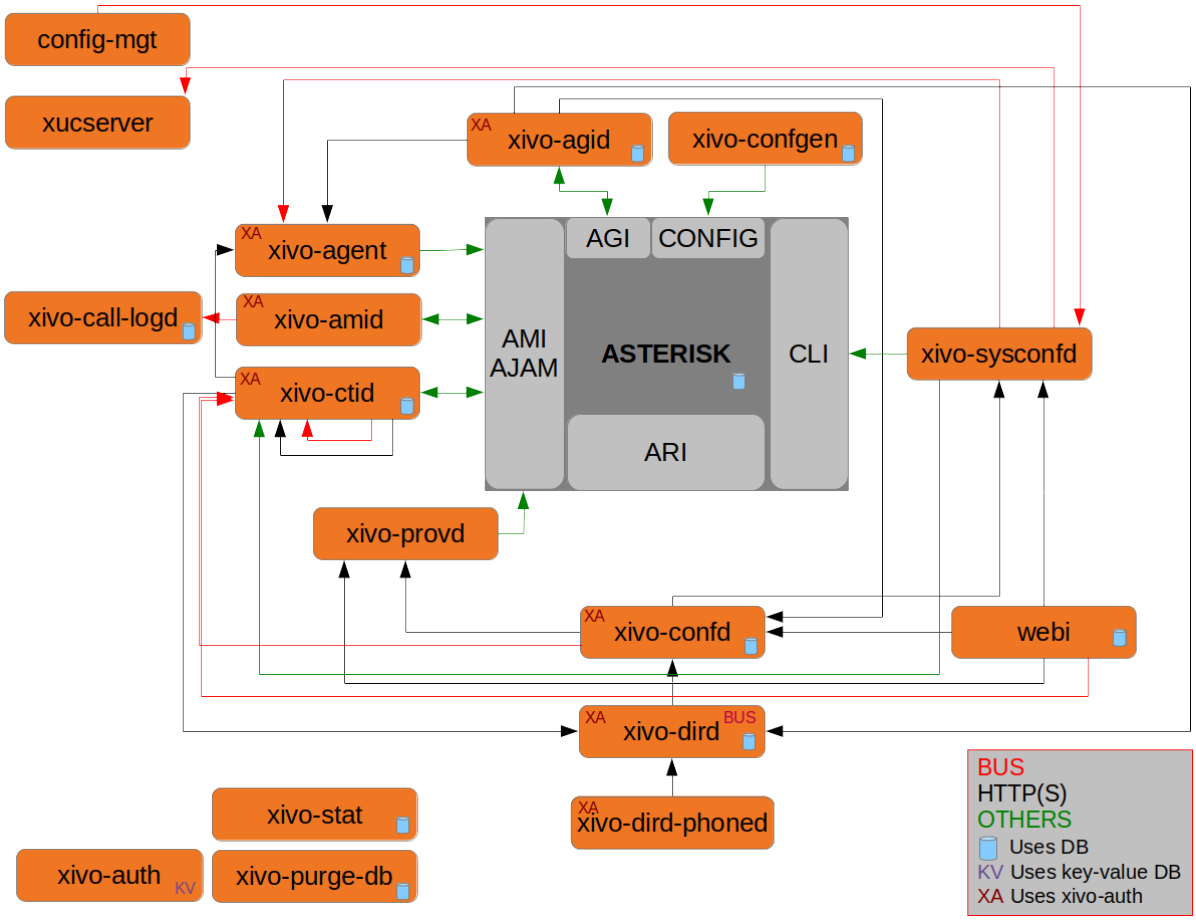


Fig. 1: Relationships between the components of Xivo. (source)

10.13.2 Architecture

10.14 Provisioning

This section describes the informations and tools for xivo-provd.

10.14.1 Managing DHCP server configuration

This page considers the configuration files of the DHCP server in `/etc/dhcp/dhcpd_update/`.

Who modifies the files

The files are updated with the command `dhcpd-update`, which is also run when updating the provisioning plugins. This commands fetches configurations files from the `provd.xivo.solutions` server.

How to update the source files

Ensure your modifications are working

- On a XiVO, edit manually the file `/etc/dhcp/dhcpd_update/*.conf`
- `service isc-dhcp-server restart`
- If errors are shown in `/var/log/daemon.log`, check your modifications

Edit the files

- Edit the files in the Git repo `xivo-provd-plugins`, directory `dhcp/`
- Push your modifications
- Go in `dhcp/`
- Run `make upload` to push your modifications to `provd.xivo.solutions`. There is no `testing` version of these files. Once the files are uploaded, they are available for all XiVO installations.

10.14.2 Managing Plugins

Git Repository

Most plugin-related files are available in the xivo-provd-plugins repository. Following examples are relative to the repository directory tree. Any modifications should be preceeded by a *git pull*.

Updating a Plugin

We will be using the *xivo-cisco-spa* plugins family as an example on this page

There is one directory per family. Here is the directory structure for `xivo-cisco-spa`:

```
plugins/xivo-cisco-spa/
+-- model_name_xxx
+-- model_name_xxx
+-- common
+-- build.py
```

Every plugin has a folder called `common` which regoups common ressources for each model. Every model has its own folder with its version number.

After modifying a plugin, you must increment the version number. You can modifiy the file `plugin-info` to change the version number:

```
plugins/xivo-cisco-spa/
+-- model_name_xxx
    +-- plugin-info
```

Important: If ever you modify the folder `common`, you must increment the version number of all the models.

Use Case: Update Firmwares for a given plugin

Let us suppose we want to update firmwares for xivo-snom from 8.7.3.25 to 8.7.3.25 5. Here are the steps to follow :

1. Copy folder plugins/xivo-snom/8.7.3.25 to plugins/xivo-snom/8.7.3.25.5
2. Update VERSION number in plugins/xivo-snom/8.7.3.25.5/entry.py
3. Update VERSION number in plugins/xivo-snom/8.7.3.25.5/plugin-info
4. Download new firmwares (.bin files from snom website)
5. Update VERSION number and URIs in plugins/xivo-snom/8.7.3.25.5/pkgs/pkgs.db (with uris of downloaded files from snom website)
6. Update sizes and sha1sums in plugins/xivo-snom/8.7.3.25.5/pkgs/pkgs.db (using helper script xivo-tools/dev-tools/check_fw)
7. Update plugins/xivo-snom/build.py (duplicate and update section 8.7.3.25 > 8.7.3.25.5)

Test your changes

You have three different methods to test your changes on your development machine.

Always increase plugin version (easiest)

If the production version is 0.4, change the plugin version to 0.4.01, make your changes and upload to testing (see below).

Next modification will change the plugin version to 0.4.02, etc. When you are finished making changes, change the version to 0.5 and upload one last time.

Edit directly on XiVO

Edit the files in `/var/lib/xivo-provd/plugins`.

To apply your changes, go in `xivo-provd-cli` and run:

```
plugins.reload('xivo-cisco-spa-7.5.4')
```

Disable plugin caching

Edit `/etc/xivo/provd/provd.conf` and add the line:

```
cache_plugin: True
```

Empty `/var/cache/xivo-provd` and restart provd.

Make your changes in provd-plugins, update the plugin version to the new one and upload to testing (see below). Now, every time you uninstall/install the plugin, the new plugin will be fetched from testing, instead of being cached, even without changing the version.

Uploading to testing

Before updating a plugin, it must be passed through the testing phase. Once it has been approved it can be uploaded to the production server

Important: Before uploading a plugin in the testing provd repository, make sure to git pull the xivo-provd-plugins git repository.

To upload the modified plugin in the testing repo on *provd.xivo.solutions*, you can execute the following command:

```
$ make upload
```

Afterwards, in the web-interface, you must modify the URL in section *Configuration* → *Provisioning* → *General* to:

```
`http://provd.xivo.solutions/plugins/1/testing/`
```

You can then update the list of plugins and check the version number for the plugin that you modified. Don't forget to install the plugin to test it.

Mass-install all firmwares related to a given plugin

Using xivo-provd-cli on a xivo server, one can mass-install firmwares. Following example installs all firmwares for xivo-snom 8.7.3.25.5 plugin (note the auto-completion):

```
xivo-provd-cli> plugins.installed().keys()
[u'xivo-snom-8.7.3.15',
 u'xivo-cisco-sccp-legacy',
 u'xivo-snom-8.4.35',
 u'xivo-snom-8.7.3.25',
 u'xivo-aastra-switchboard',
 u'xivo-aastra-3.2.2-SP3',
 u'xivo-aastra-3.2.2.1136',
 u'xivo-cisco-sccp-9.0.3',
 u'null',
 u'xivo-snom-8.7.3.25.5']
xivo-provd-cli> p = plugins['xivo-snom-8.7.3.25.5']
xivo-provd-cli> p.install_all()
```

Uploading to stable

Once checked, you must synchronize the plugin from *testing* to *stable*. If applicable, you should also update the archive repo.

To download the stable and archive plugins:

```
$ make download-stable
$ make download-archive
```

Go to the *plugins/_build* directory and delete the plugins that are going to be updated. Note that if you are not updating a plugin but you are instead removing it "once and for all", you should instead move it to the archive directory:

```
$ rm -fi stable/xivo-cisco-spa*
```

Copy the files from the directory *testing* to *stable*:

```
$ cp testing/xivo-cisco-spa* stable
```

Go back to the *plugins* directory and upload the files to the stable and archive repo:

```
$ make upload-stable
$ make upload-archive
```

The file are now up to date and you can test by putting back the *stable* url in the web-interface's configuration:

```
`http://provd.xivo.solutions/plugins/1/stable/`
```

10.14.3 Testing a new SIP phone

Let's suppose you have received a brand new SIP phone that is not supported by the provisioning system of XiVO. You would like to know if it's possible to add auto-provisioning support for it. That said, you have never tested the phone before.

This guide will help you get through the different steps that are needed to add auto-provisioning support for a phone to XiVO.

Prerequisites

Before continuing, you'll need the following:

- a private LAN where only your phones and your test machines are connected to it, i.e. a LAN that you fully control.

Configuring a test environment

Although it's possible to do all the testing directly on a XiVO, it's more comfortable and usually easier to do on a separate, dedicated machine.

That said, you'll still need a XiVO near, since we'll be doing the call testing part on it and not on a separate asterisk.

So, for the rest of this guide, we'll suppose you are doing your tests on a *Debian jessie* with the following configuration:

- Installed packages:

  ```
  isc-dhcp-server tftpd-hpa apache2
  ```

- Example content of the `/etc/dhcp/dhcpd.conf` file (restart `isc-dhcp-server` after modification):

```
ddns-update-style none;

default-lease-time 7200;
max-lease-time 86400;

log-facility local7;

subnet 10.34.1.0 netmask 255.255.255.0 {
    authoritative;

    range 10.34.1.200 10.34.1.250;

    option subnet-mask 255.255.255.0;
    option broadcast-address 10.34.1.255;
    option routers 10.34.1.6;

    option ntp-servers 10.34.1.6;
    option domain-name "my-domain.example.org";
    option domain-name-servers 10.34.1.6;

    log(concat("[VCI: ", option vendor-class-identifier, "]"));
}
```

- Example content of the `/etc/default/tftpd-hpa` file (restart `tftpd-hpa` after modifcation):

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure --verbose"
```

With this configuration, files served via TFTP will be in the `/srv/tftp` directory and those served via HTTP in the `/var/www` directory.

Testing

Adding auto-provisioning support for a phone is mostly a question of finding answers to the following questions.

1. *Is it worth the time adding auto-provisioning support for the phone ?*

   Indeed. Adding quality auto-provisioning support for a phone to XiVO requires a non negligible amount of work, if you don't meet any real problem and are comfortable with provisioning in XiVO. Not all phones are born equal. Some are cheap. Some are old and slow. Some are made to work on proprietary system and will only work in degraded mode on anything else.

   That said, if you are uncertain, testing will help you clarifying your idea.

2. *What is the vendor, model, MAC address and firmware version (if available) of your phone ?*

   Having the vendor and model name is essential when looking for documentation or other information. The MAC address will be needed later on for some tests, and it's always good to know the firmware version of the phone if you are trying to upgrade to a newer firmware version and you're having some troubles, and when reading the documentation.

3. *Is the official administrator guide/documentation available publicly on the vendor web site ? Is it available only after registering and login to the vendor web site ?*

   Having access to the administrator guide/documentation of the phone is also essential. Once you've found it, download it and keep the link to the URL. If you can't find it, it's probably not worth going further.

4. *Is the latest firmware of the phone available publicly on the vendor web site ? Is it available only after registering and login to the vendor web site ?*

   Good auto-provisioning support means you need to have an easy way to download the latest firmware of the phone. Ideally, this mean the firmware is downloadable from an URL, with no authentication whatsoever.

In the worst case, you'll need to login on some web portal before being able to download the firmware, which will be cumbersome to automatize and probably fragile. If this is the case, it's probably not worth going further.

5. *Does the phone need other files, like language files ? If so, are these files available publicly on the vendor web site ? After registering ?*

   Although you might not be able to answer to this question yet because you might not know if the phone needs such files to be either in English or in French (the two officially supported language in XiVO), you'll need to have an easy access to these files if its the case.

6. *Does the phone supports auto-provisioning via DHCP + HTTP (or TFTP) ?*

   The provisioning system in XiVO is based on the popular method of using a DHCP server to tell the phone where to download its configuration files, and a HTTP (or TFTP) server to serve these configuration files. Some phones support other methods of provisioning (like TR-069), but that's of no use here. Also, if your phone is only configurable via its web interface, although it's technically possible to configure it automatically by navigating its web interface, it's an **extremely bad** idea since it's impossible to guarantee that you'll still be able to provision the phone on the next firmware release.

   If the phone supports both HTTP and TFTP, pick HTTP, it usually works better with the provisioning server of XiVO.

7. *What are the default usernames/passwords on the phone to access administrator menus (phone UI and web UI) ? How do you do a factory reset of the phone ?*

   Although this step is optional, it might be handy later to have these kind of information. Try to find them now, and note them somewhere.

8. *What are the DHCP options and their values to send to the phones to tell it where its configuration files are located ?*

   Once you know that the phone supports DHCP + HTTP provisioning, the next question is what do you need to put in the DHCP response to tell the phone where its configuration files are located. Unless the admin documentation of the phone is really poor, this should not be too hard to find.

   Once you have found this information, the easiest way to send it to the phone is to create a custom host declaration for the phone in the `/etc/dhcp/dhcpd.conf` file, like in this example:

   ```
   host my-phone {
      hardware ethernet 00:11:22:33:44:55;
      option tftp-server-name "http://169.254.0.1/foobar.cfg";
   }
   ```

9. *What are the configuration files the phone needs (filename and content) and what do we need to put in it for the phone to minimally be able to make and receive calls on XiVO ?*

   Now that you are able to tell your phone where to look for its configuration files, you need to write these files with the right content in it. Again, at this step, you'll need to look through the documentation or examples to answer this question.

   Note that you only want to have the most basic configuration here, i.e. only configure 1 line, with the right SIP registrar and proxy, and the associated username and password.

10. *Do basic telephony services, like transfer, works correctly when using the phone buttons ?*

    On most phones, it's possible to do transfer (both attended and direct), three-way conferences or put someone on hold directly from the phone. Do some tests to see if it works correctly.

    Also at this step, it's a good idea to check how the phone handle non-ascii characters, either in the caller ID or in its configuration files.

11. *Does other "standard" features work correctly on the phone ?*

    For quality auto-provisioning support, you must find how to configure and make the following features work:

    - NTP server

- MWI
- function keys (speed dial, BLF, directed pickup / call interception)
- timezone and DST support
- multi language
- DTMF
- hard keys, like the voicemail hard key on some phone
- non-ASCII labels (line name, function key label)
- non-ASCII caller ID
- backup proxy/registrar
- paging

Once you have answered all these questions, you'll have a good idea on how the phone works and how to configure it. Next step would be to start the development of a new provd plugin for your phone for a specific firmware version.

IOT Phones

FK = Funckey

HK = HardKey

Y = Supported

MN = Menu

N = Not supported

NT = Not tested

NYT = Not yet tested

SK = SoftKey

	model
Provisioning	Y
H-A	Y
Directory XIVO	Y
Funckeys	8
Supported programmable keys	
User with supervision function	Y
Group	Y
Queue	Y
Conference Room with supervision function	Y
General Functions	
Online call recording	N
Phone status	Y
Sound recording	Y
Call recording	Y
Incoming call filtering	Y
Do not disturb	Y
Group interception	Y
Listen to online calls	Y
Directory access	Y
Filtering Boss - Secretary	Y
Transfers Functions	

Continued on next page

Table 2 – continued from previous page

	model
Blind transfer	HK
Indirect transfer	HK
Forwards Functions	
Disable all forwarding	Y
Enable/Disable forwarding on no answer	Y
Enable/Disable forwarding on busy	Y
Enable/Disable forwarding unconditional	Y
Voicemail Functions	
Enable voicemail with supervision function	Y
Reach the voicemail	Y
Delete messages from voicemail	Y
Agent Functions	
Connect/Disconnect a static agent	Y
Connect a static agent	Y
Disconnect a static agent	Y
Parking Functions	
Parking	Y
Parking position	Y
Paging Functions	
Paging	Y

10.14.4 Configuring a NAT Environment

This is a configuration example to simulate the case of a hosted XiVO, i.e. an environment where:

- the XiVO has a public IP address
- the phones are behind a NAT

In this example, we'll reproduce the following environment:

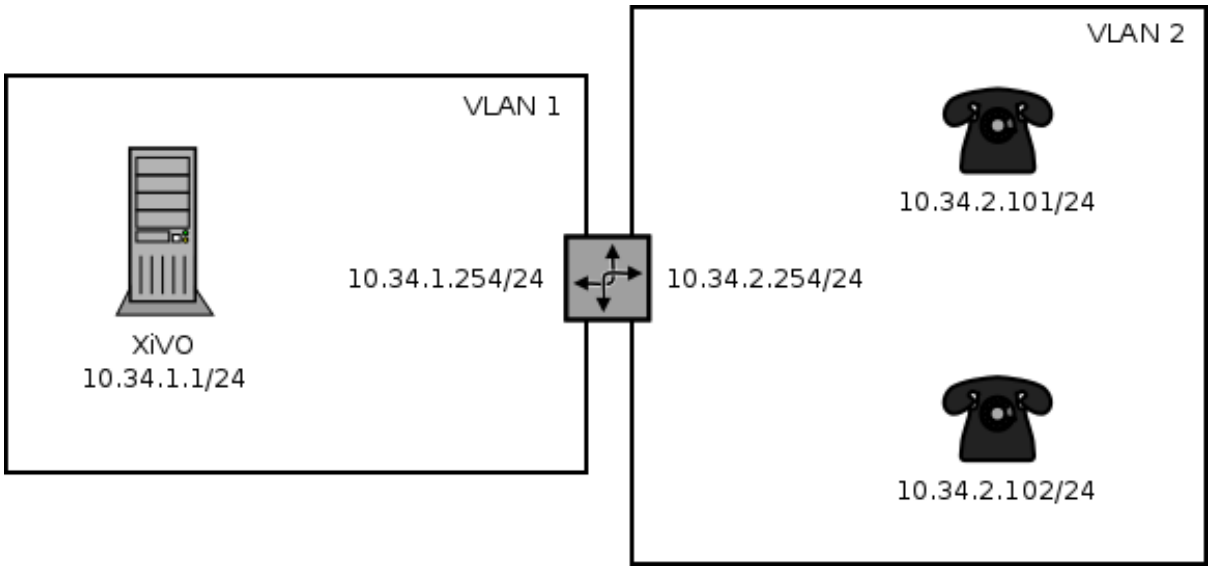


Fig. 2: Phones behind a NAT

Where:

- the XiVO is installed inside a virtual machine
- the host machine is used as a router, a NAT and a DHCP server for the phones

- the phones are in a separate VLAN than the XiVO, and when they want to interact with it, they must pass through the NAT

With this setup, we could also put some phones in the same VLAN as the XiVO. We would then have a mixed environment, where some phones are behind the NAT and some phones aren't.

Also, it's easy to go from a non-NAT environment to a NAT environment with this setup. What you usually have to do is only to switch your phone from the "XiVO" VLAN to the "phones" VLAN, and reconfiguring the lines on your XiVO.

The instruction in this page are written for Debian jessie and VirtualBox.

Prerequisite

On the host machine:

- 1 VLAN network interface for the XiVO. In our example, this will be `eth0.341`, with IP 10.34.1.254/24.
- 1 VLAN network interface for the phones. In our example, this will be `eth0.342`, with IP 10.34.2.254/24.

On the guest machine, i.e. on the XiVO:

- 1 network adapter attached to the "XiVO" VLAN network interface. In our example, this interface inside the virtual machine will have the IP 10.34.1.1/24.

Configuration

1. On the host, install the ISC DHCP server:

```
apt-get install isc-dhcp-server
```

2. If you do not want it to always be started:

```
systemctl disable isc-dhcp-server.service
```

3. Edit the DHCP server configuration file `/etc/dhcp/dhcpd.conf`. We need to configure the DHCP server to serve network configuration for the phones (Aastra and Snom in this case):

```
ddns-update-style none;

default-lease-time 3600;
max-lease-time 86400;

log-facility daemon;

option space Aastra6700;
option Aastra6700.cfg-server-name code 2 = text;
option Aastra6700.contact-rcs code 3 = boolean;

class "Aastra" {
    match if substring(option vendor-class-identifier, 0, 6) = "Aastra";

    vendor-option-space Aastra6700;
    option Aastra6700.cfg-server-name = "http://10.34.1.1:8667/Aastra";
    option Aastra6700.contact-rcs false;
}

class "Snom" {
    match if substring(option vendor-class-identifier, 0, 4) = "snom";

    option tftp-server-name = "http://10.34.1.1:8667";
    # the domain-name-servers option must be provided for the Snom 715 to work␣
↪properly
```

(continues on next page)

(continued from previous page)

```
    option domain-name-servers 10.34.1.1;
}

subnet 192.168.32.0 netmask 255.255.255.0 {
}

subnet 10.34.1.0 netmask 255.255.255.0 {
}

subnet 10.34.2.0 netmask 255.255.255.0 {
    authoritative;

    range 10.34.2.100 10.34.2.199;

    option subnet-mask 255.255.255.0;
    option broadcast-address 10.34.2.255;
    option routers 10.34.2.254;

    option ntp-servers 10.34.1.1;
}
```

4. If you have many network interfaces on your host machine, you might also want to edit `/etc/default/isc-dhcp-server` to only include the "phones" VLAN network interface in the "INTERFACES" variable.
5. Start the isc-dhcp-server:

```
systemctl start isc-dhcp-server.service
```

6. Add an iptables rules to do NAT:

```
iptables -t nat -A POSTROUTING -o eth0.341 -j MASQUERADE
```

7. Make sure that IP forwarding is enabled:

```
sysctl -w net.ipv4.ip_forward=1
```

8. Put all the phones in the "phones" VLAN on your switch
9. Activate the `NAT` and `Monitoring` options on the *Services* → *IPBX* → *General settings* → *SIP Protocol* page of your XiVO.

Note that the iptables rules and the IP forwarding setting are not persistent. If you don't make them persistent (not documented here), don't forget to reactivate them each time you want to recreate a NAT environment.

10.15 SCCP

xivo-libsccp is an alternative SCCP channel driver for Asterisk. It was originally based on chan_skinny.

This page is intended for developers and people interested in using xivo-libsccp on something other than XiVO.

10.15.1 Installation from the git repository

Warning: If you just want to use your SCCP phones with XiVO, refer to *SCCP Configuration* instead.

The following packages are required to compile xivo-libsccp on Debian.

- build-essential

- asterisk-dev

```
apt-get update && apt-get install build-essential asterisk-dev
```

```
git clone https://gitlab.com/xivo.solutions/xivo-libsccp.git
cd xivo-libsccp
make
make install
```

10.15.2 Configuration

Warning: If you just want to use your SCCP phones with XiVO, refer to *SCCP Configuration* instead.

See sccp.conf.sample for a configuration file example.

10.15.3 FAQ

```
Q. When is this *feature X* will be available?
A. The order in which we implement features is based on our client needs. Write
   us an email that clearly explain your setup and what you would like to do and we
   will see what we can do. We don't provide any timeline.
```

```
Q. I want to use the Page() application to call many phones at the same time.
A. Here a Page() example for a one way call (half-duplex):

exten => 1000,1,Verbose(2, Paging to external cisco phone)
same => n,Page(sccp/100/autoanswer&sccp/101/autoanswer,i,120 )

...for a two-way call (full-duplex):

exten => 1000,1,Verbose(2, Paging to external cisco phone)
same => n,Page(sccp/100/autoanswer&sccp/101/autoanswer,di,120 )
```

10.15.4 Network Configuration for 7920/7921

Here's how to to configure a hostapd based AP on a Debian host so that both a 7920 and 7921 Wi-Fi phone can connect to it.

The 7920 is older than the 7921 and is pretty limited in its Wi-Fi functionnality:

- 802.11b
- WPA (no WPA2)
- TKIP (no CCMP/AES)

Which means that the most secure WLAN you can set up if you want both phones to connect to it is not that secure.

1. Make sure you have a wireless NIC capable of master mode.
2. If needed, install the firmware-<vendor> package. For example, if you have a ralink card like I do:

   ```
   apt-get install firmware-ralink
   ```

3. Install the other dependencies:

```
apt-get install wireless-tools hostapd bridge-utils
```

4. Create an hostapd configuration file in `/etc/hostapd/hostapd.sccp.conf` with content: `hostapd.sccp.conf`
5. Update the following parameters (if applicable) in the configuration file:
   - interface
   - ssid
   - channel
   - wpa_passphrase
6. Create a new stanza in `/etc/network/interfaces`:

```
iface wlan-sccp inet manual
     hostapd /etc/hostapd/hostapd.sccp.conf
```

7. Up the interface:

```
ifup wlan0=wlan-sccp
```

8. Configure your 7920/7921 to connect to the network.

   To unlock the phone's configuration menu on the 7921:
   - Press the Navigation Button downwards to enter SETTINGS mode
   - Navigate to and select Network Profiles
   - Unlock the IP phone's configuration menu by pressing **#. The padlock icon on the top-right of the screen will change from closed to open.

   When asked for the authentication mode, select something like "Auto" or "AKM".

   You don't have to enter anything for the username/password.
9. You'll probably want to bridge your wlan0 interface with another interface, for example a VLAN interface:

```
brctl addbr br0
brctl addif br0 wlan0
brctl addif br0 eth0.341
ip link set br0 up
```

10. If you are using virtualbox and your guest interface is bridged to eth0.341, you'll need to change its configuration and bridge it with br0 instead, else it won't work properly.

10.15.5 Adding Support for a New Phone

This section describes the requirements to consider that a SCCP phone is working with XiVO libsccp.

Basic functionality

- Register on Asterisk
- SCCP reset [restart]
- Call history
- Date time display
- HA

Telephony

These test should be done with and without direct media enabled

- Emit a call
- Receive a call
- Receive and transfer a call
- Emit a call and transfer the call
- Hold and resume a call
- Features (*0 and others)
- Receive 2 calls simultaneously
- Emit 2 calls simultaneously
- DTMF on an external IVR

Function keys

- Redial
- DND
- Hold
- Resume
- New call
- End call
- Call forward (Enable)
- Call forward (Disable)
- Try each button in each mode (on hook, in progress, etc)

Optional options to test and document

- Phone book
- Caller ID and other display i18n
- MWI
- Speeddial/BLF

10.16 Web Interface

10.16.1 Configuration for development

Default error level for XiVO web interface is E_ALL & ~E_DEPRECATED & ~E_USER_DEPRECATED & ~E_RECOVERABLE_ERROR & ~E_STRICT

If you want to display warning or other error in your browser, edit the `/etc/xivo/web-interface/xivo.ini` and replace report_type level to 3:

```
[error]
level = E_ALL
report_type = 3
report_mode = 1
report_func = 1
email = john.doe@example.com
file = /var/log/xivo-web-interface/error.log
```

You may also edit `/etc/xivo/web-interface/php.ini` and change the error level, but you will need to restart the cgi:

```
service spawn-fcgi restart
```

10.16.2 Interactive debugging in Eclipse

Instructions for Eclipse 4.5.

On your XiVO:

1. Install php5-xdebug:

   ```
   apt-get install php5-xdebug
   ```

2. Edit the `/etc/php5/cgi/conf.d/20-xdebug.ini` (or `/etc/php5/conf.d/20-xdebug.ini` on wheezy) and add these lines at the end:

   ```
   xdebug.remote_enable=1
   xdebug.remote_host="<dev_host_ip>"
   ```

   where `<dev_host_ip>` is the IP address of your machine where Eclipse is installed.

3. Restart spawn-fcgi:

   ```
   service spawn-fcgi restart
   ```

On your machine where Eclipse is installed:

1. Make sure you have Eclipse PDT installed
2. Create a PHP project named `xivo-web-interface`:
   - Choose "Create project at existing location", using the `xivo-web-interface` directory
3. In the Window / Preferences / PHP menu:
   - Add a new PHP server with the following information:
     - Name: anything you want
     - Base URL: `https://<xivo_ip>`
     - Path Mapping:
       * Path on Server: `/usr/share/xivo-web-interface`
       * Path in Workspace: `/xivo-web-interface/src`
4. Create a new `PHP Web Application` debug configuration:
   - Choose the PHP server you created in last step
   - Pick some file, which can be anything if you don't "break at first line"
   - Uncheck "Auto Generate", and set the path you want your browser to open when you'll launch this debug configuration.

Then, to start a debugging session, set some breakpoints in the code and launch your debug configuration. This will open the page in your browser, and when the code will hit your breakpoints, you'll be able to go through the code step by step, etc.

Community Documentation

10.17 Community Documentation

This page provides links to resources on various topics around XiVO. They have been generously created by people from the community.

10.17.1 Tutorials

Please note that these resources are provided on an "as is basis". They have not been reviewed by the XiVO team, therefore the information presented may be innaccurate. We also accept resources provided in other languages besides English.

Unless specified, the license is CC BY-SA.

Tutorial	Language	Level	Author	XiVO Version
Définition de XiVO pour la communauté et tutoriel (video)	English	Beginner	XiVO	2015
Xivo pour les nuls	French	Beginner	Nicolas	2012
Installing XiVO (YouTube series)	English	Beginner	VoIP-Nuiz	14.20
Start: how to create a user with a SIP line (YouTube series)	French	Beginner	VoIP-Nuiz	2014
Start: how to popup an URL (Document)	French	Beginner		
Start: how to create a context, users, voicemails, ring group, music on hold, conf.call	French	Beginner	Networklab	2014
Tips: post-installation of XiVO on Kimsufi	French	Intermediate	NyXD Systems	2015
Tips: username and password on XiVO	French	Intermediate	NyXD Systems	2015
Tips: self-hosting and telephony with XiVO	French	Intermediate	NyXD Systems	2015
XiVO provisioning + pfSense + siproxd + OVH	French	Intermediate	NyXD Systems	2015
SCCP provisioning, unsupported phones and no DHCP	French	Intermediate	NyXD Systems	2015
Date format on SCCP 7941	French	Intermediate	NyXD Systems	2015
Installing XiVO on Raspberry Pi (Raspivo)	French	Intermediate	Iris Network	2015
`How to popup an url with CTIClient`	French	Intermediate	Assonance	14.17
`How to backup XiVO to external FTP` with `backup-ftp.sh`	French	Intermediate	Yohan Vitu	2015
`How to create a XiVO Client`	French	Intermediate	Yohan Vitu	2015
`How to configure a C610P IP on XiVO`	French	Intermediate	Yohan Vitu	2015
`How to export the phonebook of XiVO` with phonebook_csv_export.py	French	Intermediate	Yohan Vitu	2015
`How to use openVPN on XiVO`	French	Expert	Yohan Vitu	2015
`How configure SNOM M700 DECT`	French	Intermediate	Jonathan Thomas	2015
`Scripted provisioning for SNOM M700 DECT` with specific scripts	French	Intermediate	Jonathan Thomas	2015
`How to configure XiVO with Untangle firewall`	English	Intermediate	Scott McCarthy (SMS IT Group)	16.04
`How to use Keepalived with XiVO (high availability)`	English	Expert	Eric Viel (Iper Telecom)	16.11
Getting Started with XiVO	En-	Begin-	Nerd Vittles	16.07

10.17.2 Contribute

We gladly accept new contributions. There are two ways to contribute:

- The preferred way: open a pull request on Gitlab and add a line to this page (see: *Contributing to the Documentation*).

Note that we only accept documents in open formats, such as PDF or ODF.

CHAPTER 11

Release Notes

11.1 Borealis (2018.16)

Below is a list of *New Features* and *Behavior Changes* compared to the previous LTS version, Aldebaran (2018.05).

Warning: **Known limitations:** as of Borealis release these are the known limitations.

Upgrade

- XiVO PBX / UC / CC is **not installable or upgradable on XFS partition created without** `ftype=1` **option**. If the partition is XFS, you **MUST** check if the option is enabled with the `xfs_info` command.
- Upgrade for **XiVO CC** to Debian 9 **is currently not supported**.

Important: Upgrade to Borealis will upgrade your system to Debian 9 (stretch). See *Debian 9 (stretch) Upgrade Notes*.

11.1.1 New Features

- CC Agent
  - The activity color changes depending on call waiting and agent status (see *Activity Colors*).
  - A new call number badge is shown when there are incoming calls in the agent's activities (see *Activity Waiting Calls*).
  - Only display logged in agents (see *Agent list*).
- CC Manager
  - Supervisor role has now an optional 'Recording' right to enable him to connect to Recording server and to manage recording from CC Manager.
  - Recording mode of queue is now displayed in the CC Manager (see *Queue Recording*).

  - Additional informations of the queue are displayed when clicking on a queue name in the Global view.

- UC Assistant
  - On hold notification can be configured (see *On hold notifications*).
  - WebRTC: volume indication for the speaker and the microphone (see *Volume indication*).
  - WebRTC: user will hear a tone indication (a beep every 10s) warning him of the arrival of a second call.
- Recording
  - Recording of queues is now configurable in the Webi (see *Enable recording in the Queue configuration*).
  - Call to a queue may have its recording stopped after a transfer. See *Recording* feature documentation.
  - Recording on queues can be enabled/disabled via the CC Manager. See *CC Manager Queue Recording* feature documentation.
  - New *Recording Access log*.
- XiVO PBX:
  - Add SRCNUM as available information for FaxToMail application
- System
  - System for *XiVO PBX* was upgraded to Debian 9 (see *Debian 9 (stretch) Upgrade Notes*).
  - Base docker images were updated to optimize disk space taken by a XiVO UC or XiVO CC installation.
  - Support CAS SSO Authentication. See *CAS SSO Authentication Configuration*.
  - When a phone device is not reachable from the Xuc server, an error is reported in the log file.

11.1.2 Behavior Changes

- CC Agent
  - Callbacks in the callback list are now sorted by due date (instead of by status color).
- CC Manager
  - In CC Manager, in view Queues, column “Divert.” was renamed to “Timeout”.
- Desktop Application
  - Desktop application has been revamped to not have OS system window and menu (see updated *Desktop Application Configuration* user guide).
- Fax reception: upon fax reception paper format is forced to “A4” (before it was not specified).
- Reporting
  - *xivo_replic* container was moved from Reporting server to XiVO PBX server and renamed *db_replic*.
  - content from the XiVO PBX tables is now *pushed* from the XiVO PBX to the Reporting server (before it was pulled from the Reporting server)
  - replication state is now stored in a `replication_state` table in the XiVO PBX asterisk database
- System
  - the xucserver is now connecting to the asterisk AMI via the XiVO PBX VoIP address (the one configured as VoIP in the *XiVO PBX* configuration).
  - XiVO CC/UC components logrotation was changed to gzip older logs and keep 14 days of history (instead of 10).

- XUC Websocket API
  - The following properties were removed from the `Sheet` event message:
    * `internal`
    * `sheetQtui`
    * `systrayInfo`

11.1.3 Upgrade

Important: Upgrade to Borealis will upgrade your system to Debian 9 (stretch). See *Debian 9 (stretch) Upgrade Notes*.

Follow the usual procedures (**don't forget** the specific steps to upgrade to another LTS version - see *Manual steps for LTS upgrade*):

- *XiVO PBX upgrade procedure*
- *XiVO CC upgrade procedure*

11.2 Borealis Bugfixes Versions

11.2.1 Components version table

Table listing the current version of the components.

Component	current ver.
config-mgt	2018.16.04
elasticsearch	1.7.2
fingerboard	2018.16.00
kibana_volume	2018.16.00
nginx	2018.16.00
pack-reporting	2018.16.13
pgxivocc	1.3
recording-rsync	1.0
recording-server	2018.16.00
spagobi	2018.16.09
xivo-db-replication	2018.16.03
xivo-full-stats	2018.16.12
XiVO PBX	2018.16.12
xuc	2018.16.12
xucmgt	2018.16.12

11.2.2 Borealis.13

Consult the Borealis.13 Roadmap.

Components updated: **pack-reporting**

Reporting

- #3077 - [C] - log rotation for specific-stats.log does not work as expected

11.2.3 Borealis.12

Consult the Borealis.12 Roadmap.

Components updated: **xivo-full-stats**, **xivo-web-interface**, **xucmgt**, **xucserver**

CCAgent

- #2837 - Chrome / Chromium crashes when agent misses a call and focus and browser has its focus on another tab

CCManager

- #2975 - [C] - In agent view, when editing an agent, can't order on column

Reporting

- #2425 - Invalid total in agent report
- #2999 - [C] - Call history does not show calls from Group

Web Assistant

- #1961 - CallerID name display error when call comes from a Group
- #2887 - "Play audio" signal from keyboard or headset plays the xivo ringing sound

XUC Server

- #2933 - [C] - Transfer completion may fail (all is hangup) if transfer completion is done while destination is ringing

XiVO PBX

- #2710 - [C] - Error when adding a user to a group when creating this user

XiVOCC Infra

- #2986 - Fix xucmgt / desktop assistant build

11.2.4 Borealis.11

Consult the Borealis.11 Roadmap.

Components updated: **xivo-upgrade**, **xucmgt**

CCAgent * *#2757 <https://projects.xivo.solutions/issues/2757>* - Webrtc agent is still ringing after logged out on CC agent on login page

XiVO PBX

- #2727 - Download of Docker GPG key does not work behind a proxy

11.2.5 Borealis.10

Consult the Borealis.10 Roadmap.

Components updated: **xivo-agid**, **xivo-backup**, **xivo-config**, **xivo-dao**, **xivo-dbms**, **xivo-upgrade**, **xivo-web-interface**, **xivocc-installer**, **xucmgt**, **xucserver**

CCAgent

- #2529 - [C] - Cannot scroll to the end of content panel iwhen receiving a call

XUC Server

- #2534 - CAS Login doesn't work when username contains uppercase
- #2566 - AMI Failure when decoding response from other AMI applications

XiVO PBX

- #2256 - [C] - Web-I - As an admin different from root I cannot edit Queue settings
- #2307 - Can not dial numbers with 1 digit or * + 1 digit (*X* OR **X*) from uc assistant
- #2368 - Database schema may not be upgraded during upgrade
- #2381 - Add srcnum as available information when receiving a fax
- #2414 - Confgend can't generate configuration due to sqlalchemy changes for security purposes
- #2422 - Build dahdi-modules-3.16.0-8 for Jessie versions (Five, Polaris, Aldebaran)
- #2634 - When I installed DAHDI modules, it should continue to work after (minor) kernel upgrade (port to Borealis)

Important: Behavior change When upgrading, it will install new `dahdi-linux-dkms` package with necessary dependencies. The removal of the old `dahdi-linux-modules-*` packages is handled during the upgrade on XiVO.

However this removal is not handled automatically during upgrade on MDS. Therefore, *on a MDS*, you MUST remove them manually after the upgrade by running the command:

```
apt-get purge '^dahdi-linux-modules*' -y``
```

- #2641 - Improve xivo-backup adding fail2ban config AND more important: docker/xivo configuration
- #2680 - Hostname change is not taken into account

XiVO Provisioning

- #2433 - Create plugin Snom for firmware 8.9.3.80
- #2514 - Provisioning - Do not always restart Snom phone whgen synchronizing

XiVOCC Infra

- #2692 - XiVO UC Installer does not set correct pgxivocc address in xivo/custom.env for db_replic

11.2.6 Borealis.09

Consult the Borealis.09 Roadmap.

Components updated:

SpagoBi

- #2426 - Add scriplet to spagobi

11.2.7 Borealis.08

Consult the Borealis.08 Roadmap.

Components updated: **xivo-confd**, **xivo-config**, **xivo-dao**, **xivo-web-interface**

XiVO PBX

- #1968 - Set asterisk default language to fr_FR

Important: Behavior change Default asterisk language was set to *fr_FR. In this language we have all sounds file. To change it to english, one should: * verify that the pacakges 'asterisk-sounds-wav-en-us', 'xivo-sounds-en-us' * and set, in file :file:'/etc/asterisk/asterisk.conf' the *defaultlanguage* parameter to *en_US*

- #2365 - [C] Incoming call used customized Goto call leads to user not having its ringing time take into account
- #2375 - Error in ctid when using an agent created with the user api

  Important: Behavior change When creating a user using the REST API, the CTI profile is now set to a default value and the CTI client is enabled when a CTI client login and a password is set.

- #2390 - Slow installation of xivo-web-interface

11.2.8 Borealis.07

Consult the Borealis.07 Roadmap.

Components updated: **xucmgt**, **xucserver**

WebRTC

- #2373 - WebRTC - RTP flow is stopped after unhold

11.2.9 Borealis.06

Note: Bugfix release. See *Borealis (2018.16)* section for features list and behavior changes.

Consult the Borealis.06 Roadmap.

Components updated: **xivo-confd**, **xivo-dao**, **xivo-install-cd**, **xucmgt**, **xucserver**

CCAgent

- #2281 - Callbacks cannot be displayed if agent doesn't have right on any queue that contains callback requests
- #2320 - Unable to loggoff an agent with ccagent
- #2327 - Callbacks are removed from the list if I click again on the Callbacks view

CCManager

- #2285 - [C] CCManager displays wrong logout time in Agent View
- #2287 - [C] CCManager add callbacks count and oldest callback for monitoring purpose
- #2321 - Groups cannot be displayed in ccmanager
- #2336 - Group view is not refreshed if agent's group changed via ccmanager

Desktop Assistant

- #2288 - Add border to desktop application
- #2306 - Tray icon sometimes disapears from notification bar
- #2328 - Desktop assistant language is hardcoded
- #2341 - German UI translations

Web Assistant

- #2300 - Second call Bip is not always working and global key not always working

XUC Server

- #1529 - When leaving an outbound queue, agent can not longer emit a call
- #2284 - xuc does not re-connect to the right IP address of XiVO for the AMi

- #2302 - Agent state not properly initialized after XUC start
- #2311 - WebAssistant - XuC - xivo-auth User cannot toggle DND or enable/disable forward because his token doesn't exist anymore
- #2340 - Phone status incorrect after xuc restart
- #2356 - [C] - When xuc loses connection to AMI (Ami failure) ghost call may appear in user interfaces

XiVO PBX

- #2351 - Cannot create func key for agent when created with confd API

> **Important:** **Behavior change** Agents created with the user api in Xivo Version 2018.16.05 need to be recreated manually to fix this issue. New agents will be created with no issue.

XiVOCC Infra

- #2106 - Update ISO build for Debian 9 Stretch

11.2.10 Borealis.05

Note: Bugfix release. See *Borealis (2018.16)* section for features list and behavior changes.

Consult the Borealis.05 Roadmap.

Components updated: **xucmgt**

Desktop Assistant

- #2279 - Select-to-call gives an error
- #2280 - It is not possible to discard easily desktop settings

11.2.11 Borealis.04

Note: Bugfix release. See *Borealis (2018.16)* section for features list and behavior changes.

Consult the Borealis.04 Roadmap.

Components updated: **config-mgt**, **xivo-agid**, **xivo-confd**, **xivo-dao**, **xivo-lib-python**, **xivo-sysconfd**, **xivo-web-interface**, **xivocc-installer**, **xucmgt**, **xucserver**

CCAgent

- #2245 - Callbacks are not sorted by due date
- #2246 - CCagent queue (activity) state is hardcoded to green, should be more dynamic to call flow

> **Important:** **Behavior change** The activity color changes depending on call waiting and agent status. See *Activity Colors* documentation for details.

- #2259 - Not possible to hangup call in ccagent reduced action bar
- #2265 - CCagent real time waiting calls display in action menu and for each activity

Important: **Behavior change** A new call number badge is shown when there are incoming calls in the agent's activities. See *Activity Waiting Calls* documentation for details.

- #2267 - CCagent add way to filter connected agents

Important: **Behavior change** Only display logged in agents of your group. See *Agent list* documentation for details.

Config mgt

- #1499 - [C] Config Mgt / rights server is not available when a lot of Agents connect in a short time

Desktop Assistant

- #2164 - Be able to move ccagent main frame with electron on windows if reduced

Important: **Behavior change** Desktop application has been revamped to not have OS system window and menu. See updated *Desktop Application Configuration* user guide.

- #2273 - Null association with desktop assistant on Windows 10 avoids callto and tel protocol to work

Web Assistant

- #2257 - Problem when importing personal contacts from CSV file in Firefox web assistant

WebRTC

- #1496 - Webrtc user can't resume calls after trying to start third call
- #2262 - Play call waiting tone when second call is incoming instead ringing tone

Important: **Behavior change** WebRTC user will hear a tone indication (a beep every 10s) warning him of the arrival of a second call.

XUC Server

- #2240 - [C] - Unable to add agent to multiple queues at once
- #2248 - [CAS] Enable lougout from CAS authentication (Port to borealis)
- #2251 - Xuc still tries to connect to CTI server
- #2278 - Xuc link restarts indefinitely

XiVO PBX

- #1594 - Should be able to create/update/delete an agent (agent API)
- #1885 - Error message when line is removed and re-added to user in call group
- #2003 - Error code when updating via CSV - confd error handling is failing when system is not in english
- #2117 - Interface without ip address produces an error
- #2154 - xivo-auth not upgraded before xivo-docker-components causes errors
- #2229 - Should be able to create/update a user with agent number via confd API (POST /users, PUT /user/{id})
- #2247 - XDS - Agid on mds1 can't reconnect to mds0
- #2252 - BTIP - Timer Session value

Important:

Behavior change New values were added to fields :

- Maximum session refresh interval
- Minimum session refresh interval

in General -> SIP Protocol and Trunk Management -> SIP Protocol

---

- #2266 - Should be able to change the group of the agent
- #2276 - Queue search in xivo web interface

XiVO Provisioning

- #1851 - Cannot install Cisco SPA8000 firmware from plugin

11.2.12 Borealis.03

Note: Bugfix release. See *Borealis (2018.16)* section for features list and behavior changes.

Consult the Borealis.03 Roadmap.

Components updated: **asterisk**, **xivo-agid**, **xivo-confd**, **xivo-confgend**, **xivo-config**, **xivo-dao**, **xivo-db-replication**, **xivo-dist**, **xivo-full-stats**, **xivo-upgrade**, **xivo-web-interface**, **xivocc-recording**, **xucmgt**, **xucserver**

Asterisk

- #2143 - Asterisk crash when core show channels command is issued too early

Desktop Assistant

- #2211 - Number display bug with too long email address
- #2237 - Can't update Electron application on Windows if ccagent is configured as URL

Reporting

- #1077 - Xivo replic stops replicating to elasticsearch
- #2206 - Db replication do not retry elasticsearch connection if failing at startup
- #2235 - [C] Agents statistic conversation_time_outgoing_calls abnormally long

Web Assistant

- #2207 - Personal contact import on windows doesn't work

XUC Server

- #1895 - Agent's call state is not correct (in ccmanager) after a xuc restart
- #2226 - Unable to decode token message from browser when login
- #2228 - [C] When re-provisioning user on another phone with another IP, StartCallback dials wrong line/device (Port to Borealis)
- #2232 - When one server is down Xuc is disconnected from all AMIs

XiVO PBX

- #2053 - xivocc-recording should be installed by default as long as is now part of webi

---

Important: Behavior change xivocc-recording package is now part of a standard xivo PBX installation

---

- #2153 - MDS installer sets wrong version in sources.list at the end

  Important: **Behavior change** XiVO distribution in the `xivo-dist.list` will not be changed after installation or upgrade.

- #2204 - Cannot delete a user with funckeys
- #2205 - Error when creating an incoming calls with pattern
- #2217 - XDS - Add XDS installation and configuration documentation
- #2219 - XDS - Incoming call (DID) to a user located on a MDS (which is not mds0) doesn't work
- #2225 - Do not check database activity on normal upgrade
- #2230 - [C] fax tiff converted to pdf have a bad format, and can't be printed

  Important: **Behavior change** Upon fax reception paper format is forced to "A4" (before it was not specified).

- #2231 - Install script uses command from non-essential package
- #2233 - Display MDS column in users listing page

11.2.13 Borealis.02

Note: Bugfix release. See *Borealis (2018.16)* section for features list and behavior changes.

Consult the Borealis.02 Roadmap.

Components updated: **xivo-agentd**, **xivo-amid**, **xivo-auth**, **xivo-client-qt**, **xivo-confd**, **xivo-config**, **xivo-ctid**, **xivo-dao**, **xivo-db-replication**, **xivo-dird**, **xivo-monitoring**, **xivo-tools**, **xivo-upgrade**, **xivo-web-interface**, **xivocc-installer**, **xucmgt**, **xucserver**

Important: **Manual procedure:** Bug #2155 must be fixed manually, but **only** if **all** these conditions match:

- Your system was in *Five* or *Polaris* version before the upgrade to Borealis.
- The system was already upgraded to Borealis - to version older than Borealis.02
- You have *XiVO CC* installed
- List of *Callbacks* was not empty before the upgrade
- The Config Mgt database was already migrated by the `xivo-migrate-configmgt-db` script run on XiVO older than Borealis.02

To make `db_replic` start, you must run the following SQL command on **XiVO CC** and remember the returned `max` value:

```
docker exec -it -u postgres xivocc_pgxivocc_1 psql -h localhost -U xuc xuc_rights \
-c 'select max(reference_number) from callback_request;'
```

Then you must replace "`...`" in the following SQL command by the `max` reference number and run the command on **XiVO PBX**:

```
cd /tmp
sudo -u postgres psql asterisk -c "update replication_state
    set val=( ... )
    where name='callback_request';"
```

Then restart the container with command `xivo-dcomp restart db_replic`.

CCAgent

- #2198 - Agent Outbound Call statistics is wrong

CCManager

- #2160 - Some agents don't have any state displayed in CCManager after login
- #2168 - [C] - Additionnal calls are counted for missed calls and offered calls stats (port to borealis)
- #2182 - [C] Wrong linkedId in Xuc PhoneEvents

Desktop Assistant

- #2165 - Redirection loop if wrong URL parameter in desktop assistant

Switchboard

- #2105 - [C] Call to Switchboard is auto-answered without action from the XiVO Client

Web Assistant

- #1779 - Error when importing and exporting contacts
- #2130 - Update sbt file and migrate to play 2.4 for xucmgt
- #2172 - migrate to play 2.6 and update to scala 2.12 for xucmgt
- #2200 - After reload or reopen, pending voicemail message are not shown - Port to Borealis

WebRTC

- #1732 - Video is lost when changing the tab and going back
- #2177 - Media capture is not released after a call is hangup with Webrtc

XUC Server

- #2199 - [C] - Voicemail remains in disabled state after deactivating DND (port to borealis)

XiVO PBX

- #956 - Incall and Exten with pattern (regex)
- #2094 - Debian 9 - Errors on Debian9 installation
- #2120 - XiVO UC link xuc-config mgt missing although required for Borealis
- #2141 - Monit fails while checking Rabbitmq service

  Important: Behavior change RabbitMQ monitoring must be manually started in order to re-enable it.

- #2155 - db_replic can't replicate callback_request table after upgrade from Five or Polaris
- #2156 - XiVO UC - docker logs are missing
- #2157 - Docker overlay2 doesn't work if the filesystem is XFS with d_type 0
- #2163 - Test & fix physical installations
- #2173 - Should be able to get a specific user (GET users/{user_id}) and have the user plus its agent_number
- #2175 - xivo-stat cron sends email with sqlalchemy warning at each execution
- #2186 - Should be able to get list of users (GET users) and have for its user its agent_number (if relevant)

XiVOCC Infra

- #2086 - When upgrading xivocc-installer docker-xivo.override.yml files are not upgraded
- #2145 - DB replic is misconfigured if XUC_HOST is domain name

- #2149 - [C] UC add-on missing installation step causes assistant to not accept incoming calls or pause all established ones
- #2180 - Update Debian PXE for Debian 9.6 release
- #2188 - XiVO UC - Errors in xuc log when connecting to config mgt
- #2202 - Add new Support team ssh RSA key for remote ssh connections

11.2.14 Borealis.01

Note: Bugfix release. See *Borealis (2018.16)* section for features list and behavior changes.

Consult the Borealis.01 Roadmap.

Components updated: **xivo-confd**, **xivo-confgend**, **xivo-config**, **xivo-upgrade**, **xivo-web-interface**, **xivocc-installer**, **xucserver**

CCAgent

- #2137 - CSS Cache is not invalidated when a new version is released

XUC Server

- #2123 - Xuc Play 2.6 Migration
- #2132 - Xuc doesn't connect to AMI if it starts before XiVO
- #2142 - XucServer - Use Xivo data ip instead of voip when xivo is not configured as XDS

XiVO PBX

- #1765 - saving a user with many funckeys is very long
- #2110 - Upgrade of XiVO with UC addon to Debian 9 (stretch) and switch from aufs to overlay2 docker storage driver
- #2114 - Error "Too many arguments" in xivo-upgrade log
- #2121 - Install overlay2 docker volume driver while upgrading from Five or Polaris
- #2134 - Be able to skip intro of echo test
- #2150 - XiVO UC - custom.env is not updated for DB Replic if upgraded from Polaris

XiVO Provisioning

- #2140 - Support Yealink T27G

XiVOCC Infra

- #1151 - Be able to use Debian 9 with XiVO CC
- #2026 - Certificate Request Error with Debian 9/Open SSL 1.1.0

11.2.15 Borealis.00

Important: System for *XiVO PBX* was upgraded to Debian 9 (stretch) - see *Debian 9 (stretch) Upgrade Notes*

Note: **LTS Release**. New features and behavior changes are listed above, under the *Borealis (2018.16)* section.

Consult the Borealis Roadmap and the Borealis.00 Roadmap.

Borealis is a LTS release of XiVO Solutions. A summary of the *New Features* and *Behavior Changes* compared to the previous LTS version, Aldebaran, is available above (see *Borealis (2018.16)*).

Components updated: **config-mgt**, **play-authentication**, **xivo-agid**, **xivo-confgend**, **xivo-config**, **xivo-ctid**, **xivo-dao**, **xivo-install-script**, **xivo-lib-rest-client**, **xivo-manage-db**, **xivo-service**, **xivo-sysconfd**, **xivo-upgrade**, **xivo-web-interface**, **xivocc-installer**, **xucmgt**, **xucserver**

Config mgt

- #2116 - Queue recording events are not received in CCmanager when toggling recording switch

Desktop Assistant

- #2027 - No call notification with Desktop Application on Windows10 (Windows 10 16299.19)

Reporting

- #2118 - Db replication fails to insert callback requests

Web Assistant

- #2068 - UC Assistant - Phone Conference button are misaligned
- #2119 - Personal contact not translated in history

WebRTC

- #2008 - Add visual indication of audio streams for WebRTC calls

XUC Server

- #1999 - XDS - (UC) Update Phone Status from Asterisk and send ami request to all MDS
- #2078 - Call can't be answered with Polycom from CCAgent or Assistant
- #2080 - Finish Play 2.5 Migration
- #2103 - Remove Test Warnings in Play 2.5

XiVO PBX

- #1573 - Online call recording does not appear in web interface
- #2016 - Cannot call a queue if Config-mgt is not started
- #2107 - Unable to answer call as switchboard on Polycom
- #2109 - Upgrade fails from Aldebaran with apt get option AutomaticRemove to true
- #2112 - Update pyopenssl version in lib rest client to 16.2.0
- #2113 - Add next LTS codename in list of named distributions of xivo-dist

XiVO Distributed System

- #2006 - XDS - Generate configuration for a MDS
- #2079 - XDS - Unable to call from a WebRTC User
- #2081 - XDS - Routing agi breaks dialplan when call is emitted by SCCP peer
- #2087 - XDS - On MDS xivo-service script does not work properly
- #2093 - XDS - Be able to route calls depending on the MDS of the peer
- #2111 - Update MDS install to be independent on APT system options
- #2115 - Music on hold is not installed when installing from script (xivo or mds)

System

- #2089 - Update PXE to install Debian 9 for Borealis > 2018.13

11.3 Borealis Intermediate Versions

11.3.1 XiVO Borealis Intermediate Versions

2018.14

Consult the 2018.14 Roadmap.

Components updated: **asterisk**, **rabbitmq**, **xivo-agid**, **xivo-confgend**, **xivo-config**, **xivo-dao**, **xivo-db-replication**, **xivo-dist**, **xivo-manage-db**, **xivo-monitoring**, **xivo-service**, **xivo-sysconfd**, **xivo-upgrade**, **xivo-web-interface**, **xivocc-installer**

Important: In this version the system was upgraded from Debian 8 to **Debian 9 (stretch)**. Read carefully the *Debian 9 (stretch) Upgrade Notes*.

XiVO PBX

- **Asterisk**: Build asterisk for Debian Stretch
  - #2023 - XDS - Build Asterisk for Debian Stretch
- #1150 - Be able to use Debian 9 with XiVO
- #2067 - Upgrade webi to PHP7
- #2063 - APT is not updated after installation by script
- #2077 - Upgrade from Aldebaran to 2018.12 or 2018.13 is broken
- #2083 - Configmgt log are not written in the correct directory

XiVO Distributed System

- #2025 - XDS - Migrate MDS to Debian9
- #2031 - Adapt xivo-manage-db to be applicable also to MDS/Debian9
- #2036 - XDS - asterisk configuration must be ready after boot/reboot on MDS
- #2041 - XDS - Generate confgend container configuration at startup
- #2057 - XDS - Generate systemname according to the mds name
- #2076 - MDS db_replic is missing in the list of xivo-service
- #2085 - XDS - use extra_host to make agid, confgend and rabbitmq containers aware of xivo ip
- #2090 - Create view for all media server including main mds
- #2092 - Harmonize rabbitmq-server version on XiVO and MDS

System

- #2091 - Optimize spagobi docker image build to enhance download time in case of upgrade

2018.13

Consult the 2018.13 Roadmap.

Components updated: **asterisk**, **config-mgt**, **xivo**, **xivo-agid**, **xivo-config**, **xivo-dao**, **xivo-db-replication**, **xivo-install-script**, **xivo-manage-db**, **xivo-monitoring**, **xivo-service**, **xivo-web-interface**, **xivocc-installer**, **xucmgt**, **xucserver**

CCAgent

- #2042 - On hold notification is sometimes not hidden when having multiple calls

Reporting

- #2045 - XDS - Finish db_replic
- #2050 - XDS - Cleanup install and upgrade for db_replic

Important: **Behavior change** db_replic (a.k.a xivo_replic) is now only installed when a CC or UC is installed on top of XiVO PBX When doing so, a compose override file is copied in /etc/docker/xivo to enable the service automatically.

XUC Server

- #2037 - Xucserver Library upgrade

XiVO PBX

- **Asterisk** Ships with new asterisk version 13.22.0
  - #1793 - Wrapup expired event is sent whereas Wrapup was not started
  - #1993 - Integrate asterisk 13.22.0
  - #2058 - Build and package asterisk 13.22.0 with patch for #1793, #2065, #2066
  - #2065 - Compile asterisk with bundled pjproject
  - #2066 - Remove patch xivo_dtls_negotiation_delay (audio delayed on webrtc incoming call due to DTLS negotiation increasing timeout)
- #1978 - XDS - Be able to route calls depending on the MDS of the peer
- #1979 - XDS - Be able to declare Media Server in XiVO Webi
- #2049 - XDS - Finish db_replic integration
- #2051 - Cleanup MDS installer
- #2052 - Cleanup XiVO (xivo-configmgt) installer
- #2060 - XDS - Install script sets wrong apt source after installation

2018.12

Consult the 2018.12 Roadmap.

Components updated: **debian-pxelinux**, **rabbitmq**, **xivo-agid**, **xivo-confgend**, **xivo-config**, **xivo-install-script**, **xivo-service**, **xivo-sysconfd**, **xivo-web-interface**, **xivocc-installer**, **xucserver**

CCAgent

- #2033 - When selecting a callback the wrong callback is displayed an called

Reporting

Important: **Behavior change** for replication (see #1990).

- *xivo_replic* container was moved from Reporting server to XiVO PBX server and renamed *db_replic*.
- content from the XiVO PBX tables is now *pushed* from the XiVO PBX to the Reporting server (before it was pulled from the Reporting server)
- replication state is now stored in a `replication_state` table in the XiVO PBX asterisk database
- while updating, a prompt will ask you reporting server address to be able to replicate data.

XUC Server

- #2018 - Agent is not added or removed from ccmanager

XiVO PBX

- #1691 - Glitches in XiVO web-interface

XiVO Distributed System

- #1980 - XDS - Install a Media Server
  - #1995 - XDS - Convert xivo-agid to docker container
  - #2002 - XDS - Convert xivo-confgend to docker container
  - #2024 - XDS - Convert rabbitmq to docker container
  - #2009 - XDS - MDS installation using script installer
  - #2013 - XDS - Be able to use xivo-service on MDS
  - #2028 - XDS - Add dockerized rabbitmq as part of xivo-service
  - #2039 - XDS - Confgend container does not restart if pid file is present
  - #2040 - XDS - confgend, agid and rabitmq does not have the correct timezone/localtime
- #2010 - XDS - Enable and configure RabbitMQ federation plugin
- #2030 - XDS - Split xivo-config and rabbitmq-server
- #1990 - XDS - Should be able to replicate cel and queue_log

  Note: Container *db_replic* replicate the `cel` and `queue_log` tables from the Media Server towards the Reporting Server.

- #2000 - XDS - Generate Group/Queue/Meetme configuration for a MDS

System

- #1997 - Add Debian Stretch to PXE

2018.11

Consult the 2018.11 Roadmap.

Components updated: **config-mgt**, **xivo-backup**, **xivo-bus**, **xivo-sysconfd**, **xivo-web-interface**, **xucmgt**, **xuc-server**

CCAgent

- #1914 - [UGI] Popup on when call is on hold for a long time
- #1994 - Open sheet url in multiple tabs

  Important: Behavior change See *Screen Popup* and *Third Party Integration API* for more informations.

CCManager

- #1986 - Warning message when queue recording feature is not available does not appear in right state

Web Assistant

- #1996 - Exported personal contacts can't be imported if commas are missing at line ends
- #2005 - Web assistant - handle automatic reconnection from different networks

XUC Server

- #1896 - Agent penalty update in Webi is not propagated in CC Manager

XiVO PBX

- #1925 - Make xivo-backup write temporary backup file in /var/tmp instead of /tmp
- #1998 - XDS - Reload configurations using rabbitmq
- #2015 - Unable to manage agents by Xuc (and from ccmanager) because of sysconfd answer

2018.10

Consult the 2018.10 Roadmap.

Components updated: **asterisk**, **xivo-agid**, **xivo-confd**, **xivo-config**, **xivo-dao**, **xivo-db-replication**, **xivo-full-stats**, **xivo-manage-db**, **xivo-upgrade**, **xivo-web-interface**, **xucmgt**, **xucserver**

CCAgent

- #1953 - Sort is not working on callback view in CCAgent
- #1954 - Callback period change when taking callback in CCAgent
- #1956 - One Callback still displayed in CCAgent when processing is finished

CCManager

- #1928 - [ITLS] Display intermediate switch status when queues are not all recorded
- #1933 - CCmanager popup display queue information when clicking on name

Fingerboard

- #1854 - Configmgt must be accessible from fingerboard

Reporting

- #1955 - Specific agent stats takes too much time to compute
- #1971 - Missing rights for user stats on new tables "qualifications, qualifications_answer, subqualifications, transfert" in xivo_stats database
- #1972 - Add replication of "queue_qualification"

Web Assistant

- #1348 - Handle custom line in Web Assistant and CC Agent
- #1792 - personal contact accentuated chars are not encoded properly when exported on Windows
- #1960 - Support Aastra phone using the web assistant

XUC Server

- #1921 - 3rd party application should allow setting pause type
- #1936 - Rabbitmq message seem to not be processed on each instance of xuc if two xuc are running

XiVO PBX

- **Asterisk**: Ship version 13.19.0 of asterisk (see #1981) with patch for:
  - #1970 - Insert REASON of QueueMemberPause AMI command to data1 of PAUSEALL queue_log event
- #1948 - Incoming call to user are denied if user has a denied all
- #1949 - On internal calls, wrong call rights are applied

Important: Behavior change See notes on *Internal Calls* and *Forwarded Calls* in *Call Permissions* documentation page for additional information.

- #1950 - Call rights check fails when calling from external an user with a custom line
- #1951 - Cannot add CTI information (profile, login . . .) to a user through a CSV file update

- #1952 - Wrong line status for user imported via CSV with a sip_username in uppercase

> **Important: Behavior change** When importing users with a CSV file, `sip_username` in uppercase will be set to lowercase during import.

- #1963 - Configmgt is not upgraded in case of minor upgrade (e.g. from 2018.05.XX to 2018.05.YY)
- #1966 - Can't download backups files when XiVO UC is installed on XiVO
- #1967 - Can't access administration page through an SSH redirection when XiVO UC is installed
- #1969 - SQL error when trying to filter queue skills
- #1973 - User stats can't access tables in xc schema

2018.09

Consult the 2018.09 Roadmap.

Components updated: **config-mgt**, **recording-server**, **xivo-config**, **xivo-dao**, **xivo-db-replication**, **xivo-full-stats**, **xivo-lib-python**, **xivo-manage-db**, **xivo-upgrade**, **xivocc-installer**, **xucmgt**, **xucserver**

CCAgent

- #1912 - Callback list incorrect when multiple lists on the same queue
- #1927 - Web Agent Callback icon : The notification of existing callback is not present anymore (Borealis port)

CCManager

- #1790 - [ITLS] Display recording status for each queue on CCManager
- #1870 - [ITLS] Dynamic recording queue config notification

Recording

- #1791 - [ITLS] Can manage recording of queues using xivo web interface and cc manager
- #1888 - [ITLS] Add recording rights to Superviseur profile

> **Important: Behavior change** RightProfile Cti Message has been enriched with list of permission access

Reporting

- #681 - xivo replic stops to replicate in case of network problem (routing, iptables...)
- #1924 - Agent statistic (stat_agent_specific) conversation_time abnormally long (port to borealis)
- #1935 - Transfered call offered to and answered by an agent are not counted

XUC Server

- #1920 - Use AMI instead of xivo-ctid to set agent pause reason

XiVO PBX

- #1926 - Remove outgoing call redirection database migration script fails if user contains non-ascii character

2018.08

Consult the 2018.08 Roadmap.

Components updated: **config-mgt**, **recording-server**, **xivo-config**, **xivo-dao**, **xivo-web-interface**, **xucmgt**, **xucserver**

CCManager

- #1815 - Rename statistic "Divert." on ccmanager(queue_view)

> **Important: Behavior change** In CC Manager, in view Queues, column "Divert." was renamed to "Time-out".

- #1890 - Call qualification export should be completely in JSON

Config mgt

- #1900 - Cannot give the role Teacher to an user in configmgt

Recording

- #1874 - Add audio control to recording server
- #1881 - Be able to download consultation logs in recording server

> **Note:** See *Recording Access log*.

- #1882 - Ordering of agents and queues in description of a transfered call recording

XUC Server

- #1871 - High memory & cpu usage triggered by call loops and outbound queue calls
- #1903 - Disable AMI log in xucserver
- #1906 - Can't download callback tickets in ccmanager
- #1909 - Ws result future are sometimes not handled if they fail and so errors are swallowed

> **Note:** When a phone device is not reachable from the Xuc server, an error is reported in the log file.

XiVO PBX

- **Asterisk**: Ships with asterisk 13.19.0 with patch for issue #1846
- #1846 - Asterisk freeze if attended transfer from desktop assistant is hangup during initialization
- #1788 - Button for mass synchronization disappears from web interface
- #1873 - Outgoing caller id and call permissions have wrong user id when unc. fwd
- #1876 - Recording is not stopped when call is direct transferred from phone
- #1899 - Doc - Live reload deactivation implications

> **Note:** See new table in *General settings* section.

- #1907 - Support CAS with SSO

> **Note:** See *CAS SSO Authentication Configuration*.

XiVO Provisioning

- #1893 - Can't provisionning yealink dect with firmware v81

XiVOCC Infra

- #1734 - Docker XiVO CC images uses deprecated base layers and must be updated

Note: Gain of at least 1 GB on downloaded docker images

Important: **Behavior change** Application logs in docker are now gzipped by default when rotation happens.

2018.07

Consult the 2018.07 Roadmap.

Components updated: **config-mgt**, **recording-server**, **xivo-config**, **xivo-manage-db**, **xivo-web-interface**, **xucmgt**, **xucserver**

CCManager

- #1751 - Enable/disable recording on queue from CCManager

Note: See *CC Manager Queue Recording* feature documentation.

Recording

- #1809 - Can stop/start recording after xfer to queue according to queue configuration

Important: **Behavior change** Call to a queue may have its recording stopped after a transfer. See *Recording* feature documentation.

- #1859 - Outgoing transfer doesn't have the correct recording audio file link in list of recorded files

XiVO PBX

- #1830 - Incorrect rights for stats role on qualifications table after upgrade
- #1844 - Destination list hidden when creating a func key with boss-secretary filter
- #1850 - Voicemail Email - For a given voicemail, customizing email body doesn't accept some "ponctuation" characters

Important: **Behavior change** Some new error messages in XiVO web interface are displayed in the default language of web browser. The language can be different.

- #1868 - Web interface doesn't indicate that system configuration must be applied

XiVOCC Infra

- #1877 - Update Java openjdk docker images for Play applications

Note: Gain of at least 1 GB on downloaded docker images

2018.06

Consult the 2018.06 Roadmap.

Components updated: **config-mgt**, **recording-server**, **xivo-agid**, **xivo-config**, **xivo-dao**, **xivo-dist**, **xivo-manage-db**, **xivo-solutions-doc**, **xivo-web-interface**, **xivocc-recording**, **xucmgt**, **xucserver**

CCAgent

- #1802 - Cannot remove activity from favorites when showQueueControls is not set
- #1836 - Be able to use XUC API from xucmgt 3rd party app

CCManager

- #1838 - Call Qualifications: export as csv

Config mgt

- #1831 - Use Config Mgt with HTTPS protocol

Important: Behavior change All calls to configmgt API (callbakcs, user rights...) should now be prefixed by *configmgt*. For examples for requesting callbacks you would request *api/1.0/callback_lists* now you have to request *configmgt/api/1.0/callback_lists*.

- #1837 - Disable PlayFramework CSRF protection for REST API request with X-Auth-Token
- #1856 - configmgt redirection to login is not working

Recording

- #1807 - Enable recording (in dialplan) according to queue configuration

---

Important: Behavior change

- When upgrading it will replace xivocc-recording.conf file. You need to set again the `ipbx_name` variable to the previous value and reload the dialplan.
- Queue recording MUST now be set in the queue configuration. Following the upgrade you MUST edit the queues, remove the subroutine used to start the recording and replace them by the correct configuration in the queue (see *Recording configuration*).

- #1816 - As supervisor I don't see disk space usage
- #1821 - Recording disk usage is inverted in log
- #1824 - Cannot use recording control on web agent during an outgoing call or incoming call
- #1839 - Recording call status is not always reported on acd outbound calls

SpagoBi

- #1848 - java null key when executing the spagobi report "Stat_queue"

Web Assistant

- #1800 - UC assistant should not allow to hold second call if one is already holded

XUC Server

- #1840 - Connected line number and name is not always correct
- #1841 - Channel vars not propagated from queue to agent (borealis port)
- #1852 - XuC user right is not using correct url for configmgt

XiVO PBX

- #1803 - Add borealis in the named distribution of xivo-dist
- #1813 - Qualifications table not created on fresh xivo install

CHAPTER 12

Indices and tables

- genindex
- search

Index

C

ctiserver, 213

D

devices, 220

I

interconnections, 253, 255, 256, 258

M

mail, 86

N

network, 87

U

users, 331

V

VLAN, 87

W

wizard, 13