XiVO Solutions Documentation

Avencall

Jun 09, 2026

CONTENTS

1 Introduction **3**
1.1 XiVO History . . . 3
1.2 GDPR . . . 4

2 Getting Started **5**

3 Installation & Upgrade Guide **11**
3.1 XiVO Installation & Upgrade . . . 11
3.2 XiVOcc Installation & Upgrade . . . 64
3.3 XiVO Distributed System . . . 92
3.4 XiVO Resilience PRO . . . 103

4 Administrator's Guide **125**
4.1 XiVO Administration . . . 125
4.2 XiVOcc Administration . . . 370
4.3 XiVO Connect Administration . . . 481
4.4 Troubleshooting . . . 484

5 User's Guide **501**
5.1 UC Assistant . . . 501
5.2 Switchboard . . . 532
5.3 Desktop Applications . . . 540
5.4 Mobile Application . . . 550

6 Addons PRO **587**
6.1 XiVO PBX . . . 587
6.2 XiVO UC . . . 605
6.3 XiVO CC . . . 615
6.4 XiVO Edge . . . 618
6.5 Meeting Rooms . . . 647
6.6 Mobile Application . . . 661
6.7 XiVO Connect . . . 677
6.8 Metabase . . . 680
6.9 XiVO Resilience . . . 683

7 API and SDK **685**
7.1 High-Level APIs . . . 687
7.2 Low-Level & Admin APIs . . . 820

8 Telemetry **885**
8.1 Usage Writer . . . 885
8.2 Usage Collector . . . 885
8.3 System Architecture and Data Flow . . . 885
8.4 Data retention . . . 885

9 Contributing **887**
9.1 Contributing to the Documentation . . . 887
9.2 Debugging Asterisk . . . 890
9.3 Debugging Daemons . . . 893
9.4 Generate your own prompts . . . 894
9.5 XiVO Guidelines . . . 895
9.6 Debian packaging for XiVO . . . 896
9.7 Profiling Python Programs . . . 896
9.8 Style Guide . . . 898
9.9 Translating XiVO . . . 904
9.10 XiVO Package File Structure . . . 904
9.11 CTI Server . . . 906
9.12 Diagrams . . . 945
9.13 Provisioning . . . 945
9.14 Web Interface . . . 955
9.15 Community Documentation . . . 956
9.16 Experimental Features . . . 958
9.17 Deprecated Features . . . 959

10 Release Notes **993**
10.1 Quantum . . . 993
10.2 Quantum Bugfixes Versions . . . 1016
10.3 Quantum Intermediate Versions . . . 1020

11 Indices and tables **1027**

Index **1029**

Important: What's new in this version ?

Features

- **User Presence**: Do Not Disturb (DND) status visibility
- **WebRTC**:
  - Audio device testing within the application
  - Yealink headsets call control support (WebHID)
- **Simultcalls**: calls emitted by user are taken into account for the *Simultaneous Calls Limit*
- **Resilience** - see *Resilience Architecture*:
  - can chose between Automatic or Manual failover
  - new Resilience management page in XIVO PBX Admin Web interface
  - with Edge, users can automatically reconnect (or with a refresh) to the active stack in case of failure
- **Debian 13 (Trixie)**: Migration for XiVO PBX, XiVO CC, Edge, and Meeting Rooms
- **Metabase BI**: New business intelligence tool with modern dashboards for agents, queues, and groups (replaces SpagoBI)
- **Schedule**:
  - Complete CRUD API for schedule management
  - in XiVO PBX Admin Web Interface schedules can now be filtered and sorted
- **Meetingroom**: User joining via sharing link joins as guest

Deprecations

- End of Support for **LTS Kuma (2023.05)**
- **SpagoBI** deprecated in favor of Metabase (frozen at version 2024.05.00)

See *Quantum* page for the complete list of **New Features** and **Behavior Changes**.

XiVO solutions 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

ONE

INTRODUCTION

XiVO solutions is a suite of PBX applications based on several free existing components including Asterisk Play Pekko 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 was owned by Avencall SAS after a merge between Proformatique SARL and Avencall SARL in 2010. Avencall was acquired by Wisper group in 2020. Subsequently, in 2024, Avencall was further acquired by Volaris group.

The XiVO core team now works for Volaris in Dardilly (France)

- 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.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.
- XiVO solutions 2017.11 in October 2017, second LTS release known as Polaris
- XiVO solutions 2018.05 in April 2018, third LTS release known as Aldebaran,
- XiVO solutions 2019.16 in October 2018, 4th LTS release known as Borealis,
- XiVO solutions 2019.05 in April 2019, 5th LTS release known as Callisto,
- XiVO solutions 2019.12 in October 2019, 6th LTS release known as Deneb,

- XiVO solutions 2020.07 in April 2020, 7th LTS release known as Electra that set Five version as out of support,
- XiVO solutions 2020.18 in October 2020, 8th LTS release known as Freya that set Polaris version as out of support,
- XiVO solutions 2021.07 in April 2021, 9th LTS release known as Gaia running on Asterisk 18 that set Aldebaran version as out of support,
- XiVO solutions 2021.15 in October 2021, 10th LTS release known as Helios that set Borealis version as out of support,
- XiVO solutions 2022.05 in April 2022, 11th LTS release Known as Izar that set Callisto version as out of support,
- XiVO solutions 2022.10 in October 2022, 12th LTS release Known as Jabbah that set Deneb version as out of support,
- XiVO solutions 2023.05 in April 2023, 13th LTS release Known as Kuma that set Electra version as out of support,
- XiVO solutions 2023.10 in October 2023, 14th LTS release Known as Luna that set Freya version as out of support,
- XiVO solutions 2024.05 in April 2024, 15th LTS release Known as Maia that set Gaia version as out of support,
- XiVO solutions 2024.10 in October 2024, 16th LTS release Known as Naos that set Helios version as out of support,
- XiVO solutions 2025.05 in April 2025, 17th LTS release Known as Orion that set Izar version as out of support,
- XiVO solutions 2025.10 in October 2025, 18th LTS release Known as Pollux that set Jabbah version as out of support,
- XiVO solutions 2026.05 in April 2026, 19th LTS release Known as Quantum that set Kuma version as out of support,

Next release is on the way, will be called XiVO R* and will be available in October 2026.

1.2 GDPR

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

CHAPTER

TWO

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.

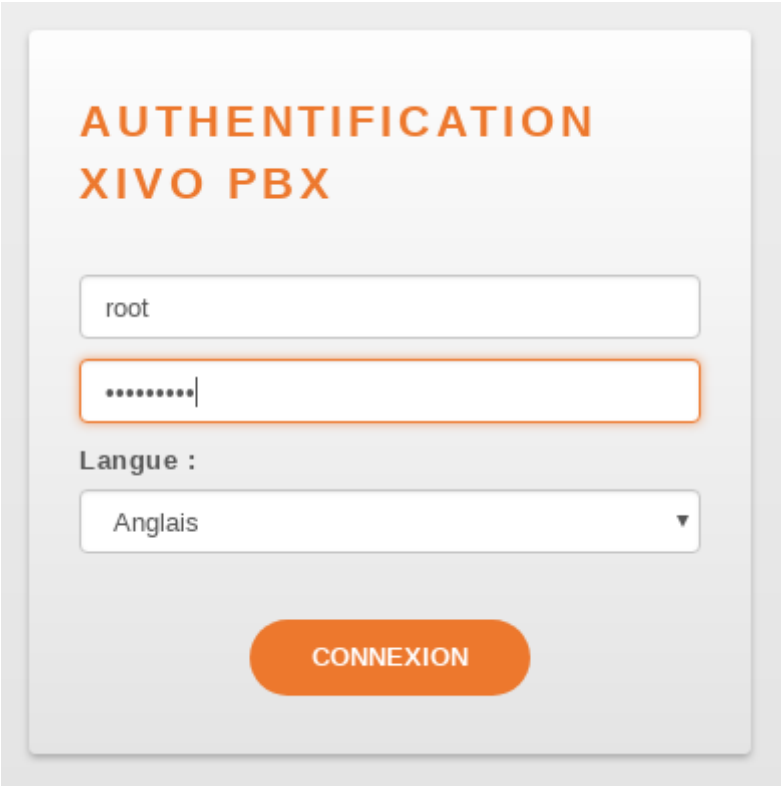


Fig. 1: Logging into the XiVO

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*.

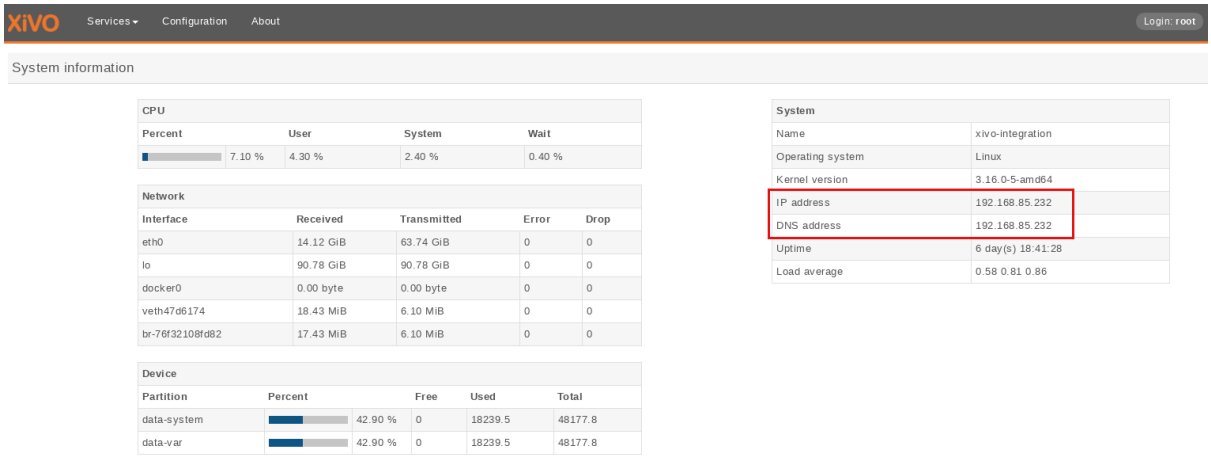


Fig. 2: System informations

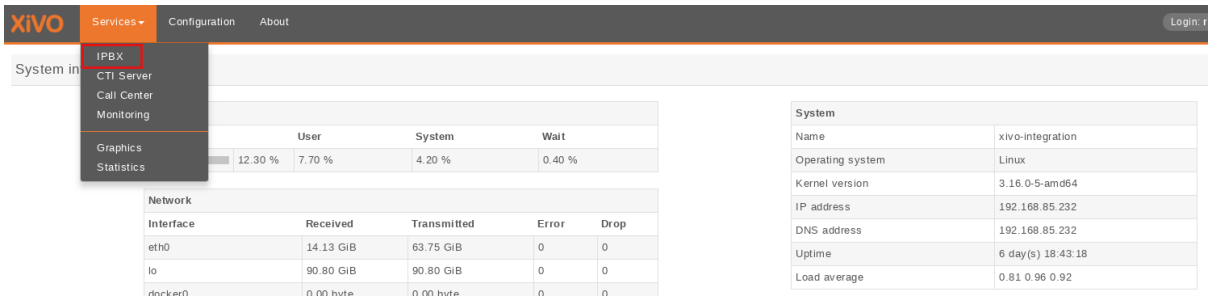


Fig. 3: Menu IPBX

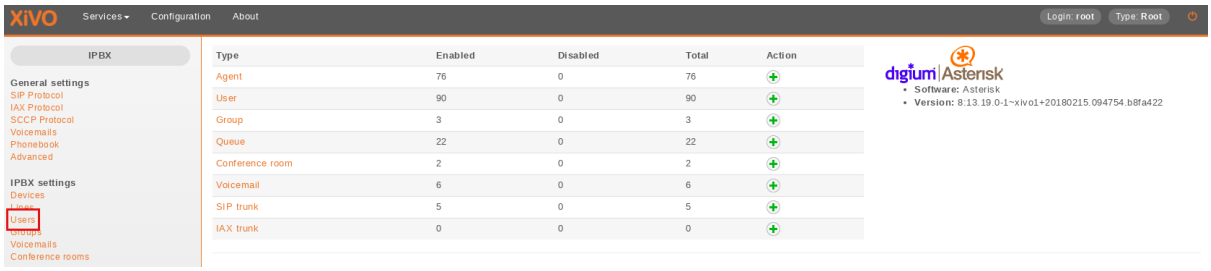


Fig. 4: Users settings

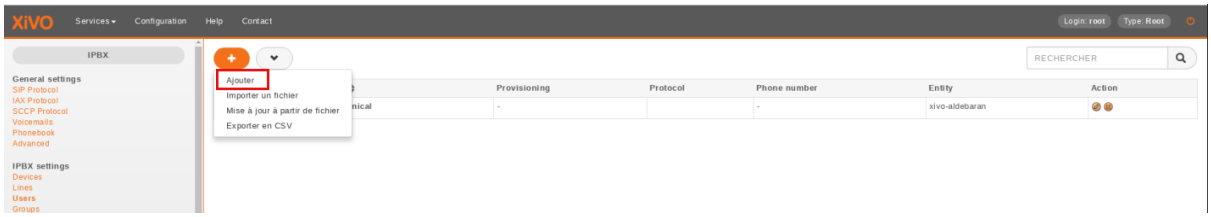


Fig. 5: Adding a new line

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'.

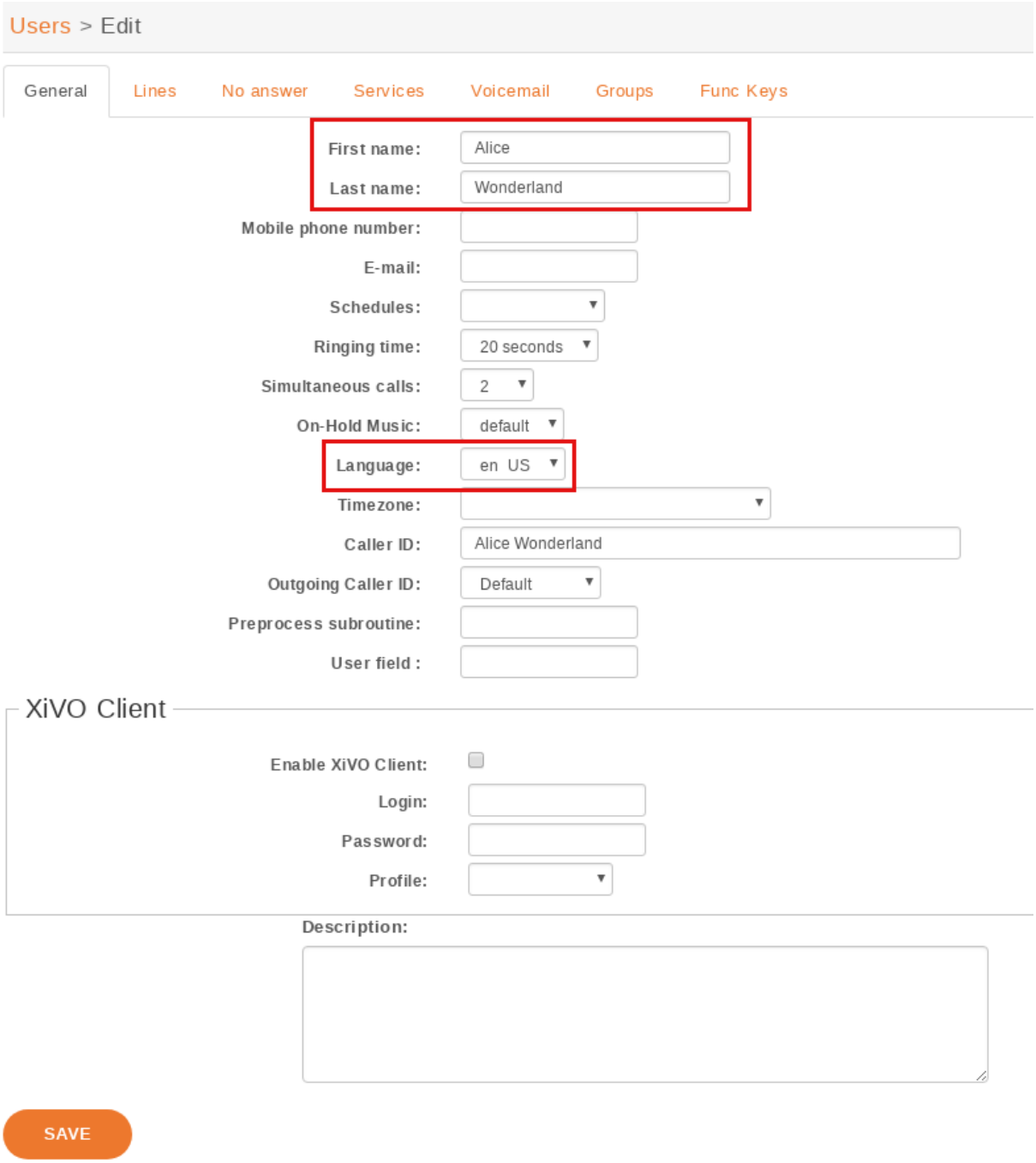


Fig. 6: User information

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'.

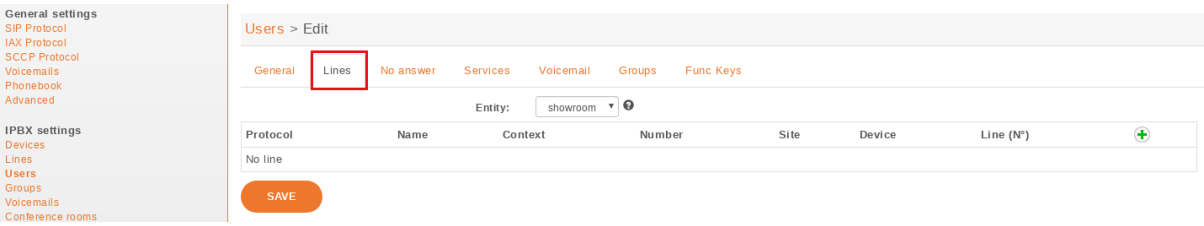


Fig. 7: Lines menu

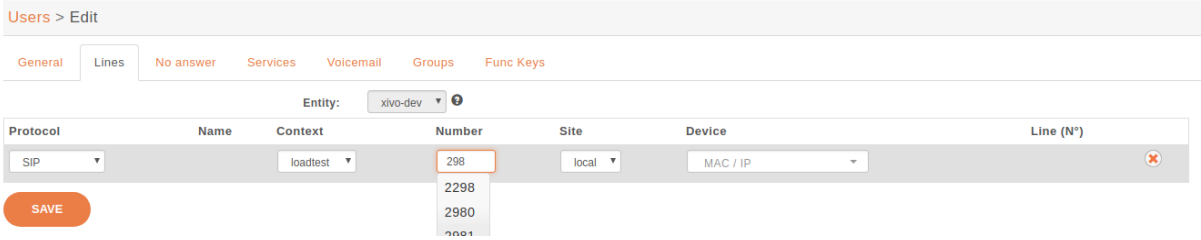


Fig. 8: Line information

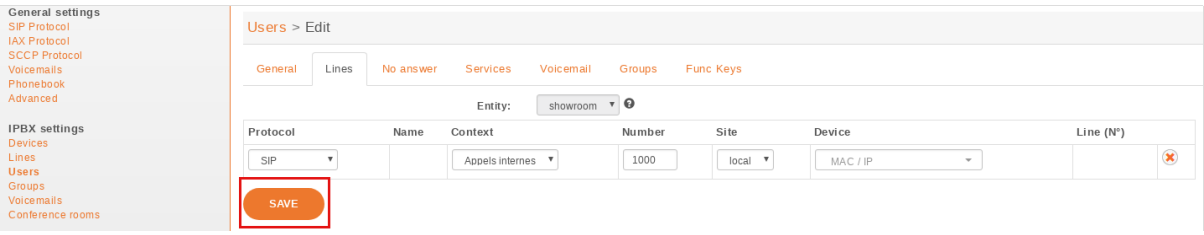


Fig. 9: Save

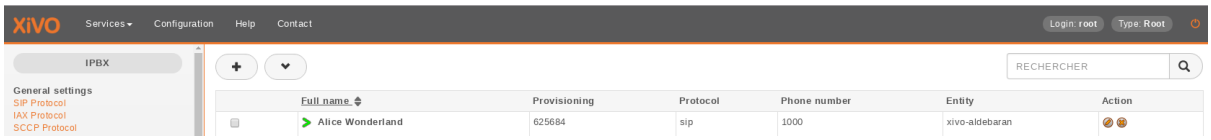


Fig. 10: User added information

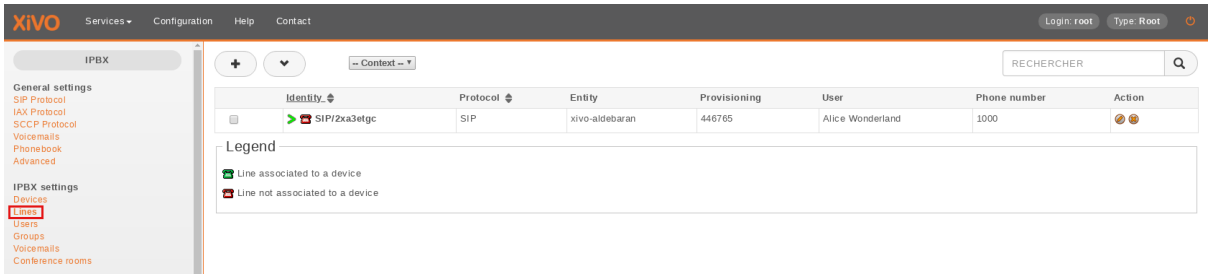


Fig. 11: Lines information

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

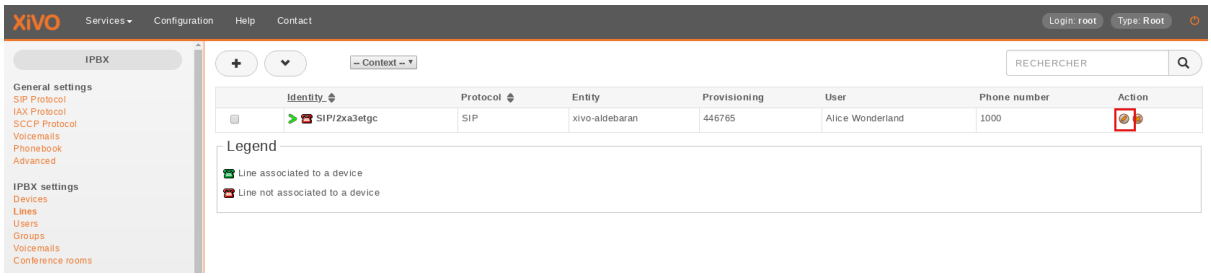


Fig. 12: Edit line

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.

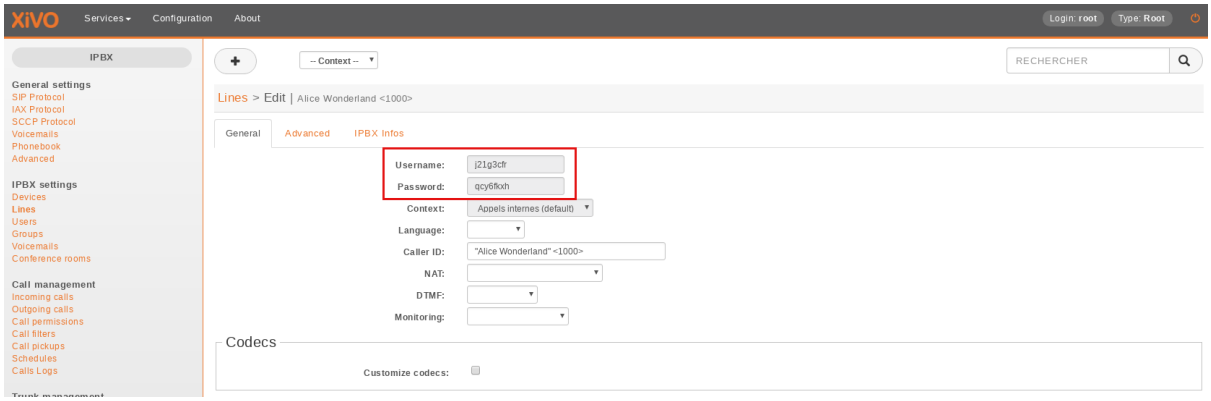


Fig. 13: General line information

CHAPTER

THREE

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

Install XiVO by using a minimal Debian installation and the XiVO installation script as described here.

The debian host may run on virtual (QEMU/KVM, VirtualBox, ...) or physical machines. That said, since Asterisk is sensitive to timing issues, you should 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.

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

Warning: XiVO used to be installable using a dedicated ISO image/PXE. This has been deprecated with Maia (2024.05) so we may focus on the install script.

Requirements

The latest XiVO LTS requires a **64-bit Debian 13 (Trixie)**. It is strongly advised to install on with a clean and minimal installation of that distribution. Retrieve the latest installation image for Debian **Trixie** at https://www.debian.org/releases/trixie/debian-installer.

The installed Debian must:

- not have caps in the hostname
- 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 properly installed, download the XiVO installation script and make it executable:

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

Then run it with:

```
./xivo_install.sh -a 2026.05-latest
```

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

Older 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 2020.18-latest:

```
./xivo_install.sh -a 2020.18-latest
```

When installing an archive version, note that:

- older versions of XiVO may require older versions of Debian too
- 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
```

3.1.2 XiVO PBX Architecture

XiVO PBX Services Links

XiVO PBX Network Table

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

continues on next page

Table 1 – continued from previous page

Daemon Name	Service	Protocol	Port	Listen	Authentication	Enabled
asterisk	SIP	UDP	5060	0.0.0.0	yes	yes
asterisk	SCCP(unsupported)	TCP	2000	0.0.0.0	yes	yes
asterisk	AMI	TCP	5038	0.0.0.0	yes	yes
asterisk	HTTP	TCP	5039	0.0.0.0	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
ntp	NTP	UDP	123	0.0.0.0	no	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
rabbitMQ	AMQP	TCP	25672	0.0.0.0	??	yes
rabbitMQ	AMQP/EPMD	TCP	4369	0.0.0.0	??	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
configmgt	HTTPS	TCP	9100	0.0.0.0	yes	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

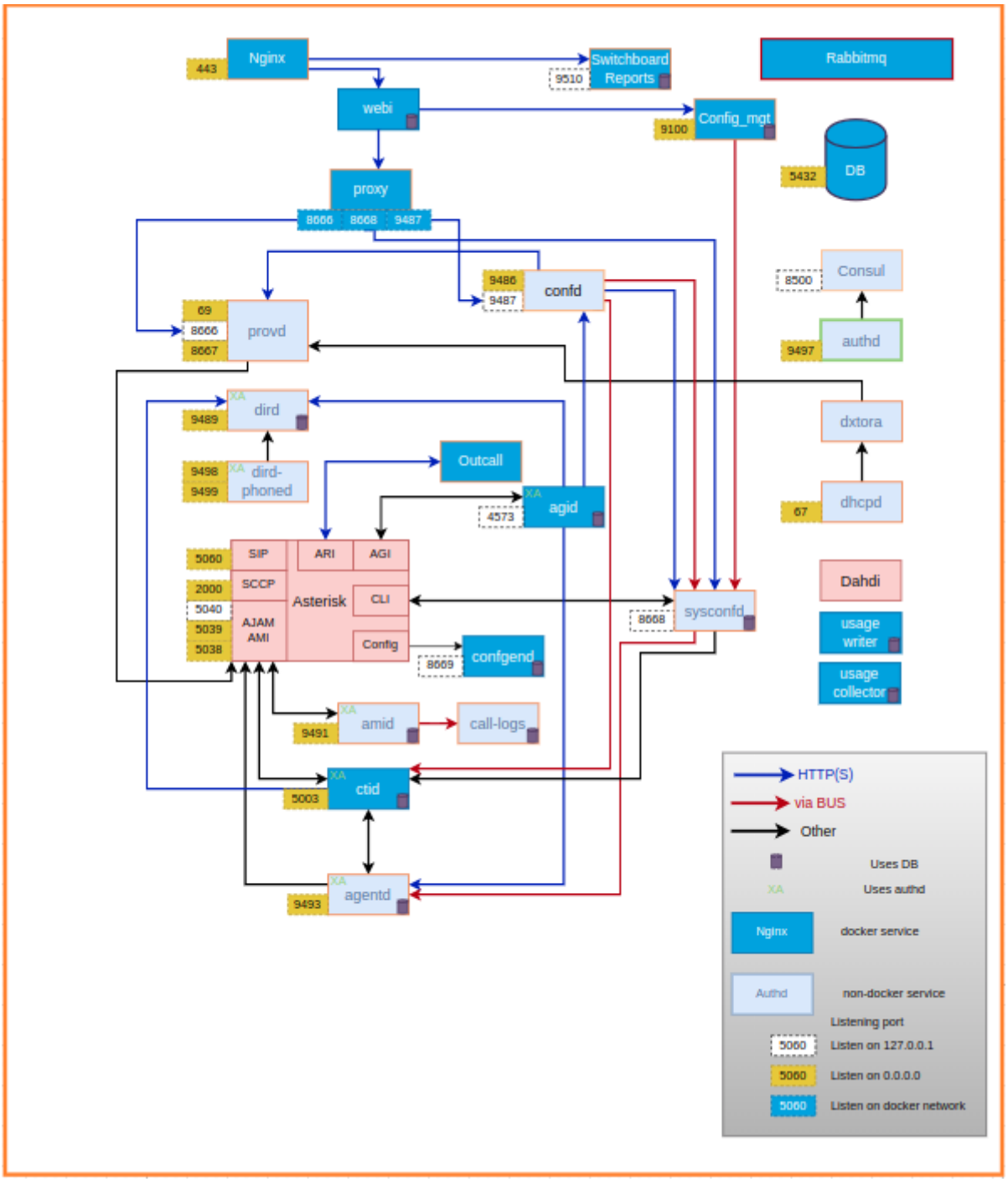
3.1.3 Running the Wizard

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)

An other method for passing the wizard is describe in *Pass Wizard with a JSON*

Language

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


443
Nginx
Switchboard
9510
Reports
Rabbitmq
webi
9100
Config_mgt
5432
DB
proxy
8666
8668
9487
9486
9487
confd
8500
Consul
69
8666
8667
provd
9497
authd
XA
9489
dird
dxtora
9498
9499
XA
dird-
phoned
Outcall
XA
agid
4573
67
dhcpd
5060
SIP
ARI
AGI
2000
SCCP
Asterisk
CLI
5040
5039
5038
AJAM
AMI
Config
8668
sysconfd
Dahdi
usage
writer
usage
collector
8669
confgend
XA
9491
amid
call-logs
XA
5003
ctid
XA
9493
agentd
HTTP(S)
via BUS
Other
Uses DB
XA
Uses authd
Nginx
docker service
Authd
non-docker service
Listening port
5060
Listen on 127.0.0.1
5060
Listen on 0.0.0.0
5060
Listen on docker network

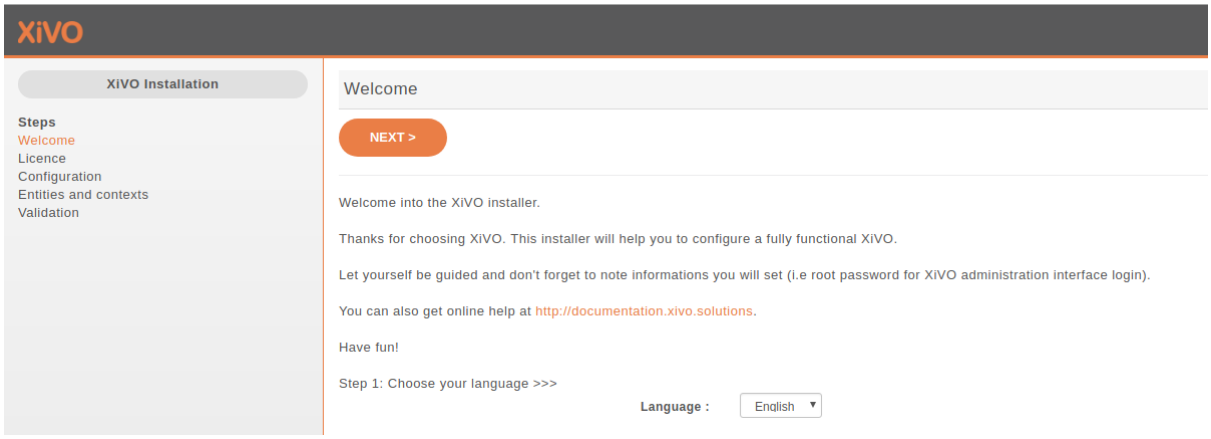


Fig. 1: Select the language

License

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

Configuration

1. Enter the hostname (Allowed characters are : `a-z 0-9 -`, do not user uppercase letter)
2. Enter the domain name (Allowed characters are : `a-z 0-9 - .`, do not user uppercase letter)
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*).

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

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.

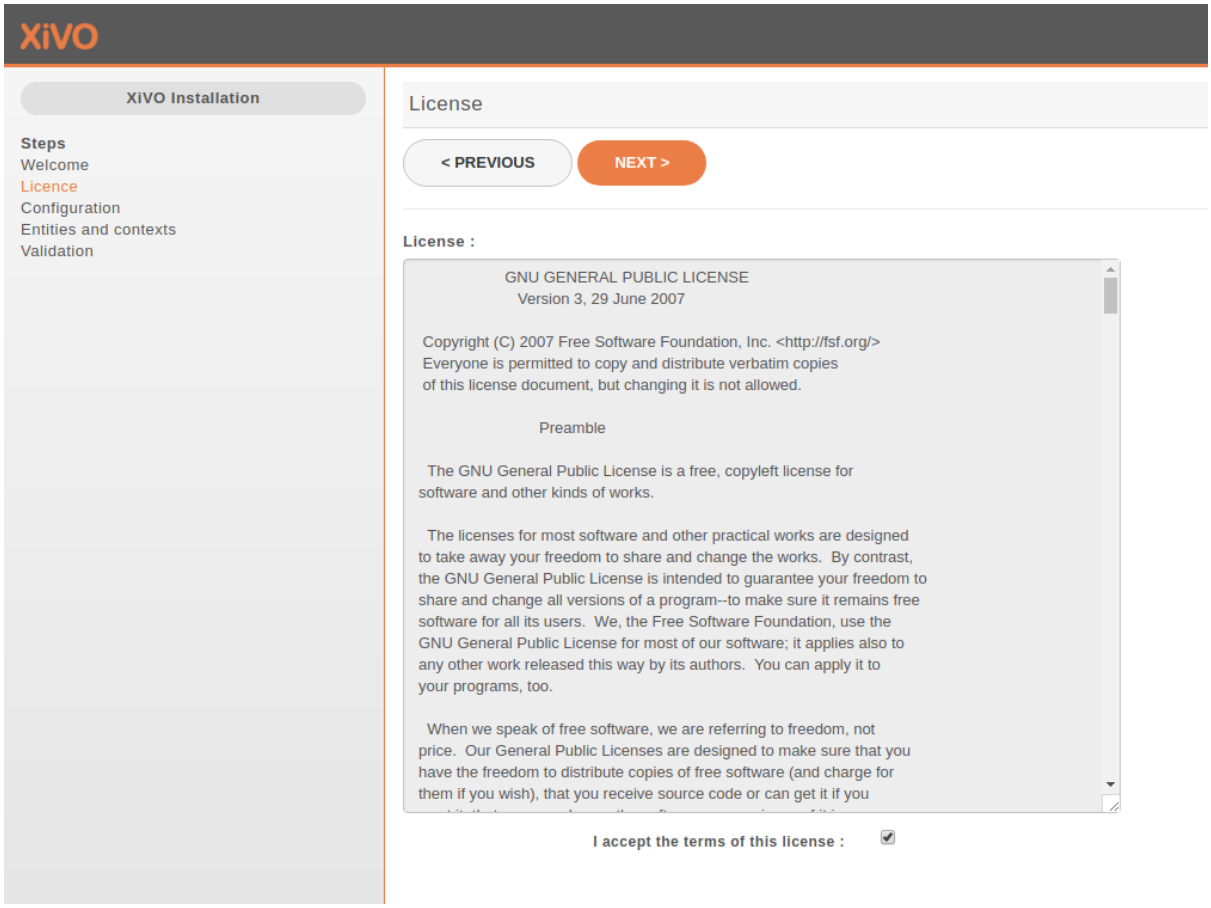


Fig. 2: Accept the license

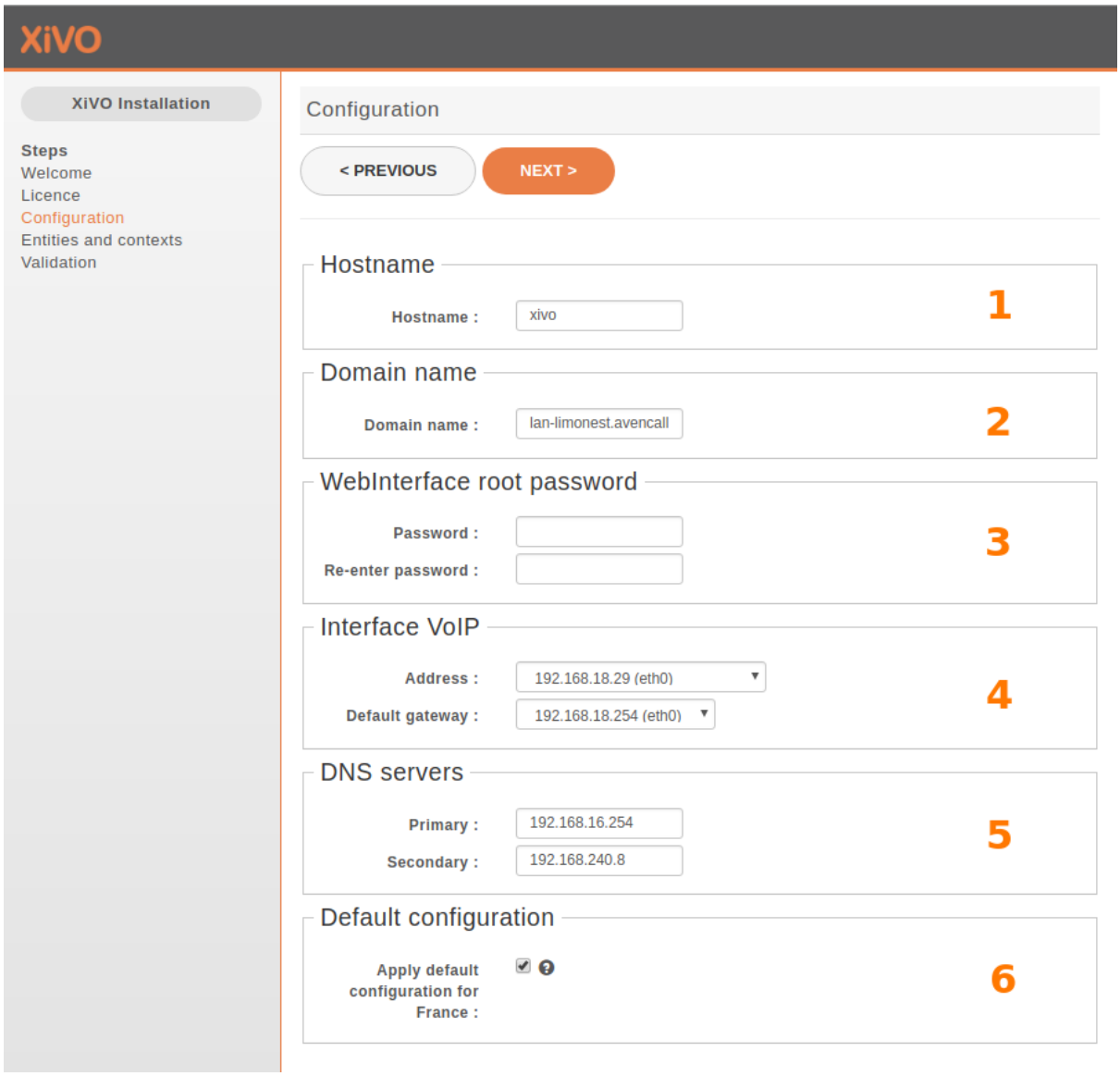


Fig. 3: Basic configuration

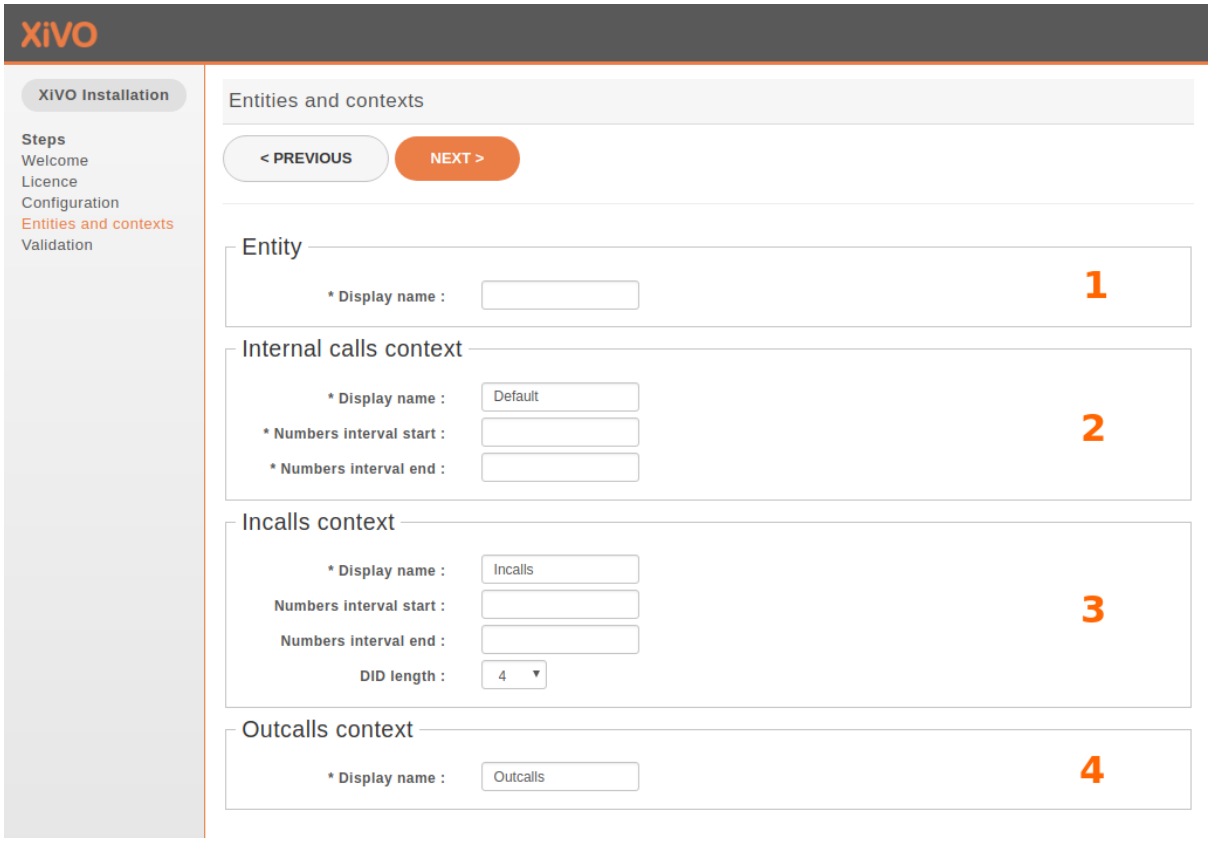


Fig. 4: Entities and Contexts

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*.

3.1.4 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.

3.1.5 Post Installation

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

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

Docker Compose Files

XiVO have some services running under docker container.

XiVO docker configuration is stored in `/etc/docker/xivo` directory. Here the list of all the "standard" files which can be present (depends on what you installed): - `docker-xivo.yml` - `docker-xivo.override.yml` => present if you have XiVOCC or UC installed with XiVO - `docker-xivo-uc.override.yml` ==> present if you have UC installed - `docker-xivo-ivr.override.yml` ==> present if you have IVR Editor installed

We advise you to not touch those files for maintnability reasons (files are replaced in case of upgrade).

If you want to customize docker container definition, you can add files with the following pattern `[0-9]{2}-.*\.override.yml`.

Example : 00-add-something.override.yml

They will be read after all the "standard" files.

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 provider. It can be modified via the *Caller Number Normalization*.

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
```

3.1.6 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

3.1.7 XiVO UC Embedded

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

Prerequisites

Important: On top of the *XiVO PBX requirements* it **MUST** also meet the following requirements:

- **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

Architecture

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

To avoid interactive prompts during installation, pass the required parameters directly. The following variables must be defined in the `/etc/docker/compose/custom.env` file for silent installation:

```
XIVO_HOST=<XIVO_IP_ADDRESS>
XUC_HOST=<XIVOCC_FQDN>
WEEKS_TO_KEEP=<Number_of_weeks>
RESTART_REPLY=<true|false>
```

The `xivouc-installer` script will install packages *xivouc* and *xivocc-docker-components*.

These packages contain these docker compose files:

- `/etc/docker/compose/docker-xivocc.yml` adds UC components which are managed by `xivocc-dcomp` script
- `/etc/docker/xivo/docker-xivo-uc.override.yml` which adds the UC env variable to nginx container so it is started in UC-mode.

Install XiVO UC

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

- either interactively:

  ```
  xivouc-installer
  ```

- or silently (file `/etc/docker/compose/custom.env` must be prefilled with some values - see above):

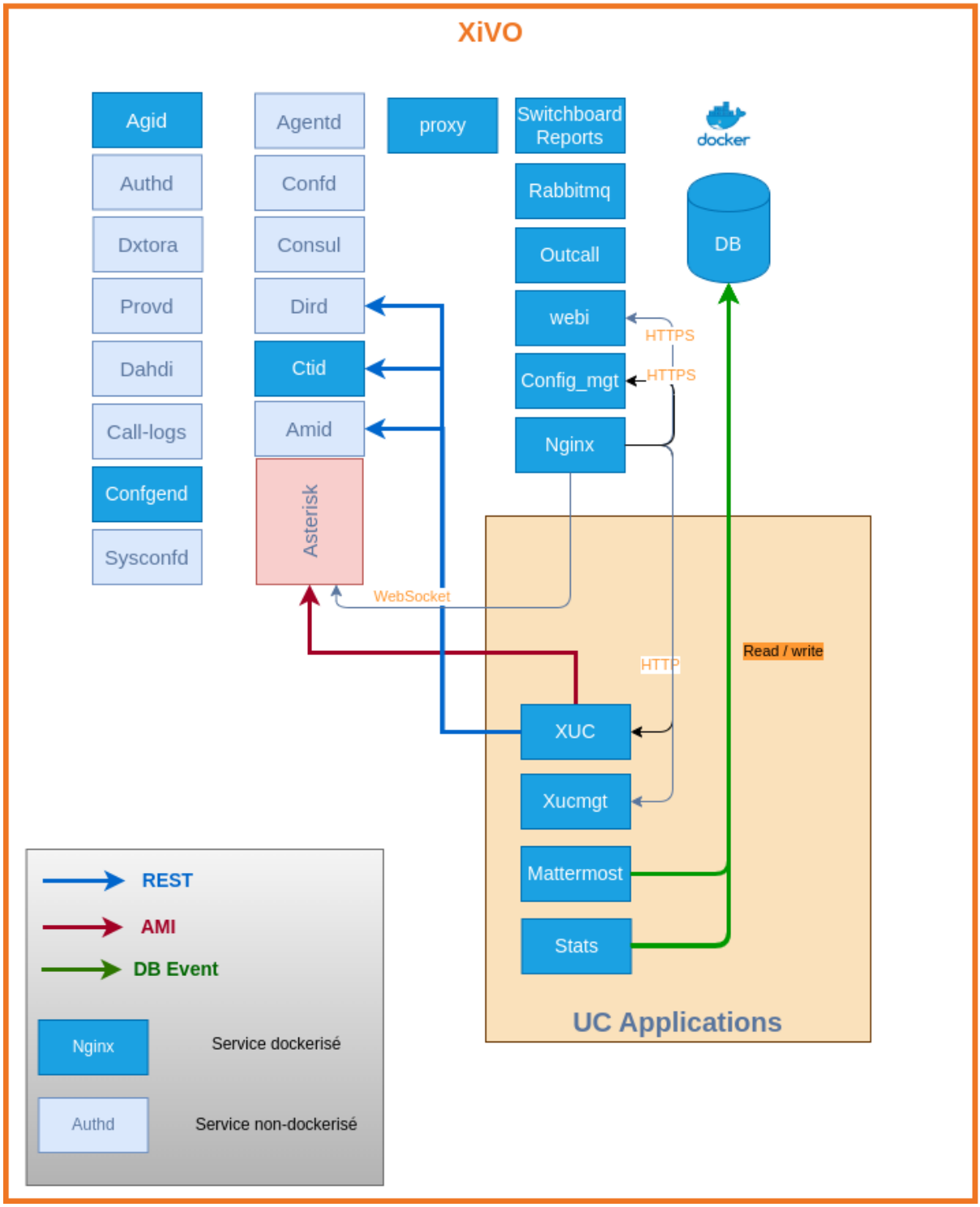
  ```
  xivouc-installer -s
  ```

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
```

XiVO will start with Nginx container configured for XiVO UC.


XiVO
Agid
Authd
Dxtora
Provd
Dahdi
Call-logs
Confgend
Sysconfd
Agentd
Confd
Consul
Dird
Ctid
Amid
Asterisk
proxy
Switchboard Reports
Rabbitmq
Outcall
webi
Config_mgt
Nginx
docker
DB
HTTPS
HTTPS
WebSocket
HTTP
Read / write
XUC
Xucmgt
Mattermost
Stats
UC Applications
REST
AMI
DB Event
Nginx
Service dockerisé
Authd
Service non-dockerisé

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.

Install Chat Backend

Chat Backend

Install trusted certificates

See *Install Trusted Certificate for Nginx (and UC apps in UC mode)*

Upgrade

Packages *xivouc* and *xivocc-docker-components* 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 *UC Assistant* using the XiVO PBX IP address. For example you can access it 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. stop Chat backend: `xivocc-dcomp stop mattermost`
2. purge Chat backend: `apt-get purge xivo-chat-backend`
3. purge the xivouc package: `apt-get purge xivouc`
4. purge DB Replic compose file and monit check: `apt-get purge xivocc-docker-components`
5. remove docker network: `docker network rm xivocc_default`

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

Upgrading

3.1.8 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.

Important: You must stop all services (`xivo-service stop all`) on Media Servers linked to your XiVO Main before the upgrade.

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 Gaia LTS version:

   ```
   xivo-dist xivo-gaia
   ```

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.
5. And then upgrade, see *Upgrading*

Upgrade to latest Bugfix release of an LTS version

After the release of an **LTS** *version* (e.g. *Freya*) we may backport some bugfixes in this version. We will then create a **subversion** (e.g. Freya **.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. Freya.00) to the latest bugfix release (e.g. Freya.04).
2. Verify that the debian sources list corresponds to your *installed LTS* or refix it, for example for Freya:

   ```
   xivo-dist xivo-freya
   ```

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
   ```

Post Upgrade

When finished:

- (*Izar only*) If you're using `chan_sip` on your Izar XiVO you must do again the *Fallback to chan_sip SIP channel driver* procedure.
- 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 etc.

Manual steps for LTS upgrade

This section was moved *here*.

Specific procedures

Resilience Upgrade

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.

Here are the steps for upgrading a cluster, i.e. two XiVO with *Resilience Architecture*:

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

   ```
   xivo-standby:~$ xivo-upgrade
   ```

4. When the standby has finished, start the upgrade on the main:

   ```
   xivo-main:~$ xivo-upgrade
   ```

5. When done, launch the database replication manually:

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

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

Upgrading XDS

Upgrading an XDS implies upgrading all components:

- XiVO (see *XiVO Server Upgrade*)
- Media Server (MDS) (one or more, see *Media Server Upgrade*)

XiVO Server Upgrade

1. Stop all services (`xivo-service stop all`) on **Media Servers** linked to your XiVO Main
2. Perform the upgrade as documented in *Upgrade* section

Media Server Upgrade

Important:

- This procedure must be done on all media servers belonging to the upgraded XDS.
- **Before upgrading a MDS, the MDS Main must be fully upgraded**

The upgrade process requires to run the following command on a shell prompt on each media server.

1. Switch version using `xivo-dist` utility and specifying the LTS or specific version you want to upgrade to. For example:

```
xivo-dist xivo-freya
```

2. Launch the upgrade process. **All MDS services will be stopped during the process**:

```
mds-upgrade
```

3. Update monitoring configuration by launching the script:

```
xivo-monitoring-update-mds
```

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 production repositories

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

- The distribution **xivo-LTS** 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 *all* previous stable Asterisk versions : therefore you can always fallback to one of the previous stable version.

Example

Repository	Distribution	Section	Content
mirror.xivo.solutions/debian	xivo-kuma-candidate	main	latest-built-and-tested-version (e.g. asterisk-20.11.0)
mirror.xivo.solutions/debian	xivo-kuma	main	most-stable-known-version (e.g. asterisk-20.6.0)
mirror.xivo.solutions/debian	xivo-kuma-oldstable	main	all-old-stable-known-version (e.g. asterisk-20.3.0)

Upgrade to asterisk candidate

Before integrating a new version of asterisk in *XiVO PBX*, we first build it and deliver it in the `xivo-LTS-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-LTS-candidate` distribution (see **last line entry** below):

```
# xivo-kuma
deb http://mirror.xivo.solutions/debian/ xivo-kuma main
# deb-src http://mirror.xivo.solutions/debian/ xivo-kuma main
deb http://mirror.xivo.solutions/debian/ xivo-kuma-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: 8:20.6.0-1~xivo2+deb12+20240522.081947.010dbfa
  Candidate: 8:20.11.0-1~xivo1+deb12+20241211.170138.e1c1701
  Version table:
     8:20.11.0-1~xivo1+deb12+20241211.170138.e1c1701 500
        500 http://mirror.xivo.solutions/debian xivo-kuma-candidate/main amd64␣
↪Packages
  *** 8:20.6.0-1~xivo2+deb12+20240522.081947.010dbfa 500
        500 http://mirror.xivo.solutions/debian xivo-kuma/main amd64 Packages
```

(continues on next page)

(continued from previous page)

```
        100 /var/lib/dpkg/status
     8:20.3.0-1~xivo1+deb12+20240311.170138.ea14f89 500
        500 http://mirror.xivo.solutions/debian xivo-kuma-oldstable/main amd64␣
↪Packages
```

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

```
apt-get -t xivo-LTS-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-LTS-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-LTS-oldstable` distribution (see **last line entry** below):

```
# xivo-kuma
deb http://mirror.xivo.solutions/debian/ xivo-kuma main
# deb-src http://mirror.xivo.solutions/debian/ xivo-kuma main
deb http://mirror.xivo.solutions/debian/ xivo-kuma-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: 8:20.6.0-1~xivo2+deb12+20240522.081947.010dbfa
  Candidate: 8:20.11.0-1~xivo1+deb12+20241211.170138.e1c1701
  Version table:
     8:20.11.0-1~xivo1+deb12+20241211.170138.e1c1701 500
        500 http://mirror.xivo.solutions/debian xivo-kuma-candidate/main amd64␣
↪Packages
  *** 8:20.6.0-1~xivo2+deb12+20240522.081947.010dbfa 500
        500 http://mirror.xivo.solutions/debian xivo-kuma/main amd64 Packages
        100 /var/lib/dpkg/status
     8:20.3.0-1~xivo1+deb12+20240311.170138.ea14f89 500
        500 http://mirror.xivo.solutions/debian xivo-kuma-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='8:20.3.0-1~xivo1+deb12+20240311.170138.ea14f89'
```

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-LTS`:

```
xivo-dist xivo-kuma
```

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: 8:20.3.0-1~xivo1+deb12+20240311.170138.ea14f89
  Candidate: 8:20.11.0-1~xivo1+deb12+20241211.170138.e1c1701
  Version table:
     8:20.11.0-1~xivo1+deb12+20241211.170138.e1c1701 500
        500 http://mirror.xivo.solutions/debian xivo-kuma-candidate/main amd64␣
↪Packages
     8:20.6.0-1~xivo2+deb12+20240522.081947.010dbfa 500
        500 http://mirror.xivo.solutions/debian xivo-kuma/main amd64 Packages
  *** 8:20.3.0-1~xivo1+deb12+20240311.170138.ea14f89 500
        500 http://mirror.xivo.solutions/debian xivo-kuma-oldstable/main amd64␣
↪Packages
        100 /var/lib/dpkg/status
```

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

```
apt-get -t xivo-LTS 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

Debian 10 (Buster) Upgrade Notes

Debian was upgraded to Debian 10 (Buster) in XiVO 2020.10 release.

Warning: Upgrade from versions *earlier* than XiVO Boréalis (2018.16) are not supported. You MUST first upgrade to at least XiVO Boréalis (2018.16) or more before upgrading to XiVO Freya.

Before the upgrade

Important: 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.

- It is not possible to upgrade from XiVO below Boréalis (2018.16) version. You first need to upgrade to XiVO Boréalis.
- 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` `pgdg.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/ stretch main
deb-src http://ftp.fr.debian.org/debian/ stretch main

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

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

  If collectd is installed, you will need to do the same with the `/etc/collectd/collectd.conf` 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

No major change to advertise here that would come with XiVO on Debian 10.

External Links

Official Debian 10 Release Notes

Debian 11 (Bullseye) Upgrade Notes

Debian was upgraded to Debian 11 (Bullseye) in XiVO 2022.XX release.

> **Warning:** Upgrade from versions *earlier* than XiVO Freya (2020.18) are not supported. You MUST first upgrade to at least XiVO Freya (2020.18) or more before upgrading to XiVO Izar.

Before the upgrade

Important: **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.

- It is not possible to upgrade from XiVO below Freya (2020.18) version. You first need to upgrade to XiVO Freya.
- 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` `pgdg.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/ bullseye main
deb-src http://ftp.fr.debian.org/debian/ bullseye main

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

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

Changes

No major change to advertise here that would come with XiVO on Debian 11.

External Links

Official Debian 11 Release Notes

Debian 12 (Bookworm) Upgrade Notes

Debian was upgraded to Debian 12 (Bookworm) in XiVO 2022.XX release.

Warning: Upgrade from versions *earlier* than XiVO Freya (2020.18) are not supported. You MUST first upgrade to at least XiVO Freya (2020.18) or more before upgrading to XiVO Izar.

Before the upgrade

Important: **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.

- It is not possible to upgrade from XiVO below Freya (2020.18) version. You first need to upgrade to XiVO Freya.
- 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` `pgdg.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/ bookworm main
deb-src http://ftp.fr.debian.org/debian/ bookworm main

deb http://security.debian.org/ bookworm-security main
deb-src http://security.debian.org/ bookworm-security main

# stretch-updates, previously known as 'volatile'
```

(continues on next page)

(continued from previous page)

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

Changes

No major change to advertise here that would come with XiVO on Debian 12.

External Links

Official Debian 12 Release Notes

Debian 13 (Trixie) Upgrade Notes

Debian was upgraded to Debian 13 (Trixie) in XiVO 2026.XX release.

Warning: Upgrade from versions *earlier* than XiVO Maia (2024.05) are not supported. You MUST first upgrade to at least XiVO Maia (2024.05) or more before upgrading to XiVO Maia.

Before the upgrade

Important: 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.

- If using a separate `/boot` partition check there is enough space - see Ensure /boot has enough free space
- It is not possible to upgrade from XiVO below Maia (2024.05) version. You first need to upgrade to XiVO Maia.
- 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` `pgdg.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://deb.debian.org/debian/ trixie main
deb-src http://deb.debian.org/debian/ trixie main

deb http://security.debian.org/debian-security trixie-security main
deb-src http://security.debian.org/debian-security trixie-security main

deb http://deb.debian.org/debian/ trixie-updates main
deb-src http://deb.debian.org/debian/ trixie-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

- Check if network interface name might change - see Network interface names may change

Changes

Major change with XiVO on Debian 13:

- file */etc/sysctl.conf* is no longer read. Use files in */etc/sysctl.d/*.conf* instead - see /etc/sysctl.conf is no longer honored
- file stored in */tmp* are now stored in memory ; **be careful to not store big** files there ! - see The temporary-files directory /tmp is now stored in a tmpfs
- note that the directories /tmp and /var/tmp are now regularly cleaned (note that it is not activated by default after the upgrade)
- `/etc/timezone` file is deprecated and not present on newer Debian 13 install

External Links

Official Debian 13 Release Notes

Outcall to Route Upgrade Guide

This page gives some indication to migrate the Outgoing Calls configuration to the new *Routes introduced in LTS Callisto*.

Deal with the warning in migration script

Could not find an internal context matching the outgoing context

Outcall WAS NOT MIGRATED TO ROUTE because it is deactivated ib db

Deal with your configuration

ELK 7 Upgrade Notes

ELK was upgraded to version 7 in XiVO 2019.10 release.

Data/Dashboard of previous Elastic/Kibana version:

- were backuped before upgrade
- but **were not upgraded**.

This page describes the procedure to

1. activate the old Elastic/Kibana version
2. so it can be used during the time the dashboard are created in the new Kibana
3. and then how to remove backup and old Elastic/Kibana version

Use Previous Version of Totem Panels

With Deneb, new version of the ELK stack is included. The old Kibana panels are disabled, but you can still reactive them manually while loosing access to the new one. To do so, you have to rename the docker compose override file on the *XiVO CC* server and recreate services:

```
mv /etc/docker/compose/docker-xivocc.override.yml.elk17 /etc/docker/compose/docker-
↪xivocc.override.yml
xivocc-dcomp up -d
```

These commands will restart the **old** Elasticsearch and Kibana on the XiVOCC.

Then, on *XiVO PBX*, you need to edit the `/etc/docker/xivo/docker-xivo.override.yml` file to reactivate the replication from `db_replic` to Elasticearch.

```
-    - DISABLEELASTIC
+    - DISABLEELASTIC=false
```

Then you'll need to do a `xivo-dcomp up -d`. This will reacreate the `db_replic` on the *XiVO PBX*.

You'll then be able to access the data using the URL http://XIVOCC_SERVER_ADDRESS/prevKibana

Important: While using the previous version of Totem Panels, these are not accessible from the fingerboard, only through the http://XIVOCC_SERVER_ADDRESS/prevKibana link.

Put Back the New ELK 7

To put back the new version of ELK:

- On *XiVO CC*, remove the override file:

```
rm /etc/docker/compose/docker-xivocc.override.yml
```

- and re-create the containers:

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

- On *XiVO PBX*, set the `DISABLEELASTIC` env variable to `true` directly in the `/etc/docker/xivo/docker-xivo.override.yml`:

```
- DISABLEELASTIC=true
```

- and re-create the containers:

```
xivo-dcomp up -d
```

Cleaning Backup of Old Dashboards

If you're not going to use the previous version of the Elasticsearch or if you're done creating the Dashboard in the new Kibana, please delete the folder with backuped data (on *XiVO CC*):

```
rm -rf /var/local/elasticsearch-1.7/
```

Asterisk `chan_sip` to `pjsip` Migration Guide

During upgrade to Izar, the SIP channel driver is switched from `chan_sip` to `res_pjsip`.

- *Migration Guide*
  - *Manipulating SIP Headers*
    - *Major difference whith PJSIP_HEADER function*
    - *Utility functions in XiVO dialplan*
    - *XiVO header manager for simplified PJSIP header management*
  - *Retrieving peer information*
- *References*

Note: For Izar LTS, rollback to `chan_sip` is supported. For this, see *Fallback to chan_sip SIP channel driver*.

Migration Guide

This section describes some of the checks you have to do and documents some changes you'll have to make.

Manipulating SIP Headers

With `chan_sip` you could use:

- dialplan application SIPAddHeader to add SIP headers
- dialplan application SIPRemoveHeader to remove SIP headers
- and dialplan function SIP_HEADER to read SIP headers

If you use one of them in one of your custom dialplan you **have to change them** to use the dialplan function PJSIP_HEADER (please read carefully the PJSIP_HEADER documentation).

Major difference whith PJSIP_HEADER function

Important: PJSIP_HEADER dialplan function comes with the major difference (compared to SIPAddHeader or SIPRemoveHeader) to operate on the current channel (i.e. the **caller's** channel). See detail below.

The major difference between SIPAddHeader or SIPRemoveHeader and PJSIP_HEADER is that PJSIP_HEADER operates on the current channel (i.e. **caller's incoming** channel): `PJSIP_HEADER(add,X-My-Hdr,2)` will add the header `X-My-Hdr` on the **caller's** channel.

If you want to add a header on the (still to be created) **callee's outgoing** channel, you must use pre-dial handler. See example in PJSIP_HEADER documentation.

Utility functions in XiVO dialplan

Note that currently XiVO supplies utility subroutines to do these read/add/remove action whatever the SIP channel driver:

- `xivo-generic-sip-get-header` which takes the following arg:
  - arg1: the name of the SIP Header to read
  - arg2: the name of the dialplan variable in which the header content will be set
- `xivo-generic-sip-remove-header` which takes the following arg:
  - arg1: the name of the SIP Header to remove

Reading a SIP Header
Old dialplan

```
same  = n,Set(MY_CONTEXT=${SIP_HEADER(X-My-Context)})
```

New dialplan starting from Izar, compatible with `res_pjsip` (and `chan_sip`)

```
same  = n,GoSub(xivo-generic-sip-get-header,s,1(X-My-Context,MY_CONTEXT))
```

Removing a SIP Header

Old dialplan

```
same  = n,SIPRemoveHeader(X-My-Context)
```

New dialplan starting from Izar, compatible with `res_pjsip` (and `chan_sip`)

```
same  = n,GoSub(xivo-generic-sip-remove-header,s,1(X-My-Context))
```

XiVO header manager for simplified PJSIP header management

The module is named `xivo_header_mgr` and is available anywhere in the dialplan. You can call it anywhere in your dialplan to add your headers except in predial handler options `b()`. It works with any SIP channel driver. It will store the modifications and apply then inside a predial handler already set in the default dialplan.

Warning: **Important limitation:** Since the predial handler was added by default in every `Dial` application calls, calling a predial handler in your preprocess subroutines will have the priority over the default one and will most likely break the header manager. You will need to call gosub `set_header_on_channel` in your predial handler to make it work.

Adding a customized preprocess subroutine

Old dialplan

```
[my-pre-dial-handler]
exten = s,1,NoOp()
same = n,...
same = Return()

[my-subr]
exten = s,1,NoOp()
same = n,Set(XIVO_CALLOPTIONS=${XIVO_CALLOPTIONS}b(my-pre-dial-handler))
same = n,Return()
```

New dialplan starting from Jabbah, compatible with `res_pjsip` (and `chan_sip`)

```
[my-pre-dial-handler]
exten = s,1,NoOp()
same = n,...
same = n,Gosub(xivo-user-predial,s,1)
same = Return()

[my-subr]
exten = s,1,NoOp()
same = n,Set(XIVO_CALLOPTIONS=${XIVO_CALLOPTIONS}b(my-pre-dial-handler))
same = n,Return()
```

The following functions are availables :

- `add_header` to add your header to the upcoming `Dial`, which takes the following arg:
  - arg1: the name of the SIP Header to add, with quotes

- arg2: the value of the SIP Header to add, with quotes

Adding a SIP Header
Old dialplan
`same = n,SIPAddHeader(X-My-Context: ${XIVO_BASE_CONTEXT})`
New dialplan starting from Izar, compatible with `res_pjsip` (and `chan_sip`)
`same = n,Gosub(xivo_header_mgr,add_header,1("X-My-Context","${XIVO_BASE_CONTEXT}"))`

Note: If you need to add a piece of dialplan ending with your own call to the `Dial` application, don't forget to add the call to the header manager in the `b()` predial handler, even if you did not add any header during your custom subroutine.

For example :

```
same = n,Dial(PJSIP/qwerty-006a,,b(xivo_header_mgr^set_headers_on_channel^1))
```

Retrieving peer information

With `chan_sip` you could use the dialplan function SIPPEER to retrieve the ip address of the peer or stuff like that.

Dependeing on the use case, a replacement can be found with:

- dialplan function CHANNEL: for example `CHANNEL(pjsip,remote_addr)` will retrieve the peer IP of the current channel
- or with dialplan functions PJSIP_AOR and/or PJSIP_CONTACT. For example, the following will give you the registration status of peer `abcd`:

  ```
  ${PJSIP_CONTACT(${PJSIP_AOR(abcd,contact)},status)}
  ```

References

- https://wiki.asterisk.org/wiki/display/AST/Migrating+from+chan_sip+to+res_pjsip
- https://wiki.asterisk.org/wiki/display/AST/PJSIP+Configuration+Sections+and+Relationships

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: *
```

(continues on next page)

(continued from previous page)

```
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

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 *XiVO CC Architecture* diagram.

3.2.1 Installation

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

- *Overview*
- *Prerequisites*
- *SSH Key configuration*
- *XiVO PBX Restrictions and Limitations*
  - *General Configuration*
  - *Queue Configuration*
  - *User And Agent Configuration*
- *Install process overview*
- *Install from repository*
  - *Configuration setup*
    - *For regular install*
    - *For silent install*
  - *Download installer script*
  - *Launch installation*
  - *XiVO CC reconfigures XiVO PBX*
- *After-install steps*
  - *Configure ntp server*
  - *Adjust xuc memory*
  - *Launch the services*

- *Reinstallation*
- *Known Issues*
- *Checking Installed Version*
- *Using XivoCC*
- *Chat Backend*
  - *Chat Backend Installation*
- *Post Installation*
  - *User Configuration*
  - *Post Installation Check List*
- *Recording*

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:

- 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 subnets 172.17.0.0/16 and 172.18.0.0/16 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
- 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 13** (Trixie), 64 bits.
- ssh key on XiVO CC to access XiVO PBX : see *SSH Key configuration*
- having a *XiVO PBX* server to work with (see below)

The **XiVO PBX** must be:

- having OpenSSH `PermitRootLogin` set to `yes` (you could revert to `no` after installation of XivoCC)
- installed in a compatible version (basically *XiVO PBX* and *XiVO CC* have to be in the **same** version)

- either having **passed the wizard** or you do the reconfiguration later, see *XiVO CC reconfigures XiVO PBX*

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

- the *XiVO PBX* has the IP 192.168.0.1
- the *XiVO CC* server has the IP 192.168.0.2

A part of the XiVO CC install aims to reconfigure XiVO PBX. You can start installing XiVO CC without XiVO PBX ready at first (or installing both in parallel). But if you do so you will need to do the reconfiguration part later, once XiVO PBX is properly set up (XiVO CC cannot work without it). See more details below.

SSH Key configuration

It is mandatory that your XiVO CC can access XiVO PBX through ssh using an ssh key. The ssh key name or type does not matter, use whichever you like. On XiVO CC, you can generate it this way :

```
ssh-keygen
```

You can then use ssh-copy-id once to configure that key as a way to connect to XiVO PBX.

```
ssh-copy-id root@192.168.0.1
```

Note that ssh-copy-id requires the user prompt, PermitRootLogin on the XiVO PBX VM and (obviously) the XiVO PBX root password.

Alternatively, if you do have a ssh key on XiVO PBX, you can manually register XiVO PBX among XiVO CC's known hosts. This does not requires user prompt. Run on XiVO CC :

```
ssh-keyscan 192.168.0.1 | grep -P "^192.168.0.1" >> /root/.ssh/known_hosts
```

If those methods did not work, you may configure the ssh access on XiVO CC to the XiVO PBX with xivocc's ssh key in *~/.ssh/config* with an entry such as

```
Host 192.168.0.1
  IdentityFile ~/.ssh/id_rsa
```

You know it's working well when the following commands executes on XiVO CC with neither errors nor cli prompt :

```
ssh root@192.168.0.1 'echo "hello $(uname -n)"' | cat
```

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 in line* (i.e. asterisk queue parameter *ringinuse*) in XiVO queue advanced configuration
- When creating a new queue, this queue will not appear immediately in *CCAgent* and *CCManager* until the agent or the manager is not relogged to these applications accordingly.
- When deleting an existing queue, this queue will still appear in *CCAgent* and *CCManager* until the Xuc server is not restarted.

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 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 four 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 : *Install from repository*
2. The second part is to execute the after-install steps : *After-install steps*
3. The third part is to install the extra package for the chat : *Chat Backend*
4. The fourth part is to install the extra package for the recording : *Recording*

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.

Configuration setup

For regular install

Beforehand, check that you know the following information, as they will be prompted :

- *XiVO PBX*'s IP address (XIVO_HOST)
- *XiVO CC* IP address (XUC_HOST)
- Number of weeks to keep statistics (WEEKS_TO_KEEP)
- Number of weeks to keep recordings (beware of disk space) (RECORDING_WEEKS_TO_KEEP)

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.

For silent install

You can pass those parameters in the custom.env on your XiVO CC if you don't want to be prompted:

Installation Types

The variable `XIVOCC_TYPE` defines the installation type:

- `full`: Complete installation including all features.
- `app`: Application-only installation including **xuc**, **xucmgt**, and **nginx**.
- `backend`: Installation of reporting and recording modules only.

Variables Common to All Installation Types

The following variables are mandatory regardless of the chosen installation type:

```
XIVOCC_TYPE=<full|app|backend>
XIVO_HOST=<XIVO_IP_ADDRESS>
XUC_HOST=<XIVOCC_IP_ADDRESS>
CONFIGURE_REPLY=<true|false>
RESTART_REPLY=<true|false>
```

- **CONFIGURE_REPLY** tells whether or not to connect on the XiVO PBX and finish its reconfiguration during the CC install.
- **RESTART_REPLY** tells whether or not to restart the XiVO PBX services after the reconfiguration.

According your XiVO PBX is ready (**wizard is passed**), set CONFIGURE_REPLY to true. If you're doing the installations in parallel, set both variables to false and see *XiVO CC reconfigures XiVO PBX* after the silent install.

> **Warning:** The variable XUC_HOST must always be an IP address, never a FQDN

Variables Specific to full Installation

Mandatory:

```
WEEKS_TO_KEEP=<Number_of_weeks>
RECORDING_WEEKS_TO_KEEP=<Number_of_recording_weeks>
```

Optional:

```
XUC_FQDN=<XUC_FQDN>
```

Example `full` Configuration:

```
mkdir -p /etc/docker/compose
echo "XIVOCC_TYPE=full
XIVO_HOST=192.168.10.10
XUC_HOST=192.168.10.20
CONFIGURE_REPLY=true
RESTART_REPLY=true
XUC_FQDN=full.xivocc.local
WEEKS_TO_KEEP=52
RECORDING_WEEKS_TO_KEEP=26" > /etc/docker/compose/custom.env
```

Variables Specific to app Installation

Mandatory:

```
XIVOCC_BACKEND=<XIVOCC_BACKEND_ID_ADDRESS>
```

Optional:

```
XUC_FQDN=<XUC_FQDN>
```

Example app Configuration:

```
mkdir -p /etc/docker/compose
echo "XIVOCC_TYPE=app
XIVO_HOST=192.168.10.10
XUC_HOST=192.168.10.20
CONFIGURE_REPLY=true
RESTART_REPLY=true
XUC_FQDN=app.xivocc.local
XIVOCC_BACKEND=192.168.10.30" > /etc/docker/compose/custom.env
```

Variables Specific to backend Installation

Mandatory:

```
WEEKS_TO_KEEP=<Number_of_weeks>
RECORDING_WEEKS_TO_KEEP=<Number_of_recording_weeks>
```

Example backend Configuration:

```
mkdir -p /etc/docker/compose
echo "XIVOCC_TYPE=backend
XIVO_HOST=192.168.10.10
XUC_HOST=192.168.10.30
CONFIGURE_REPLY=true
RESTART_REPLY=true
WEEKS_TO_KEEP=52
RECORDING_WEEKS_TO_KEEP=26" > /etc/docker/compose/custom.env
```

Download installer script

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

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

Launch installation

```
./xivocc_install.sh -a 2026.05-latest
```

Launch installation in silent mode using the flag -s:

```
./xivocc_install.sh -s -a 2026.05-latest;
```

If you chose not to configure XiVO PBX during the installation, you have one more step to run here *XiVO CC reconfigures XiVO PBX*

XiVO CC reconfigures XiVO PBX

If you chose to reconfigure XiVO PBX with XiVO CC later, this is the additional step you have to do. First verify the following points :

- you installed xivocc by running xivocc_install.sh without errors : *Install from repository*
- you can access XiVO PBX from XiVO CC correctly : *SSH Key configuration*
- your XiVO CC custom.env contains the correct configuration : *Configuration setup*
- your XiVO PBX is ready (*wizard is passed*)

You must then run this on XiVO CC :

> **Warning:** Even if you're installing an edge, XUC_HOST must be the XIVOCC IP at installation time. You'll have to change it to the edge's FQDN later

```
configure-pbx
```

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/ntpsec/ntp.conf`:

```
server 192.168.0.1 iburst
```

Adjust xuc memory

Xuc memory must be increased on these installations:

Condition	XUC Xmx Default (CC)	Required
> 50 agents or > 500 users	2048m	4096m

Configure by setting *JAVA_OPTS_XUC* variable in the `/etc/docker/compose/custom.env` file:

```
JAVA_OPTS_XUC=-Xms512m -Xmx4g
```

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

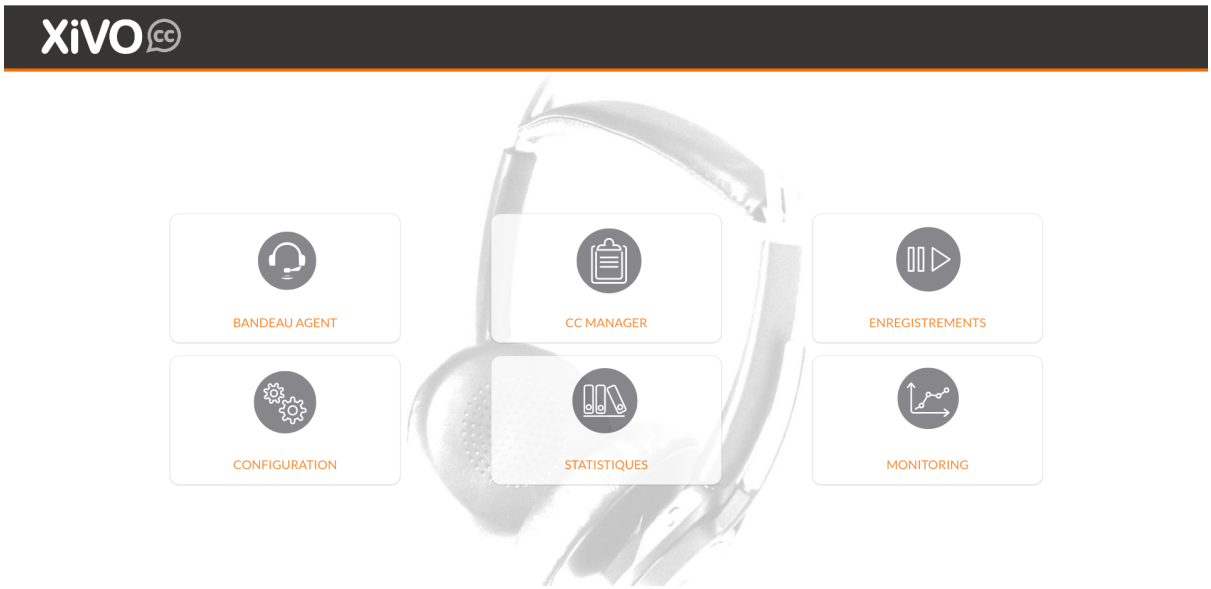
Version of the running docker containers can be displayed by typing (see *Show containers and images versions* for other commands):

```
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: https://192.168.0.2
- SpagoBI: https://192.168.0.2/SpagoBI
- Config Management: https://<XiVO IP Address>/configmgt/ or https://192.168.0.2/configmgt/
- Recording server: https://192.168.0.2/recording
- Fingerboard: https://192.168.0.2/fingerboard

Chat Backend

Since *Electra* version, you **MUST install and configure** the chat backend to have the *Chat feature* working properly.

Installation type:

- *UC Embedded*: the chat backend package must be installed on the XiVO PBX with the UC embedded.
- *CC/UC mono-server*: the chat backend package must be installed on your CC/UC server.
- *CC/UC multi-server*: the chat backend package must be installed on the server which hosts the `xuc`. You will be asked to give the IP Address of the server hosting the `pgxivocc`.

Warning: Installing the Chat backend will configure a linux user on the host with UID 2000. Therefore you should check that no user with UID 2000 (you can do it with command `id 2000`) is existing on the host before installing the Chat backend.

Warning: XiVO CC containers will be recreated. Therefore you must not install the chat backend before initialization of all databases in pgxivocc was completed. DB replication to the stats database must be also completed before installing the chat backend.

Chat Backend Installation

1. Install the `xivo-chat-backend` package on your XIVO CC (on the server hosting the `xuc` server):

```
apt-get install xivo-chat-backend
```

2. When done, run the configuration script:

```
/var/lib/xivo-chat-backend/scripts/xivo-chat-backend-initconfig.sh
```

Note: This will configure

- the database
- the chat backend (currently mattermost server)
- and the link between xuc and mattermost services

Post Installation

User Configuration

You should configure users and their rights in the Configuration manager http://<XiVO IP Address>/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.

Post Installation Check List

- All components are running : xivocc-dcomp ps
- Xuc internal database is synchronized with xivo check status page with https://xivoccserver:8443/
- CCManager is running, log a user and check if you can see and manage queues : https://xivoccserver/ccmanager
- Check database replication status using spagobi system report https://xivoccserver/SpagoBI
- Check that you can listen to recordings https://xivoccserver/recording

Recording

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

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

3.2.2 XiVO CC Architecture

This diagram shows the architecture between the different components inside XiVO CC and also interactions with XiVO PBX components that can be found in details at *XiVO PBX Architecture*.

XiVO CC Network Table

Network Flow table (IN) :

Daemon Name	Service	Protocol	Port	Listen	Authentication	Enabled
-	ICMP	ICMP	-	0.0.0.0	no	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
pgxivocc	SQL	TCP	5443	0.0.0.0	yes	yes
xucmgt	HTTPS	TCP	8070	0.0.0.0	yes	yes
xuc	HTTPS	TCP	8090	0.0.0.0	yes	yes
mattermost	HTTP	TCP	8000	127.0.0.1	no	no
recording-server	HTTPS	TCP	9400	0.0.0.0	yes	yes
metabase	HTTPS	TCP	3000	0.0.0.0	yes	yes
spagobi	HTTPS	TCP	9500	0.0.0.0	yes	no
recording-rsync	HTTP	TCP	873	0.0.0.0	no	yes

3.2.3 Components Configuration

Important: When reading this section, keep in mind the *XiVO CC Architecture* diagram.

Overview

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

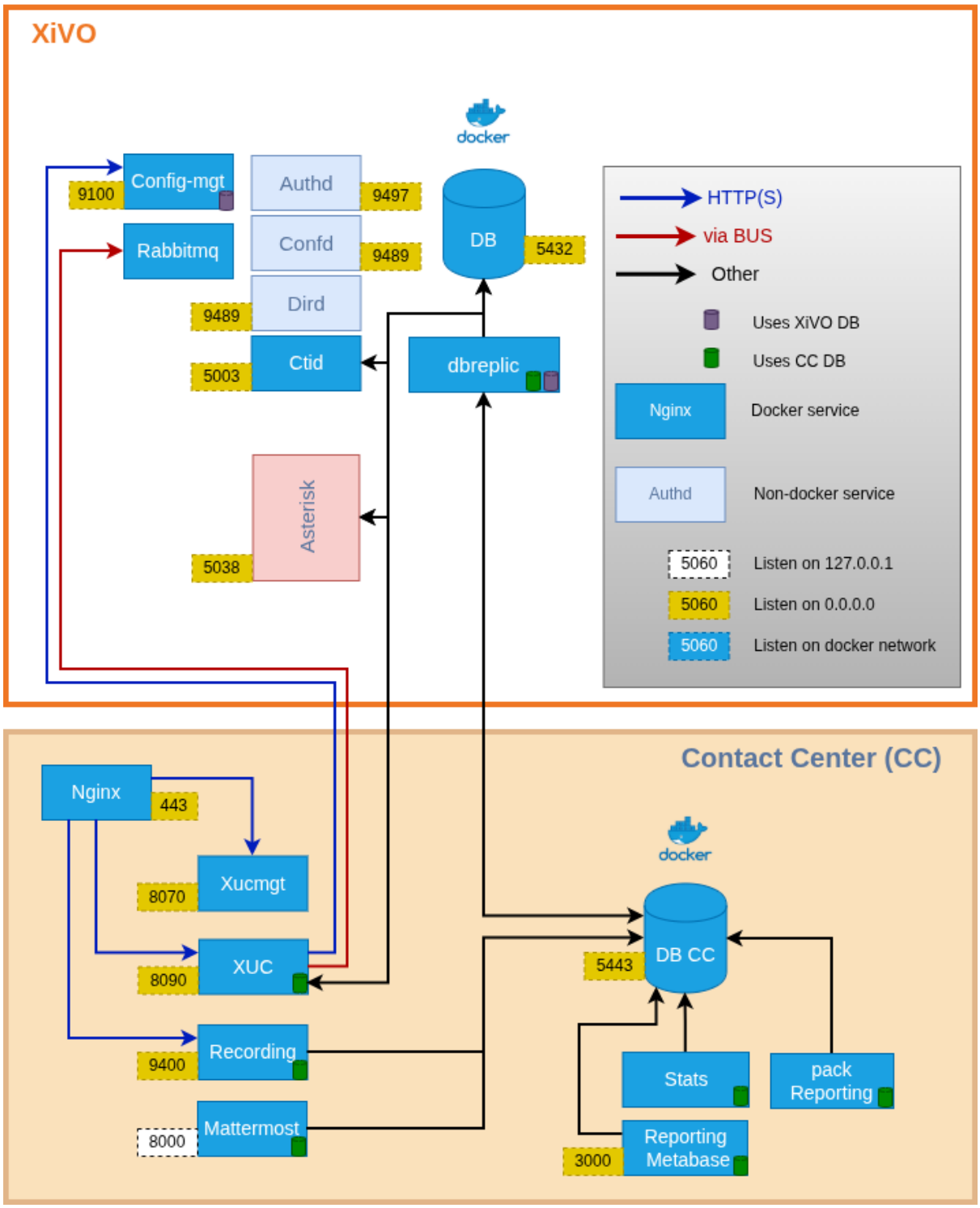
- *Editing basic XiVO CC configuration*
- *Components customization*
- *Multi-server installation*

XiVO CC system configuration is stored in the `/etc/docker/compose` directory. All files are 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 **MUST 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.

If you want to change the default configuration, you can override the `docker-xivocc.yml` file with `docker-xivocc.override.yml`.

You can also create files with the following pattern : `[0-9]{2}-.*\.override.yml` Those files will be read by alphabetical order and added after `docker-xivocc.yml` and `docker-xivocc.override.yml` Example : 00-add-something.override.yml 01-add-other-features.override.yml


XiVO
docker
Config-mgt
9100
Authd
9497
Rabbitmq
Confd
9489
DB
5432
9489
Dird
5003
Ctid
dbreplic
Asterisk
5038
HTTP(S)
via BUS
Other
Uses XiVO DB
Uses CC DB
Nginx
Docker service
Authd
Non-docker service
5060
Listen on 127.0.0.1
5060
Listen on 0.0.0.0
5060
Listen on docker network
Contact Center (CC)
Nginx
443
docker
Xucmgt
8070
XUC
8090
DB CC
5443
Recording
9400
Stats
pack Reporting
Mattermost
8000
Reporting Metabase
3000

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

Factory env file

`factory.env`

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

Custom env file

`custom.env`

Editing basic XiVO CC configuration

In this file you can edit main configuration of XiVO CC originally set by the xivocc-installer:

- `XIVO_HOST`
- `XUC_HOST`
- `CONFIG_MGT_HOST`
- `WEEKS_TO_KEEP`
- `RECORDING_WEEKS_TO_KEEP`
- `PLAY_AUTH_TOKEN` (see *Shared token*)

Components customization

You can also change value of any option defined in the compose file. For example:

- `SHOW_RECORDING_CONTROLS`
- `SHOW_QUEUE_CONTROLS`

The option value must not be assigned inside the compose file, otherwise it will not apply.

Multi-server installation

Some XiVO CC components can run a separate server, but the installation procedure is not documented. For this purpose these variables can be set:

- `REPORTING_HOST`
- `RECORDING_SERVER_HOST`
- `PLAY_AUTH_TOKEN` (see *Shared token*)

3.2.4 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()'`**

Configuration Customization

If you changed, via the *XiVO PBX* Web Interface in *Configuration* → *Provisioning*→ *Template Device*, the phone **administrator username** or **administrator password**, you need to customize the xuc server configuration.

For this you need to:

1. Include the a specific configuration file for the xucserver onment variable to specify the alternate config file location

```
xuc:
...

  environment:
  ...
  - CONFIG_FILE=/conf/xuc.conf

  volumes:
  - /etc/docker/xuc:/conf
```

2. Create the directory `/etc/docker/xuc/`:

```
mkdir -p /etc/docker/xuc/
```

3. Create the `/etc/docker/xuc/xuc.conf` configuration file with the following content:
   - For **Snom**: change the `user` and `password` values accordingly:

```
include "application.conf"

Snom {
    user="guest"
    password="guest"
}
```

   - For **Polycom**: change the `user` and `password` values accordingly:

```
include "application.conf"

Polycom {
    user="guest"
    password="guest"
}
```

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

3.2.5 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 *XiVO CC Architecture* diagram.

Check XiVOcc Configuration

Check the prerequisites

- the OS must be Debian 11 (Bullseye), 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-pack-reporting-1     /bin/sh -c echo "WEEKS_TO_ ...   Up
xivocc-pgxivocc-1           docker-entrypoint.sh postgres    Up             0.0.0.
↪0:5443->5432/tcp
xivocc-recording_rsync-1    /usr/local/sbin/run-rsync.sh     Up             0.0.0.
↪0:873->873/tcp
xivocc-recording-server-1   bin/recording-server-docker      Up             0.0.0.
↪0:9400->9000/tcp
xivocc-spagobi-1            /bin/sh -c /root/start.sh        Up             0.0.0.
↪0:9500->8080/tcp
xivocc-xivo-stats-1         /usr/local/bin/start.sh /o ...   Up
xivocc-mattermost-1         /entrypoint.sh mattermost        Up (healthy)   8000/tcp
xivocc-nginx-1              /bin/sh -c /bin/bash -c "e ...   Up             0.0.0.
↪0:443->443/tcp, 0.0.0.0:80->80/tcp,
                                                                            0.0.0.
↪0:8443->8443/tcp, 0.0.0.0:9100->9100/tcp
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

- Connection from the XiVO CC for user asterisk must be authorized. See file `/var/lib/postgresql/17/data/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 : <randomly generated password>
- 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*).

Check the ldap definition

In *Services → CTI Server → Directories → Definitions* make sure your definitions matches with the new display filter.

For specific installations

3.2.6 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 13** (Trixie), 64 bit
- the latest stable version of Docker is installed
- the latest stable version of Docker-compose-plugin 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

Add this line to `/var/lib/postgresql/17/data/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 : <randomly generated password>
- 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*,
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
```

(continues on next page)

(continued from previous page)

```
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_fingerboard_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
```

Upgrading

3.2.7 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 the version via `xivo-dist` utility
- upgrade via the `xivocc-upgrade` utility: it will upgrade the system (Debian packages) and the XiVOCC containers

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 sources to the new XiVO CC **LTS** version with `xivo-dist`, for example, to switch to Gaia LTS version:

   ```
   xivo-dist xivo-gaia
   ```

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. *Freya (2020.18)*) we may backport some bugfixes in this version. We will then create a **subversion** (e.g. Freya **.04** (2020.18 **.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. Freya.00) to the latest bugfix release (e.g. Freya.04).
2. Verify that the debian sources list corresponds to your *installed LTS* or refix it, for example for Freya:

```
xivo-dist xivo-freya
```

3. Verify that the `/etc/docker/compose/factory.env` file has
   - `XIVOCC_TAG=VERSION` (where `VERSION` is your current installed *version* - e.g. *2020.18*)
   - and `XIVOCC_DIST=latest`
4. And then upgrade, see *Upgrading*

Upgrading

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

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

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

```
xivocc-upgrade
```

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
```

XiVO UC/CC Debian 9 (Stretch) Upgrade Procedure

This page describes what to do to upgrade your XiVO UC/CC to Debian 9 (Stretch).

Important: Upgrade your XiVO UC/CC to Debian 9 (Stretch) is a **mandatory** step when upgrading to Electra because system freezes were detected during test of Electra version with kernel 3.16. Problem was solved with kernel 4.9 and higher.

Warning: In Debian 9 (Stretch) docker storage driver changed from `aufs` to `overlay2`. Therefore all containers and images need to be recreated. Note that `overlay2` is incompatible with **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.

Checks

What to check before upgrading to Debian 9 (Stretch):

- your XiVO UC/CC **MUST** be in Deneb version otherwise all the database data **will be lost !**
- the partition where docker data is stored must not be using XFS file system with ftype different from 1. You can use e.g. this command to check file system:

  ```
  docker info 2> /dev/null | grep "Backing Filesystem"
  ```

  If it is XFS, you can check ftype with the following command (assuming that docker data dir is `/var/lib/docker`):

  ```
  apt-get install xfsprogs
  xfs_info /var/lib/ 2> /dev/null | grep ftype
  ```

When

The upgrade to Debian 9 (Stretch) should be done **before** upgrading to Electra (if you are already in Deneb).

It can be done *after* upgrading to Electra, but it should be done *just after*. Otherwise your system won't be stable.

Backups

During upgrade all **Kibana configuration** (including the dashboard) will be lost (they are stored in *elasticsearch* container).

You **MUST** backup Kibana configuration before the upgrade.

Since Deneb version, the database data was exported to the host. Therefore no data loss should happen even when removing the pgxivocc container.

Though we **strongly advise** to *backup the database* for safety.

Upgrade

After having backup your configuration you're ready to upgrade the host to Debian 9 (Stretch). We don't cover the Debian upgrade procedure itself here, this can be found on Debian 9 release notes.

Here we'll only give the main steps and **where specific actions are to be performed in a XiVO UC/CC context**.

The upgrade consist of:

1. Updating the host to latest Debian 8 (Jessie) version
2. Switching the sources list to Debian 9 (Stretch). **Specifically on a XiVO UC/CC** you must also update the sources list of the docker repo:

```
sed -i 's/jessie/stretch/' /etc/apt/sources.list /etc/apt/sources.list.d/docker.
↪list
```

3. Launch the Debian 8 to Debian 9 upgrade
4. **Specifically on a XiVO UC/CC** and **before** rebooting you **MUST** remove the `aufs` dir. **All** docker data will be removed - it is safe only in Deneb version:

```
xivocc-dcomp stop
xivocc-dcomp rm
systemctl stop docker
rm -rf /var/lib/docker/aufs
```

5. Reboot the machine
6. Check if the docker storage driver was changed to overlay2:

```
docker info 2> /dev/null | grep "Storage Driver"
```

7. If overlay2 is not used yet, you must repeat the previous steps to remove (again) the aufs dir
8. **Specifically on a XiVO UC/CC** and **after** the reboot you must pull the images again:

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

Restore

At last you must restore the Kibana configuration.

Upgrade notes

See *Release Notes* for version specific informations.

3.3 XiVO Distributed System

3.3.1 Installing XDS

Important: Before installing, make sure you understand the *XDS Architecture* and links between components.

- *Requirements*
- *XiVO Configuration*
  - *AMI configuration*
  - *Define Media Servers*
  - *Define Media Servers for Provisionning*
  - *Media Servers connection to the XIVO database*
  - *SMTP relay configuration*
  - *XDS File Synchronization*
- *Edge configuration*
- *Media Server Configuration*
  - *Requirements*
  - *Installation*
  - *Configuration*
  - *Mail configuration*
- *Outgoing Call Configuration*
  - *Create the Provider Trunk*
  - *Create Outgoing Call Rule*
- *XiVO CC Configuration*
  - *Enable WebRTC on MDS*
- *Known Limitations*
  - *Agent states after XUC restart*

The XDS architecture has the following components:

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

An XDS needs also:

- a XiVO UC/CC with the UC/CC features (i.e. the CTI Server),
- a XiVO UC/CC with the Reporting features for the centralized call history (i.e. the Reporting Server).

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. and the configuration of the UC/CC server (CTI and Reporting 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
Role	XiVO	Media Server	UC/CC (CTI/Reporting Server)
Name	mds0	mds1	cc
IP Data	10.32.0.1	10.32.0.101	10.32.0.9
IP VoIP	10.32.5.1	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, the *xucserver* from UC/CC Server will immediately start to use the VoIP interface for AMI connection to all media servers **and also to XiVO**. Therefore we must ensure that UC/CC Server is able to connect to XiVO AMI via its VoIP interface.

Warning: If a XiVO UC/CC is already installed, you **MUST** 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 *xucserver** of UC/CC Server (CTI Server):
   - `permit` to authorize the VoIP IP of the UC/CC Server (CTI 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

Once you define a media server, you will be able to create local SIP trunks that exist only there. The location can be set in *tab General* → *Media server* in the SIP trunk configuration.

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)

Media Servers connection to the XIVO database

Note: The MDS need to connect to the XiVO database.

In file `/var/lib/postgresql/17/data/pg_hba.conf` add an authorization **per mds** to connect to the db. Here you will probably want to use the Data IP of mdsX:

```
host      asterisk    all     10.32.0.101/32      md5
```

And reload the database configuration:

```
xivo-dcomp reload db
```

SMTP relay configuration

Note: This step is specific to **XiVO Main**

SMTP relay must be configured to receive voicemail notifications from media servers. *Mail* on XiVO must be configured also.

1. Create custom template for postfix configuration:

```
mkdir -p /etc/xivo/custom-templates/mail/etc/postfix/
cp /usr/share/xivo-config/templates/mail/etc/postfix/main.cf /etc/xivo/
↪custom-templates/mail/etc/postfix/
```

2. Open `/etc/xivo/custom-templates/mail/etc/postfix/main.cf` for editing
3. Add MDS Data IP value to the `mynetworks` option:

```
mynetworks =    127.0.0.0/8
                [::1/128]
                #XIVO_DOCKER_NET#
+               10.32.0.101/32
```

4. Update configuration

```
xivo-update-config
```

XDS File Synchronization

Note: You need to do this manual step on **XiVO Main** for the file sync to take place. See *File Synchronization* page for the feature description.

1. Rename files in synced dir (and mainly custom dialplans in `/etc/asterisk/extensions_extra.d/`) which should not be synchronized to be prefixed with `xds_override`. All files named with this prefix will be excluded from synchronization.
2. Initialize synchronization of dialplans by running the command:

```
xivo-xds-sync -i
```

Note: The initialization will:

1. generate a ssh key pair: `~/.ssh/rsync_xds` and `~/.ssh/rsync_xds.pub`
2. copy the public key to `/usr/share/xivo-certs/`
3. make the public key available at `https://XIVO_HOST/ssh-key`
4. create cron job `/etc/cron.d/xivo-xds-sync` to schedule the synchronization

Edge configuration

If an edge is already configured and you are switching from a normal setup to an XDS one, you must do the following configuration on the edge side:

- For edge 3 VM : *SIP Proxy*.
- For edge on a single VM : *Web Proxy, SIP Proxy and TURN Server*.

Media Server Configuration

Requirements

On *server2* install a **Debian 13** 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)

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 (i.e. the XiVO UC/CC with reporting features - database, xivo_stats ...)

To perform a silent installation, you can use debconf to set these variables based on their corresponding environment values:

- $XIVO_HOST (e.g. 10.32.0.1)
- $MDS_NAME (e.g. mds1)
- $DB_HOST (corresponding to the MDS IP, e.g. 10.32.0.101)
- $REPORTING_DB_HOST (e.g. 10.32.0.9)

```
echo "xivo-mds-installer xivo-mds-installer/XIVO_HOST string $XIVO_HOST" | debconf-
↪set-selections
echo "xivo-mds-installer xivo-mds-installer/REPORTING_DB_HOST string $REPORTING_DB_
↪HOST" | debconf-set-selections
echo "xivo-mds-installer xivo-mds-installer/DB_HOST string $DB_HOST" | debconf-set-
↪selections
echo "xivo-mds-installer xivo-mds-installer/MDS_NAME string $MDS_NAME" | debconf-set-
↪selections
echo "xivo-mds-installer xivo-mds-installer/DROP_REPLICATION_SLOT boolean true" |␣
↪debconf-set-selections
```

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 2026.05-latest
```

When prompted:

- give the IP of XiVO (mds0): <XiVO Data IP> (e.g. 10.32.0.1)
- 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. 10.32.0.9)

Important: In case of `re-installation`, you will be prompted to drop the existing database replication slot.

Configuration

To finalize the MDS configuration you have to:

1. Configure NTP to synchronize on XiVO (mds0) by replacing preconfigured servers/pools in file `/etc/ntpsec/ntp.conf` by:

   ```
   server 10.32.0.1 iburst
   ```

2. And restart NTP:

   ```
   systemctl restart ntp
   ```

3. 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 UC/CC Server (CTI 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
     ```

4. Restart the services:

   ```
   xivo-service restart all
   ```

Mail configuration

The install script installs and configures postfix:

- to relay mails towards the XiVO Main - see `relayhost` parameter in `/etc/postfix/main.cf` file,
- and use the content of `/etc/mailname` file as the domain part of the from address - see `myorigin` parameter in `/etc/postfix/main.cf`.

Note: Note that the content of `/etc/mailname` file is taken during installation from the domain if set or the hostname.

Therefore if mail sent from the MDS are not correctly relayed by the XiVO Main, you should check and play with the value of the `/etc/mailname` file. And then reload the postfix if needed `service postfix reload`.

Outgoing Call Configuration

Create the Provider Trunk

Add on the XiVO the trunk towards your provider (it can be an ISDN or SIP trunk). When creating the trunk, select the Media Server on which it will be located.

Create Outgoing Call Rule

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

In XiVO Webi:

1. Go to *Services* → *IPBX* → *Call Management* → *Outgoing calls*
2. Create a route for XDS:
    1. Call pattern: X.
    2. Trunks: <your provider trunk>
    3. (after opening the advanced form) Caller ID: <main DID of the system>

XiVO CC Configuration

On *server3*:

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

Enable WebRTC on MDS

Note: The manual procedure has been automated in kuma.

WebRTC users can be configured on the MDS. When you add a new mds, you need to update the xivocc nginx configuration as shown below.

1. Either refresh the sip proxy configuration on the nginx and reload it:

```
xivocc-dcomp exec nginx /docker-entrypoint.d/50-mds-sip-proxy.sh
xivocc-dcomp reload nginx
```

2. Or restart the container:

```
xivocc-dcomp restart nginx
xivocc-dcomp up -d nginx
```

3. You can check out your configuration here

```
xivocc-dcomp exec nginx cat /etc/nginx/sip_proxy/sip_proxy.conf
```

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.

3.3.2 XDS Architecture

The following diagram presents the XDS architecture with:

- one XiVO Main
- one CTI / Reporting Server (XiVO CC)
- and three MDS

As you can see:

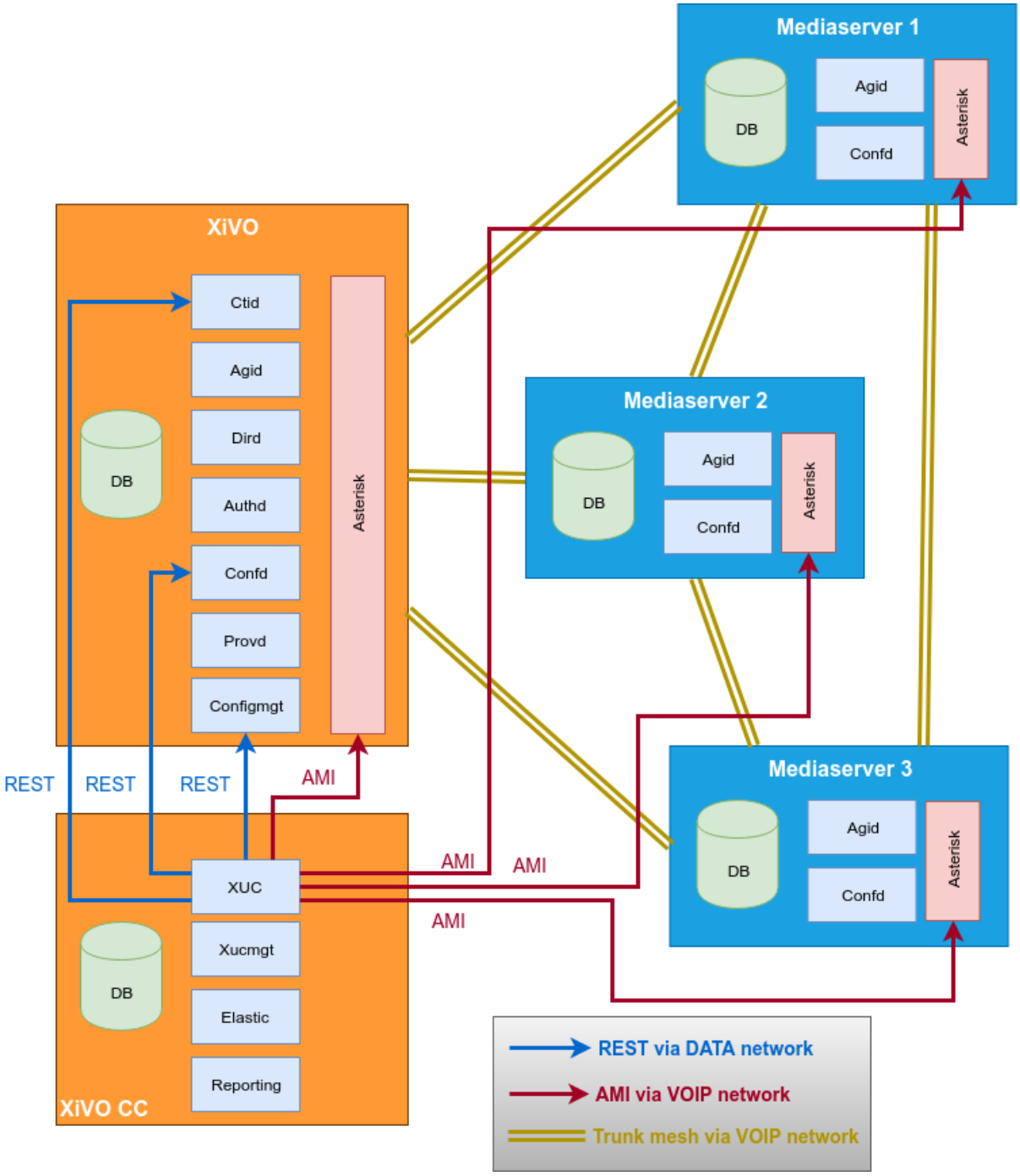
- each MDS and the Main are linked together via an internal SIP trunk (the yellow lines) - via the VoIP IP Address of the MDS
- the *xuc* components of the CTI / Reporting Server is connected to each MDS AMI - via the VoIP IP Address of the MDS

Database Replication

Database replication is employed by the mds to receive replicated data from XiVO Main.

- The mds use PostgreSQL replication to synchronize data with XiVO Main.
- To control the replication process, the *max_slot_wal_keep_size* parameter is set to 1G. This means that if a mds becomes unavailable and falls too far behind, it will not be able to continue replication.

For more detailed information on PostgreSQL replication configurations, please refer to the PostgreSQL documentation.


Mediaserver 1
DB
Agid
Confd
Asterisk
XiVO
Ctid
Agid
Dird
Authd
Confd
Provd
Configmgt
DB
Asterisk
Mediaserver 2
DB
Agid
Confd
Asterisk
Mediaserver 3
DB
Agid
Confd
Asterisk
REST
REST
REST
AMI
AMI
AMI
AMI
XUC
Xucmgt
Elastic
Reporting
DB
XiVO CC
REST via DATA network
AMI via VOIP network
Trunk mesh via VOIP network

3.3.3 Media Server Uninstallation

This procedure is to remove a media server permanently.

On XIVO

1. Disassociate all Users/Groups/SIP Trunk from the Media server you want to remove (remove these objects or associate them to another MDS),
2. Then remove the media server from XDS by removing it from webi (if needed refer to related sections in *XiVO Configuration*):
   - remove the media server in *Configuration -> Management -> Media Servers*
   - remove the corresponding template line in *Configuration -> Provisioning -> Template Line*
3. Clean connection permissions from all places and apply new configuration (see related sections in *XiVO Configuration*):
   - in AMI configuration
   - in Postgres conf (pg_hba.conf)
   - and in SMTP conf if applicable
4. And remove replication slot on XiVO Main server. Connect to postgres database find the correct replication slot and drop it:

```
select * from pg_replication_slots;
select pg_drop_replication_slot('slot_name');
```

On Media Server

Only reinstallation is supported.

3.3.4 Media Server Reinstallation

On XIVO

All settings can remain if they are still valid - if the new media server has the same name and IP as a previous (for example crashed) media server.

On Media Server

If the mds install script failed with error, you can re-run it.

But to fully reinstall media server, you need to reinstall xivo-mds-installer package and recreate database on the media server.

- Uninstall MDS:

```
xivo-service stop all
apt purge xivo-mds-installer
rm -rf /var/lib/postgresql/17
```

- Reinstall MDS: the `custom.env` will be recreated with new values:

```
apt install xivo-mds-installer
xivo-service start all
xivo-dcomp up -d
```

At the end you need also to reinstall the *XiVO Main* ssh public key for *File Synchronization*:

```
XIVO_HOST=$(grep XIVO_HOST /etc/docker/mds/custom.env | awk -F "=" '/1/ {print $2}')
mkdir -p /root/.ssh
wget https://$XIVO_HOST/ssh-key --no-check-certificate -O /root/.ssh/rsync_xds.pub
cat /root/.ssh/rsync_xds.pub >> /root/.ssh/authorized_keys
apt install -y rsync
```

3.4 XiVO Resilience PRO

3.4.1 Resilience Architecture

- *Prerequisites*
- *Failover Mode*
  - *Automatic Failover*
  - *Manual Failover*
- *Resilience in XiVO deployement*
- *Resilience in CC deployement*
- *Resilience in XDS deployement*
- *Resilience with Edge deployement*
- *Replication*
  - *DB Replication*
  - *File Replication*
- *Limitations*

The Resilience 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 Resilience solution in XiVO is based on a 2-nodes "main and standby" architecture. In the normal situation, both the main and standby nodes are running in parallel, the standby acting as a "hot standby", and all the telephony services are provided by the main node. If the main fails or must be shutdown for maintenance, then the standby node automatically takes over the telephony services. Supported telephony devices automatically communicate with the standby node instead of the main one. Once the main is up again, the standby node stops itself and the telephony devices failback to the main node.

Currently, resilience is supported with:

- XIVO deployment
- CC deployment
- XDS deployment

Prerequisites

- Phones must be able to reach the main and the standby
- Main and Standby nodes must be in the same subnet
- If firewalling, the main must be allowed to join the standby on ports 22 and 5432
- If firewalling, the standby must be allowed to join the main with an ICMP ping
- Trunk registration timeout (`expiry`) should be less than 300 seconds (5 minutes)
- The standby must have **no** provisioning plugins installed.

The Resilience solution is guaranteed to work correctly with the following devices.

Failover Mode

Resilience comes in two failover mode, automatic or manual.

Automatic Failover

Note: Refer to *Administration in Automatic Failover Mode* for more information

When chosing to operate in **Automatic failover** mode:

- Standby XiVO will actively ping the Main XiVO
  - if Standby XiVO cannot ping Main XiVO, it will automatically activate and start the services
  - when Standby XiVO correctly ping Main XiVO, it will automatically disable and stop the services

Manual Failover

Note: Refer to *Administration in Manual Failover Mode* for more information

When chosing to operate in **Manual failover** mode:

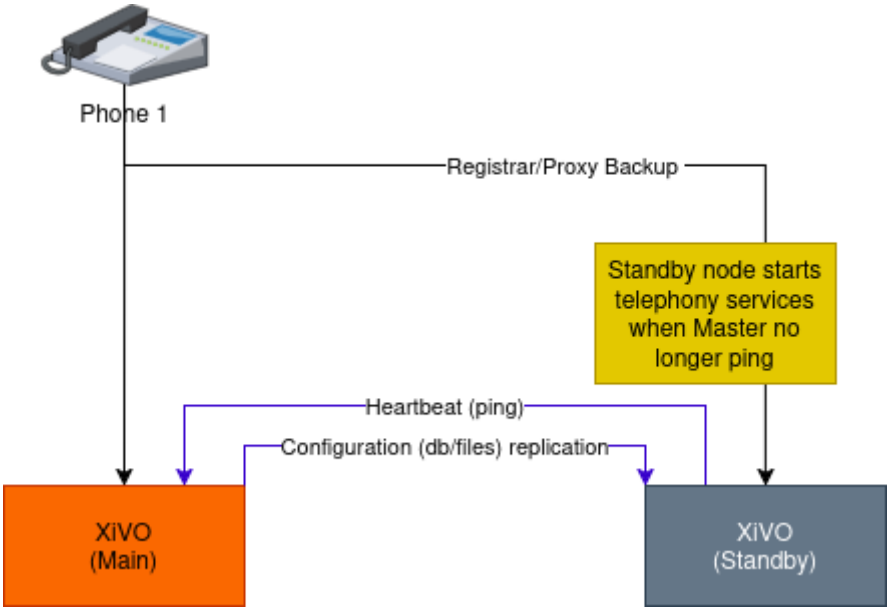
- there is no heartbeat between Standby and Main XiVO
- the failover must be triggered by an admin via the Web interface

Resilience in XiVO deployement

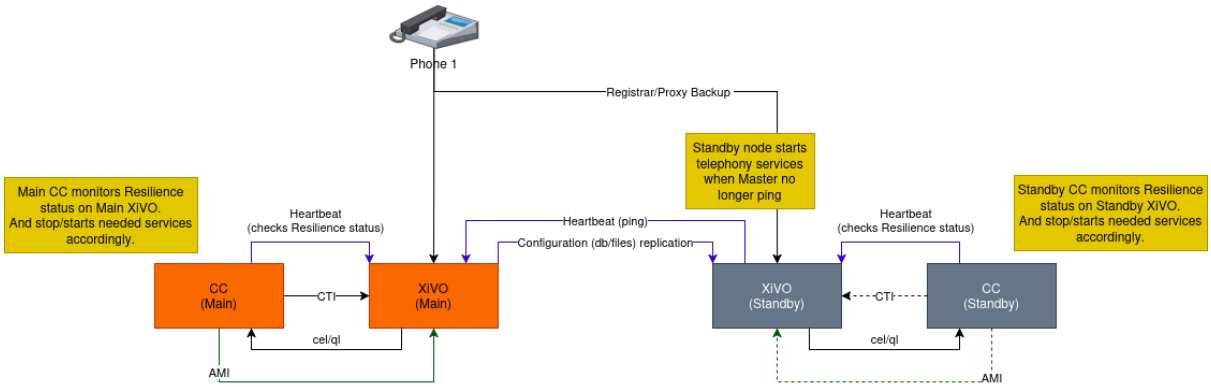
In a simple XiVO deployment another XiVO is added in standby mode.

In standby mode the XiVO ping the main XiVO every minutes.

- if the ping succeeds (main is up): the standby XiVO shutdown the telephony services
- if more than 3 ping fails (main is down): the standby XiVO startup the telephony services



Resilience in CC deployement



In a CC deployment another CC is configured with the standby XiVO.

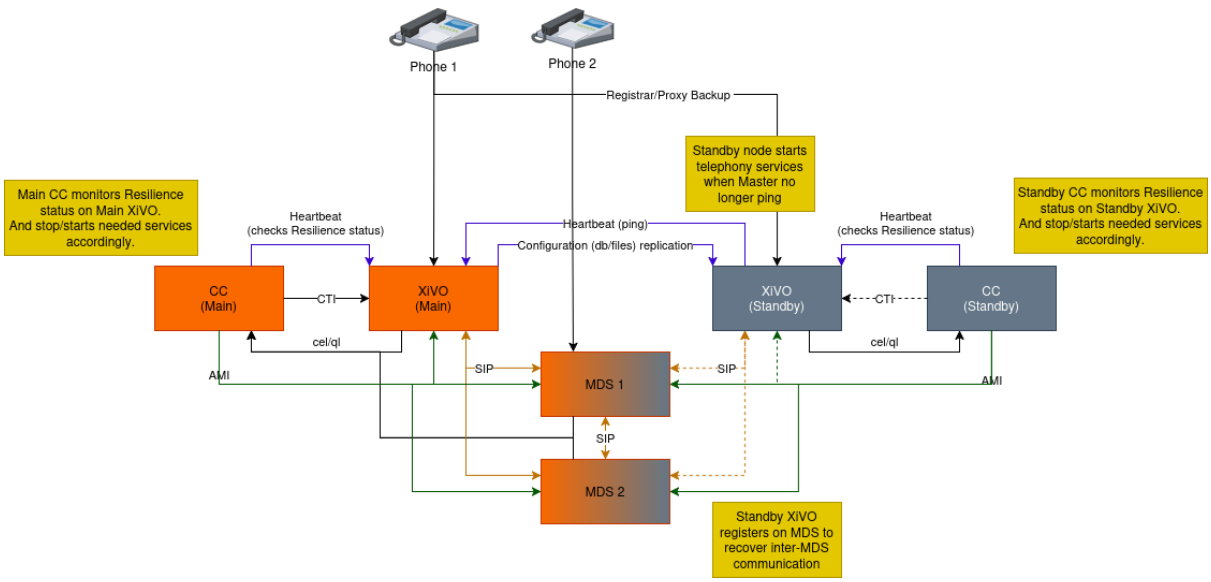
The standby CC is up and running.

Resilience in XDS deployement

In an XDS deployment no other node is added.

MDS will be shared between the main and the standby XiVO.

When the standby starts the telephony services, it will register the inter-mds trunk with the other MDS so the inter-MDS call can work.



Resilience with Edge deployement

Note: Resilience can work with single Edge deployment

Edge must know where the CC and the XiVO are located. Therefore a reconfiguration of the Edge is needed when standby takeover.

XiVO and CC standby must be configured with Edge information (mainly TURN secret).

Replication

Once main standby configuration is completed, XiVO configuration (DB and files) is replicated from the main node to the standby every hour (:00).

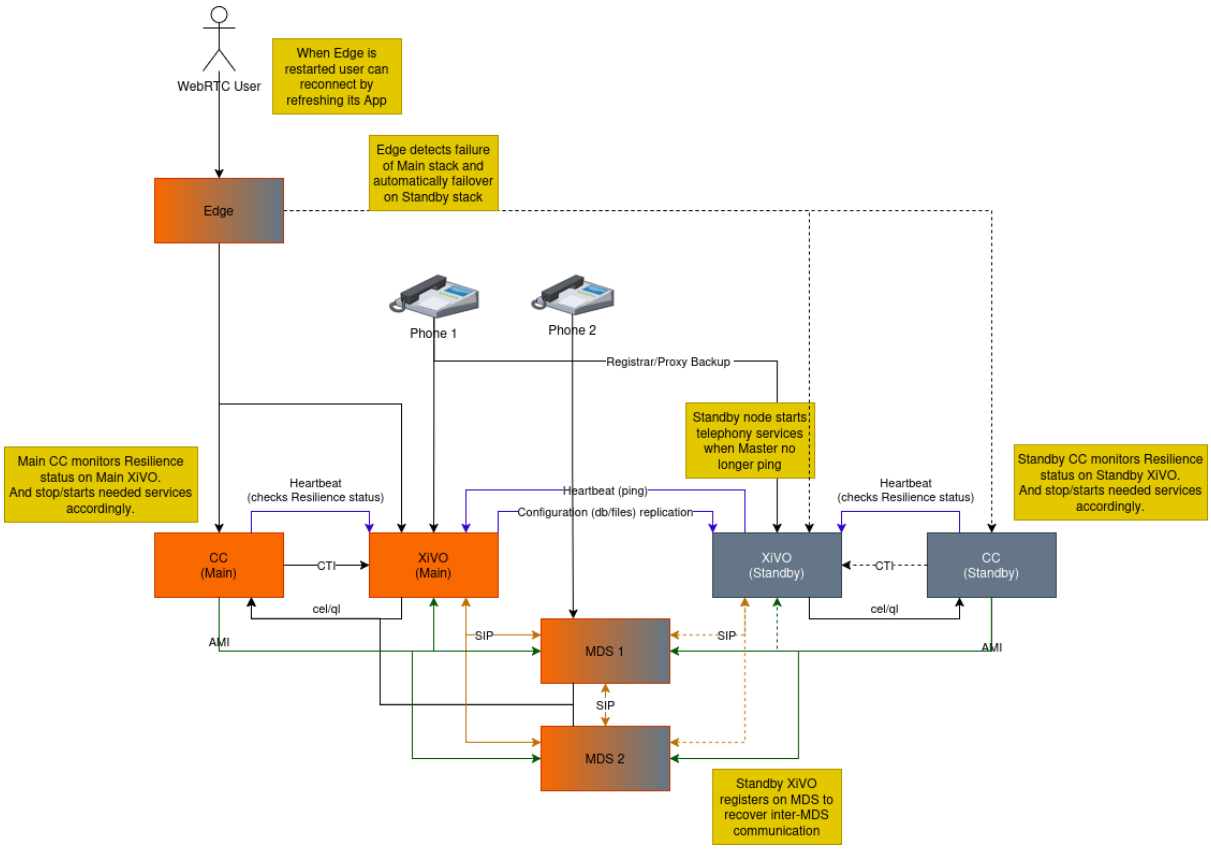
DB Replication

The replication does not copy the full XiVO configuration of the main. 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)
- Resilience settings
- Access Web Services configuration
- Provisioning configuration
- Voicemail messages

These event data are also excluded:

- Queue logs
- CELs



File Replication

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

Limitations

Architecture:

- Since DHCP parameters are replicated, Main and Standby node MUST be on the same VoIP network.

When the main node is down, some features are not available and some behave a bit differently. This includes:

- Call history / call records are not recorded.
- Voicemail messages saved on the main node are not available.
- Custom voicemail greetings recorded on the main 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 main.

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 main goes down. Since it's hard to know when the main goes down, if your CTI client disconnects and you can't reconnect it, then it's a sign the main might be down.

Additionally, only on failback:

- Voicemail messages are not copied from the standby to the main, i.e. if someone left a message on your voicemail when the main was down, you won't be able to consult it once the main 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 main 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.

3.4.2 Resilience Installation

Important: Before installing, make sure you understand the *Resilience Architecture* and links between components.

- *Overview*
- *Resilience Standby Stack Installation*
  - *All Deployement*
  - *CC Deployement*
  - *XDS Deployement*
  - *Edge Deployement*
- *Resilience Configuration Details*
  - *Configure one XiVO as Main*
  - *Configure the other XiVO as Standby*
  - *Configure file synchronization*
  - *Post-Configugration Steps*
  - *Specific steps*
    - *CC Deployment*
    - *XDS Deployment*

This page describes the necessary steps to install and configure the resilience solution in XiVO.

Overview

1. First the chosen XiVO deployment must be ready: whether it is intended to deploy a XiVO, a CC or an XDS solution, the deployment must be ready and configured.
2. Then a standby stack must be installed and configured - see *Resilience Standby Stack Installation*:
    1. a second XiVO must be installed (which will be the standby node).
    2. and for CC and XDS deployment: a second CC stack linked to the standby XiVO node must be installed.
3. Finally configure the Resilience itself - see *Resilience Configuration Details*:
    1. Configure one XiVO as main and fill in the standby address (**VoIP interface**)
    2. Configure the other XiVO as standby and fill in the main address (**VoIP interface**)
    3. Configure file synchronization
    4. Follow specific steps depending on the deployment type (XiVO, CC, XDS)

That's it, you now have a HA configuration, and every hour all the configuration done on the main will be reported to the standby.

Resilience Standby Stack Installation

All Deployement

Important: This is a mandatory step which applies to all deployment types: XiVO, CC and XDS.

Install a standby XIVO and configure (i.e. pass the Wizard) it.

CC Deployement

Note: This applies to deployment types: CC and XDS.

Install and configure a standby CC linked to the standby XiVO previously installed: this means that when requested fill in the XiVO Standby IP address etc.

XDS Deployement

Note: This applies to deployment types: XDS.

Nothing to install

Edge Deployement

Note: This applies to deployment types: Edge.

- On **XiVO Standby** configure the TURN secret for asterisk:
  - Run the following script to generate the credentials for the TURN server (the <TURN_SECRET> is the secret generated during *TURN Server Secret*):

    ```
    xivo-edge-gen-turn-cred <TURN_SECRET>
    ```

    Note: This script will generate new turn username and password in `/etc/asterisk/rtp.d/01-xivo-edge-turn-cred.conf`

- On **CC Standby** follow *XiVO CC configuration* to make it Edge aware.
- On **Edge** add two variables in edge *.env* for XiVO Standy and XiVOCC Standby (replace <…_IP> with their actual values):

  ```
  XIVO_HOST_STANDBY=<XIVO_STANDBY_IP>
  XIVOCC_HOST_STANDBY=<XIVOCC_STANDBY_IP>
  ```

- Ensure **both** XiVO Main and Standby are registered in the edge *.env* for TURN configuration:

  ```
  TURN_ALLOWED_PEERS="--allowed-peer-ip=<XIVO_MAIN_IP> --allowed-peer-ip=
  ↪<XIVO_STANDBY_IP> --allowed-peer-ip=... (MDS_IP, MEETINGROOM_IP, etc)"
  ```

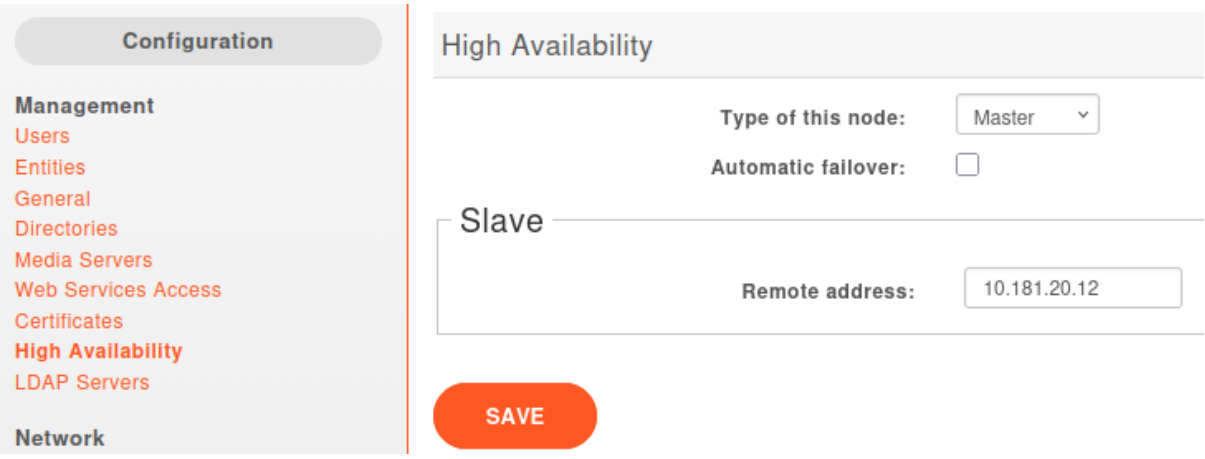
Resilience Configuration Details

Configure one XiVO as Main

Note: Step to be run on **XiVO Main**

1. In XiVO Web interface go to *Configuration* → *Management* → *High Availability* page
2. Select *Type of this node* to `Main`
3. Chose whether to activate or not *Automatic failover* - see *Failover Mode*
4. In the *Remote address* enter the VoIP address of the **Standby** node

   HA Dashboard Main
5. Restart all services on XiVO: `xivo-service restart all`



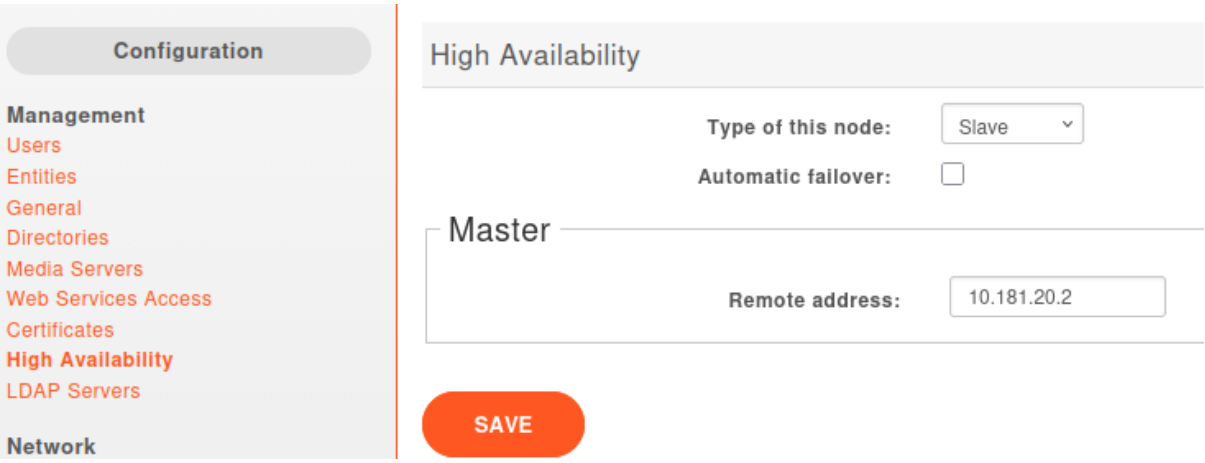
Configure the other XiVO as Standby

Note: Step to be run on **XiVO Standby**

1. In XiVO Web interface go to *Configuration* → *Management* → *High Availability* page
2. Select *Type of this node* to `Standby`
3. Chose whether to activate or not *Automatic failover* - see *Failover Mode*

   Important: Chose the same mode as on the **XiVO Main**

4. In the *Remote address* enter the VoIP address of the **Main** node



HA Dashboard Standby

Configure file synchronization

Note: Step to be run on **XiVO Main**

1. Configure file synchronization: `xivo-sync -i`

Post-Configugration Steps

When HA nodes is configured and file synchronization is setup here are the remaining manual steps to do:

- On **XiVO Main**:
  1. Force a first synchronization:

     Important: For the command below, `jq` must be installed and the file sync between Main and Standby must have been initiated - see *Configure file synchronization*.

     ```
     XSTANDBY_ADDR=$(cat /etc/xivo/ha.conf |jq .remote_address|tr -d '"')
     /usr/sbin/xivo-master-slave-db-replication $XSTANDBY_ADDR
     /usr/bin/xivo-sync
     ```

- On **XiVO Standby**:
  1. Force an update of the XIVO_UUID:

     ```
     function remove_xivo_uuid_from_env() {
        local sed_args_forsystemd=(-i 's/"XIVO_UUID=[a-f0-9\-]\{36\}"//g' /etc/
     ↪systemd/system.conf)
        local sed_args_env=(-i '/export XIVO_UUID/d' /etc/default/xivo)
        if sed "${sed_args_forsystemd[@]}" && sed "${sed_args_env[@]}" && rm -f /
     ↪etc/profile.d/xivo_uuid.sh; then
           true
        else
           false
        fi
     }
     echo ""
     echo "Current UUID in env and conf"
     grep XIVO_UUID /etc/profile.d/xivo_uuid.sh /etc/systemd/system.conf /etc/
     ↪default/xivo
     echo ""
     echo "Removing current UUID from env and conf"
     remove_xivo_uuid_from_env
     echo ""
     echo "Updating UUID in conf"
     xivo-configure-uuid
     ```

  2. Restart services:

     ```
     xivo-service restart all
     ```

Specific steps

Note: Specific steps depending on the deployment type.

CC Deployment

Nothing to do yet.

XDS Deployment

The XiVO Main & Standby must share the same AMI secret. If you need to change it, adapt the standby stack with the master secret this way:

- On **XiVO Main** execute the following which will configure the same AMI secret for xuc user on Main and Standby:

  Important: For the command below, `jq` must be installed and the file sync between Main and Standby must have been initiated - see *Configure file synchronization*.

```
XMASTER_AMI_SECRET=$(grep -oP '^\s*secret\s*=\K.*' /etc/asterisk/manager.d/02-
↪xivocc.conf)
XSTANDBY_ADDR=$(cat /etc/xivo/ha.conf |jq .remote_address|tr -d '"')
ssh -i /root/.ssh/xivo_id_rsa root@$XSTANDBY_ADDR "sed -i 's/secret.*/secret =
↪$XMASTER_AMI_SECRET/' /etc/asterisk/manager.d/02-xivocc.conf; asterisk -rx
↪'manager reload'"
XSTANDBY_AMI_SECRET=$(ssh -i /root/.ssh/xivo_id_rsa root@$XSTANDBY_ADDR "grep -
↪oP '^\s*secret\s*=\K.*' /etc/asterisk/manager.d/02-xivocc.conf")
if [ "$XMASTER_AMI_SECRET" == "$XSTANDBY_AMI_SECRET" ]; then
  echo "Good! AMI secret are the same !"
else
  echo "Wrong! AMI secret are still not the same. Please check manually"
fi
```

- On CC Standby:
  - edit `/etc/docker/compose/custom.env` and set `XIVO_AMI_SECRET` with the same value as in the XiVOCC Main
  - recreate xucserver:

```
xivocc-dcomp up -d xuc
xivocc-dcomp restart xuc # To be sure xuc is restarted anyway
```

Each MDS needs the Standby CC to be able to establish connection as well as the Main Xuc.

Note: Step to be done on each MDS nodes

- On each MDS:
  - edit the existing `/etc/asterisk/manager.d/02-xuc.conf` to add a new permit line (replace `XIVOCC_STANDBY_IP` with actual Standby XiVOCC IP address):

```
permit=XIVOCC_STANDBY_IP/255.255.255.255`
```

- reload AMI:

```
asterisk -rx "manager reload"
```

3.4.3 Resilience Administration

- *Administration*
- *General Information on Resilience*
  - *Resilience Modes*
    - *Node type and statuses*
    - *Disable node*
    - *Main node*
    - *Standby node*
    - *Visuals*
  - *Scripts*
  - *Automatic Configuration Change*
    - *SIP Expiry*
    - *Provisioning: Template Line*
    - *Logging*

This page describes the administration part of the Resilience.

Warning: On Failover, some manual steps are required. When moving to Standby/Main, remember to wait:

- for the services to be ready on the target xivo: *xivo-service status all*
- for the restarted xuc to be ready: *xivocc-dcomp logs -f xuc*

Finally the users must refresh their browser windows before reconnecting to the UC Apps (UC/CCAgent/Switchboard/…).

Administration

Administration in Automatic Failover Mode

In Automatic failover mode, there is an heartbeat link between Standby and Main XiVO. As soon as Standby XiVO detects Main as unreachable, it will become active - see *Failover Mode*. This page describes when the failover and fallback are triggered and what manual steps might be necesseray in the different deployment type.

- *XiVO Deployment*
  - *Failover to Standby*
  - *Failback to Main*
- *CC Deployment*
  - *Failover to Standby*
  - *Failback to Main*

- *XDS Deployment*
  - *Failover to Standby*
  - *Failback to Main*
- *Edge Deployment*
  - *Failover to Standby*
  - *Failback to Main*

XiVO Deployment

Failover to Standby

What happens:

- When Standby does not ping Main anymore, Standby node starts the telephony service
- The supported phones should automatically register and send their calls towards the Standby

Manual actions: None

Failback to Main

What happens:

- When Standby pings Main again, the standby shutdown the telephony services
- The supported phones should automatically register and send their calls towards the main

Manual actions: None

CC Deployment

Failover to Standby

What happens:

- Standby CC starts needed services when it detects Standby XiVO went to status *Active*
- Main CC stops needed services when it detects Main XiVO went to status *Inactive*

Manual actions:

- The users must reconnect manually to the CC Standby by entering the new CC Standby FQDN in their browser

Failback to Main

What happens:

- Standby CC stops needed services when it detects Standby XiVO went to status *Inactive*
- Main CC starts needed services when it detects Main XiVO went to status *Active*

Manual actions:

- The users must reconnect manually to the CC Main by entering the new CC Main FQDN in their browser

XDS Deployment

Failover to Standby

What happens:

- When the standby takes over the telephony service, it registers to the other MDS

Manual actions: None

Failback to Main

What happens:

- When the main is back again, it registers to the other MDS

Manual actions: None

Edge Deployment

- A single Edge stack is relaying its services to either the Main or the Standby XiVO machines.

Failover to Standby

What happens:

- Edge detects that Main CC/XiVO are not working and redirects the traffic to Standby CC/XiVO

Manual actions:

- The users must refresh its Web page to reconnect (no need to change FQDN)

Failback to Main

What happens:

- Edge detects that Standby CC/XiVO are not working and redirects the traffic to Main CC/XiVO

Manual actions:

- The users must refresh its Web page to reconnect (no need to change FQDN)

Administration in Manual Failover Mode

In Manual failover mode, there is no heartbeat link between Standby and Main XiVO. Activation of the Standby XiVO is manual - see *Failover Mode*. This page describes how to trigger the failover and fallback in the different deployment type.

- *XiVO Deployment*
  - *Failover to Standby*
  - *Failback to Main*
- *CC Deployment*
  - *Failover to Standby*
  - *Failback to Main*

- *XDS Deployment*
  - *Failover to Standby*
  - *Failback to Main*
- *Edge Deployment*
  - *Failover to Standby*
  - *Failback to Main*

XiVO Deployment

Failover to Standby

On **Standby** XiVO:

- An admin connects to the Web interface in *Services* → *IPBX* → *Resilience Management*
- And triggers the activation of the Standby node by clicking on *Start services*
- The standby will then activate and start the services

on **Main** XiVO:

- An admin must connects to the Web interface in *Services* → *IPBX* → *Resilience Management*
- And triggers the deactivation of the Main node by clicking on *Stop services*
- The Main will then deactivate and stop the services

Failback to Main

On **Standby** XiVO:

- An admin connects to the Web interface in *Services* → *IPBX* → *Resilience Management*
- And triggers the deactivation of the Standby node by clicking on *Stop services*
- The standby will then deactivate and stop the services

on **Main** XiVO:

- An admin connects to the Web interface in *Services* → *IPBX* → *Resilience Management*
- And triggers the activation of the Main node by clicking on *Start services*
- The Main will then activate and start the services

CC Deployment

Failover to Standby

What happens:

- Standby CC starts needed services when it detects Standby XiVO went to status *Active*
- Main CC stops needed services when it detects Main XiVO went to status *Inactive*

Manual actions:

- The users must reconnect manually to the CC Standby by entering the new CC Standby FQDN in their browser

Failback to Main

What happens:

- Standby CC stops needed services when it detects Standby XiVO went to status *Inactive*
- Main CC starts needed services when it detects Main XiVO went to status *Active*

Manual actions:

- The users must reconnect manually to the CC Main by entering the new CC Main FQDN in their browser

XDS Deployment

Failover to Standby

What happens:

- When the standby takes over the telephony service, it registers to the other MDS

Manual actions: None

Failback to Main

What happens:

- When the main is back again, it registers to the other MDS

Manual actions: None

Edge Deployment

- A single Edge stack is relaying its services to either the Main or the Standby XiVO machines.

Failover to Standby

What happens:

- Edge detects that Main CC/XiVO are not working and redirects the traffic to Standby CC/XiVO

Manual actions:

- The users must refresh its Web page to reconnect (no need to change FQDN)

Failback to Main

What happens:

- Edge detects that Standby CC/XiVO are not working and redirects the traffic to Main CC/XiVO

Manual actions:

- The users must refresh its Web page to reconnect (no need to change FQDN)

General Information on Resilience

Resilience Modes

Node type and statuses

Node type and status can be seen in the admin web interface header.

Disable node

Default status of Resilience is disabled:

> **Warning:** You should not disable an HA node in production as it will break the configuration and restart some services.

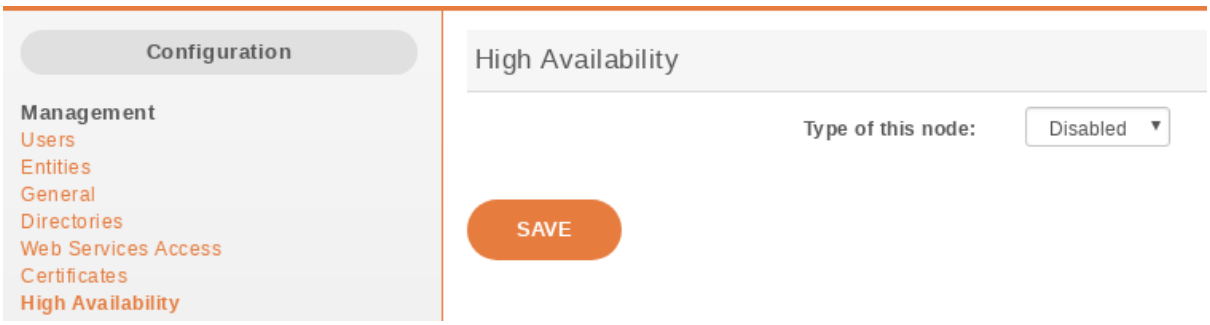


Fig. 5: HA Dashboard Disabled (default state)

Important: You have to restart services (xivo-service restart) once the main node is disabled.

Main node

In choosing the method `Main` you must enter the IP address **of the VoIP interface** of the standby node.

Important: You have to restart all services (xivo-service restart) once the main node is configured.

Standby node

In choosing the method `Standby` you must enter the IP address **of the VoIP interface** of the main node.

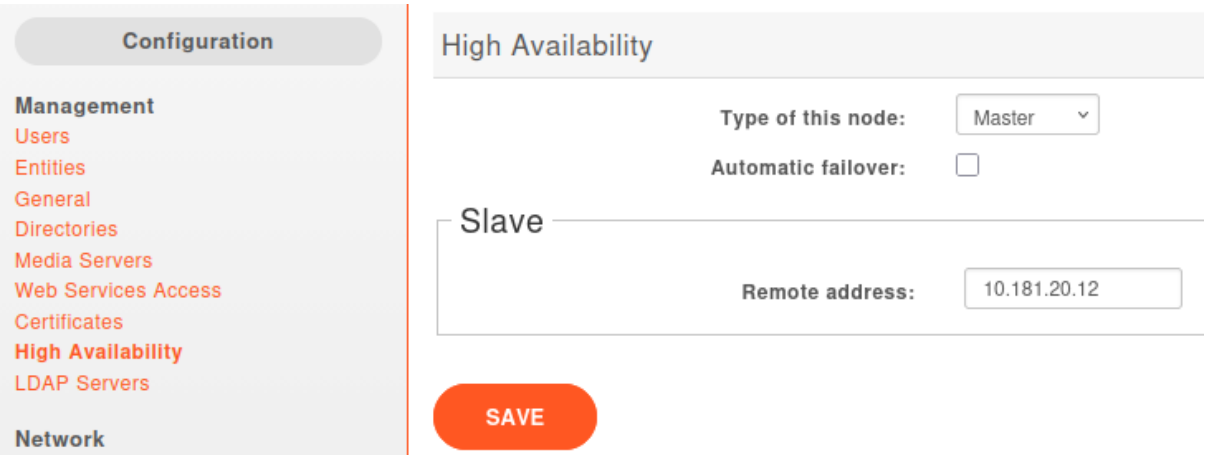


Fig. 6: HA Dashboard Main

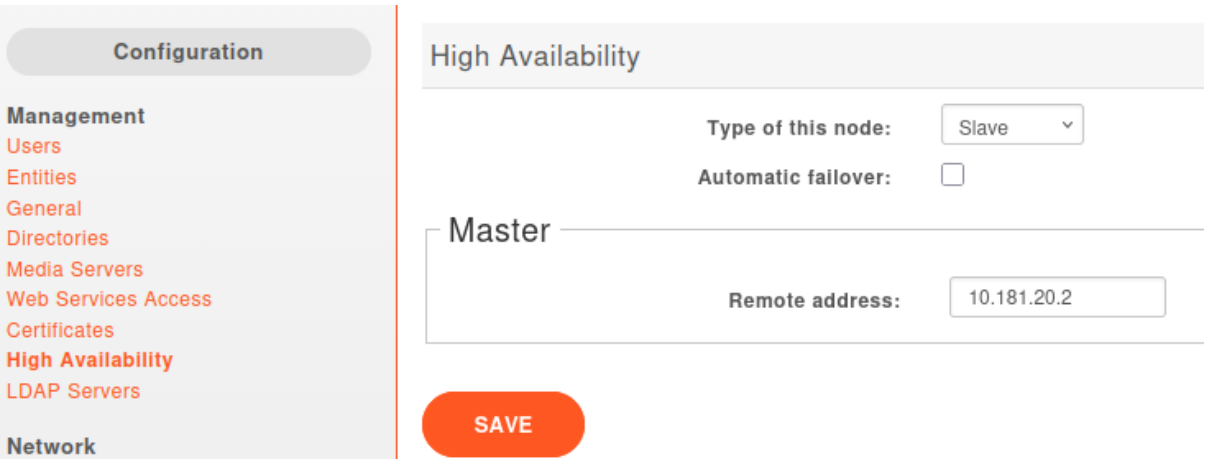
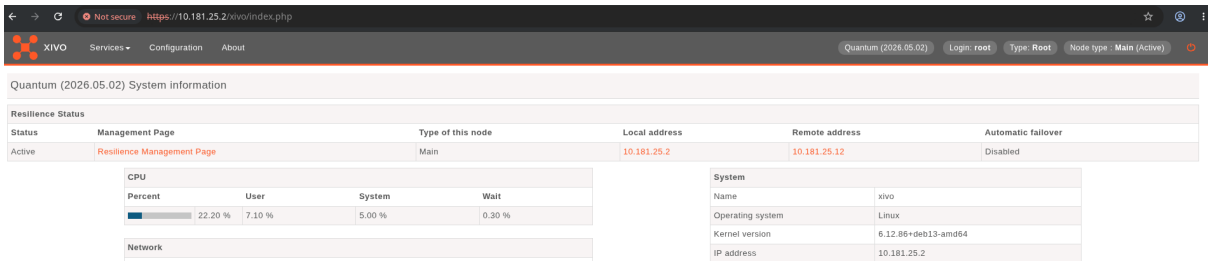


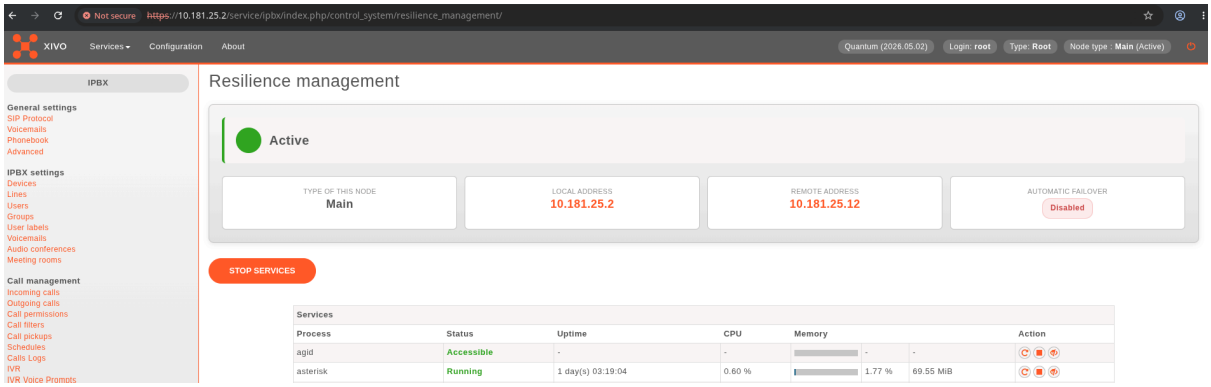
Fig. 7: HA Dashboard Standby

Visuals

Once resilience is activated either as `Main` or `Standby`, an extra table becomes available in the monitoring page.



The `Resilience Management Page` provides more informations. It can be accessed by link on the monitoring page or under *Services -> IPBX -> Control -> Resilience management*



Scripts

4 scripts are used to manage services and data replication.

- `xivo-master-slave-db-replication <standby_ip>` is used on the main to replicate the main's data on the standby server. It runs on the main.
- `xivo-manage-slave-services {start,stop}` is used on the standby to start, stop monit and asterisk. The services won't be restarted after an upgrade or restart.
- `xivo-check-master-status <main_ip>` is used to check the status of the main and enable or disable services accordingly.
- `xivo-sync` is used to sync directories from main to standby.
- On **CC** `xivocc-ha-monitoring` is used to monitor Resilience state on its linked XiVO to start or stop nginx service.

Replication can be started manually by running the replication scripts on the main:

```
xivo-master-slave-db-replication <standby_ip>
xivo-sync
```

Running `xivo-sync -i` will create a passwordless SSH key on the main, stored under the `/root/.ssh` directory, and will add it to the `/root/.ssh/authorized_keys` file on the standby.

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@<standby_ip>`

Automatic Configuration Change

> **Warning:** When the HA is configured, some changes will be automatically made to the configuration of XiVO. **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.

SIP Expiry

SIP expiry value on main and standby will be automatically updated:

- min: 1 minutes (default: 1 minute)
- **max: 5 minutes** (default: 1 hour)
- default: 2 minutes (default: 2 minutes)

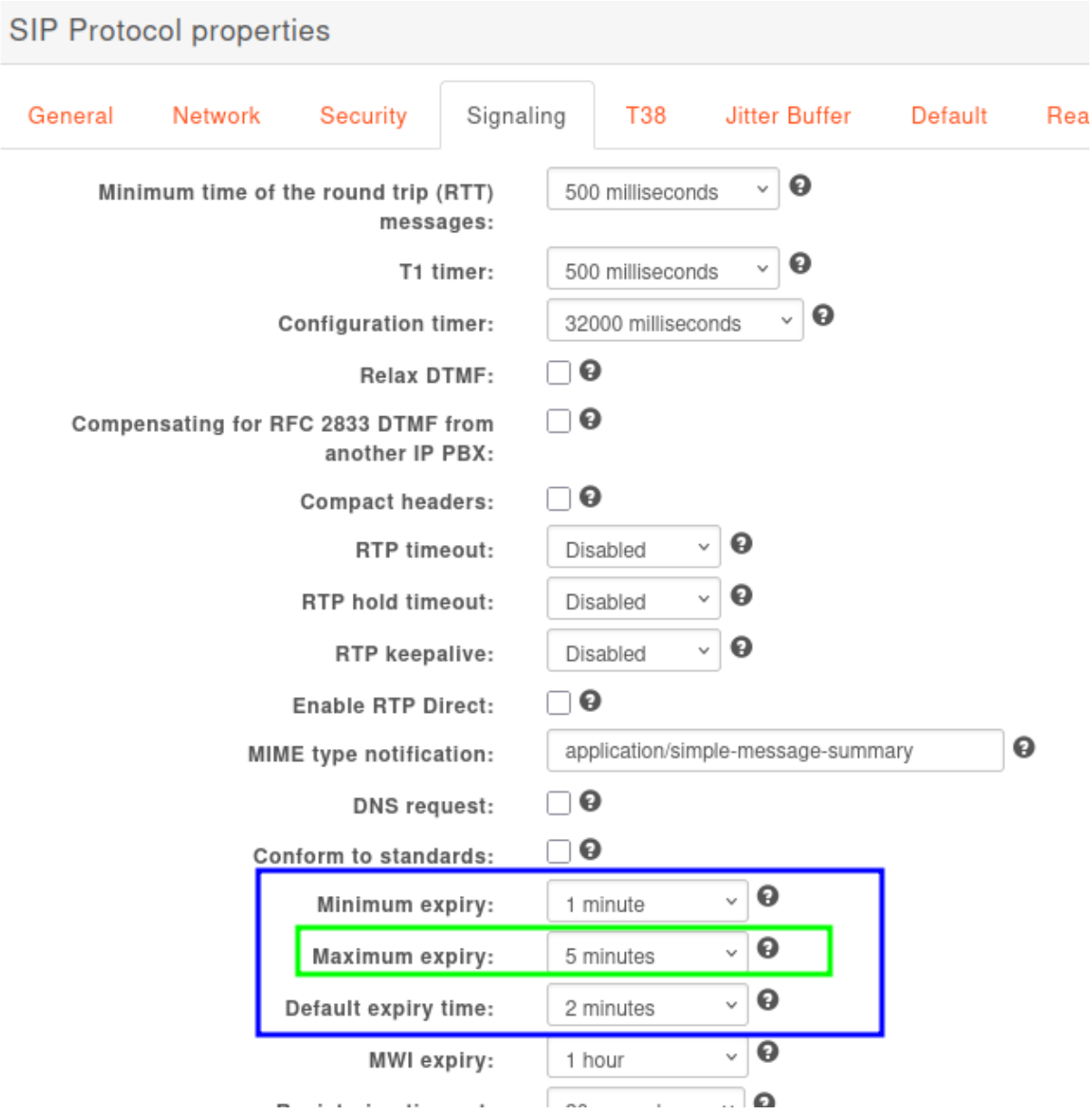


Fig. 8: *Services → IPBX → General Settings → SIP Protocol*

Provisioning: Template Line

The provisioning server configuration will be automatically updated in order to allow phones to switch from XiVO power failure.

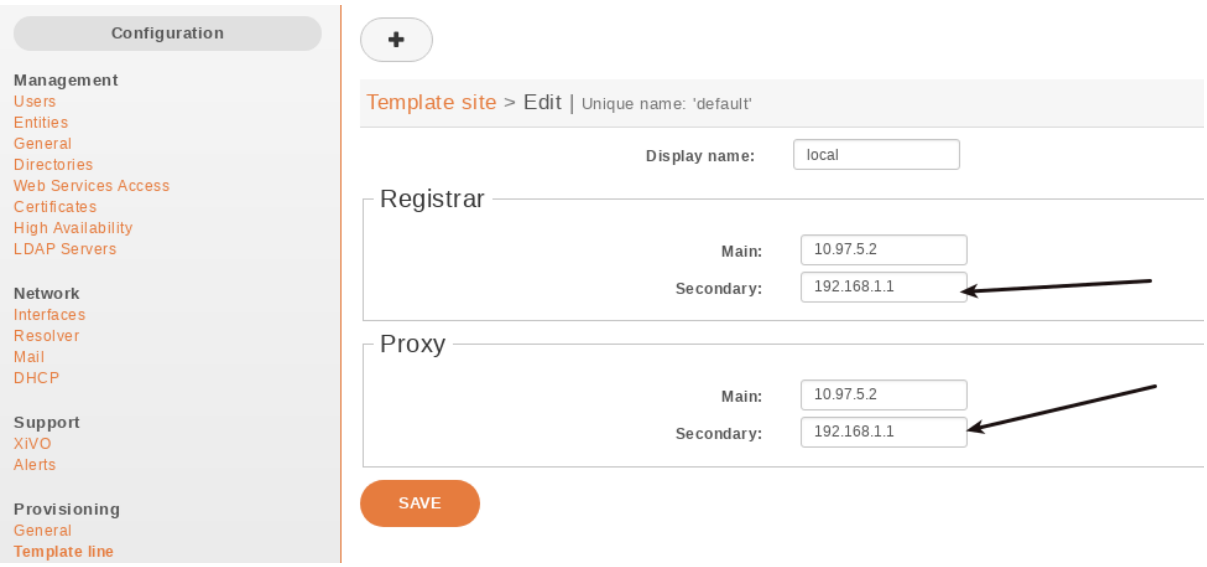


Fig. 9: *Configuration → Provisioning → Template Line → Edit default*

Important: For the telephony devices to take the new proxy/registrar settings into account, you must *resynchronize the devices* or restart them manually.

Logging

Reslience operation can be monitored in the following logs:

- On XiVO PBX (Main or Standby): `/var/log/xivo-sysconfd.log`:

```
grep 'Updating resilience state' /var/log/xivo-sysconfd.log

2026-04-23 17:47:22,395 [9857] (INFO) (xivo_sysconf.modules.ha): Updating␣
↪resilience state, from: False to: True
2026-04-23 18:04:56,490 [9857] (INFO) (xivo_sysconf.modules.ha): Updating␣
↪resilience state, from: True to: False
```

- On XIVO CC (Main or Standby): `/var/log/syslog`:

```
grep xivocc-ha /var/log/syslog

2026-04-23T17:12:01.088629+02:00 haXivocc xivocc-ha: [RESILIENCE] Querying HA␣
↪status at https://10.181.20.12/configmgt/api/2.0/sysconfd/get_ha_config
2026-04-23T17:12:01.228730+02:00 haXivocc xivocc-ha: [RESILIENCE] HA config␣
↪shows active=false
2026-04-23T17:12:01.285168+02:00 haXivocc xivocc-ha: [RESILIENCE] Nginx already␣
↪stopped
```

- On Edge: in nginx and kamailio logs:

```
edge-dcomp logs -t kamailio |grep "WAR.*Destination"

kamailio-1  | 2026-04-23T15:55:14.222395375Z 13(20) WARNING: <script>:␣
↪Destination down: OPTIONS sip:10.181.20.2:5060;transport=udp (<null>)
kamailio-1  | 2026-04-23T16:05:44.238528368Z 13(20) WARNING: <script>:␣
↪Destination down: OPTIONS sip:10.181.20.12:5060;transport=udp (<null>)
kamailio-1  | 2026-04-23T16:06:03.739168324Z 12(19) WARNING: 10.181.20.2:5060 -␣
↪4f36849712bb6f27-20@0.0.0.0 - OPTIONS - <null> <script>: Destination up:␣
↪OPTIONS sip:10.181.20.2:5060;transport=udp

edge-dcomp logs -t nginx|grep "temporarily disabled"

nginx-1  | 2026-04-23T16:00:40.612739761Z 2026/04/23 16:00:40 [warn] 108#108:␣
↪*166 upstream server temporarily disabled while connecting to upstream,␣
↪client: 10.182.20.3, server: edge-bis.test.avencall.com, request: "GET /config/
↪showAppDownload HTTP/1.1", upstream: "https://10.181.20.3:443/config/
↪showAppDownload", host: "edge-ealbis.test.avencall.com", referrer: "https://
↪edge-bis.test.avencall.com/"
```

CHAPTER

FOUR

ADMINISTRATOR'S GUIDE

In-depth documentation on administration of XiVO solution systems.

4.1 XiVO Administration

4.1.1 System

xivo-config configuration templating

Concept

The xivo-config package is deployed on xivo pbx and on mds.
It embeds default templates (files and directories) for various configuration on the machine.
Those configuration can be safely overriden and extended as described below.
They are applied on calling the script **xivo-update-config**.

Provided configs

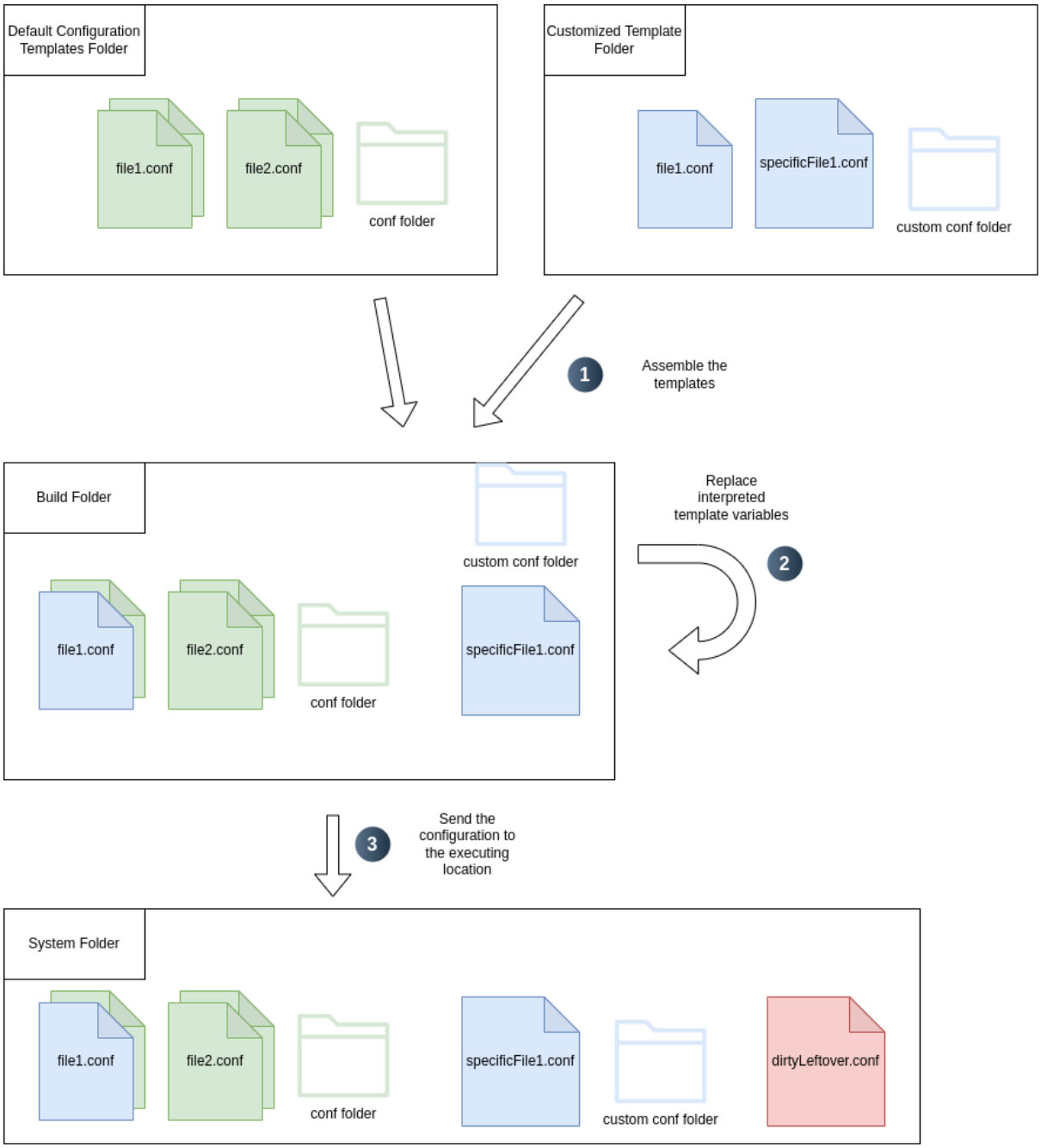
- dhcp
- mail
- provd
- system
- dialplan (xivo pbx & mds, new with NAOS)

Dialplan specifically is loaded to **/usr/share/xivo-config/dialplan/asterisk**, is checked after being loaded and can be configured on MDS and XiVO. The other confs work only on XiVO.

Behavior

xivo-config installs the config templates in the **base template directory** : **/usr/share/xivo-config/templates**
For each config, there is a directory and .restart file which contains what to do once the config is updated
You should not modify this folder, as it might be altered by future xivo-config updates
Config modifications should be defined instead in the **custom template directory** : **/etc/xivo/custom-templates**
You shall replicate/add there the file(s) you want to customized using the same structure as in base template directory. Check the examples.

On running **xivo-update-config**, the files from those directories will be thrown together and sent to the system directory where their configuration is applied. For files existing in both directory only the custom version will be taken.




```
root@xivo-naos:~# xivo-update-config
Generating configuration
(Re)templating
'dhcp' configuration OK, skipping.
'dialplan' configuration OK, skipping.
File asterisk/dirtyLeftover.conf lives outside the templated dialplan configuration
'mail' configuration OK, skipping.
'provd' configuration UPDATED.
'system' configuration OK, skipping.
(Re)configuring dahdi
(Re)configuring dhcp
```

(continues on next page)

(continued from previous page)

```
DONE
```

Notes:

- the build folder is temporary
- **xivo-update-config** does not destroy anything. It's up to you to clean the target of dirty/old config files

Variable replacement

To be detailed

Examples

Override existing dialplan conf

- Copy/Create the file you want from the base templates folder to the custom template folder

```
mkdir -p /etc/xivo/custom-templates/dialplan/asterisk
cp /usr/share/xivo-config/templates/dialplan/asterisk/extensions_lib_did.conf /
↪etc/xivo/custom-templates/dialplan/asterisk/
```

- Edit the file in custom-templates
- Run **xivo-update-config** (dialplan will be reloaded)
- The modified file is loaded to /usr/share/xivo-config/templates/asterisk/ & you can verify your modifications from asterisk cli

Override existing dhcp conf

- Copy/Create the file you want from the base templates folder to the custom template folder

```
mkdir -p /etc/xivo/custom-templates/dhcp/etc/default/dhcp
cp /usr/share/xivo-config/templates/dhcp/etc/default/dhcp/dhcp.conf /etc/xivo/
↪custom-templates/dhcp/etc/default/dhcp/
```

- Edit the file in custom-templates
- Run **xivo-update-config**
- The modified file is loaded to /etc/default/dhcp/dhcp.conf

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* :

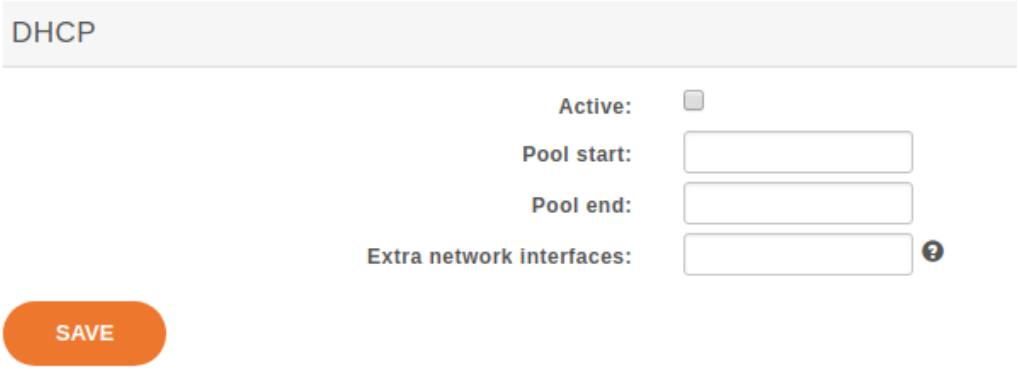


Fig. 1: *Configuration → Network → DHCP*

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 issuing 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:

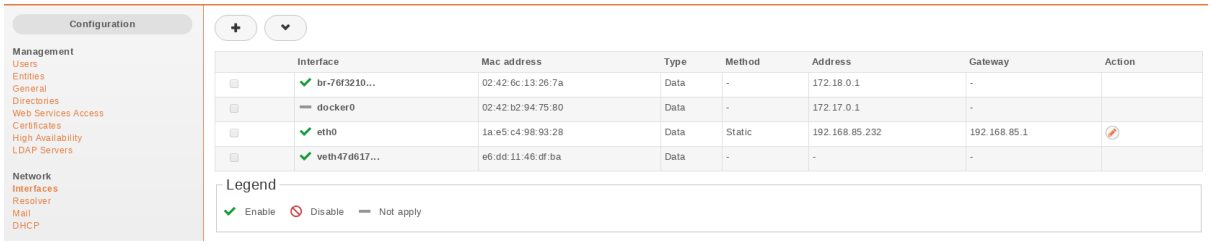


Fig. 2: *Configuration → Network → Interfaces*

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

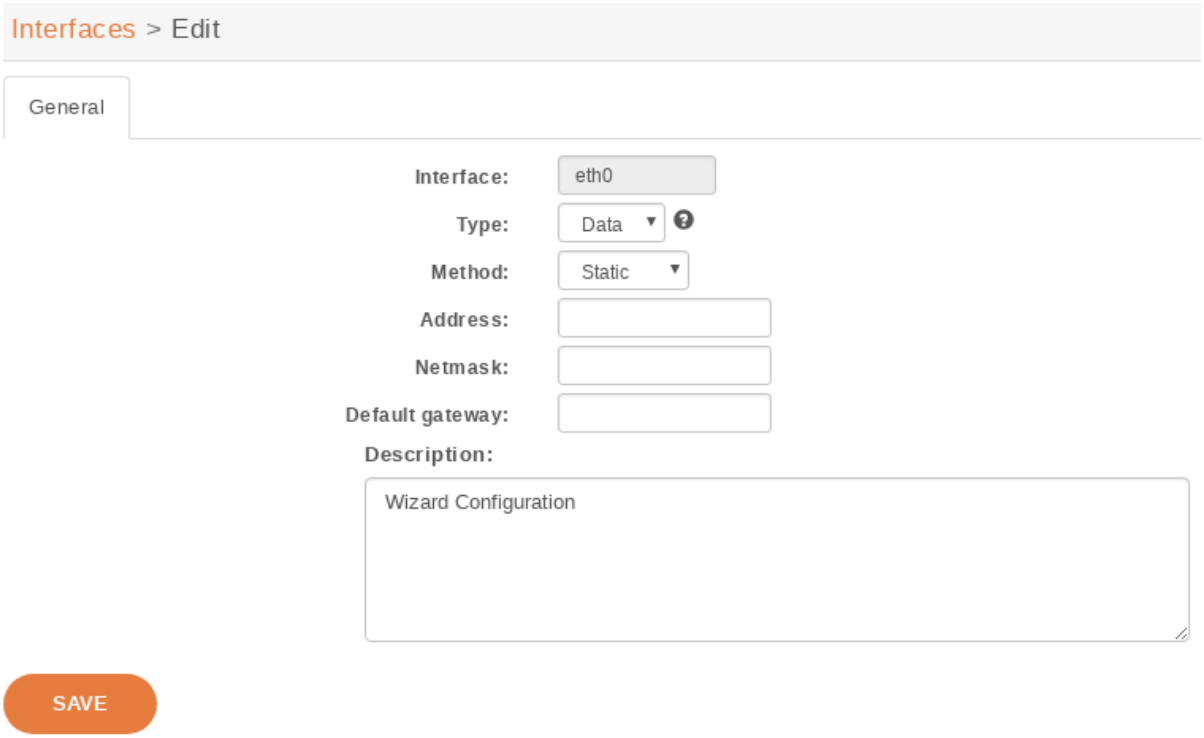


Fig. 3: *Configuration → Network → Interfaces → eth1 → Add*

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

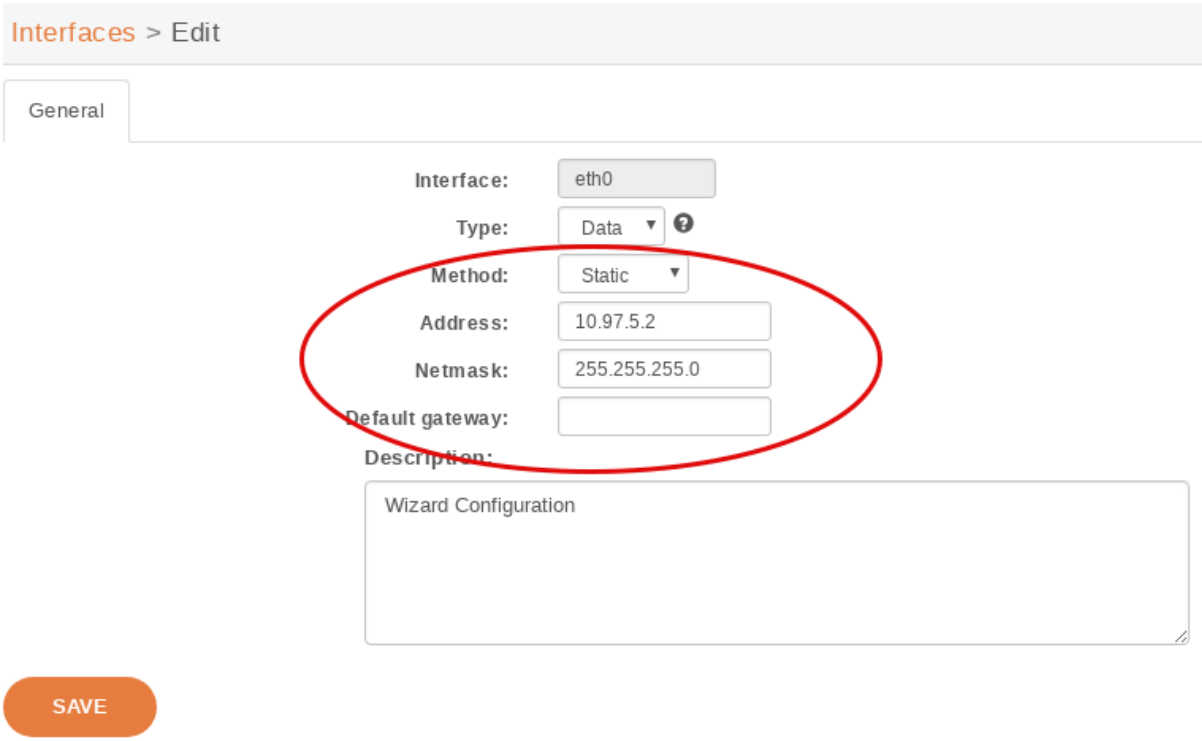


Fig. 4: *Configuration → Network → Interfaces → eth1 → Add*

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

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

and we get to this page:

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:

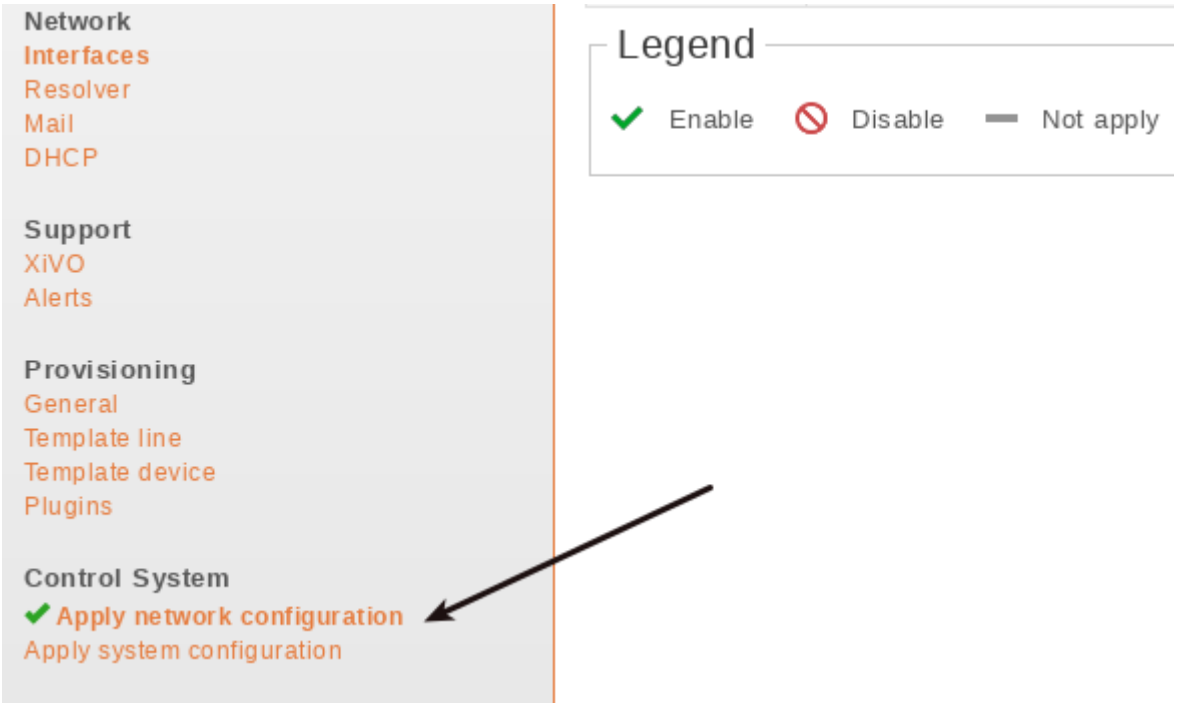
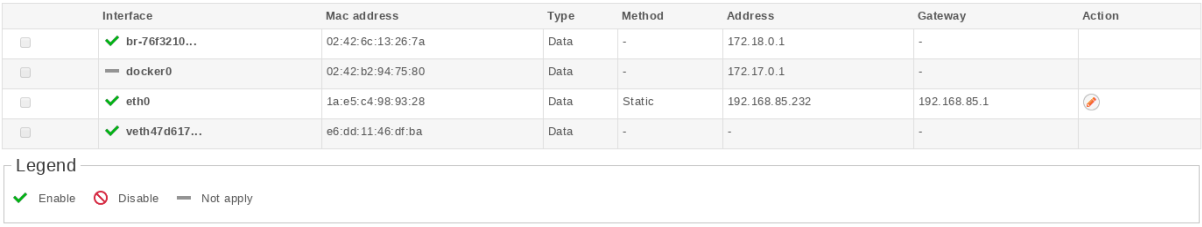


Fig. 5: Apply after modify interface



	Interface	Mac address	Type	Method	Address	Gateway	Action
	br-76f3210...	02:42:6c:13:26:7a	Data	-	172.18.0.1	-	
	docker0	02:42:b2:94:75:80	Data	-	172.17.0.1	-	
	eth0	1a:e5:c4:98:93:28	Data	Static	192.168.85.232	192.168.85.1	
	veth47d617...	e6:dd:11:46:df:ba	Data	-	-	-	

Legend
Enable Disable Not apply

Fig. 6: *Configuration → Network → Interfaces*

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

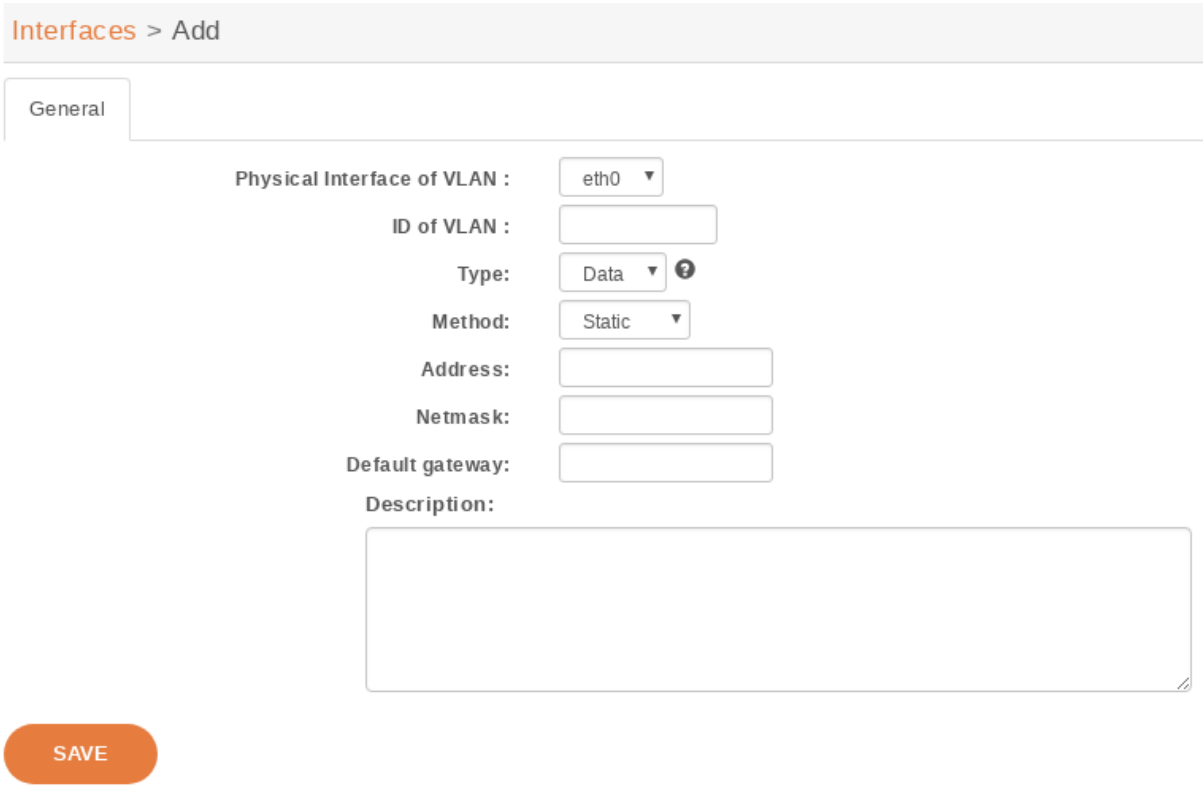


Fig. 8: *Configuration → Network → Interfaces → Add*

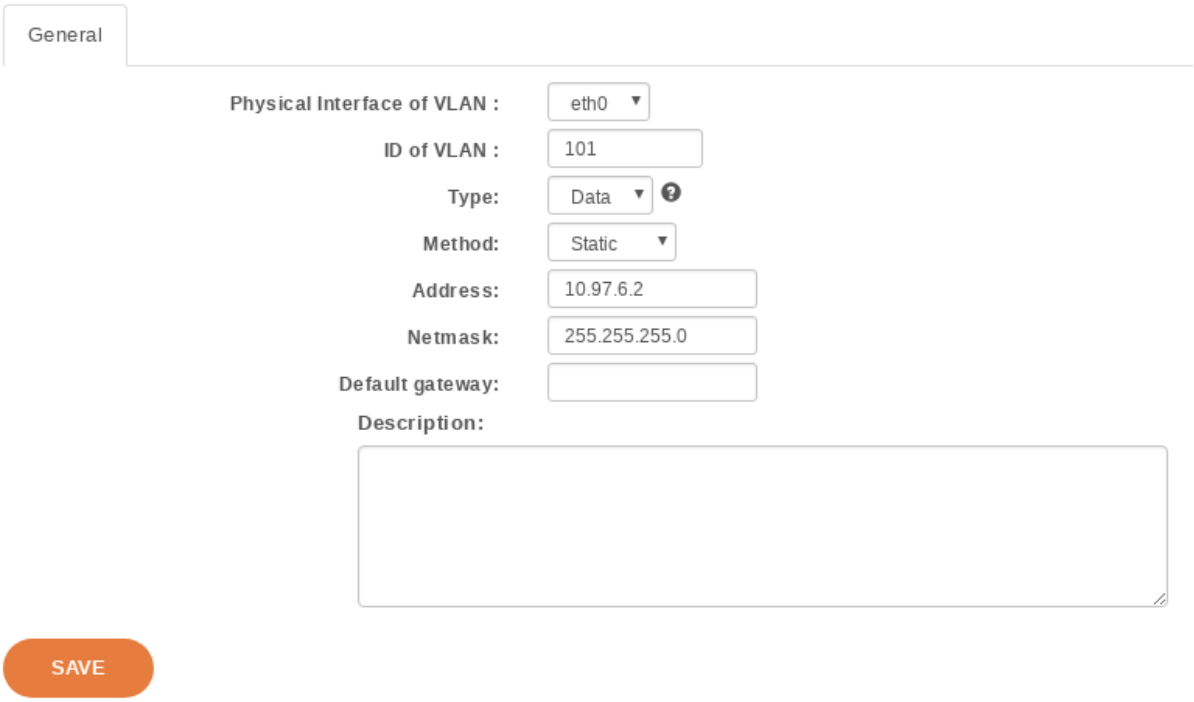


Fig. 9: *Configuration → Network → Interfaces → Add*

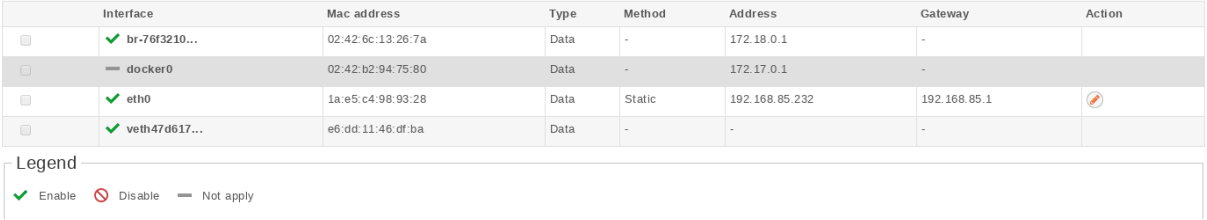


Fig. 10: *Configuration → Network → Interfaces*

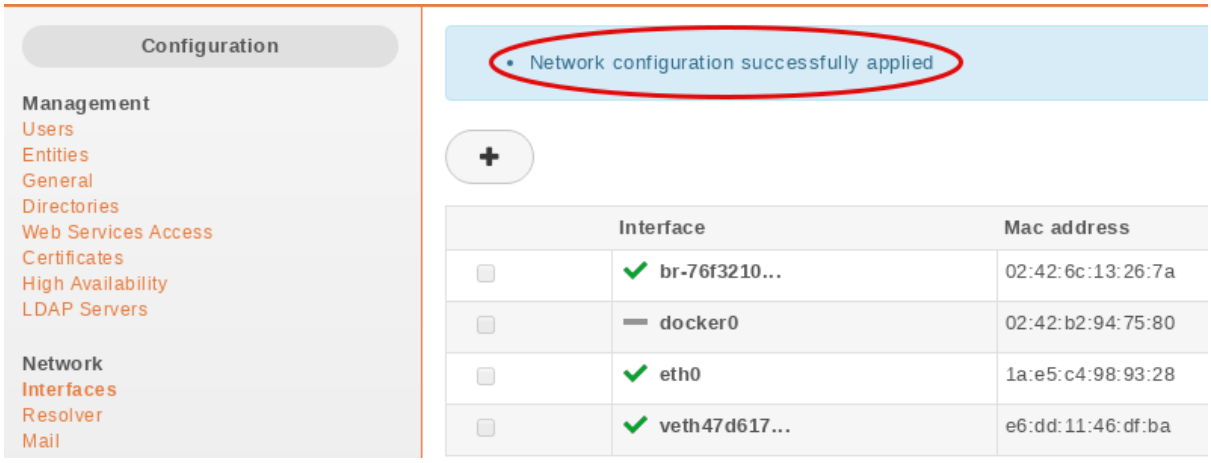


Fig. 11: Network configuration successfully apply

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).

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>
```

Database

Creation and Initialization

Starting from the Callisto version, the database on the **XiVO PBX** and **all mediaservers** is started inside a docker container based on custom image. On first startup the database will be initialized and the required structure and data will be created inside a host mounted folder `/var/lib/postgresql/17/data/`.

Custom database configuration

The database image contains a default postgres configuration and some specific defaults required by our application:

- the postgres default configuration is located in `/var/lib/postgresql/17/data/postgresql.conf`
- and our custom defaults are in `/var/lib/postgresql/17/data/conf.d/00-xivo-default.conf`.

Warning: Do not change these files!

If you need to change some parameters, create another file in `/var/lib/postgresql/17/data/conf.d/` prefixed with a number like `01` or upper and with a `.conf` extension. This file will be loaded after all defaults and can override any parameter.

Sample configuration

Here is an example to increase the default number of concurrent connection to the database:

`/var/lib/postgresql/17/data/conf.d/10-custom-max-connection.conf`:

```
max_connections = 300
```

Apply the configuration

Run this command to reload postgres configuration:

```
xivo-dcomp reload-db
```

It will not restart db container despite the message *"Killing xivo_db_1"*.

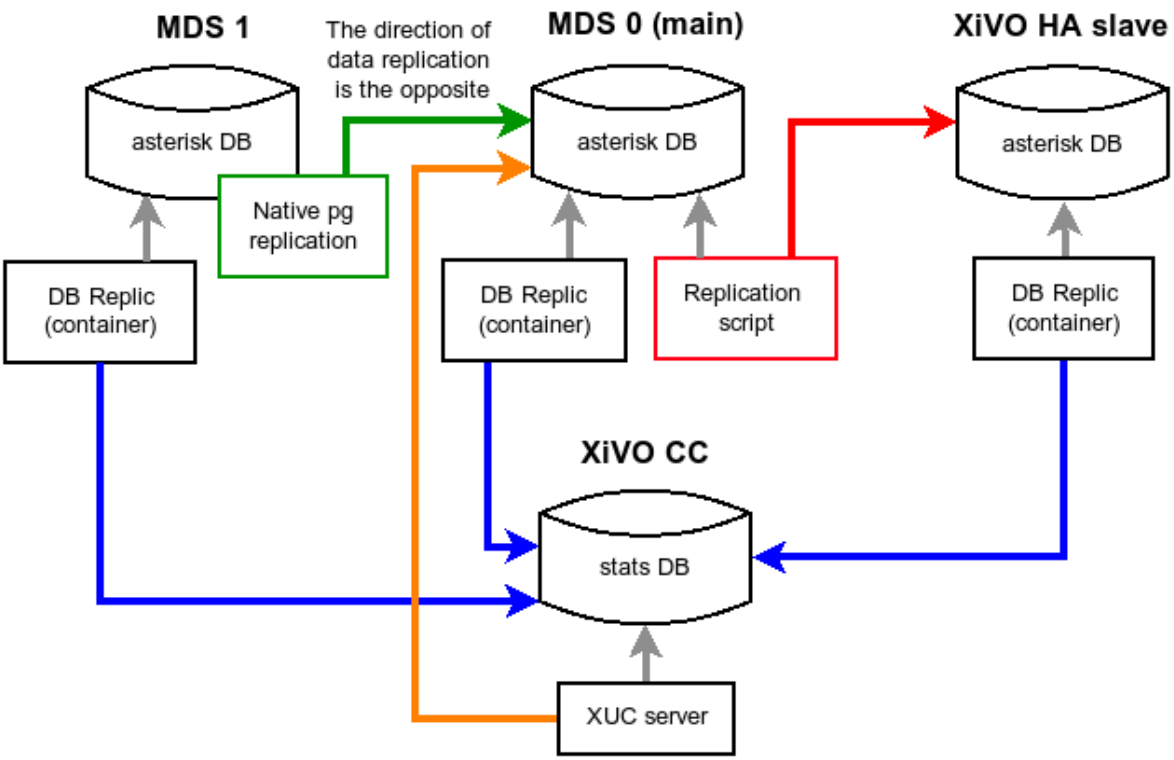
Files path summary table

File	Path	Comment
postgresql.conf	/var/lib/postgresql/17/da	Please, do not edit this file but use overriding mechanism explained in *Custom database configuration*
conf.d files	/var/lib/postgresql/17/da	Files are handled in Lexicographical order
pg_hba.conf	/var/lib/postgresql/17/da	Configure connection authorization in this file
postgresql-15-main.log	/var/log/postgresql/	Database log file.

Connections to database and replication schema

This schema shows required database connections between different types of servers. These connections must be authorized in `pg_hba.conf` file. It is being done automatically or it is described in installation or upgrade documentation.

It also shows how asterisk database is replicated. Only *configuration* tables are replicated between asterisk databases and only *event* tables (cel and queue_log) are replicated to stats database.



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/
- /var/lib/postgresql/17/data/*.conf
- /var/lib/postgresql/17/data/conf.d/*.conf

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.

> **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
```

Before restoring the database, all other services connected to it (XiVO CC, MDS) must be stopped also. You can list active connections: `sudo -u postgres psql -c "SELECT * FROM pg_stat_activity WHERE pid <> pg_backend_pid()"`

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

- *Default certificate*
  - *The default certificate is untrusted*
- *Regenerating Default Certificate*
  - *Default NGINX Certificate*
  - *Default Backend Certificate*
- *Replacing Default Certificate with a Trusted Certificate*
  - *Install Trusted Certificate for Nginx (and UC apps in UC mode)*
  - *Install Trusted Certificate for Backend services*

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 certificates shipped by default in XiVO:

- `/usr/share/xivo-certs/server.crt`: the default certificate used for backend services and REST APIs
- `/etc/docker/nginx/ssl/xivoxc.crt`: the default certificates used by the web interface (and also by the UC Application when in *UC Embedded mode*)

Default certificate

XiVO uses HTTPS where possible. The certificates are generated at install time. The main certificates are located

- in `/usr/share/xivo-certs/server.crt` (used by backend services and REST APIs)
- and in `/etc/docker/nginx/ssl/xivoxc.crt` (used ny Nginx and therefore the Webi)

However, these certificates are 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 Default Certificate

Default NGINX 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:

```
ssldir=/etc/docker/nginx/ssl
openssl req -nodes -newkey rsa:2048 \
  -keyout "${ssldir}/xivoxc.key" \
  -out "${ssldir}/xivoxc.csr" \
  -subj "/C=FR/ST=Rhone-Alpes/L=Limonest/O=Avencall/CN=$(hostname --fqdn)"
openssl x509 -req -days 3650 \
  -in "${ssldir}/xivoxc.csr" \
  -signkey "${ssldir}/xivoxc.key" \
  -out "${ssldir}/xivoxc.crt"
```

3. Reload Nginx service `xivo-dcomp reload nginx`.

Default Backend 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`.

Replacing Default Certificate with a Trusted Certificate

You can use a valid certificate for both Nginx service and Backend services. Though, in production environement, **the main need** is to deploy a valid certificate for Nginx (and the UC apps if you are in *UC mode*). To do this follow *Install Trusted Certificate for Nginx (and UC apps in UC mode)*

If you want also to change the certificate for Backend service follow *Install Trusted Certificate for Backend services*.

Install Trusted Certificate for Nginx (and UC apps in UC mode)

This certificate is used for XiVO Webi and, *more importantly*, for UC Application when in *UC mode*. In this case this is mandatory (at least to use WebRTC) to install trusted certificate.

For this, follow these steps:

1. Replace the following files with your own private key/certificate pair:
   - Private key: `/etc/docker/nginx/ssl/xivoxc.key`
   - Certificate: `/etc/docker/nginx/ssl/xivoxc.crt`
2. Reload Nginx configuration with:

```
xivo-dcomp reload nginx
```

3. Check that the certification chain is complete - see *What is a complete certificate chain*

Install Trusted Certificate for Backend services

Note: This is probably useless in most production environment. If you want to change the certificate on XiVO you probably want to change the NGINX one - see *Install Trusted Certificate for Nginx (and UC apps in UC mode)*.

If you want to use your certificate on backend services, 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. Set correct 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. 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
```

(continues on next page)

(continued from previous page)

```
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
```

4. 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.
5. 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
6. 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.

7. 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 by xivo-sysconfd. To make the change persistent, you need to create **custom templates**:
     1. in `/etc/xivo/custom-templates/system/etc/hosts` (based on template `/usr/share/xivo-config/templates/system/etc/hosts`)
     2. and in `/etc/xivo/sysconfd/custom-templates/resolvconf/hosts` (based on template `/usr/share/xivo-sysconfd/templates/resolvconf/hosts`)
8. Your X.509 certificate must have `subjectAltName` defined. See the example at cacert.org or detailed documentation at ietf.org.
9. Restart all XiVO services:

```
xivo-service restart all
```

Asterisk HTTP server

Asterisk HTTP server is used for WebRTC and xivo-outcall.

> **Warning:** **Security warning:** Asterisk HTTP server is configured to accept websocket connections from outside. You must ensure that the configuration is secure.

- ARI connection **must** be secured. Open file `/etc/asterisk/ari.conf` and check if the default password `Nasheow8Eag` was changed. If not, change it:
  - Generate a password (e.g. with `pwgen -s 16`)
  - replace:

    ```
    password = Nasheow8Eag
    ```

    by:

    ```
    password = <YOUR_GENERATED_PASSWORD>
    ```
- Open file `/etc/asterisk/http.conf` and check if `bindaddr` option is set to `0.0.0.0` (to accept outside websocket connections).
- You should also secure (e.g. via an external firewall) access to the asterisk HTTP server (which listens on port 5039).

WebSocket connection limit

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
```

Asterisk & SIP

Asterisk default SIP channel driver is since XiVO Izar *PJSIP*.

Note: Please note the *Asterisk chan_sip to pjsip Migration Guide*.

- *PJSIP Configuration*
  - *Where the configuration is stored ?*
  - *When the configuration is generated ?*
  - *How the configuration is generated ?*
  - *How to customize the configuration ?*
  - *How to reload the configuration ?*
- *Debugging*
- *Fallback to* `chan_sip` *SIP channel driver*

PJSIP Configuration

Note: This applies to >=Jabbah.07, >= Kuma.03 and forward

Where the configuration is stored ?

The pjsip configuration is stored in `/etc/asterisk/pjsip.d/01-pjsip.conf`.

When the configuration is generated ?

Warning: Since *Jabbah.07* the conf is only re-generated after an edition in the Webi.

Launching manually a `module reload res_pjsip.so` in asterisk CLI **will not** reload the **current** conf from the db, it will only re-load the current generated conf (see *How to reload the configuration ?*).

When you save an object in the Admin Web interface (a User, or a Line or ...) the Web Interface calls `xivo-sysconfd` service which will reload the configuration.

How the configuration is generated ?

1. Admin changes some configuration in Webi
2. Webi calls `xivo-sysconfd` service to ask for a `pjsip` reload
3. Then `xivo-sysconfd`:
   1. Calls a script (`/usr/sbin/xivo-generate-pjsip-conf`) which itself calls `xivo-confgen` to generate pjsip conf
   2. Calls `xivo-fix-paths-rights` script to ensure correct rights
   3. Then puts the genertated conf in `/etc/asterisk/pjsip.d/01-pjsip.conf`

4. When it's done, launch a module reload: `module reload res_pjsip.so`

4. And finally asterisk re-reads its pjsip conf

How to customize the configuration ?

If you want to customize the configuration:

1. add your own configuration file in the directory `/etc/asterisk/pjsip.d/`

   Note: Follow the asterisk documentation on how to override an existing section.

2. then see *How to reload the configuration ?*

How to reload the configuration ?

Two cases:

1. Reload the **generated** configuration (it will also reload what you might have changed in `pjsip.d` dir - see *How to customize the configuration ?*), run:

```
asterisk -rx 'module reload res_pjsip.so'
```

2. Reload the **current** configuration (from the db and from `pjsip.d` dir) - otherly said, regenerate & reload the whole configuration:

```
curl --insecure -XPOST -H 'Content-Type: application/json' 'http://
↪localhost:8668/exec_request_handlers' -d '{"ipbx": ["module reload res_pjsip.so
↪"]}'
```

Debugging

See useful links:

- The upgrade notes on the dialplan differences: *Asterisk chan_sip to pjsip Migration Guide*
- https://www.asterisk.org/new-pjsip-logging-functionality/
- https://wiki.asterisk.org/wiki/display/AST/Asterisk+PJSIP+Troubleshooting+Guide

Fallback to `chan_sip` SIP channel driver

Warning: If you chose to fallback to `chan_sip` you will need to redo it for each bugfix upgrade.

For Izar LTS a fallback mechanism has been implemented to be able to fallback to deprecated `chan_sip` SIP channel driver.

On your XiVO, run the following command to switch your XiVO from `res_pjsip` to `chan_sip`:

```
xivo-switch-sip-driver SIP
```

And follow the instructions displayed.

Note: If you're running an XDS installation you MUST switch all the XiVO and MDS of the installation to the same SIP channel driver.

Configuration Files

This section describes some of the XiVO configuration files.

- *XiVO Daemons Configuration*
  - *Configuration priority*
  - *File configuration structure*
  - *xivo-agentd*
  - *xivo-amid*
  - *xivo-auth*
  - *xivo-confgend*
  - *xivo-ctid*
  - *xivo-dao*
  - *xivo-dird-phoned*
- *Other Configurations*
  - *xivo_in_callerid.conf*
  - *xivo_ring.conf*
  - *ipbx.ini*

XiVO Daemons Configuration

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`

Other Configurations

xivo_in_callerid.conf

- Path: `/etc/xivo/asterisk/xivo_in_callerid.conf`
- Purpose: This file is used to normalize the caller number received from the provider.

> **Warning:** Please read *Caller Number Normalization* to understand what are the impacts of changing or not changing the caller number.

This file `xivo_in_callerid.conf` consists of:

- sections of configuration like `[national1]`
- each section having:
  - `comment` a comment to explain the role of this section
  - `callerid` a pattern matching a number
  - `strip` the number of digits to strip from the caller number
  - `add` what to add to the caller number

How the file is processed:

- Sections are only used to make differentations and give sense to options.
- Sections are scanned for matching in file order, so you can implement priorities easily.
- If the callerid didn't match: do nothing
- If a number matched:
- If strip only: strip specified number at the beginning of the number
- If add only: add the specified digits at the beginning of the number
- If both: begin with striping and finish with add. Usefull if incoming callerid need to be set in the "international format" : for example in France, replace (0) by (+33)

Examples (but again, take care of the important note above):

- 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:

```
systemctl restart xivo-agid
```

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

- Purpose: This file specifies various configuration options and paths related to Asterisk and used by the web interface.
- As file is embedded in docker, to edit it, you should execute:

```
docker cp xivo_webi_1:/etc/xivo/web-interface/ipbx.ini /tmp
vi /tmp/ipbx.ini # And save your modification
docker cp /tmp/ipbx.ini xivo_webi_1:/etc/xivo/web-interface/ipbx.ini
```

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

postgresql

- File location: `/var/log/postgresql/postgresql-17-main.log`
- Rotate configuration: `/etc/logrotate.d/xivo-manage-db`
- 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 uses nginx docker container 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 (webi container)
- API documentation (xivo-swagger-doc)

XiVO UC Applications

If you install the *UC Embedded* then the nginx container also listen on TCP/8443 and serves the UC application (via xuc/xucmgt container).

Notes

Customization of the services to be used through the nginx container are not supported.

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/ntpsec/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/ntpsec/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/ntpsec/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.

agid

Important: This is needed in order fo agid to access Google/Apple Push Notification servers when *Mobile application* is used

Add the following variable to `/etc/docker/xivo/custom.env` file:

```
HTTPS_PROXY=http://username:password@proxyip:proxyport
HTTP_PROXY=http://username:password@proxyip:proxyport
```

If you have *Meetingrooms* installed, you **MUST** add the `NO_PROXY` var with the default content *and* the Meeting room server IP. For example if Meetingroom server IP is 192.168.56.2 you **MUST** add the following NO_PROXY variable in the `/etc/docker/xivo/custom.env` file:

```
NO_PROXY=localhost,127.0.0.1,xivo,config_mgt,192.168.56.2
```

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*.

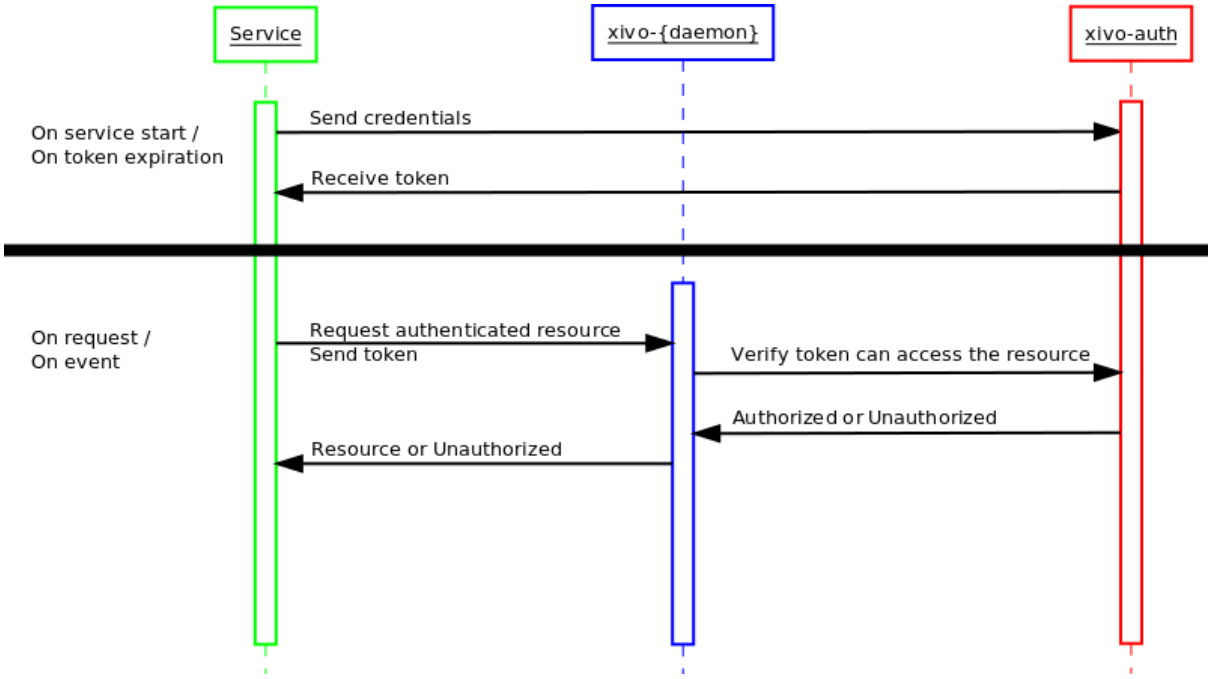


Fig. 12: Call flow of service authentication

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.

Webi admin users

Webi users with limited rights can be configured in the *Configuration → Management → Users* menu.

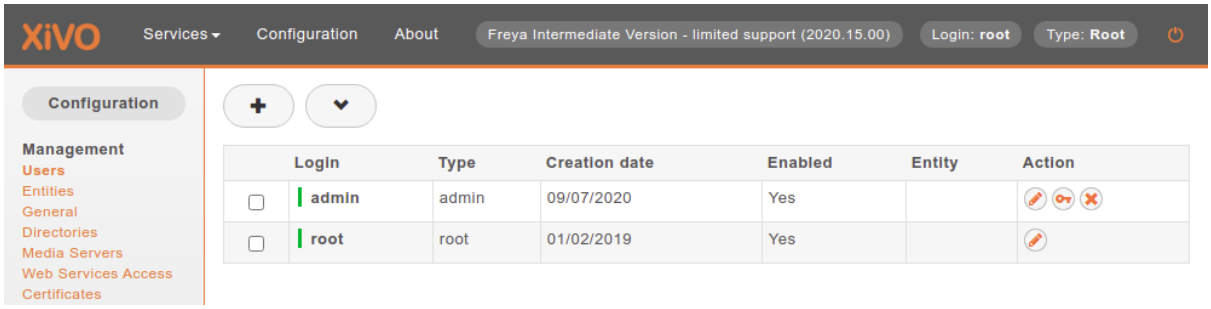


Fig. 13: *Configuration → Management → Users*

Rules

Newly created webi user can't see any webi menus. To add them, use the *key* action button. Then choose objects which can be edited by this user and save the form.

Important: These menus require additional permissions to allow webi to access ConfigMgt and external Confd port:

- *Configuration → Management → Media Servers*
- *Services → IPBX → IPBX Settings → Labels*
- *Services → IPBX → IPBX Settings → Users*
- *Services → IPBX → Call Management → Outgoing calls*
- *Services → IPBX → Trunk management → SIP Protocol*
- *Services → IPBX → Trunk management → SIP Provider*
- *Services → Call Center → Queues*

To allow admin users to use these menus, you must also give them the permission *Authenticate configuration server web services request (required for admin users)* from the *Rules → IPBX → Control System* menu.

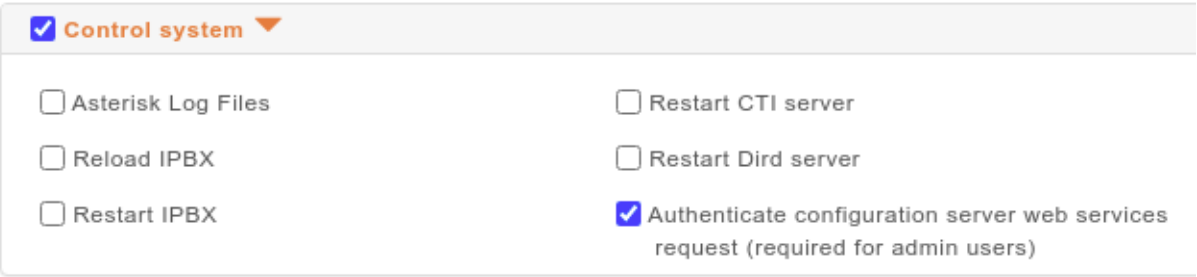


Fig. 14: *Rules → IPBX → Control System*

You can find detailed explanation in the *XiVO Session* authentication backend documentation and the *xivo-auth Developer's Guide*.

---

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

2020.16

- *XiVO Session* backend now send correct ACL for webi administrator of type admin, depending on the admin webi permissions.

2020.14

- *XiVO Session* backend can now authenticate webi administrator of type admin.

2020.12

- Added *XiVO Session* backend

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*.

Call to confd from the Webi

In legacy webi PHP code, calls to confd are done *server side* through the internal confd port (listening only on localhost). On this endpoint, no authentication is necessary.

While moving some part of the webi code towards Js, needed calls to confd are done *client side* and therefore can only be done through the external confd port (listening on all IP interfaces). On this endpoint authentication is needed.

The following schema summarizes the approach:

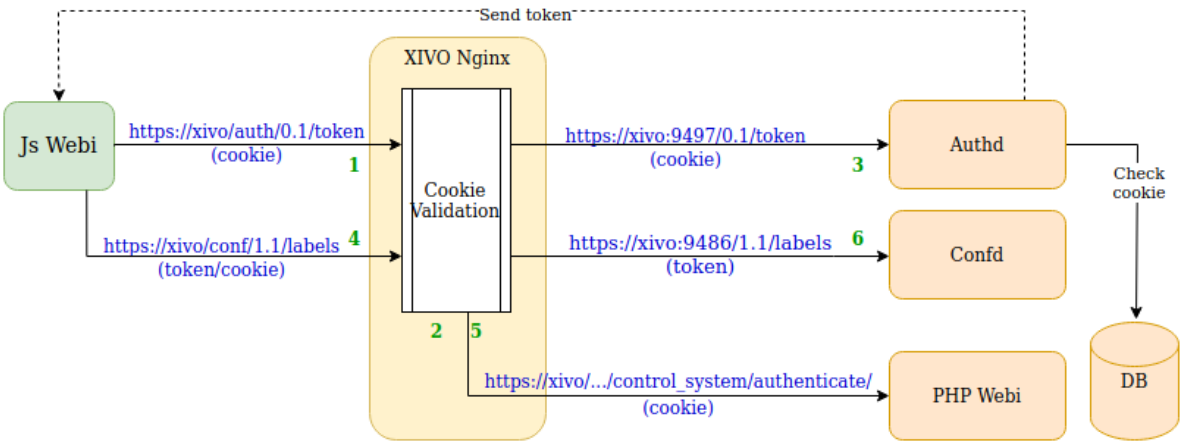


Fig. 15: Accessing confd API from Webi client side. `(source)`

Given a webi administrator is logged in the webi and opens a menu rewritten in Js:

1. Js client asks a token to *authd*, sending its Webi cookie
2. This call is proxified via Nginx to *authd* which first asks the Webi to validate the cookie before proxifying the request
3. *authd* returns a token based on the cookie. For this :
   1. it validates the cookie via the Webi
   2. it creates a token with ACL based on the logged webi admin's permission (see *XiVO Session* backend and *Webi permission to Confd ACL mapping*)
4. Js client calls *confd* API with token (and cookie)
5. This call is proxified via Nginx to *confd* which first asks the Webi to validate the cookie before proxifying the request
6. Confd returns result

Webi permission to Confd ACL mapping

Here's how the mapping between webi permission to confd ACL is done:

1. the *XiVO Session* backend retrieve the webi admin user and its configured webi permissions
2. these webi permissions are converted to a python dict
3. which is then flattened, read and compared to a mapping dict named *WEBI_PERMS_TO_CONFD_ACL* to retrieve the correct ACLs

If you want to add a mapping you simply have to edit the `WEBI_PERMS_TO_CONFD_ACL` dict to add a new mapping between a menu path to an confd ACL or list of ACL.

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 Login section.

XiVO Session

Backend name: `xivo_session`

Purpose: Authenticate a XiVO administrator's session. This backend receives an administrator's webi cookie with session id and returns a valid token with a confd ACL depending on the logged administrator webi permission:

- Webi administrator of type *root* receives a wildcard confd ACL
- Webi administrator of type *admin* receives a confd ACL depending on its webi permission (see table below)

Current mapping implemented:

Webi Permission	confd ACL
Services -> IPBX -> IPBX Settings -> User labels	`confd.labels.#`
Services -> IPBX -> IPBX Settings -> Users	`confd.labels.read`

See *Webi permission to Confd ACL mapping*.

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 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*.

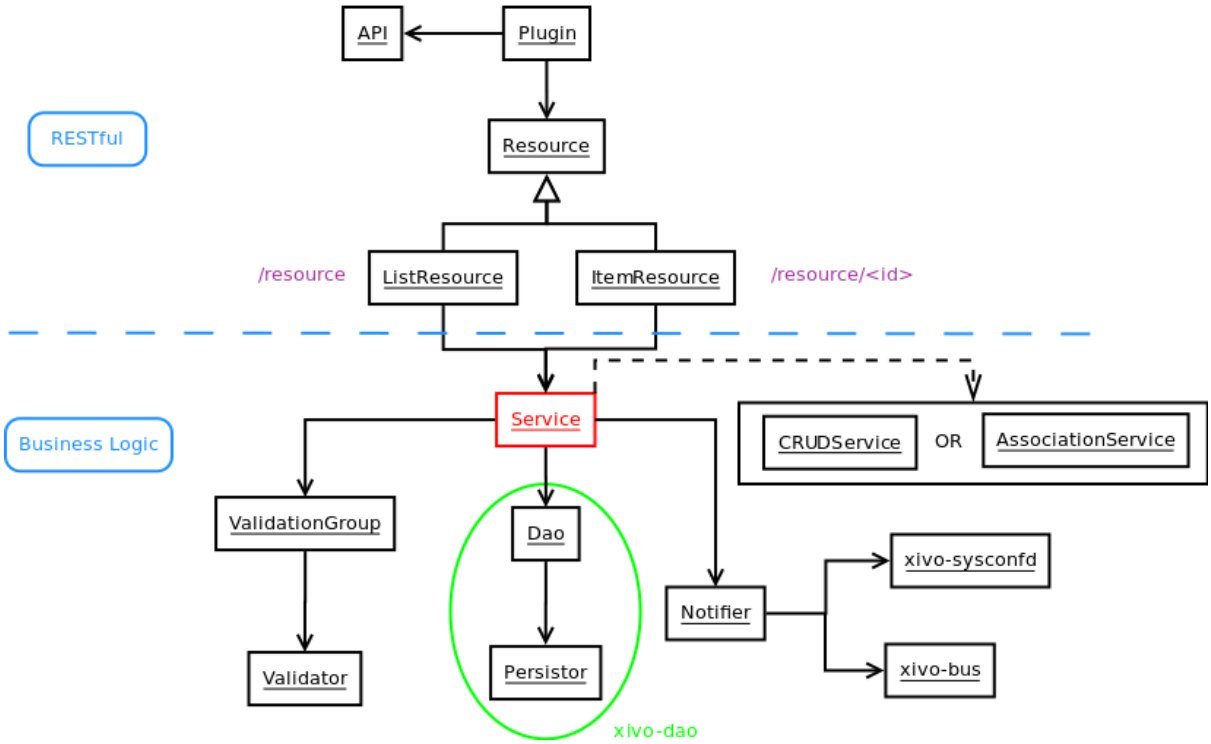


Fig. 16: Plugin architecture of xivo-confd

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)

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 *Personal contacts* 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:

```yaml
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
```

(continues on next page)

(continued from previous page)

```
   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
                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
```

(continues on next page)

(continued from previous page)

```
            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:

```yaml
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:

```yaml
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
```

(continues on next page)

(continued from previous page)

```
        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 -*-

```

(continues on next page)

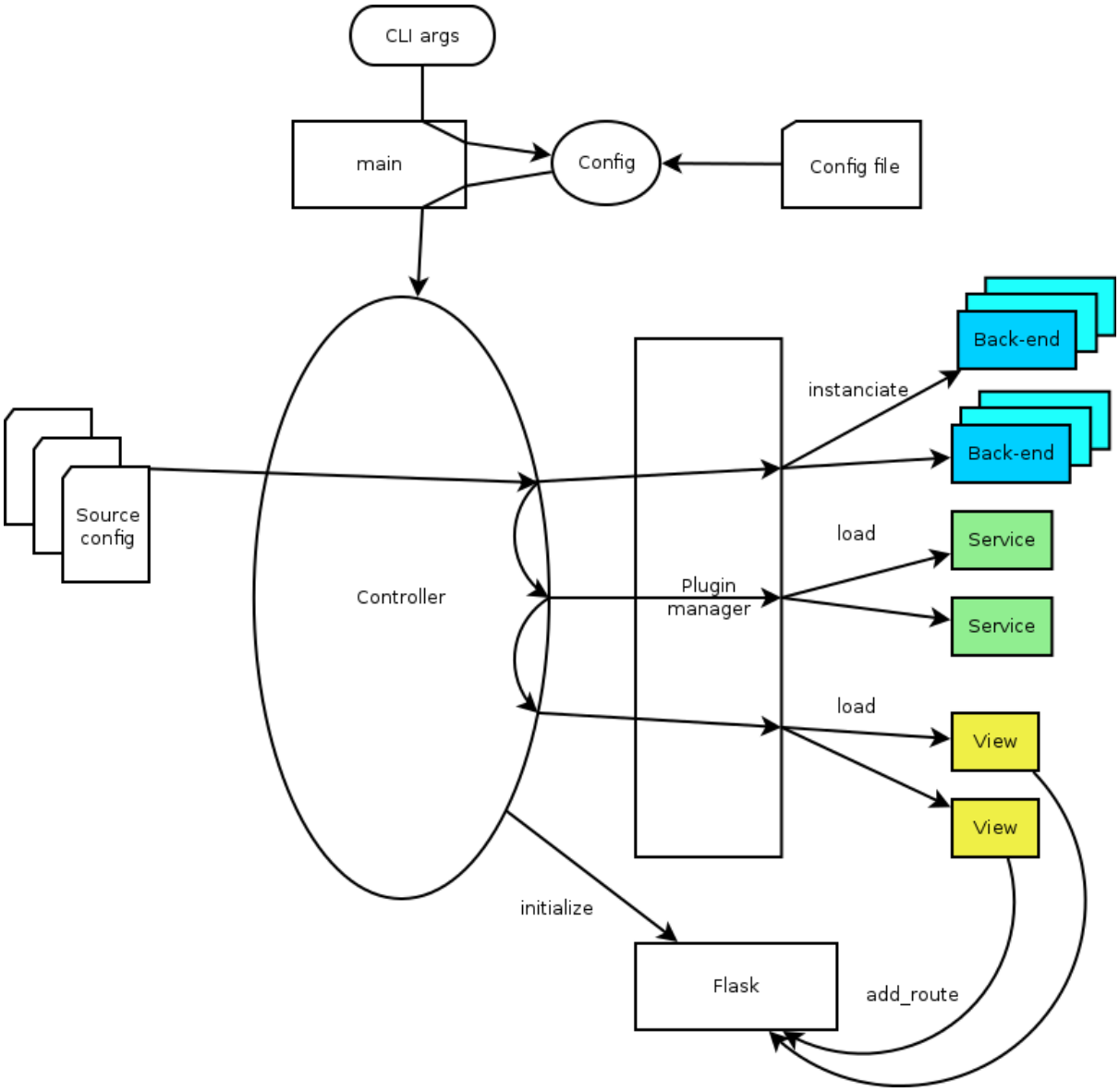


Fig. 17: xivo-dird startup flow

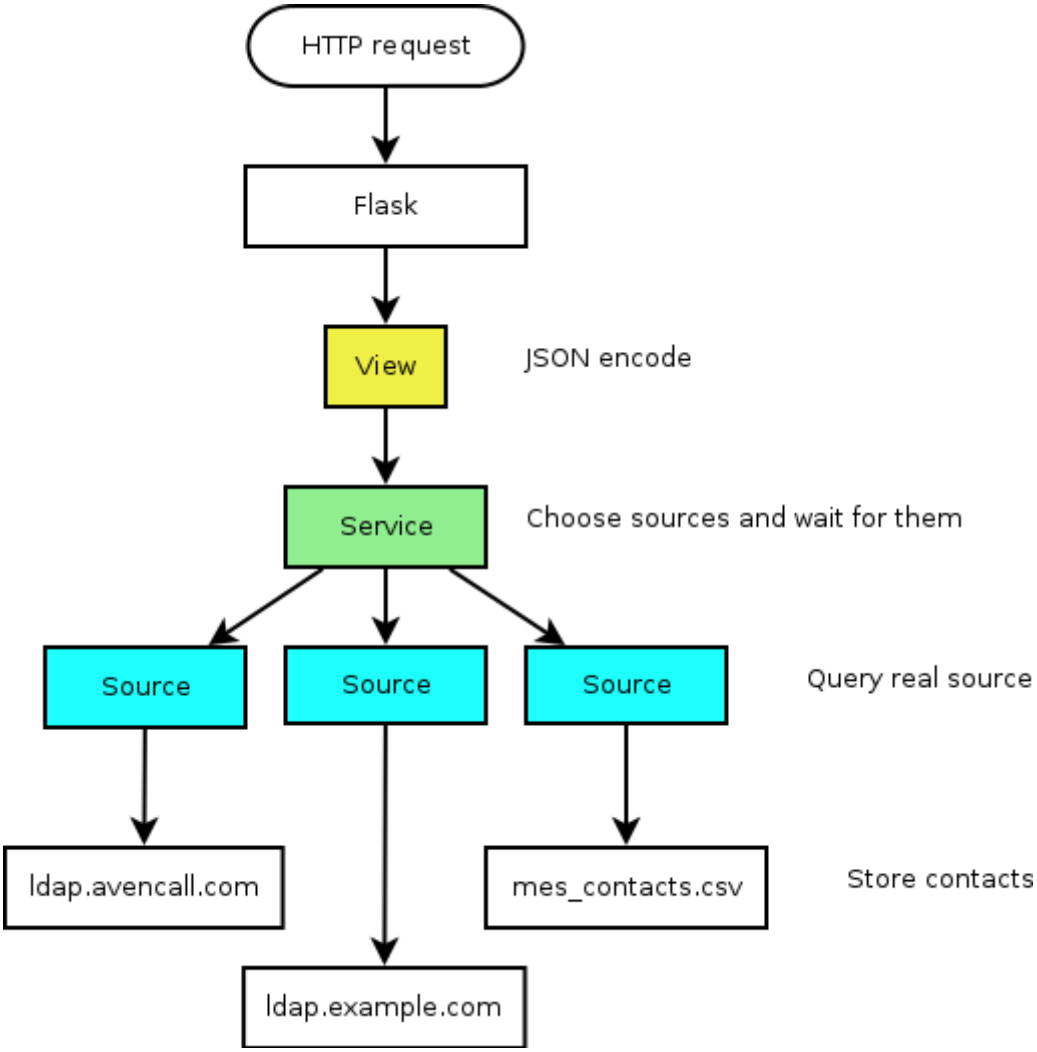


Fig. 18: xivo-dird HTTP query

(continued from previous page)

```
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(),

    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

```

(continues on next page)

(continued from previous page)

```
    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

```

(continues on next page)

(continued from previous page)

```
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')


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.

Note: For purpose of alphabetic ordering in results, accented character are converted to unaccented.

Field used to order result

Field for ordering is selected based on configuration of view/display. First non-empty field with type *name* is used. If there is no relevant value available, the result goes to end.

Relevance of results in lookup

Lookup results are ordered in way to prefer more relevant results:

1. exact matches of whole word
2. exact matches at beginning of word
3. exact matches anywhere
4. matches similar to term

---

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`):

```yaml
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"
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`):

```yaml
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.

Configuration

Example (a file inside `source_config_dir`):

```yaml
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`):

```yaml
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)

xivo_meetingroom

Back-end name: xivo_meetingroom

Purpose: search directory entries among meetingrooms (static or personal)

You should only have one source of type `xivo_meetingroom`.

Configuration

The configuration should be as follow (a file inside `source_config_dir`):

```yaml
type: xivo_meetingroom
name: xivo_meetingroom
db_uri: postgresql://asterisk:proformatique@localhost/asterisk
searched_columns:
    - name
    - display_name
    - number
format_columns:
    display_name: '{display_name}'
    name: '{display_name}'
    phone: '{number}'
get_all_rooms_keywords: conference, visio
```

get_all_rooms_keywords
keywords used to return all meeting rooms (both static and relevant personal) - should be lowercase and separated by coma `,`

`unique_column` is not configurable, its value is always `id`.

Integration of XiVO dird with the rest of XiVO

Configuration values

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.

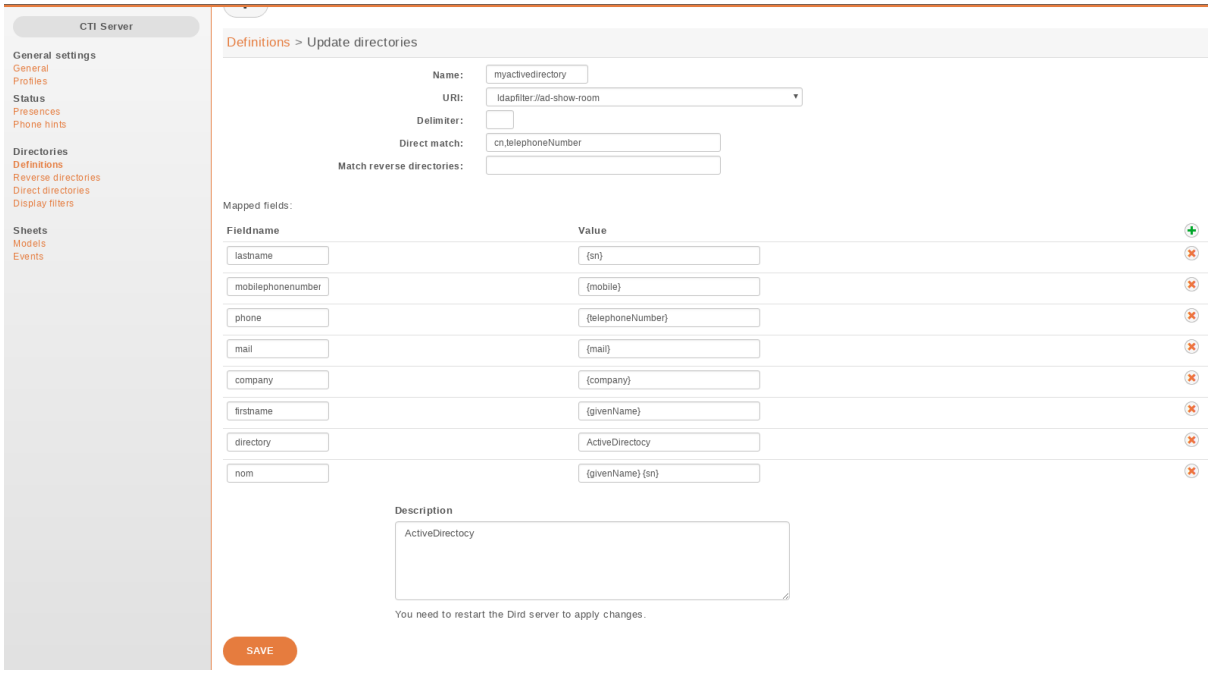
- Add a *unique_column* to your sources.
- Add a *favorite* column to your display

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*
- *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 a major Postgres version 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': [
            '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.

xivocc-upgrade script

Usage

Note:

- You can't use xivocc-upgrade if XiVOCC is split on multiple servers

This script will update *XiVOCC* 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 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

[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
```

(continues on next page)

(continued from previous page)

```
[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.

Pass Wizard with a JSON

After any clean installation of XiVO, it must be initialized through a program called wizard. Wizard is usually passed through the web interface, but there is another way - through a REST API.

You can pass the wizard with a single request to the API, by providing a specific JSON body as argument. You can use the swagger found on XIVO_IP/api > xivo_confd > wizard to send the POST request.

The body to fill has this form :

```
{
"admin_password": "string",
"license": true,
"timezone": "string",
"language": "de_DE",
"entity_name": "string",
"network": {
    "hostname": "string",
    "domain": "string",
```

(continues on next page)

(continued from previous page)

```
    "interface": "string",
    "ip_address": "string",
    "netmask": "string",
    "gateway": "string",
    "nameservers": [
    "string"
    ]
},
"context_incall": {
    "display_name": "Incalls",
    "number_start": "string",
    "number_end": "string",
    "did_length": 0
},
"context_internal": {
    "display_name": "Default",
    "number_start": "string",
    "number_end": "string"
},
"context_outcall": {
    "display_name": "Outcalls"
},
"default_french_configuration": true,
"handle_system_conf": true,
"run_scripts": false
}
```

For example, you can use curl to send the JSON:

```
curl -X POST \
-H 'Content-Type: application/json' \
-H "Accept: application/json" \
-d @/JSON/LOCATION --insecure "https://XIVO_IP:9486/1.1/wizard"
```

In this command, you need to modify :

- JSON/LOCATION by the path of you json file
- XIVO_IP by the IP of you XiVO

Some key's meaning will be detailed below.

handle_system_conf

Value: `true | false`, default is `true`

By default the XiVO appliance handles the system configuration (network interfaces, nameservers etc.).

If `handle_system_conf` is set to `false` when launching the wizard, then the XiVO **will not** touch the system configuration. It will not touch/change:

- `/etc/hosts`
- `/etc/hostname`
- `/etc/mailname`
- `/etc/network/interfaces`
- `/etc/postfix/canonical`
- `/etc/postfix/main.cf`

- `/etc/resolv.conf`

All the settings passed to the wizard API will only be set in the XiVO db, but will not be applied on the system.

> **Warning:** Beware, though, that even if you run the wizard with `handle_system_conf` to `false` the system configuration will still be accessible via the webi. If you change it there and then click on *Apply network configuration* this will touch/change the system configuration.

run_scripts

Value: `true | false`, default is `false`

If set to *true*, after the wizard is finished, all executable scripts in directory */etc/xivo/wizard.d* are going to be run.

Those scripts must be runnable by xivo-confd (file has permission set to 'x' for *www-data*) plus any necessary right to perform their additionnal actions.

The scripts are run in ascii order (0-9, A-Z, a-z). In case of unsuccessful run, an exception is raised and all non-finished scripts are stopped. You can then find the failing script's output in *xivo-confd* log. Another exception is raised if the directory */etc/xivo/wizard.d* doesn't exist.

4.1.2 IPBX Configuration Guide

Advanced Configuration

This section describes the advanced system configuration.

XiVO General Settings

XiVO offers the possibility to configure the general settings via the *Configuration* → *Management* → *General* page.

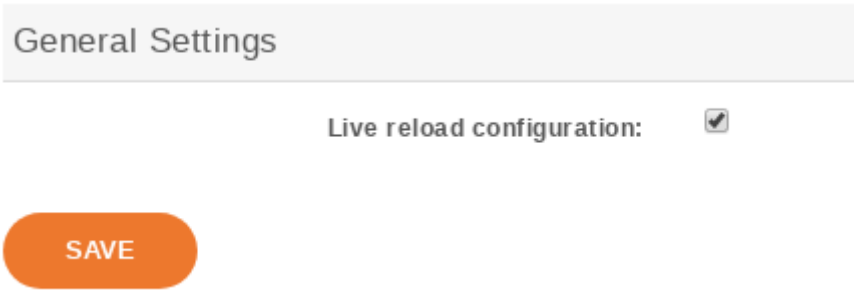


Fig. 19: Configure XiVO General Settings

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**

continues on next page

Table 1 – continued from previous page

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.
	Diversions	N.A.	N.A.
	Qualifications	N.A.	N.A.
Agents	*all*	*all*	`queue reload all`
Queues penalties	*all*	*all*	`queue reload all`

continues on next page

Table 1 – continued from previous page

Skills	*all*	*all*	`queue reload all`
Skill rules	*all*	*all*	`queue reload all`
Qualifications	*all*	N.A.	N.A.

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:

- 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.

Telephony certificates

Note: this figure is current not used and not fully described.

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 Xuc

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:

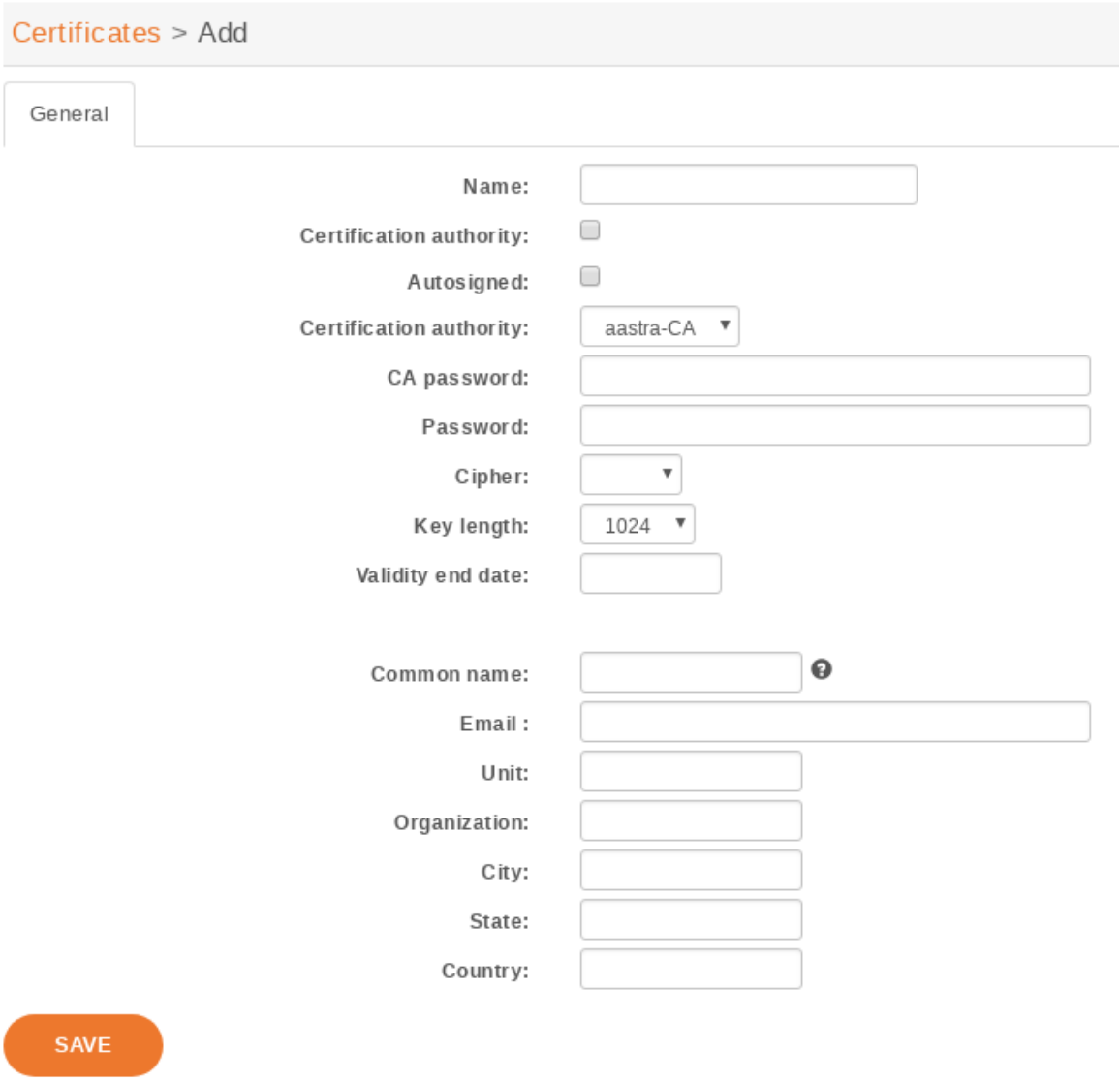


Fig. 20: Adding a certificate

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
- *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.

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.

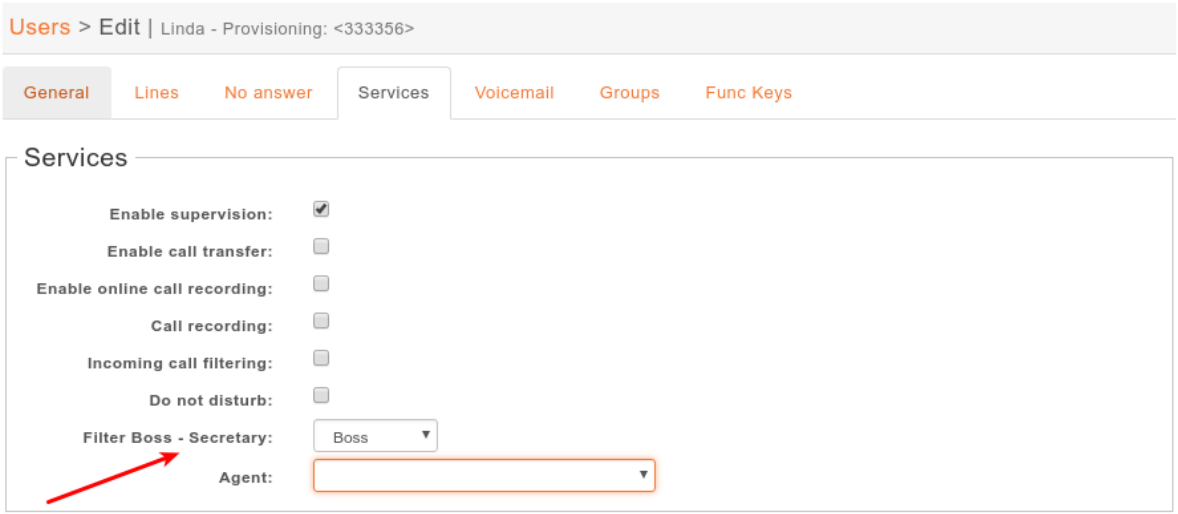
Quick Summary

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

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.

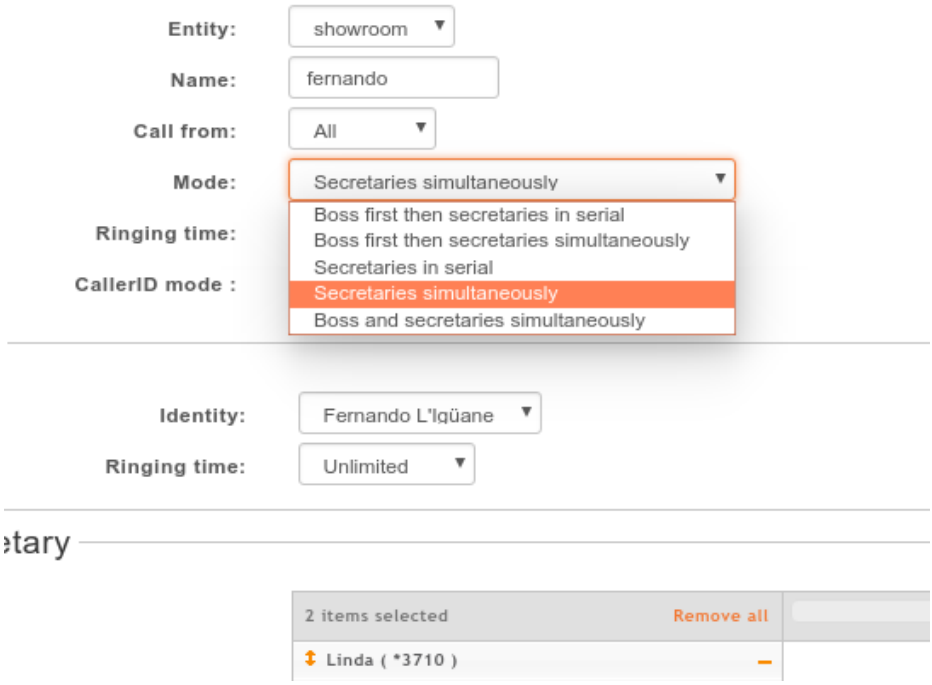


Creating a Filter

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
- 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.

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*.

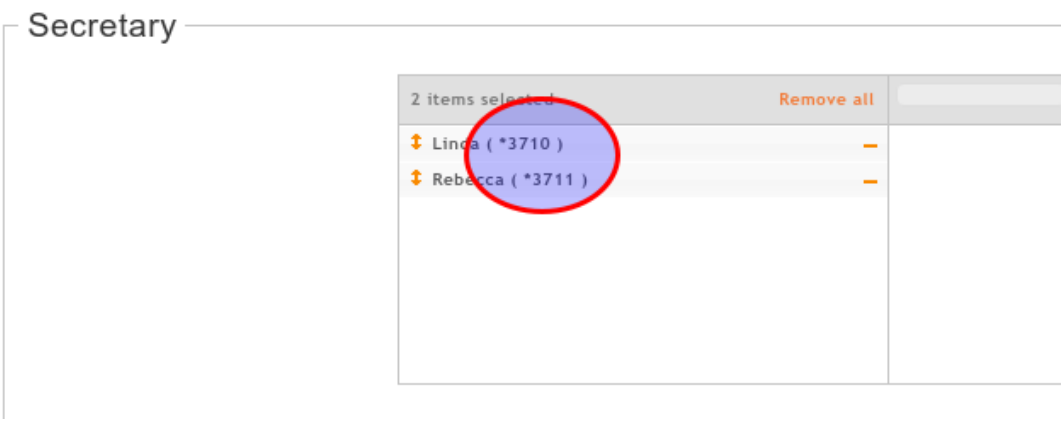
The call filter has to be activated for each secretary if more than one is defined for a given boss.

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`.



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.

Limitations

In case it is needed to switch the roles between users (eg. change secretary user to a boss user and vice versa) it is not possible to make this change by editing the call filter. In order to switch the roles the old call filter should be removed and then a new call filter with the correct role assignment should be created.

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.

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 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.

Configuration

The call completion extension is enabled via the *Services* → *IPBX* → *IPBX services* → *Extensions* page, in the *General* tab.

Fig. 21: 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.

Caller Number Normalization

Providers can send the **caller** number in various formats (e.g. for french national number you could receive 0123456789 or +33123456789 or even 33123456789). You can change the caller number to adapt to your need. For examples:

- you may want to always display incoming numbers in E.164 format (e.g. +33123456789)
- or, if you use a prefix to dial outgoing numbers (like a 0), you should add a 0 to the received caller number so you can redial it.

This caller number normalization is done via the *xivo_in_callerid.conf* file.

Important: Before modifying this file, you must know:

- that any caller number modification **will break** the *Customer Call History*,
- and that the *Reverse lookup* is done with the caller number **after** it has been modified through the rules of this `xivo_in_callerid.conf` file.

Therefore, if you need to have the *Customer Call History* (used in *CC Agent* and *Switchboard* applications), then you must void all the rules in this file (i.e. remove the content of the `add` and `strip` lines so the caller number is not changed). *But then* you must take care that, if you also use the *Reverse lookup*, the phone numbers in your directory are stored in the same format than received from the provider (i.e. if your provider sends *+33123456789* the phone number must be stored in the directory as *+33123456789*).

Default rules

By default the following rules are applied

National number with missing prefix

- rule: if caller number does not start by `0` and is 9 digits long
- action: add a leading `0`
- example: `123456789` is normalized to `0123456789`

French national number E.164 format

- rule: if caller number starts with `+33` and is followed by 9 digits
- action: replace `+33` with `0`
- example: `+33123456789` is normalized to `0123456789`

International number E.164 format

- rule: if caller number starts with + followed by at least 4 digits
- action: replace + with `00`
- example: `33123456789` is normalized to `0033123456789`

International number with missing prefix

- rule: if caller number does not start by `0` and is at least 11 digits long
- action: add a leading `00`
- example: `33123456789` is normalized to `0033123456789`

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.

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

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

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.

Search Dashboard

Call logs can be accessed using the menu *Services* → *IPBX* → *Call management* → *Call Logs* page.

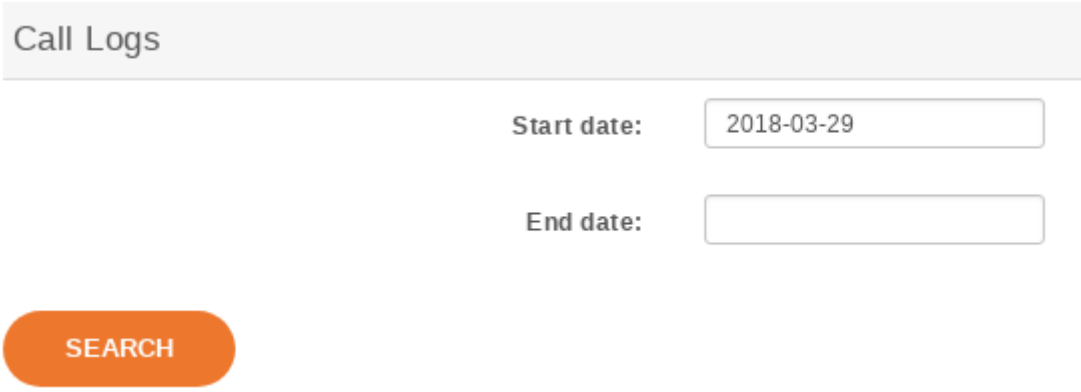


Fig. 22: Calls Records Dashboard

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
- 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

REST API

Call logs are also available from *xivo-confd REST API*.

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
```

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.

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      ␣
↪      |                     |                 |               |       |
```

(continues on next page)

(continued from previous page)

```
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             |        |
```

Conference Room

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.

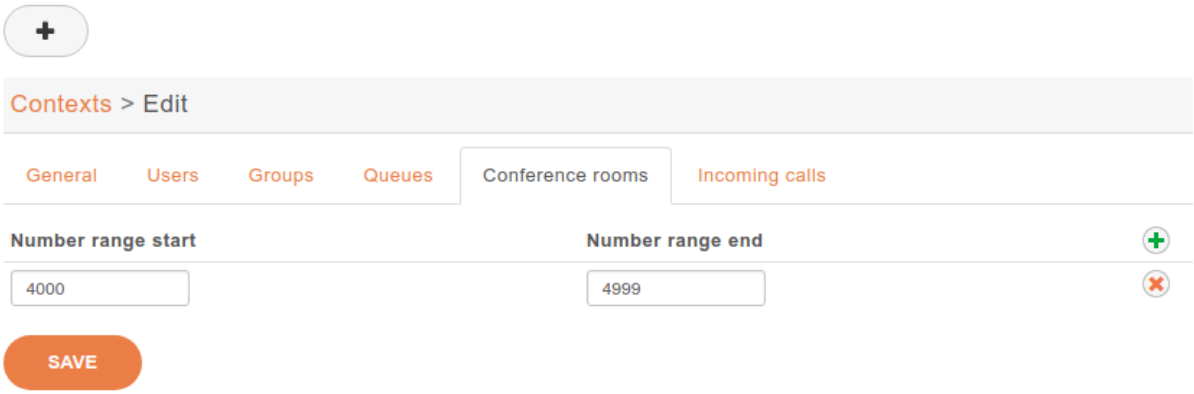


Fig. 23: Adding a conference room number range to the default context

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.

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:

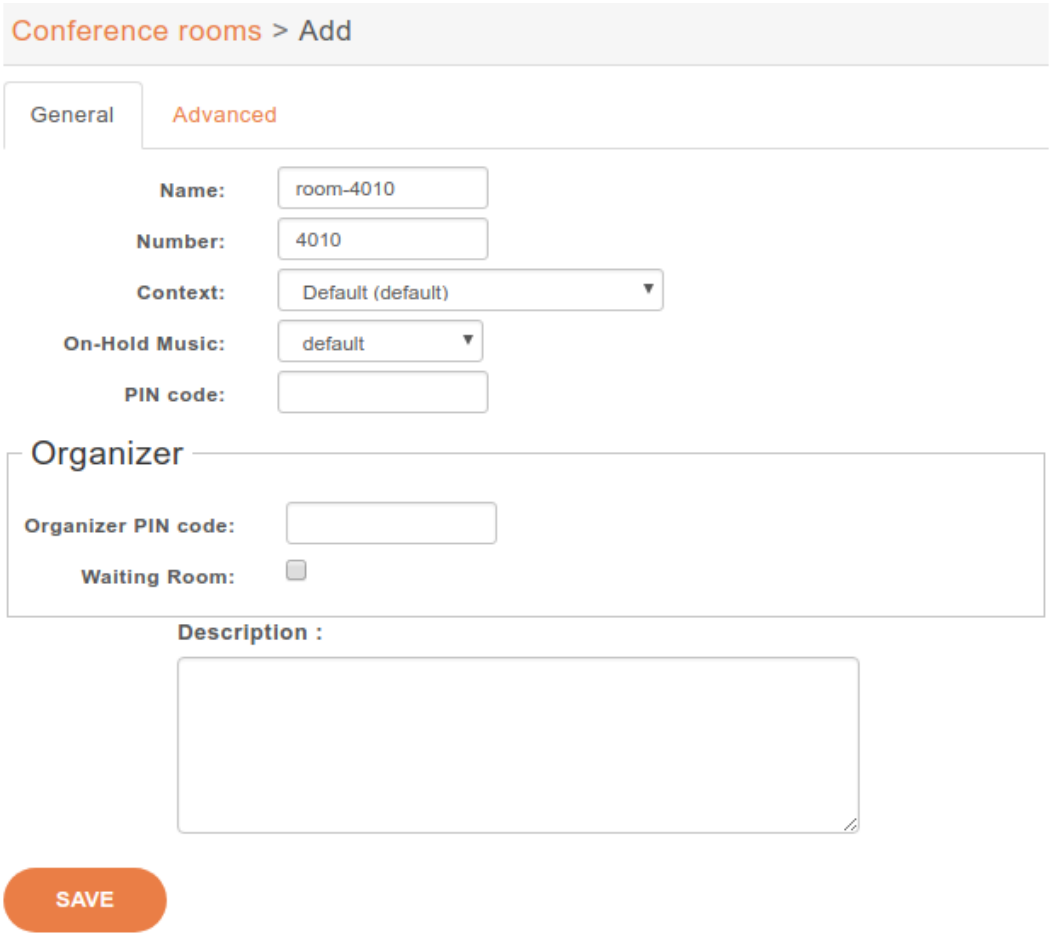


Fig. 24: Creating conference room 4010

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

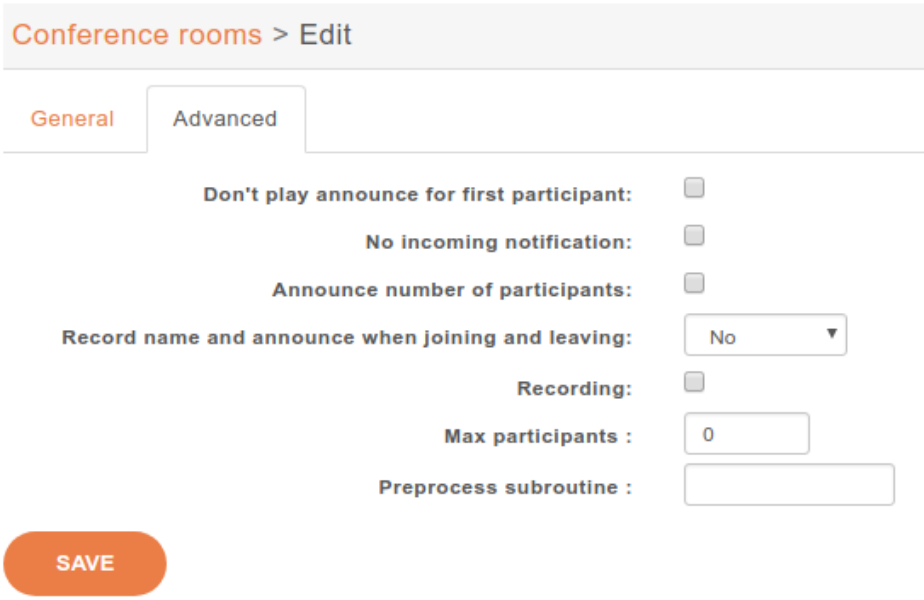


Fig. 25: Conference room configuration advanced tab

- *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*.

CTI Server

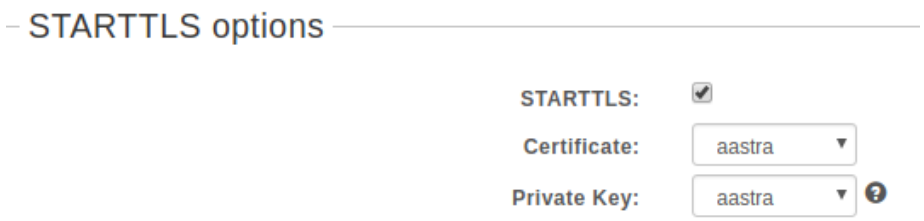
The CTI server configuration options can be found in the web-interface under the services tab.

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.

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:

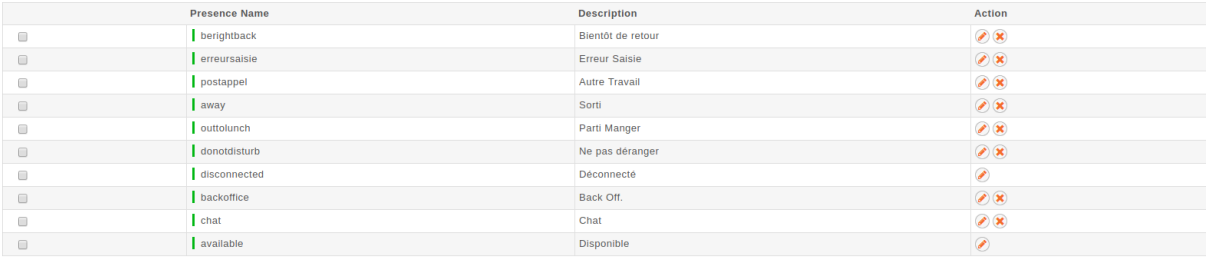
```
auth:
    backend: backend_name
```

where *backend* name is the name of an enabled *xivo-auth* *Backends Plugins*.

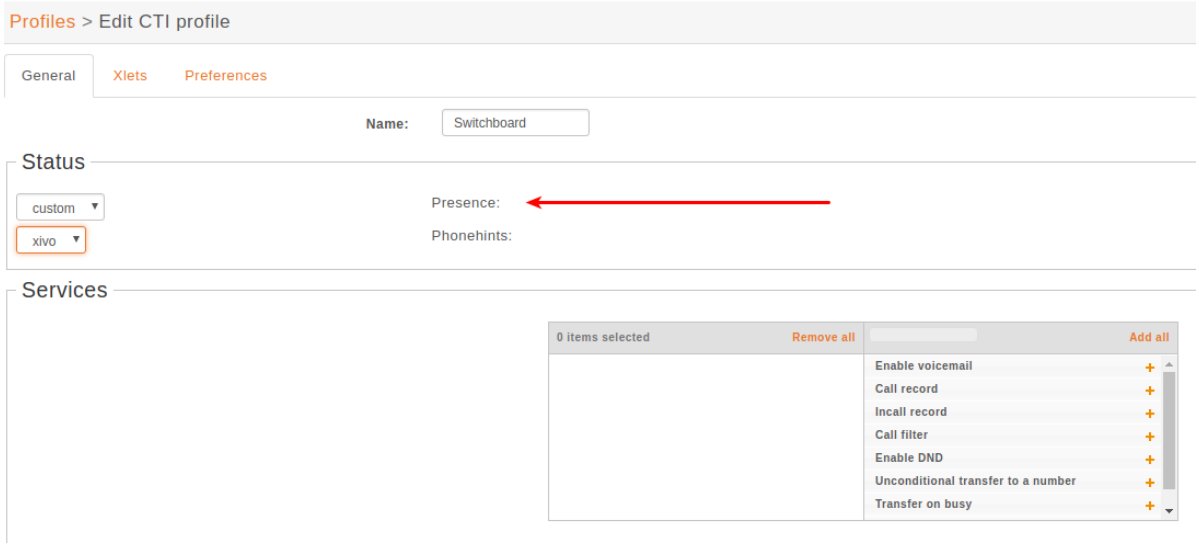
Presence Option

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.

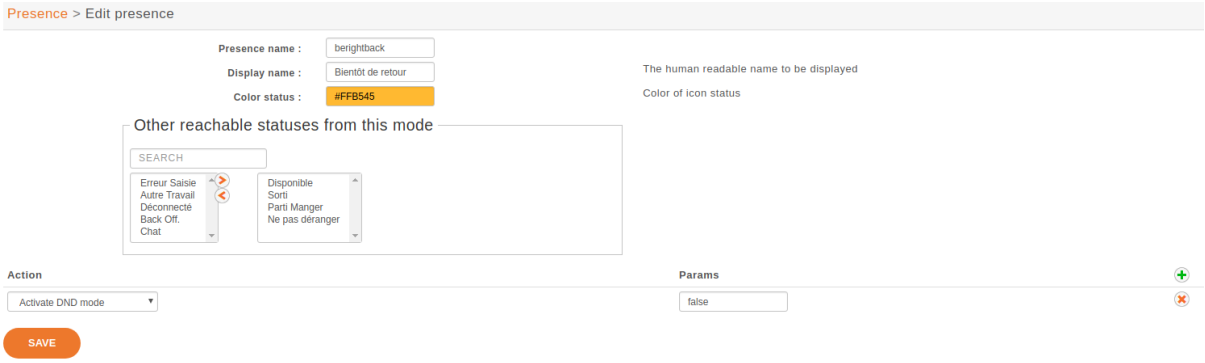


	Presence Name	Description	Action
	berightback	Bientôt de retour	
	erreursaisie	Erreur Saisie	
	postappel	Autre Travail	
	away	Sorti	
	outtolunch	Parti Manger	
	donotdisturb	Ne pas déranger	
	disconnected	Déconnecté	
	backoffice	Back Off.	
	chat	Chat	
	available	Disponible	



Available configuration

- *Presence name* is the name of the presence
- *Display name* is the human readable representation of this presence
- *Color status* is not relevant
- *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	

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.

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*:

Display customer informations

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.

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	

CTI Server

General settings
General
Profiles
Status
Presences
Phone hints

Directories
Definitions
Reverse directories
Direct directories
Display filters

Sheets
Models
Events

	Model	Description	Action
	XiVO	Modèle de fiche de base.	
	basic		
	customer		
	demoweb	Montée d'url	

General settings

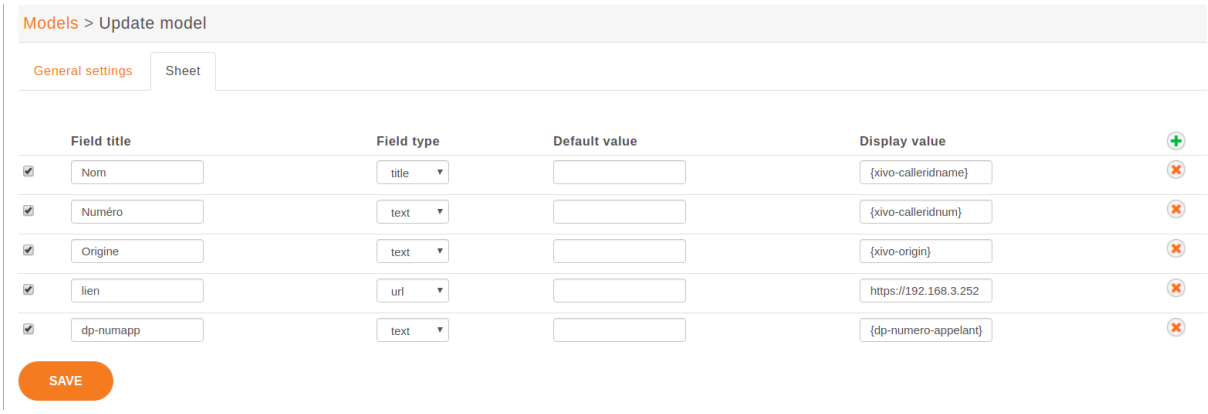
You must give a name to your sheet to be able to select it later.

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` : **not implemented**
  - `phone` : **not implemented**
  - `form` : **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")
```

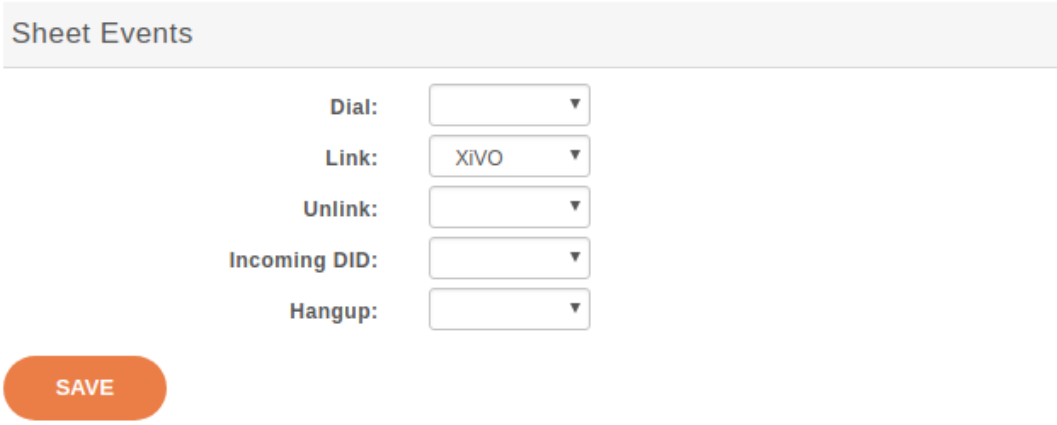
The `{dp-test}` displays Salut.

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 opens a url on new browser tab.

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. It must have a name and at least two fields : one with Field title popupUrl, Default value / and Display value {dp-mysheeturl}, the other with Field title folderNumber and Display value /. See *Sheet Configuration* for more details.

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 `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 CC Agent opened, and your browser should open the specified URL.

Devices

Synchronize a device

First you have to display the list of devices.

Fig. 26: 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. 27: List devices

You will see a pop-up to confirm synchronization

Click on the <ok> button.

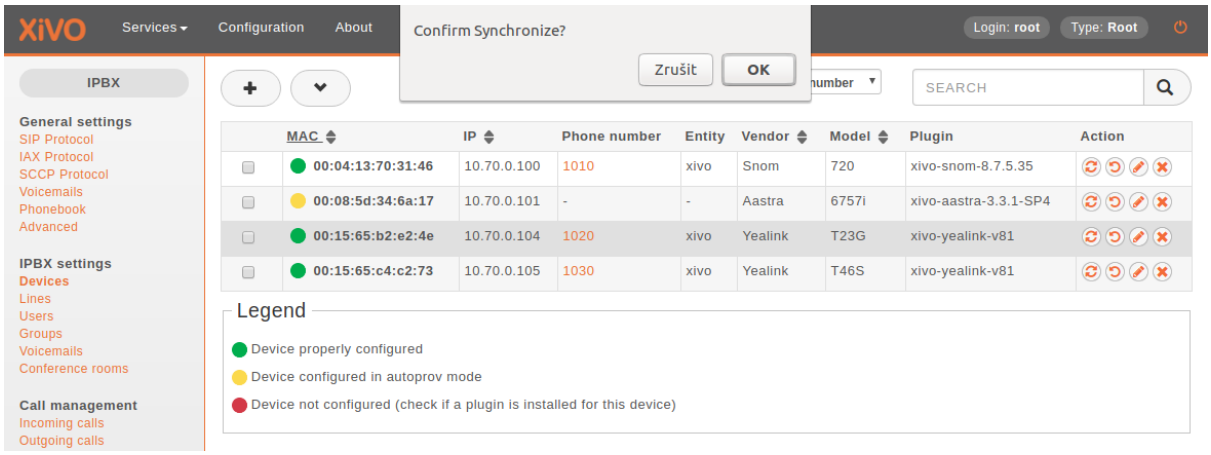


Fig. 28: Alert confirm synchronize

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.

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.

A pop-up will appear requesting confirmation.

If mass synchronization was successfully sent to the devices, a blue information balloon notifies you of success.

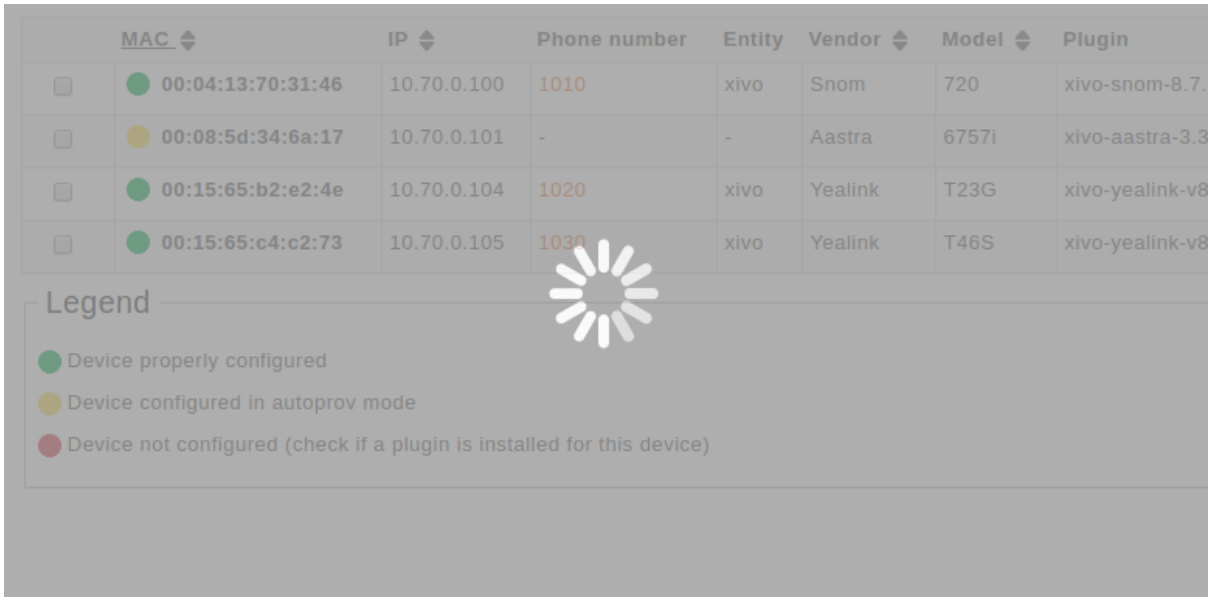


Fig. 29: Request synchronization processing

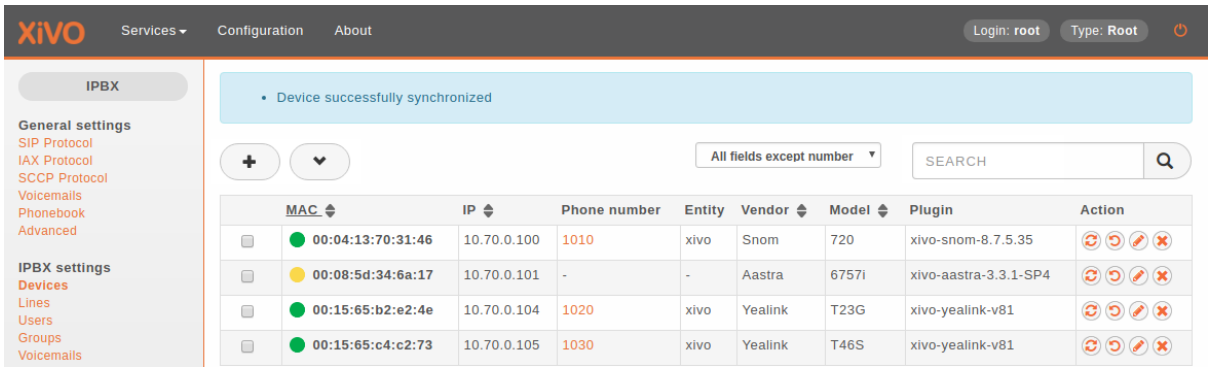


Fig. 30: Device successfully synchronized

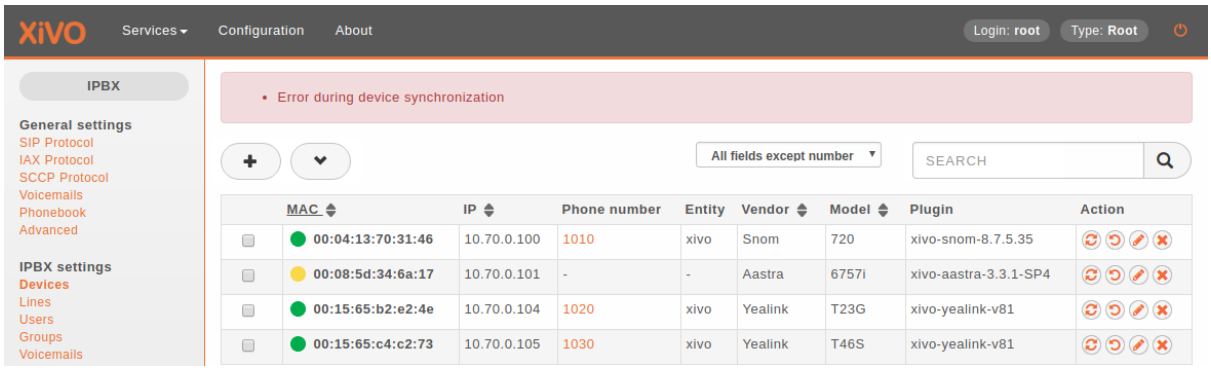


Fig. 31: Error during device synchronization

Fig. 32: Synchronize selected devices

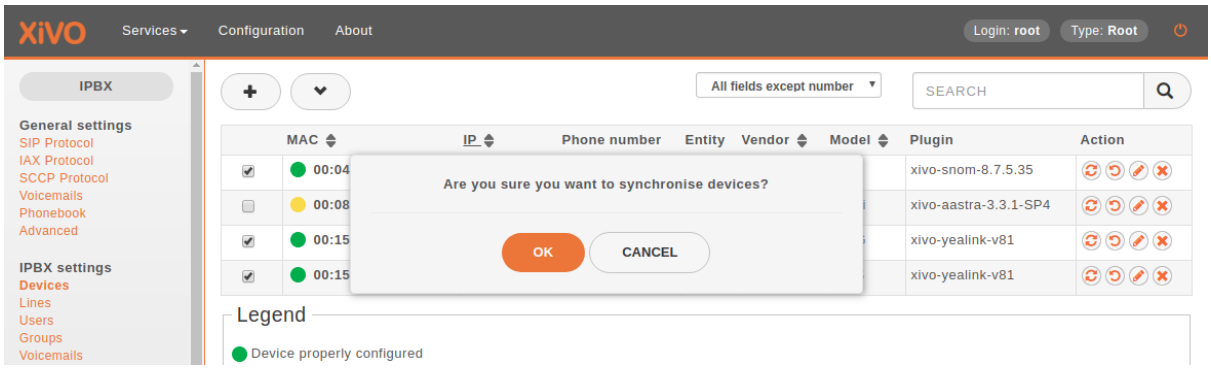


Fig. 33: Synchronize selected devices confirmation

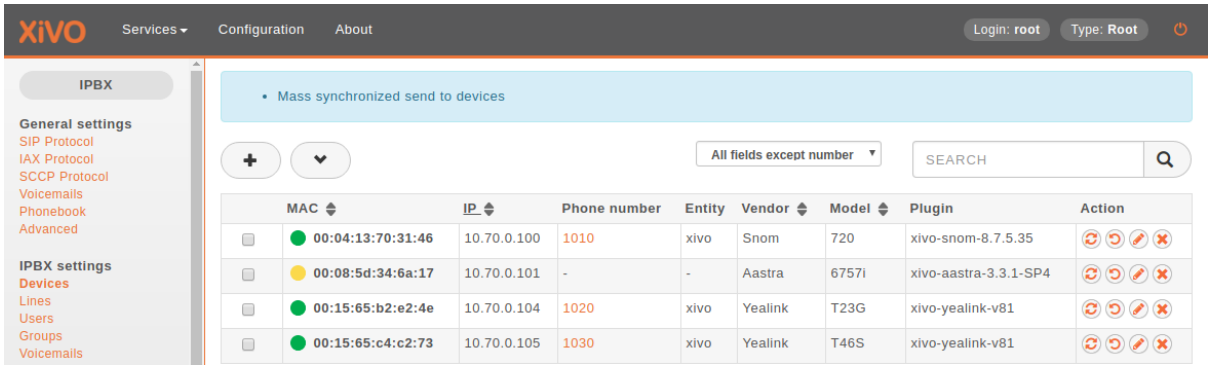


Fig. 34: Mass synchronization request sent successfully

Directories

Whenever you search a user on XiVO using any of its Web Apps or a registered phone, it uses the module Dird to query sources named Directories. If you have a custom source, such as a LDAP, that you would like XiVO to know about and rely on, you must first register it as a Directory. Then you must write the right configuration, so XiVO is able to query it and render the right informations from it.

These are the steps to do so:

- *Add the Directory itself*
- *Add the Directory Definition*
- *Register the Directory to Direct Search and/or Reverse lookup*
- *Restart Dird*

There are 5 different *Directory Types*, each having some specific configurations. You will find some exampes in those pages, but you must follow the procedure here.

Note: Phone IP should be in the authorized subnet to access the directories. See *Remote directory*.

I. Add a Directory

You define where the data source that will become a Directory is located and what is its type.

Default configuration

In *Configuration* → *Management* → *Directories*:

- *Directory name*: the directory name
- *Type*: select among:
  - *XiVO*
  - *CSV File*
  - *CSV Web service*
  - *Phonebook*
- *URI*: the data source location
- *Description*: (optional) a description of the directory

Warning: Editing the *URI* later requires you to save again the *Directory Definition* because the corresponding *URI* on Definition will be broken. After that you shall *restart Dird*

LDAPs (specific)

LDAP Directories are created differently.
Under *LDAP Server* you specify the location of your LDAP, and under *LDAP Filters* you set how you navigate it.
See *LDAP* for adding this Directory, then continue here with the remaining steps starting with *II. Add a Directory Definition*.

II. Add a Directory Definition

Under Definition, you define:

- the Directory Source Name
- the Directory Fields that will be used for the search. Directory Fields are values that shall be available on the source. If you're configuring your own custom source, you know what they are.
- the mapping between the Directory Fields and a the DisplayFilter Fields (see *Display Filter*)

In *Services* → *CTI Server* → *Directories* → *Definitions*:

- *Name*: the Directory Source Name
- *URI*: the data source, one linked to the Directory you defined above will appear
- *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) - see *Reverse lookup*
- *Mapped fields*: the mapping between your Directory Fields and the DisplayFilter Fields
  - *Fieldname*: your Directory Fields (the values you want to read from it)
  - *Value*: the associated DisplayFilter Fields

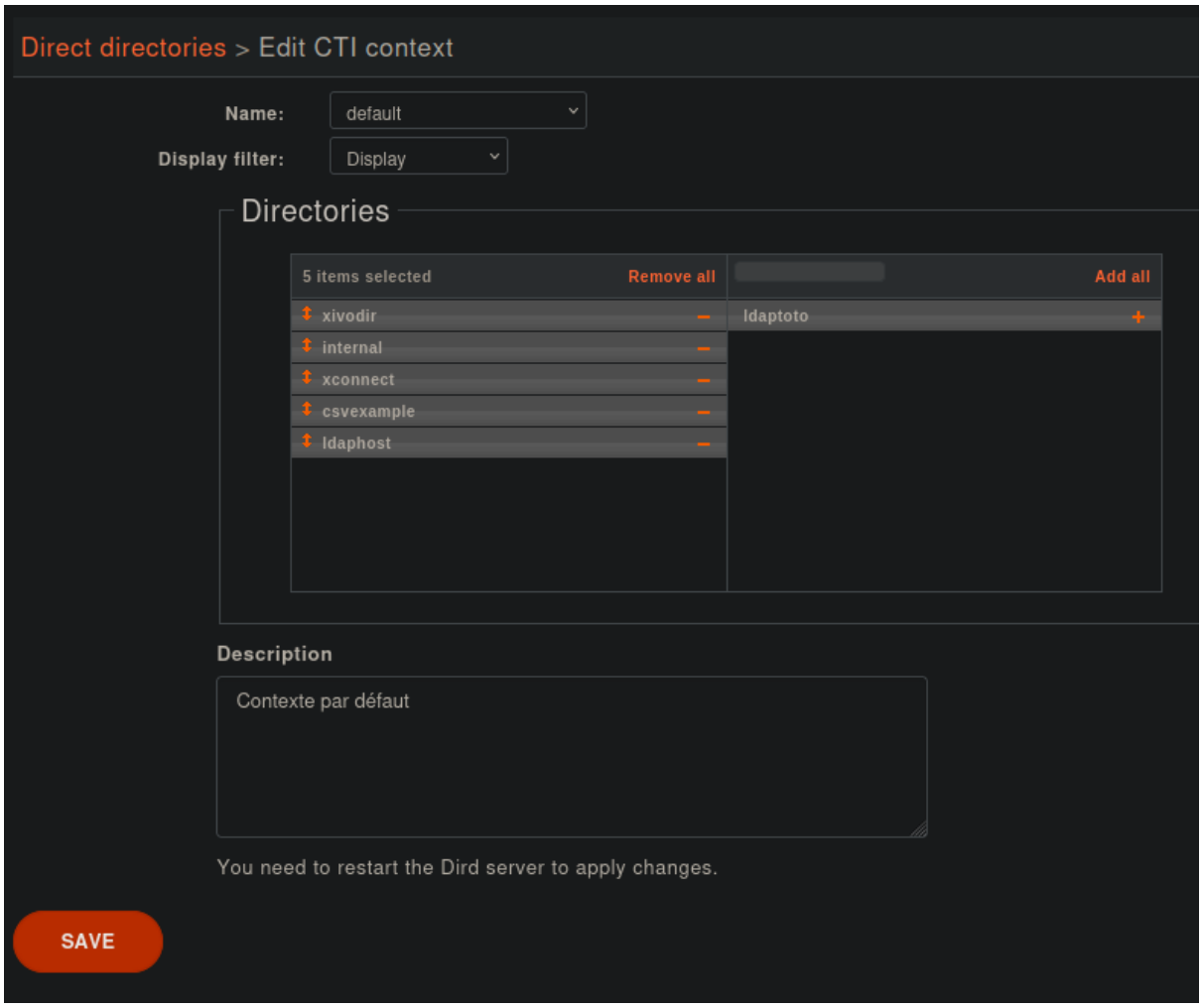
Important: The Directory Source Name cannot contain special characters, use a-z, A-Z, 0-9

III. Register the Directory

Your Directory is defined but not activated. It has to be selected, through its Directory Source Name, for direct search and/or for reverse lookup.

Register Directory for direct search

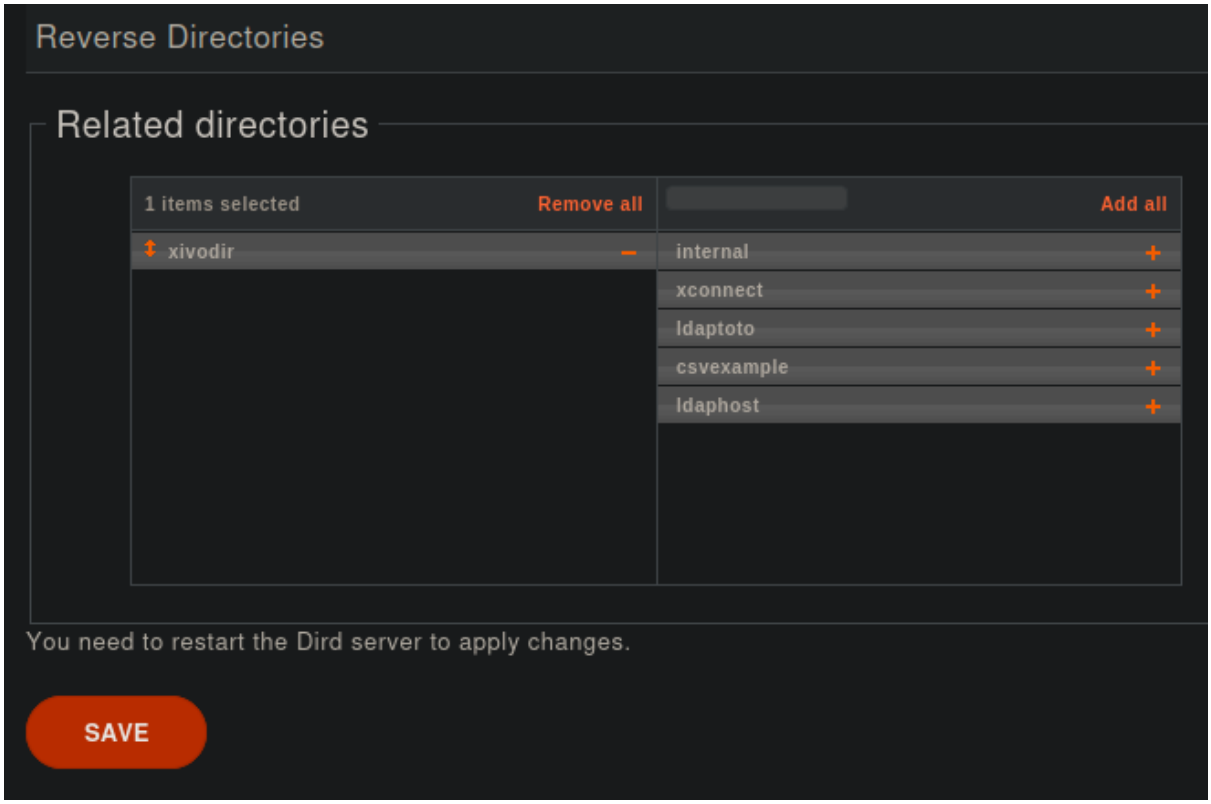
1. In *Services* → *CTI Server* → *Directories* → *Direct directories*.
2. Edit your context (usually default)
3. Select your display filter (usually Display, see *Display Filter*)
4. Add the directories in the *Directories* section.


Direct directories > Edit CTI context
Name:
default
Display filter:
Display
Directories
5 items selected
Remove all
Add all
xivodir
internal
xconnect
csvexample
ldaphost
ldaptoto
Description
Contexte par défaut
You need to restart the Dird server to apply changes.
SAVE

Register Directory for reverse lookup

See *Reverse lookup* to configure the directory for reverse lookup.

1. In *Services* → *CTI Server* → *Directories* → *Reverse directories*.
2. Add the directories to include to reverse lookups in the *Related directories* section.



IV. Restart Dird

Use *one* ofthese methods:

- *Services* → *IPBX* → *Control* → *Restart Dird server*
- console `service xivo-dird restart` on XiVO PBX

You may want to follow closely Dird logs there, so you are sure no Directory fails to reload.

Directory Types

XiVO Directory

This Directory Type is used to query the users of a XiVO.

On a fresh install, there is a default XiVO Directory defined.
Its Directory Name is **xivo** by default.
Its Directory Source Name is **internal** by default.

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

To map/search on in *II. Add a Directory Definition*

- id
- agent_id
- line_id
- firstname
- lastname
- email
- exten
- context
- mobile_phone_number
- userfield
- description
- voicemail_number

Example

Adding Directory

> *Configuration → Management → Directories*

Below the default **xivo** Directory

Adding Directory Definition

> *Services → CTI Server → Directories → Definitions*

Below the Definition for the xivo Directory, with **internal** as Directory Source Name

CSV File Directory

The source file of the directory must be in CSV format.
This directory type matches the *csv* backend in *xivo-dird*.

Directories Servers > Edit

Directory name: xivo

Type: XiVO

URI: http://localhost:9487

XiVO directory

Username:

Password:

Verify certificate: No

Custom CA certificate:

Description

XiVO internal users

SAVE

Definitions > Update directories

Name: internal

URI: http://localhost:9487

Delimiter:

Direct match: firstname,lastname

Match reverse directories: mobile_phone_number

Mapped fields:

Fieldname	Value
contact_1_callable	{exten}
contact_3_callable	{mobile_phone_number}
contact_5_email	{email}
header_1_info	{firstname} {lastname}
reverse	{firstname} {lastname}

Description

Répertoire XiVO Interne

You need to restart the Dird server to apply changes.

SAVE

Available fields

Available fields are the column names from your CSV file.

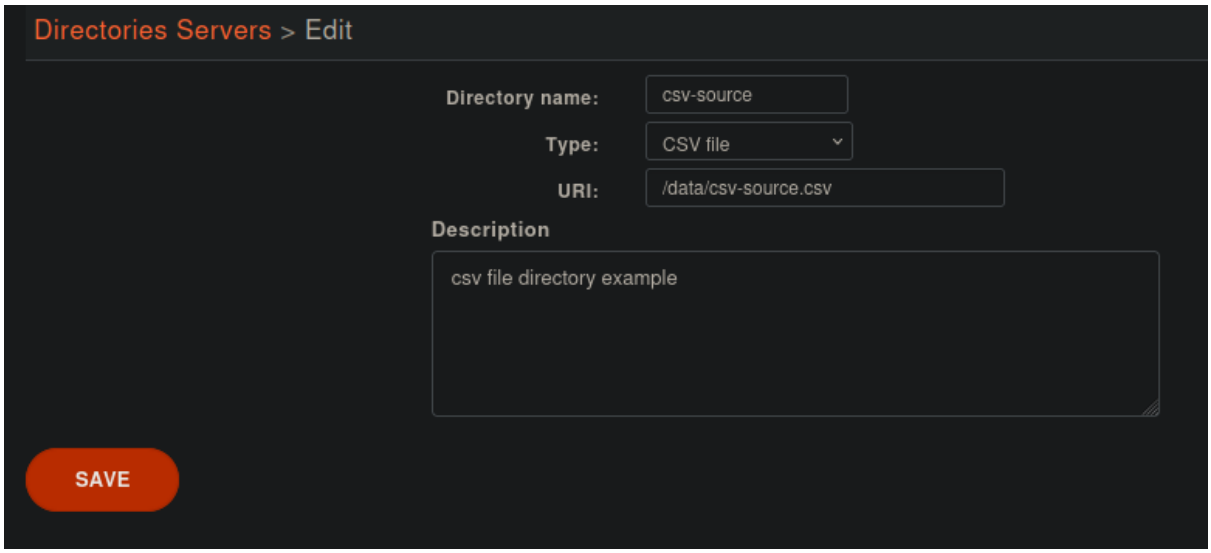
Example

csv-source.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 Directory

Configuration → Management → Directories

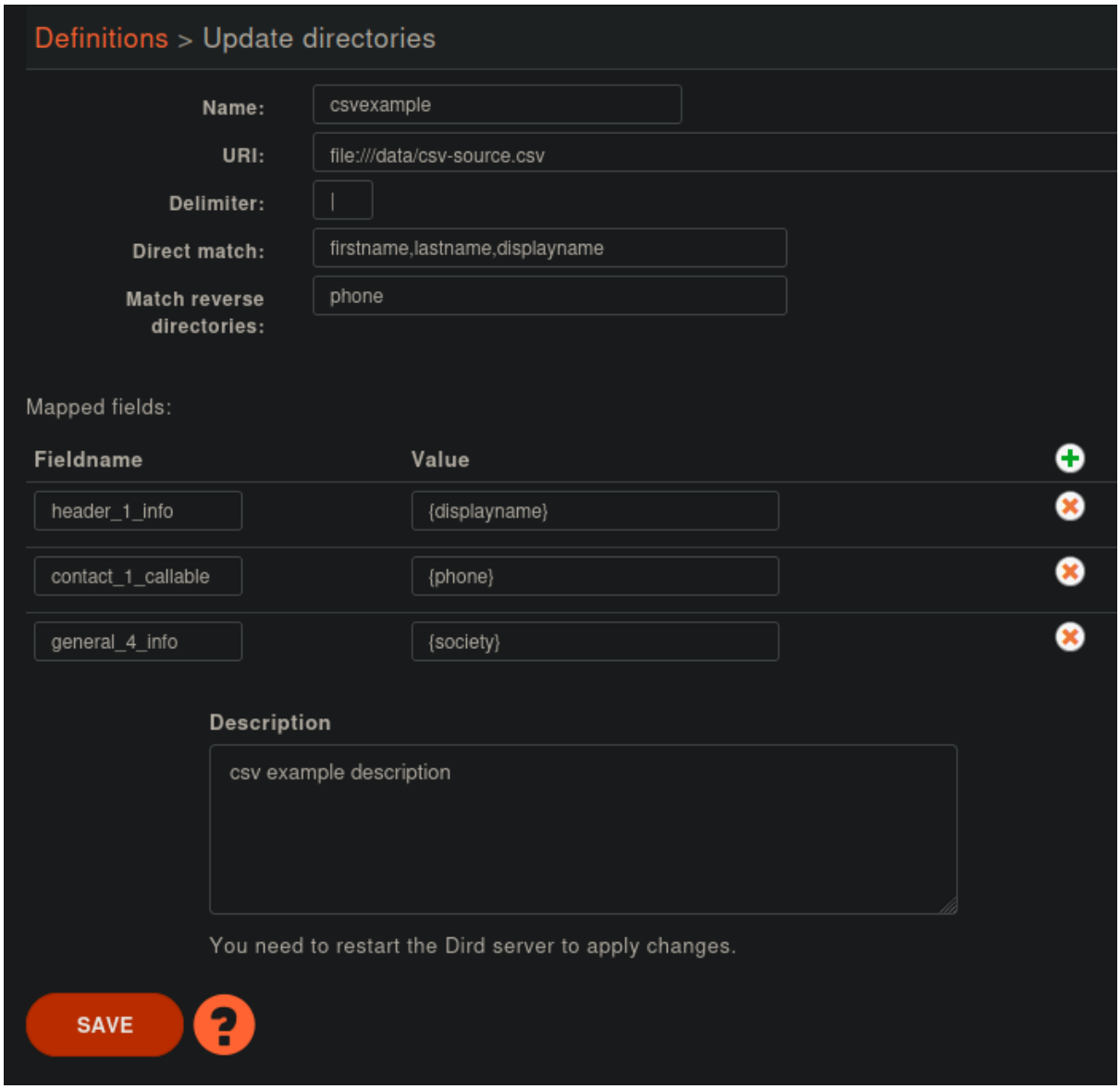


- By default, Dird looks for the file in XiVO PBX.
- It's the absolute file path you must register in URI
- You must explicitely write the extension .csv
- If the Directory fails to appear after step *IV. Restart Dird*, you might find corresponding information in dird logs

```
2024-12-10 22:37:59,646 [817094] (ERROR) (xivo_dird.plugins.csv_plugin): Could not␣
↪load CSV file content
Traceback (most recent call last):
File "/usr/lib/python3/dist-packages/xivo_dird/plugins/csv_plugin.py", line 91, in _
↪load_file
with open(filename, 'r') as f:
      ^^^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: '/data/csv-source'
```

Adding Directory Definition

Services → CTI Server → Directories → Definitions



- The Delimiter must be the separator you use in the CSV (| in the example but it may be another character like ,)

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.

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

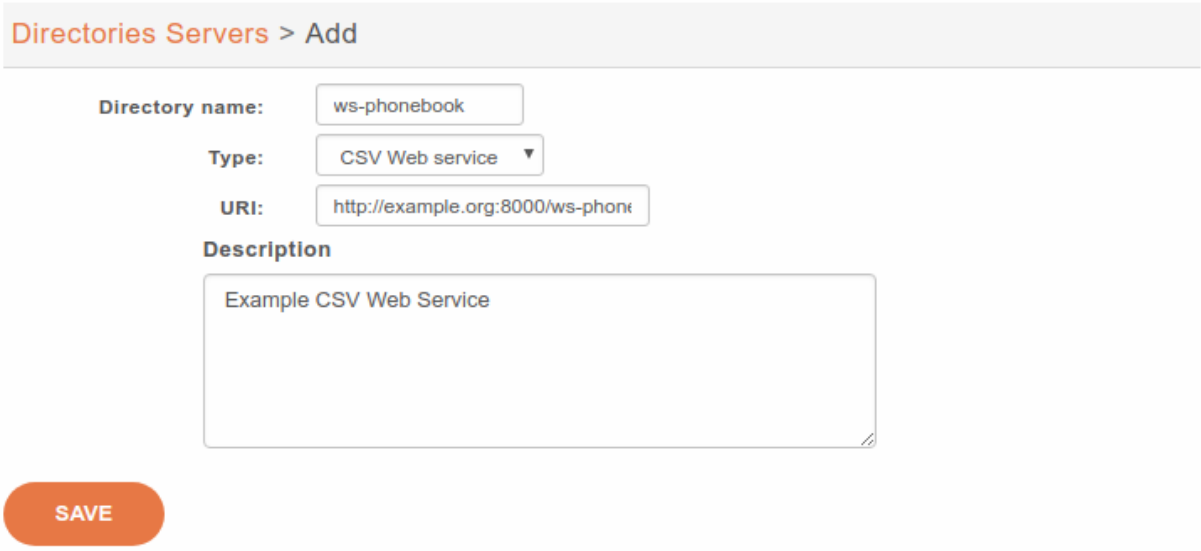


Fig. 35: *Configuration → Management → Directories*

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.

Definitions > Add directory

Name: wsphonebook

URI: http://example.org:8000/ws-phonebook

Delimiter: |

Direct match: search

Match reverse directories: phone

Mapped fields:

Fieldname	Value
directory	CSV web service example
firstname	{firstname}
lastname	{lastname}
display_name	{title} {displayname}
phone	{phone}
email	{email}
society	{society}

Description

You need to restart the Dird server to apply changes.

Fig. 36: *Services → CTI Server → Directories → Definitions*

LDAP

XiVO offers the possibility to integrate LDAP servers.
Once configured properly, you'll be able to search your LDAP servers from your XiVO and from your phones (if they support this feature).

Add a LDAP Server

In *Configuration → Management → LDAP Servers*

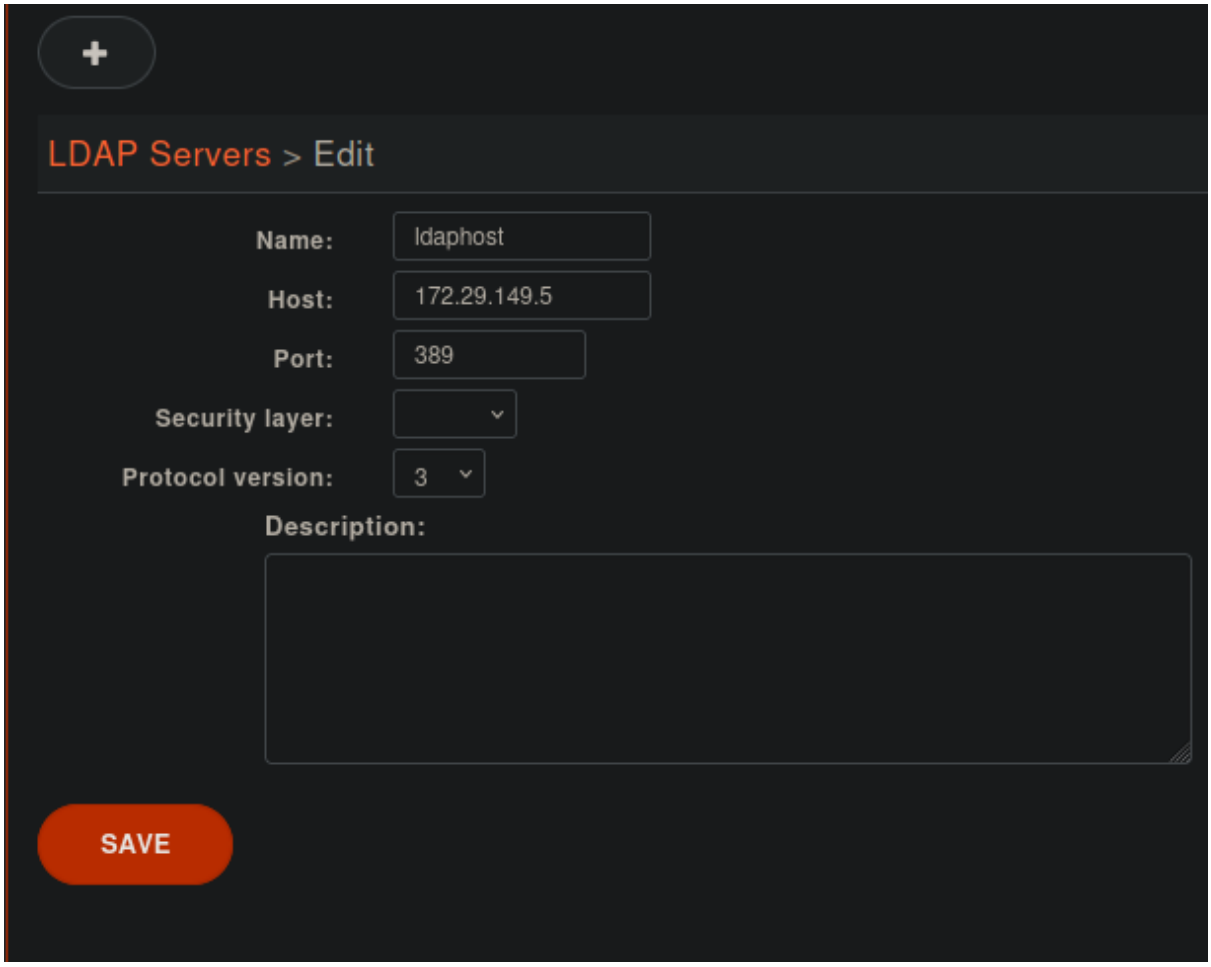


Fig. 37: Adding a LDAP server

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)

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.

In *Services* → *IPBX configuration* → *LDAP Filters* :

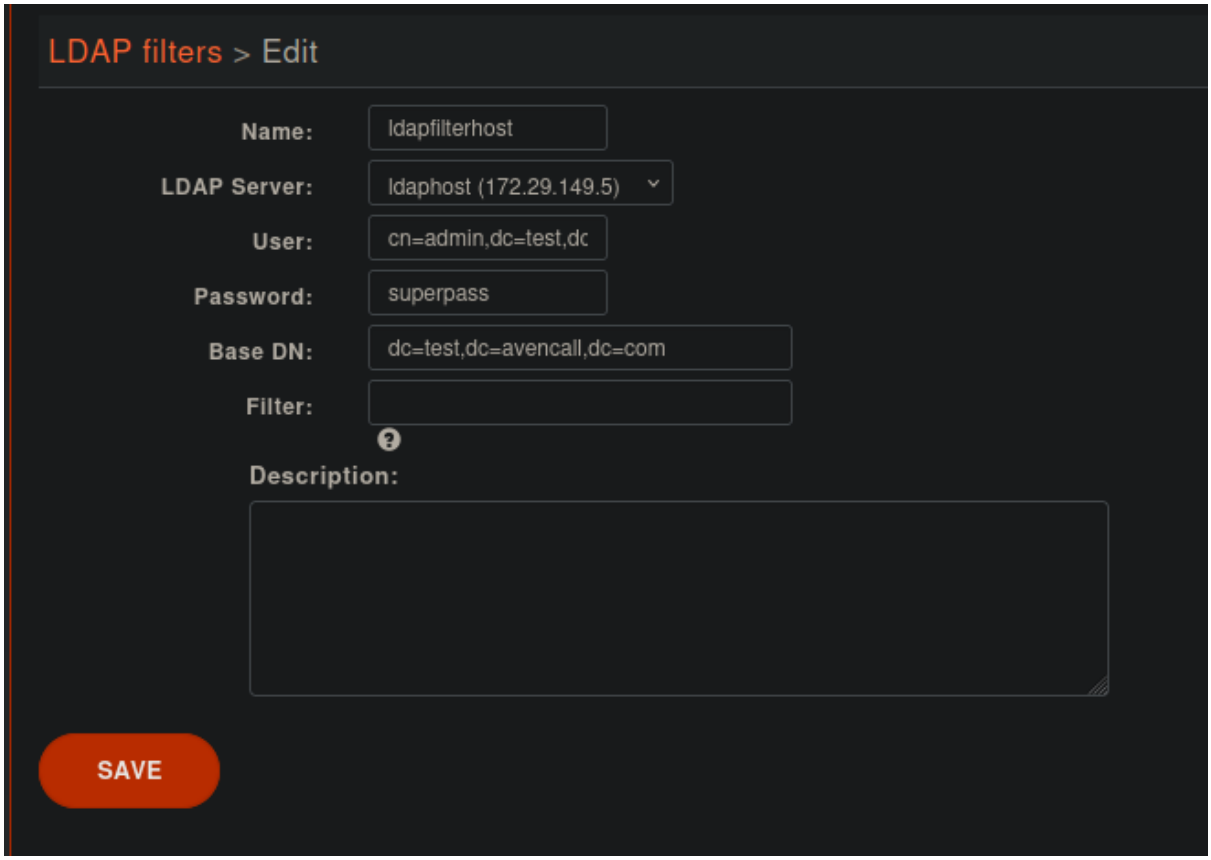


Fig. 38: Adding a LDAP Filter

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*

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, like for other Directories.

In *Services* → *CTI Server* → *Directories* → *Definitions*:

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*))`

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
- phonebook.lastname
- phonebook.society
- phonebook.title
- phonebook.url

Definitions > Update directories

Name:	ldaphost
URI:	ldapfilter://ldapfilterhost
Delimiter:	
Direct match:	givenname,cn,dn,title,sn,mail,ou
Match reverse directories:	homePhone,telephoneNumber,mobile

Mapped fields:

Fieldname	Value
contact_1_callable	{homePhone}
contact_2_callable	{telephoneNumber}
contact_3_callable	{mobile}
contact_5_email	{mail}
general_1_info	{title}
general_2_info	{ou}
header_1_info	{title}
header_2_info	{title}
header_3_info	{ou}
reverse	{cn}

Description

Important: phonebook.fullname is automatically generated as *"<firstname> <lastname>"*, e.g. *"John Doe"* unless neither is found then it fallbacks to phonebook.displayname We recommend to stop using phonebook.fullname and use phonebook.displayname, which is always provided, instead.

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

Configuring source access

Default phonebook is set in *Services → CTI Server → Directories → Definitions → xivodir*.

Important: New in Maia (2024.05), xivodir update to match the Display Filters evolution. You may clear the remaining values from the previous default phonebook or adapt them if you implemented specific usage for them.

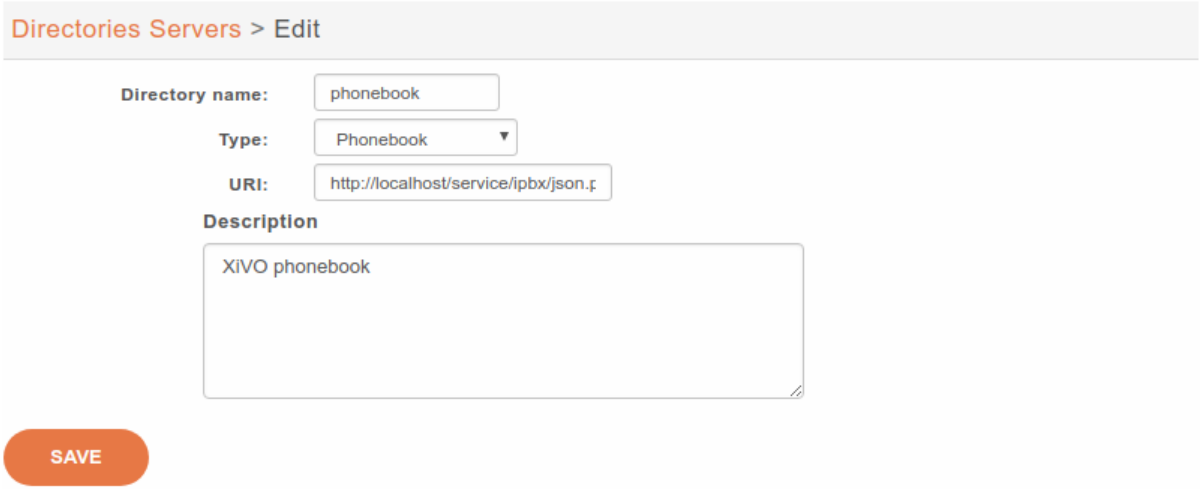


Fig. 39: *Configuration → Management → Directories*
URI : `http://localhost/service/ipbx/json.php/private/pbx_services/phonebook`

Display Filter

Location

In *Services → CTI Server → Directories → Display filters*.
Use the default **Display** or create your own on its model.
The DisplayFilters are chosen by context under *III. Register the Directory*.

Default

Fields types and Fields variables name are fixed in the default Display.
You may update the field title to better match your needs. See *Contact modal* for the visual rendering.

Attributes

Each line in the display filter will result in a header in your Web Apps.

- *Field title*: text displayed in the header.
- *Field type*: type of the column, this information is used by the Web Apps. (see *type description*)
- *Field name*: names of the variables to map in Directory Definitions to Directory Fields reverse_lookup

Reverse lookup

Reverse lookup aims to show a better caller ID name on incoming calls when the caller is in one of our directories, based on the incoming caller ID number.

It 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"

Important: Reverse lookup is performed **after** *Caller Number Normalization* (since XiVO 13.11).

Definitions > Update directories

Name: xivodir

URI: http://localhost/service/ipbx/ison.php/private/pbx_services/phonebook

Delimiter:

Direct match: phonebook.firstname,phonebook.lastname,phonebo

Match reverse directories: phonebooknumber.office.number,phonebooknumbe

Mapped fields:

Fieldname	Value
contact_1_callable	{phonebooknumber.office.number}
contact_2_callable	{phonebooknumber.other.number}
contact_3_callable	{phonebooknumber.mobile.number}
contact_4_callable	{phonebooknumber.home.number}
contact_5_email	{phonebook.email}
general_4_info	{phonebook.society}
header_1_info	{phonebook.displayname}
reverse	{phonebook.fullname}

No fields

Description

Répertoire XiVO Externe

You need to restart the Dird server to apply changes.

SAVE

Filters > Update displays

Name: Display

Field title	Field type	Field name
Headers		
Nom	name	header_1_info
Subtitle1	subtitle1	header_2_info
Subtitle2	subtitle2	header_3_info
Contacts		
Téléphone interne	number	contact_1_callable
Téléphone pro	callable	contact_2_callable
Téléphone mobile	callable	contact_3_callable
Téléphone autre	callable	contact_4_callable
E-mail	email	contact_5_email
Télécopie	callable	contact_6_callable
General		
Fonction	info	general_1_info
Service	info	general_2_info
Manager	info	general_3_info
Société	info	general_4_info
Site web	info	general_5_info
Location		
Bureau	info	location_1_info
Adresse	info	location_2_info
Others		
	favorite	favorite
	personal	
Avatar	picture	picture

Description

Affichage par défaut

You need to restart the Dird server to apply changes.

SAVE

To enable reverse lookup for a directory, you need to add in its definition (*Services* → *CTI Server* → *Directories* → *Definitions*):

- a columns list in *Match reverse directories* where the incoming caller ID number is searched
- an entry in *Mapped fields* with:
  - *Fieldname*: `reverse`
  - *Value*: how you want the caller ID name to be displayed (for instance `{phonebooknumber.society}`)

Warning: When you update this, xivo-dird must be restarted, see *IV. Restart Dird*.

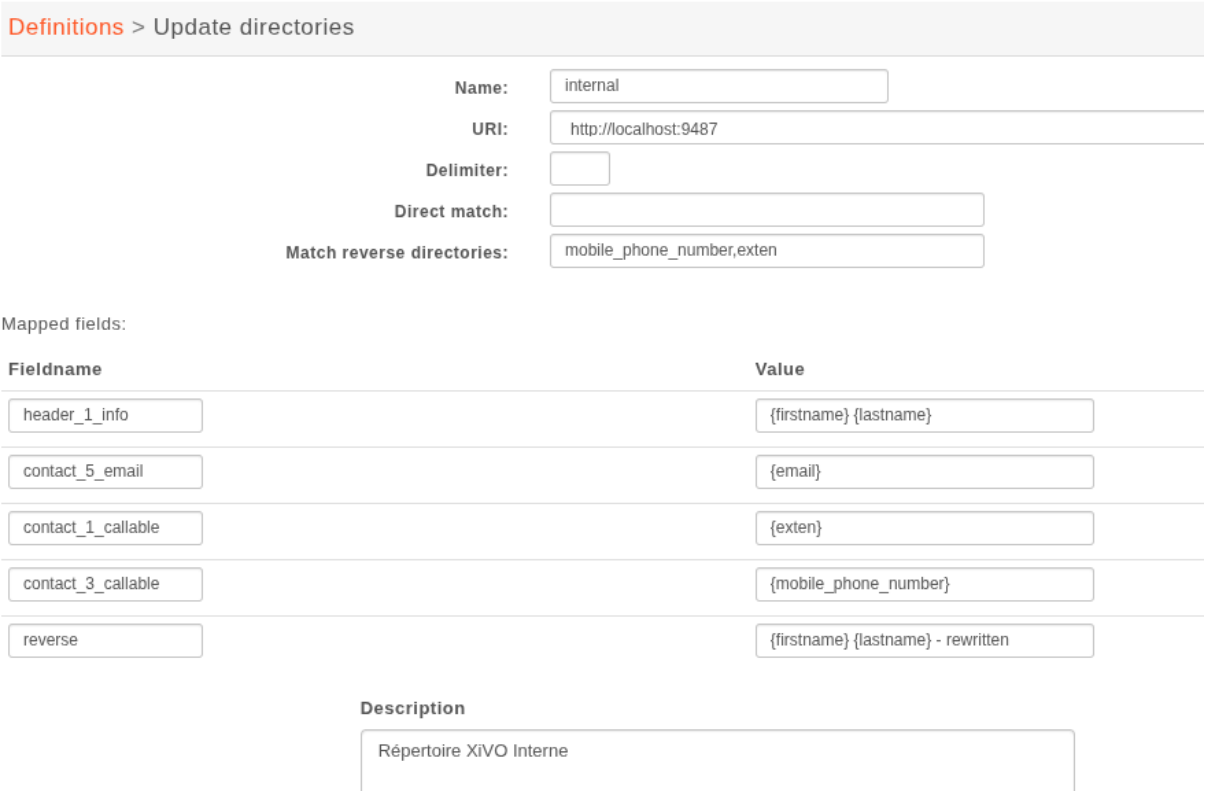
Important: The directory must also be added as Reverse Lookup Directory, see *Register Directory for reverse lookup*.

Example

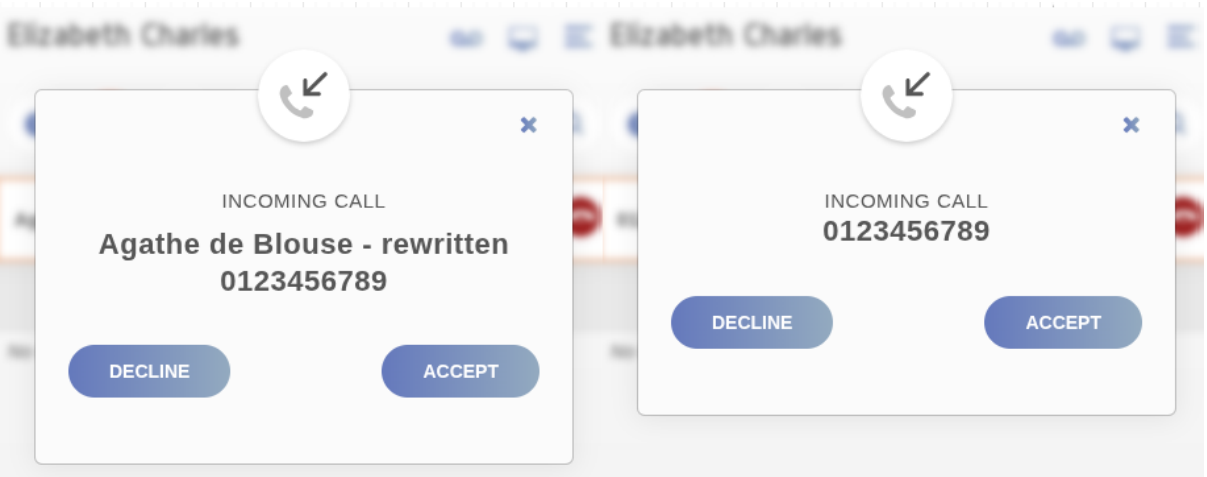
For internal directory:

- *Match reverse directories*: `mobile_phone_number,exten`
- *Fieldname*: `reverse`
- *Value*: `{firstname} {lastname} - rewritten`

This configuration will show the contact's firstname, lastname then - *rewritten* on the caller ID name, when the incoming call callerid number will match its mobile or default number.

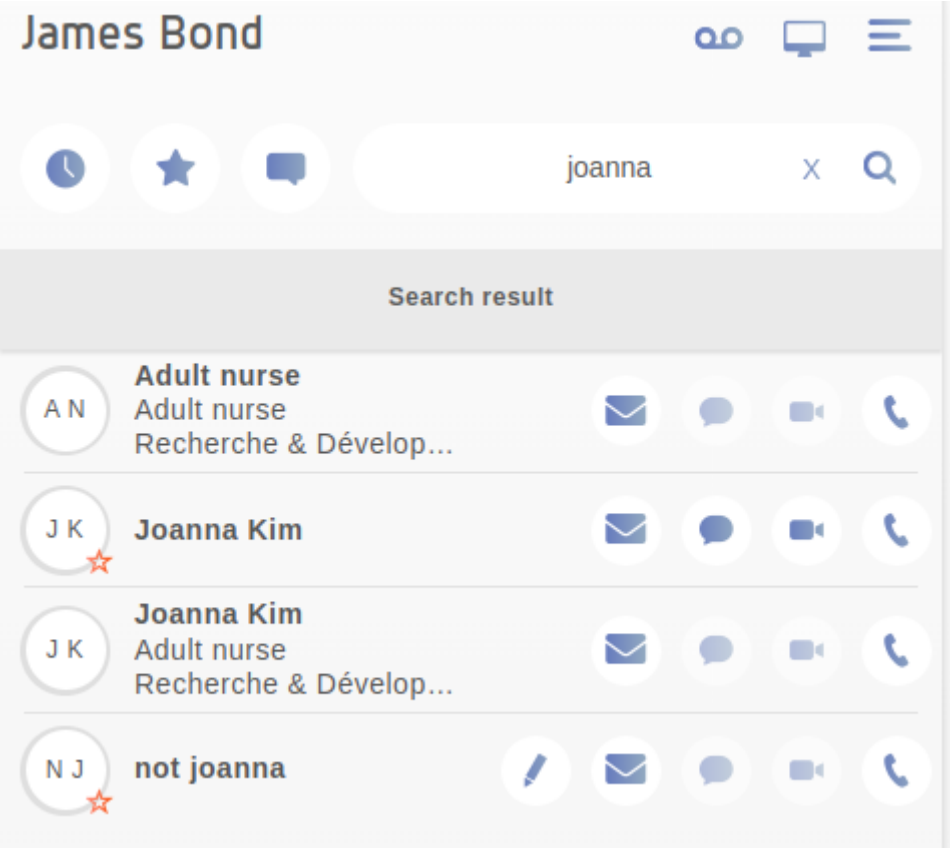


This makes the following difference:

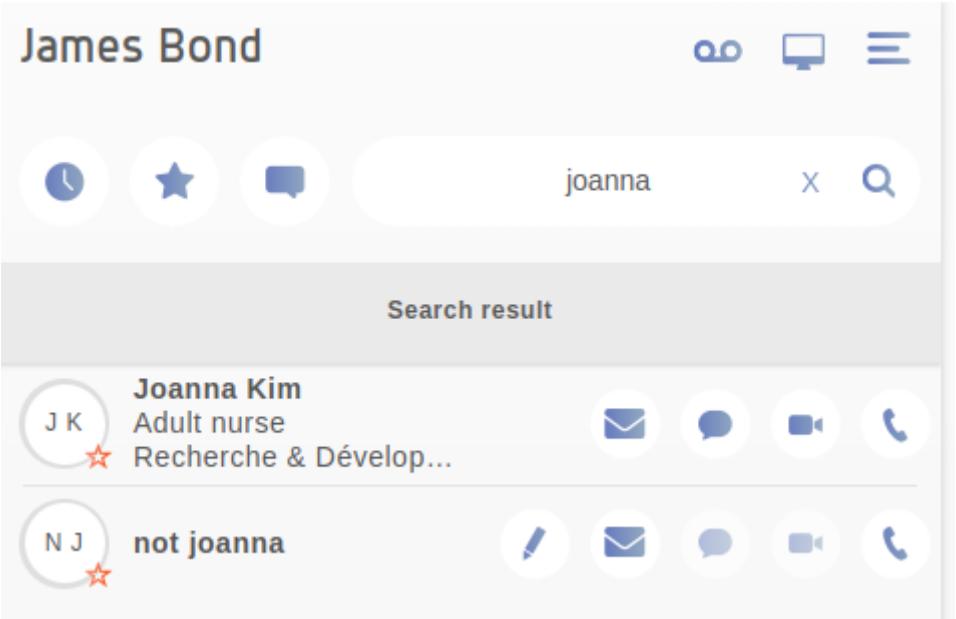


Merge Contacts

Previously, if as a company you had your employees registered in a LDAP but wanting to provide them an internal number on XiVO, you had to create XiVO users for your employees with this number.
Then when searching for them in the XiVO Web Apps (UC Assistant, CC Agent, POPC), they would appear twice in the search.
You would see this:



Starting with Orion, the XUC Server comes with a way to merge Dird results with a **common email**.
So instead, you will see this:



- Given you have several different registered Directories containing users having common emails.
- Given those Directories have for **Directory Source Name**: internal, ldap1, phonebook, CsvFileOne
- Define the Merge Order in xivocc's *custom.env* with `DIRECTORY_SOURCES_PRIORITY` and the sources you want to merge, separated by `,` :

```
DIRECTORY_SOURCES_PRIORITY="ldap1,internal,phonebook"
```

`DIRECTORY_INTERNAL_SOURCE` applies a different order for merging the phone (precisely **contact_1_callable** according to the default Display).
The common use case is that we trust LDAP informations for everything but when it comes to calling we need the internal number, defined on XiVO.

```
DIRECTORY_INTERNAL_SOURCE="internal,phonebook"
```

- `DIRECTORY_INTERNAL_SOURCE` can only contain members of `DIRECTORY_SOURCES_PRIORITY`. Any other member won't be read.
- `DIRECTORY_INTERNAL_SOURCE` merging happen after the `DIRECTORY_SOURCES_PRIORITY` merging, updating the object.

Important: If `DIRECTORY_SOURCES_PRIORITY` is not provided nothing is done. Duplicates are then maintained.

Important: If a Source provides several times a contact with the same email (for instance several people use a service email), then these won't be merged.

Important: The merge contact is case insenstive with mails. The case sensitivity can be restablished by adding the following key to xivocc's *custom.env*: `MERGE_CONTACTS_CASE_SENSITIVITY=true`

Important: You need to restart XUC for this configuration to apply: **xivocc-dcomp up -d xuc**

> **Warning:** The merge appears only on xivo-dird results. If the search term does not lead xivo-dird to retrieve all contacts you want to merge, the merge will be uncomplete. An example of this will be searching contacts by service name, where most likely, only the ldap entries will be found by xivo-dird and not be mergeable with their internal counterpart.

Advanced Dird Configuration

- *Advanced Dird Configration*.

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.

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.

Entities

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.

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

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

Fax

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

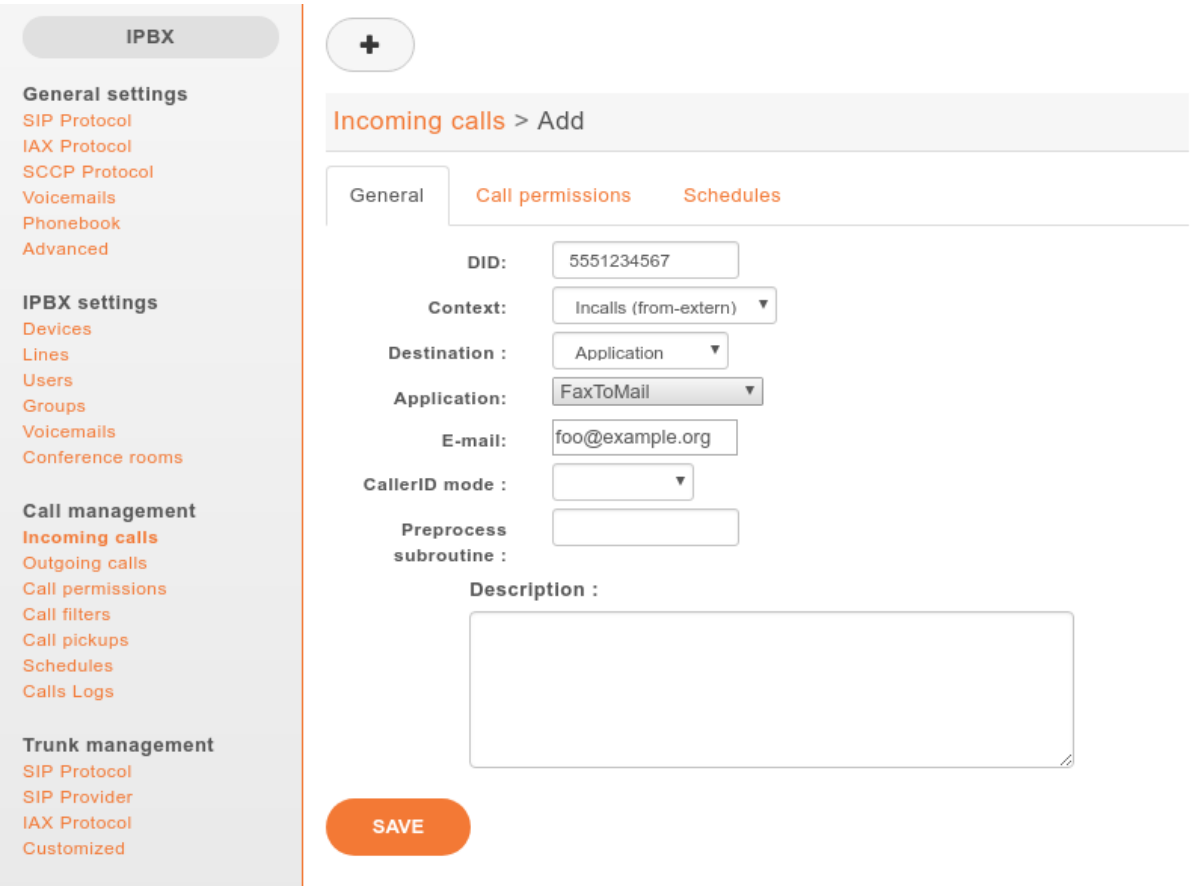
You can change the body of the email sent upon fax reception by editing `/etc/xivo/asterisk/mail.txt`.

The following variable can be included in the mail body:

- `%(dstnum)s`: the DID that received the fax
- `%(srcnum)s`: the sender number (as we received it)

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.

Fax detection

XiVO **does not currently support Fax Detection**. A workaround is described in the *Fax detection* section.

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 }
↪}_>
```

(continues on next page)

(continued from previous page)

```
<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 }
↪}_>
<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.

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.
; 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
```

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.

Groups

Groups are used to be able to call a set or users.

Group name cannot be `general` reserved in asterisk configuration.

When calling a group, if nobody is online in the group, the call will be directed toward the "fail" no-answer scenario configured for this specific group.

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.

Quick Summary

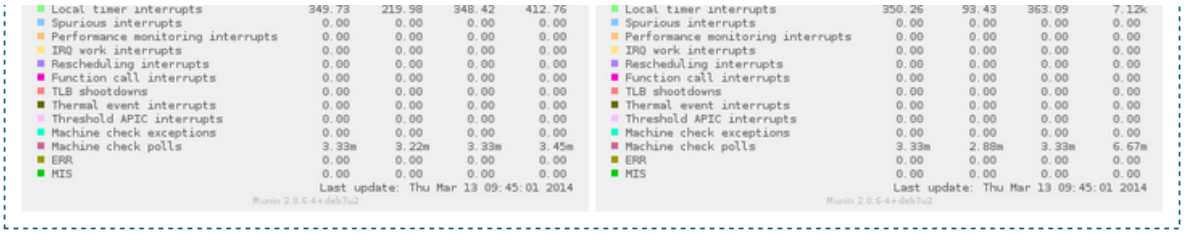
In order to be able to use group pickup you have to:

- Create a pickup group
- Enable an extension to intercept calls
- Add a function key to interceptors

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.



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

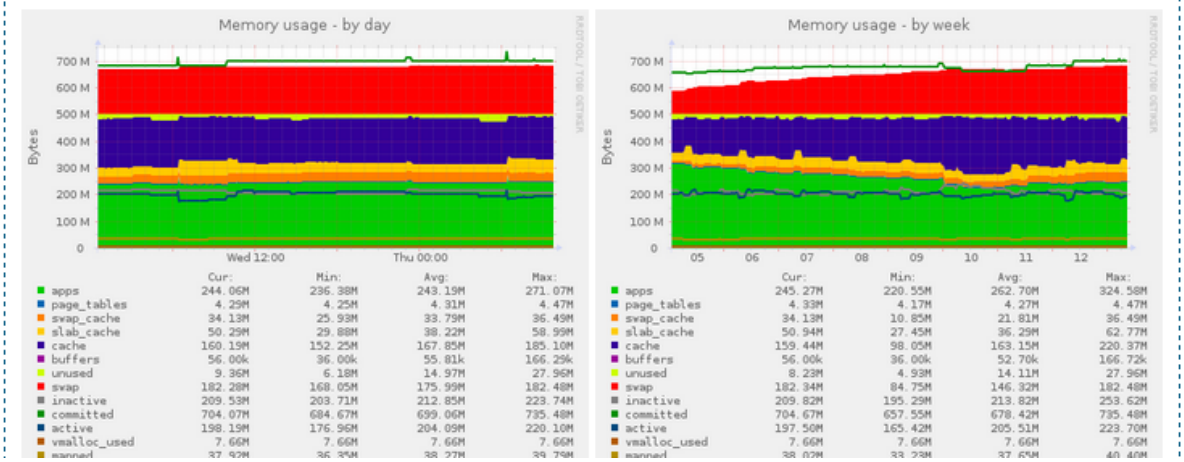
Last update: Thu Mar 13 09:45:01 2014

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

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	50.25M	29.88M	38.22M	58.99M
cache	160.15M	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	198.15M	176.96M	204.09M	220.10M
vmalloc_used	7.66M	7.66M	7.66M	7.66M
mapped	37.92M	36.35M	38.27M	39.75M

	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

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

- Père Noël (2702@default) –
- Linda (2701@default) –
- Fernando L'Igüane (2700@default) –

Add all

- 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

Enabling an Interception Extension

The pickup extension can be defined in the *Services → IPBX → IPBX services → Extensions* page.

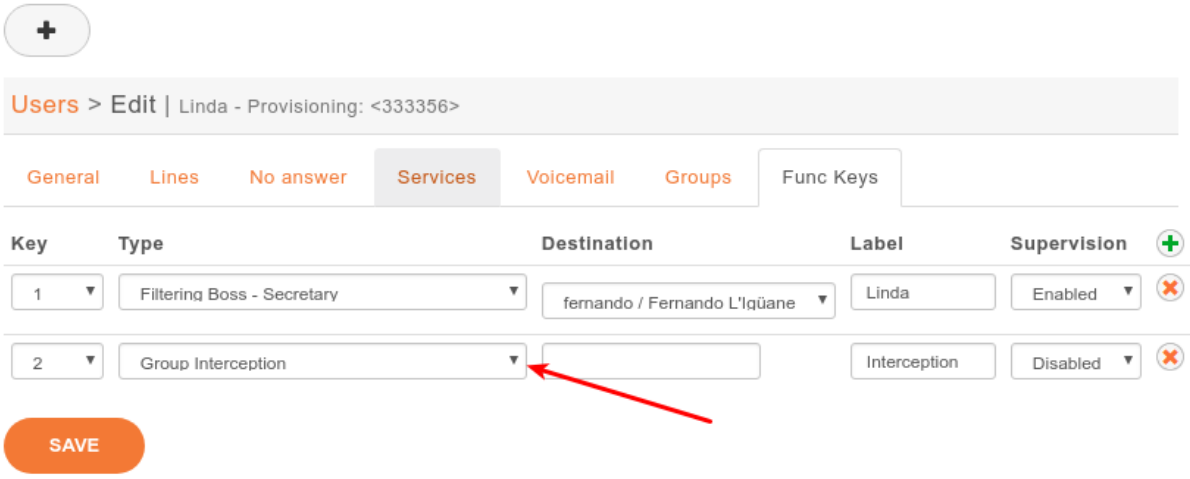
The extension used by group pickup is called *Group interception* it's default value is **8*.

> **Warning:** The extension must be enabled even if a function key is used.

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.



Incall

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
```

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.

Interconnections

Interconnect two XiVO directly

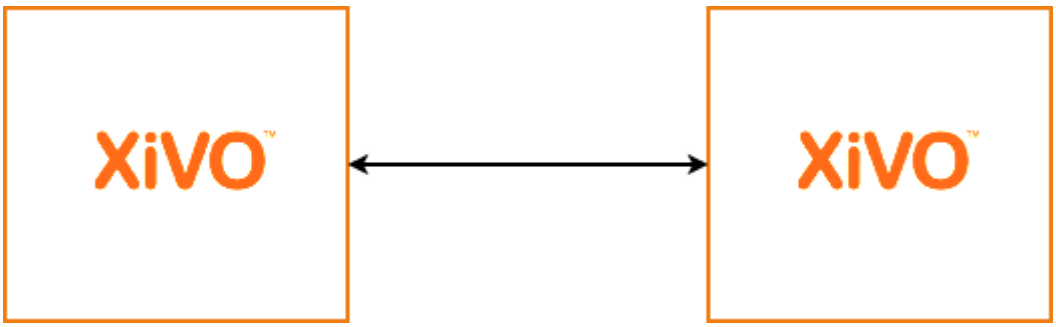


Fig. 40: Situation diagram

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>
Media server: MDS Main
```

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
Media server: MDS Main
```

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 a route.

Tab General:

```
Trunks: xivo-trunk
Extension: **99. (note the period at the end)
Target: \1
RegExp: .{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.

Interconnect XiVO with a known SIP Provider

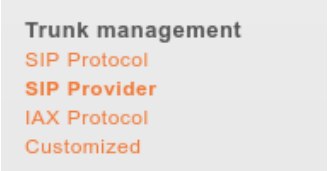
Connection to global telephony network can be configured automatically in this version of XiVO. For instructions how to configure it manually, see *Interconnect XiVO with any VoIP provider*.

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

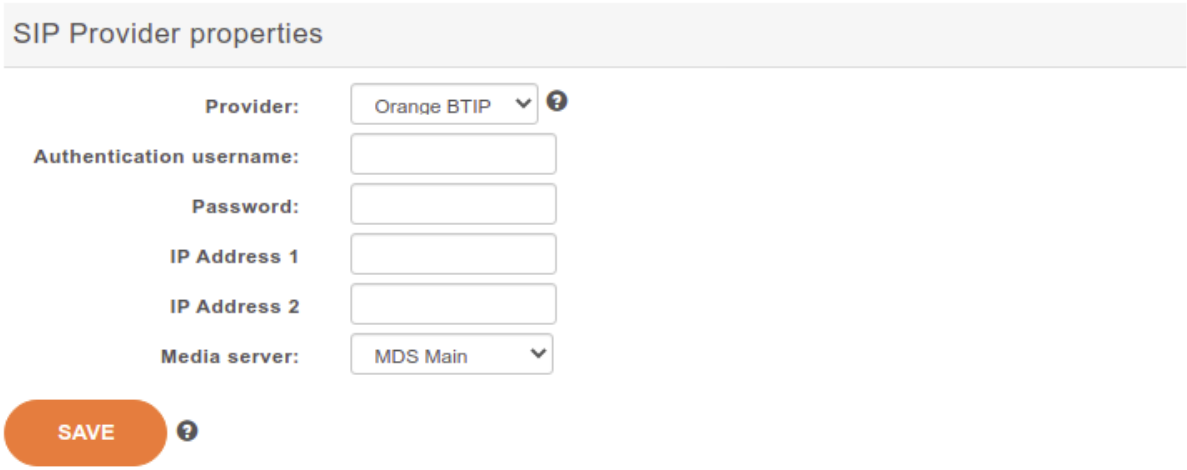
The SIP provider configuration can be applied from the SIP provider page.



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

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 trunk:

```
Name : provider_username
Username: provider_username
Password: provider_password
Connection type: Peer
```

(continues on next page)

(continued from previous page)

```
IP addressing type: voip.provider.example.com
Context: Incalls (or another incoming call context)
Media server: MDS Main
```

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 a route.

Tab General:

```
Trunks: provider_username
Extension: 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.

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*

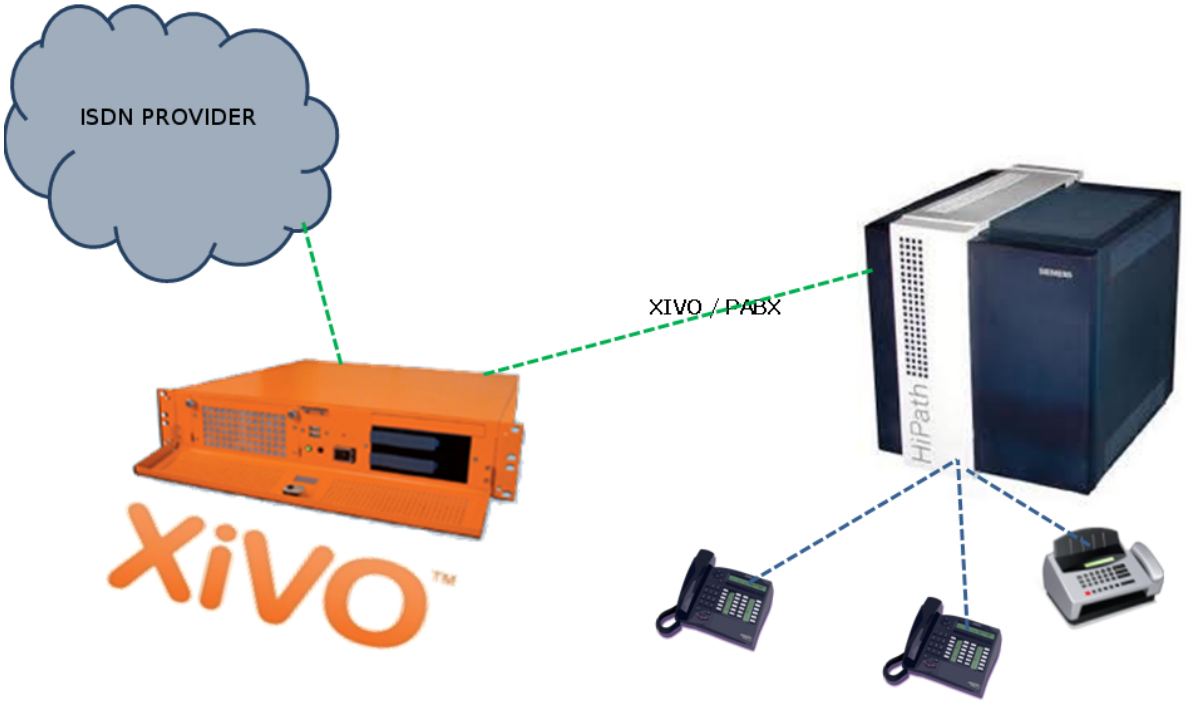


Fig. 41: Interconnect XiVO with a PBX

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)
```

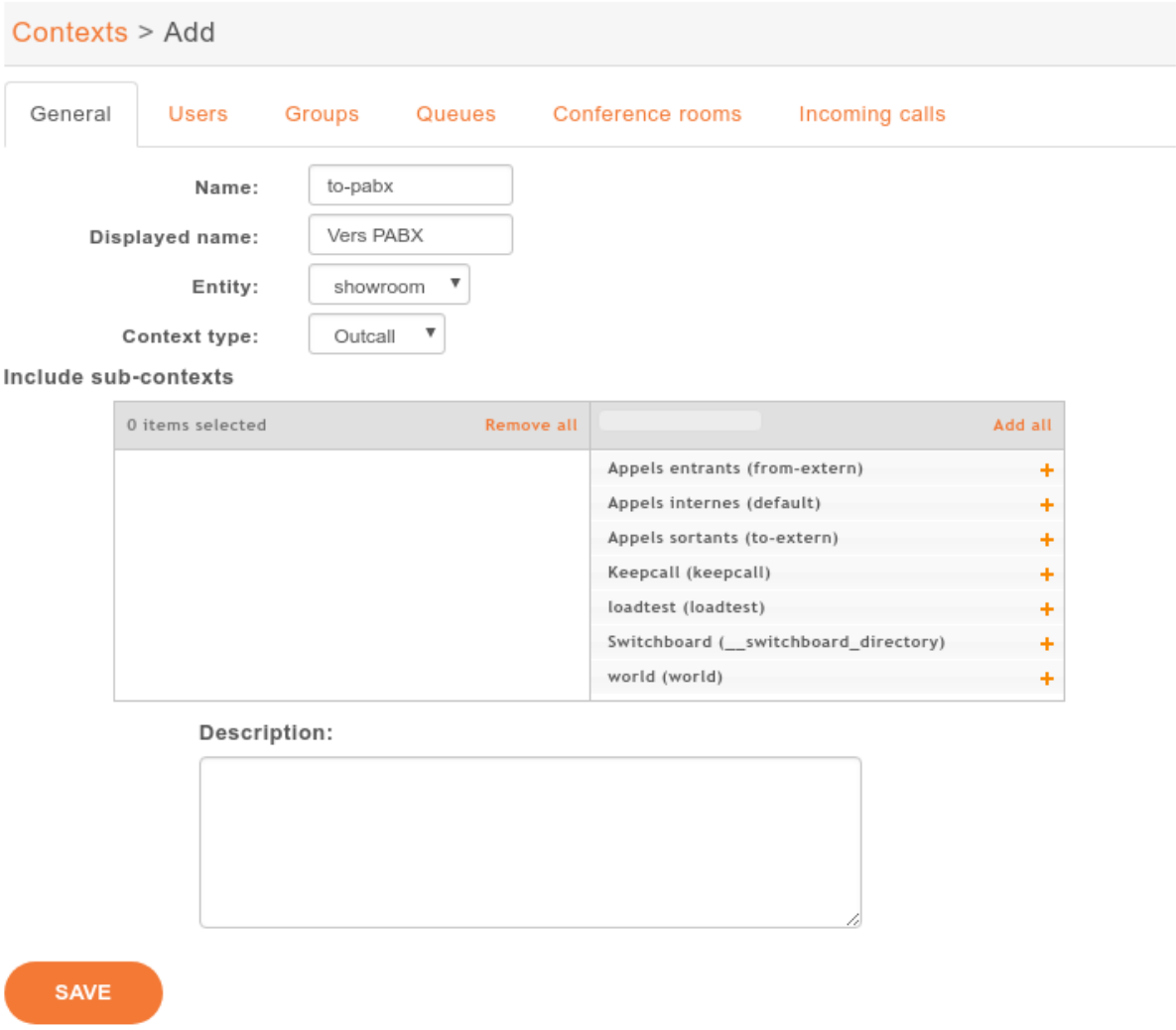
This dialplan routes incoming calls from the PBX in the `default` context of XiVO. It enables call from the PBX : * towards a SIP phone (in `default` context) * towards outgoing destniation (via the `to-extern` context included in `default` context)

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.



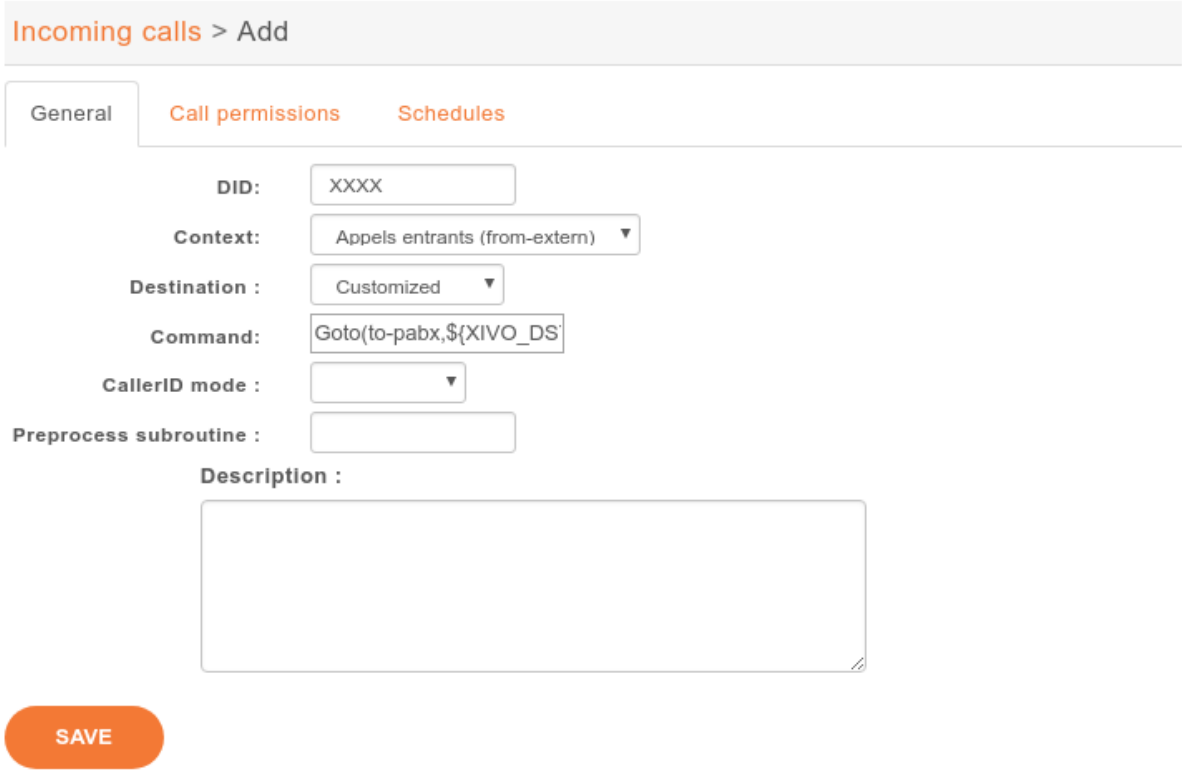
Route incoming calls to PBX

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

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

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
- Extension : XXXX
- Context : to-pabx
- Trunks : choose the *t2-pabx* interconnection

2. Create a rule "fsc-operateur":

- Name : fsc-operateur
- Extension = X.
- Context : to-extern
- Trunks : choose the "t2-operateur" interconnection

Create an interconnection

There are two types of interconnections :

- Customized
- SIP

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**

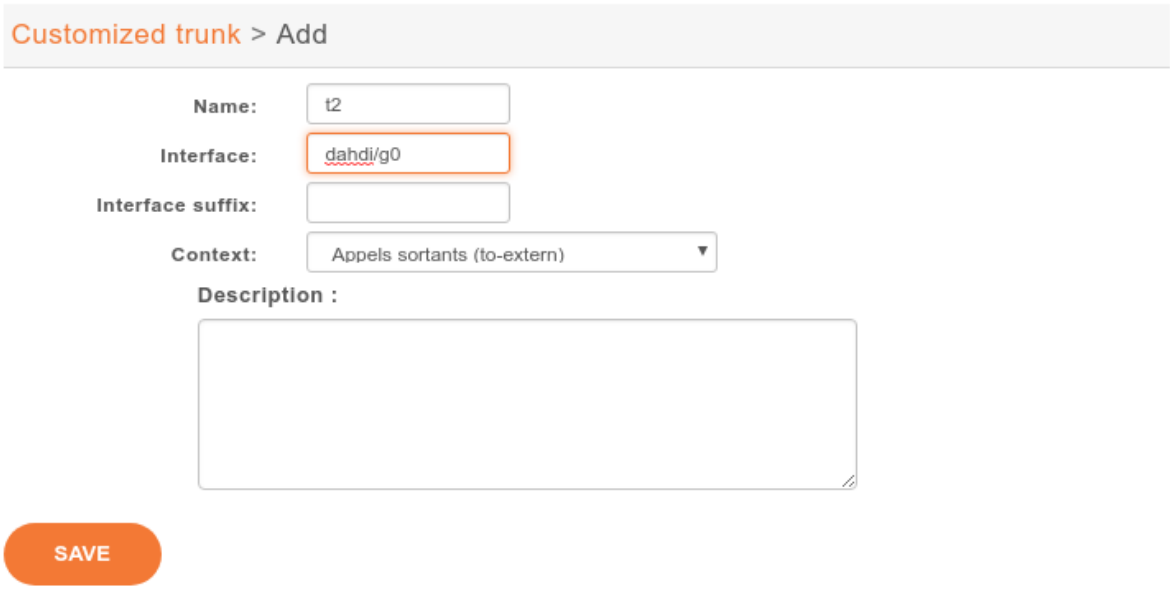
Interface : must be of the form `dahdi/[group order][group number]` where :

- `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
```



Debug

Interesting Asterisk commands:

```
sip show peers
sip show registry
sip set debug on
```

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.

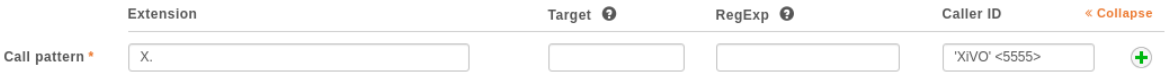
The format for a caller ID is the following `"My Name" <9999>` If you don't set the number part of the caller ID, the dialplan's number will be used instead. This might not be a good option in most cases. If you only need to set a *number* as an outgoing caller ID, you just have to put the number in the caller ID field like `0123456789`.

Outgoing call caller ID

There are several behavior for the outgoing caller ID.

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.

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

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

Interactive Voice Response

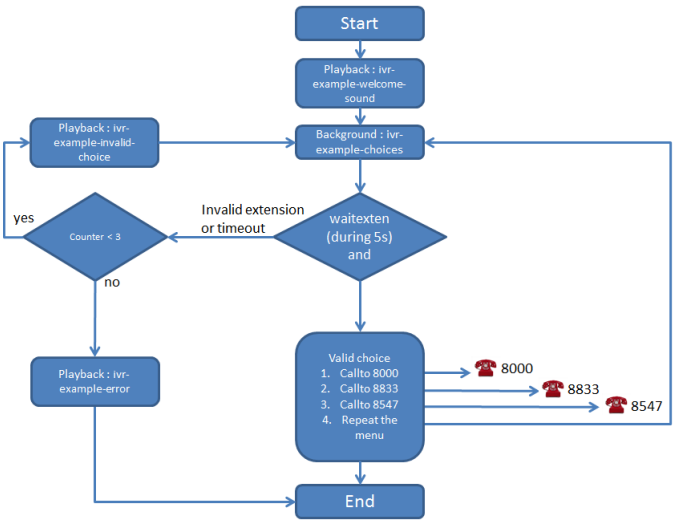
Introduction

> *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 embedded language also named dialplan .

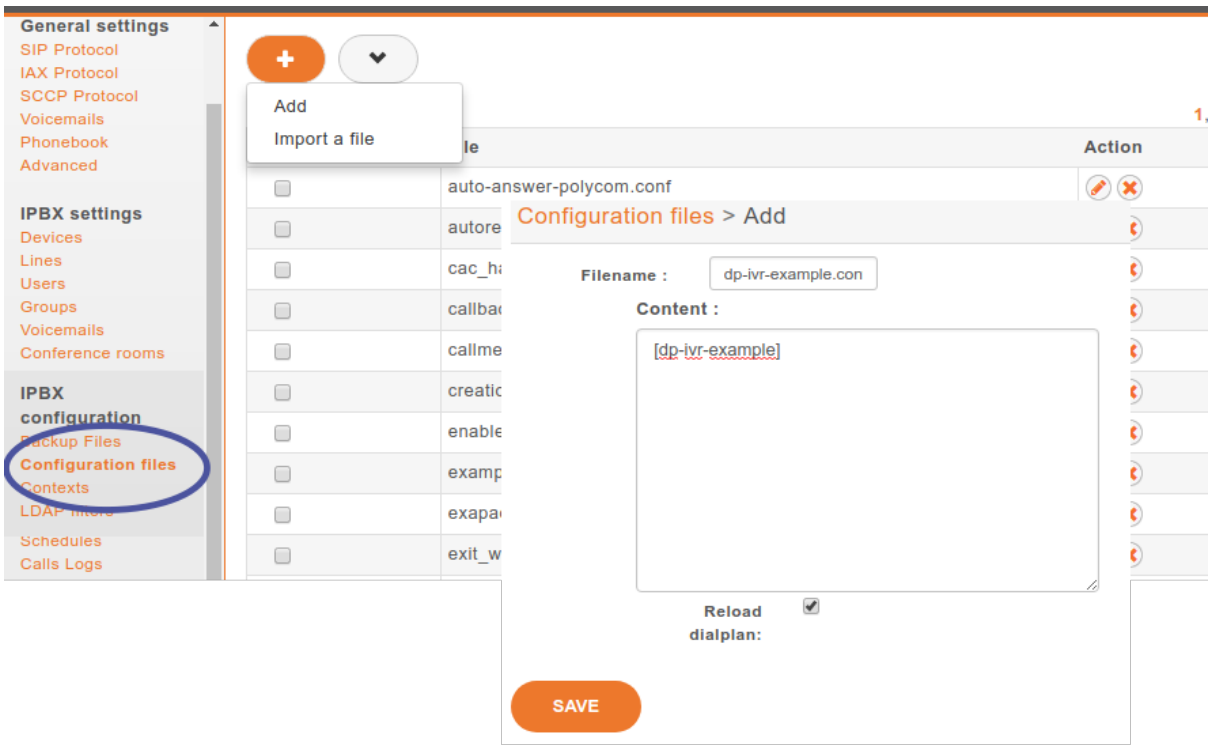
Use Case: Minimal IVR

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.



Important:

- Since WebRTC calls are initiated through an originate for UC/CC applications, you should use *xivo-pickup* subroutine to answer the channel properly in your custom dialplans like IVRs

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 ###)
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 before continuing)
same = n,GoSub(xivo-pickup,s,1)

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})
```

(continues on next page)

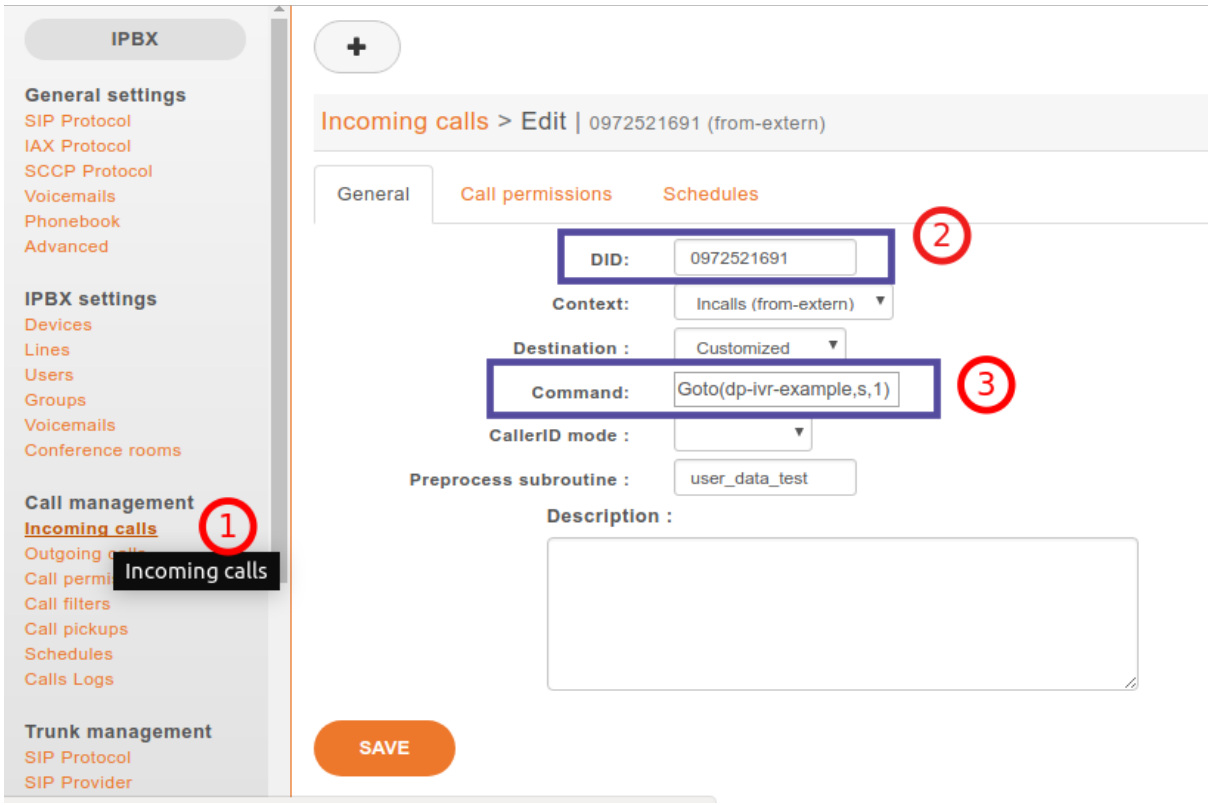
(continued from previous page)

```
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

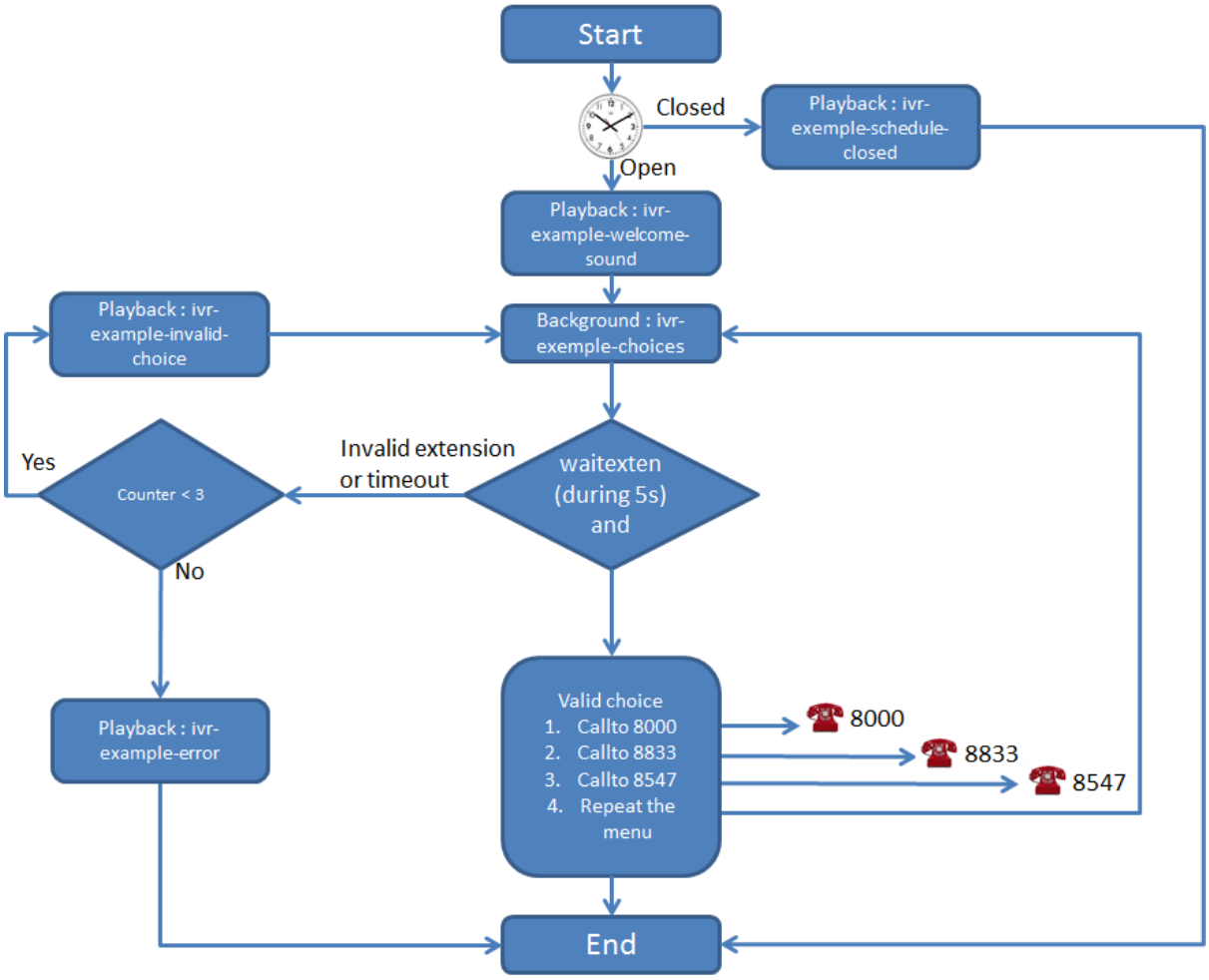
To call the script dp-ivr-example from an internal phone you must create an entry in the default context (`xivo-extrafeatures` is included in `default`). The best way is to add the extension in the file `xivo-extrafeatures.conf`.

```
exten => 8899,1,Goto(dp-ivr-example,s,1)
```

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

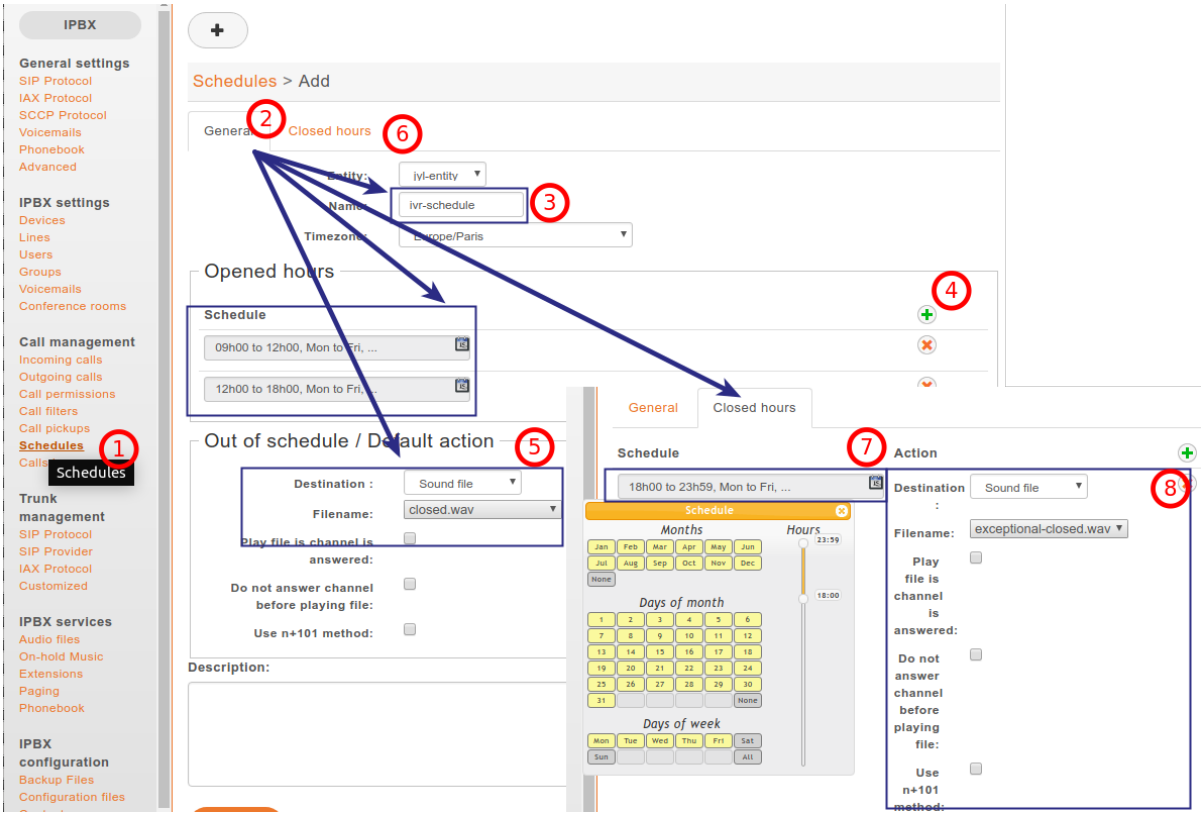


Create Schedule

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

Use Case: IVR with submenu

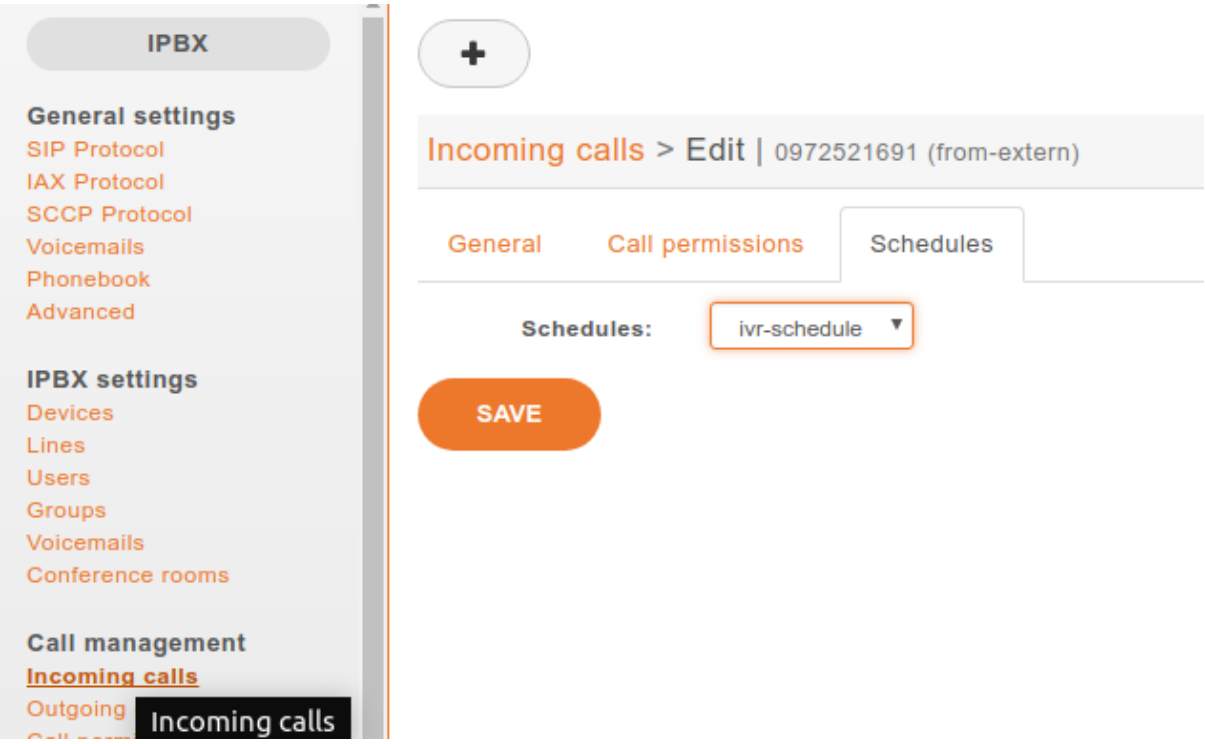
Flowchart

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 before continuing)
same = n,GoSub(xivo-pickup,s,1)

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)
```

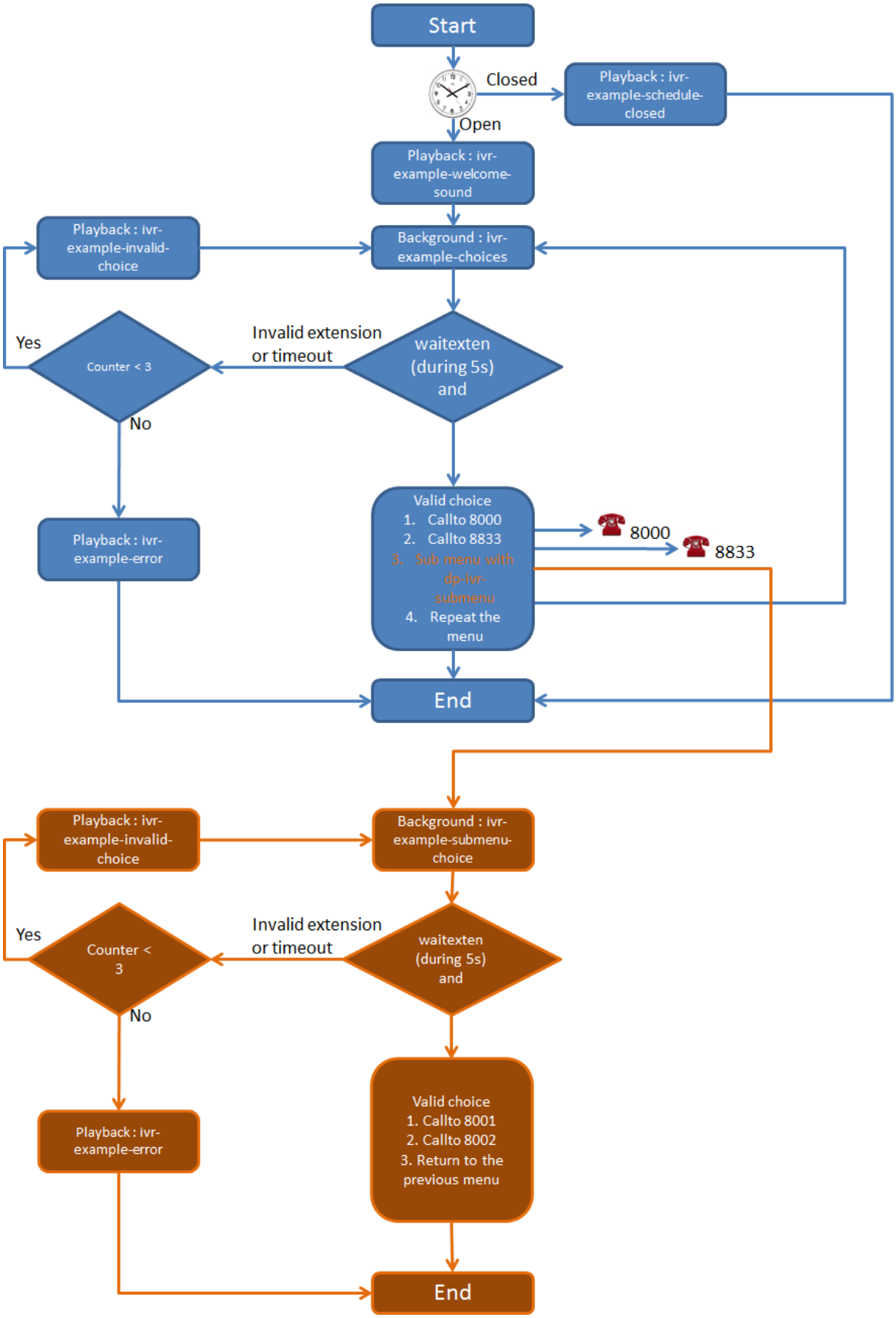
(continues on next page)



(continued from previous page)

```
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)
```

(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)

```
;##### 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()


[dp-ivr-submenu]

exten = s,1,NoOp(### dp-ivr-submenu ###)
same = n,NoOp(the system answers the call before continuing)
same = n,GoSub(xivo-pickup,s,1)

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)
```

(continues on next page)

(continued from previous page)

```
;##### 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()
```

User Labels

You can configure user labels in *Services → IPBX → IPBX Settings → User Labels*. Label allows an administrator to group its XiVO users.

Important: Note for Webi admin user

For admin users of the Webi (see *Webi admin users*) edit its permission and add permission:

- for the *IPBX -> IPBX Settings -> User Labels* menu
- **and** for *Control System –> Authenticate configuration server web services request (required for admin users)*

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*

System

Displays generic information about the operating system, network addresses, uptime and load average. Read only.

Device

Displays free/used space on physical storage partitions. Read only.

XiVO Services Configuration About Login: root Type: Root

System information

CPU				
Percent		User	System	Wait
	6.70 %	5.10 %	1.60 %	0.00 %

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

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

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	

CPU

Monitors the CPU usage. Read only.

Network

Displays network interfaces and corresponding network traffic. Read only.

Memory

Displays Physical and swap memory usage. Read only.

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.

Music on Hold

The menu *Services* → *IPBX* → *IPBX services* → *On-hold Music* leads to the list of available on-hold musics.

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 -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.

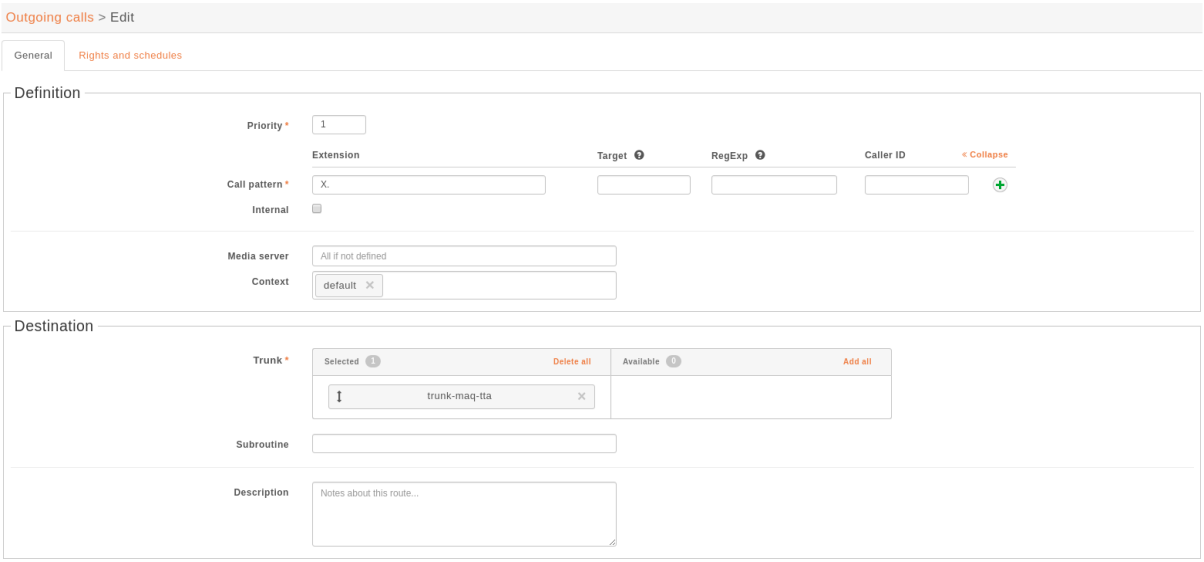
  Example process: `/usr/bin/mpg123 -s --mono -y -f 8192 -r 8000 http://streaming.example.com/stream.mp3`

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.

Outgoing Calls

You can configure outgoing calls settings in *Services → IPBX → Call Management → Outgoing calls*.

An outgoing call is composed with a **definition** of a outgoing route (the extension patterns that should match) and a **destination** (a trunk owned by a media server).



Definition

A route can define

- *Priority*: A number that will prioritize a route compare to another one, if both routes are available for a same matching extension.
- *Call Pattern*:
  - *Extension*: The pattern that will match an extension. More details on the Asterisk wiki.
  - (advanced) *RegExp*: A pattern that will match the dialed number - see *Dialed Number Transformation* below.
  - (advanced) *Target*: The transformation to apply to the dialed number - see *Dialed Number Transformation* below.
  - (advanced) *Caller ID*: Override the presented number once call is performed.
- *Internal*: set the route to forward the internal caller's caller ID to the trunk. This option is useful when the other side of the trunk can reach the user with it's caller ID number.

- *Media server*: define the route only for specific media server.
- *Context*: define the route only for specific context.

Dialed Number Transformation

The fields *RegExp* and *Target* can be used to change the dialed number before it is sent via the trunk defined in the *Route*.

Basically one uses:

- the *RegExp* field to define a pattern that will match the dialed number
- and the *Target* field to define how to transform the dialed number

Below are some examples of how you can take advantage of **RegExp** and **Target**:

Add or Remove Prefix

Regexp	Target	Result
0(.*)	\1	Delete prefix 0 from dialed number and keep the rest (e.g. 00123456789 → 0123456789)
14(.*)..	\1	Delete prefix 14 and last two digits from dialed number and keep the rest (e.g. 1455660 → 556)
.{1}(.*)	\1	Delete leading digit from dialed number and keep the rest (e.g. 03601 → 3601)
+33(.*)	0\1	Replcae leading +33 with 0 from dialed number and keep the rest (e.g. +33123456789 → 0123456789)
0(.*)	+33\1	Replace leading 0 with +33 from dialed number with 33 (e.g. 0123456789 → +33123456789)
14(.*)..	33\18	Delete prefix 14 and last two digits, and then prefix the dialed number with 33 and suffix it with 8 (1455660 → 335568)

Change the Number

Regexp	Target	Result
(.*)	\1	Get whole called number (e.g. 0123456789 → 0123456789)
(.*)	445	Replace called number with 445 number (e.g. 0123456789 → 445)

Destination

Once route is defined and therefore can be selected if an extension match the route, you need to set the wanted destination. A route destination is materialized by:

- **Trunk**: SIP or Custom trunk that will be used to reach desired network.
- **Subroutine**: Subroutine to apply once route has been selected.

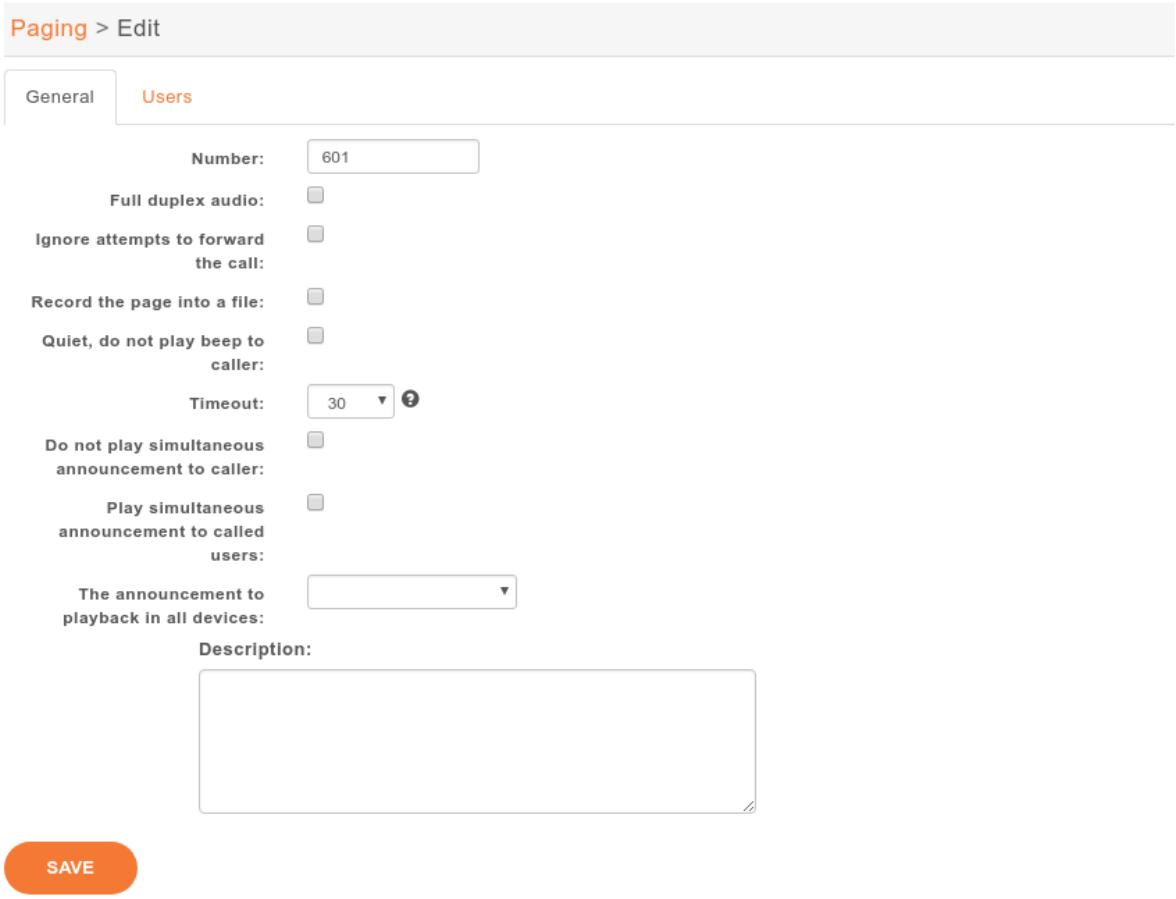
Rights and schedules

A route can define **rights** aka call permissions, it means that a call can be discarded if missing the right to use this route. The same applies for **schedules** where you can define time slots of availability of the route.

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.



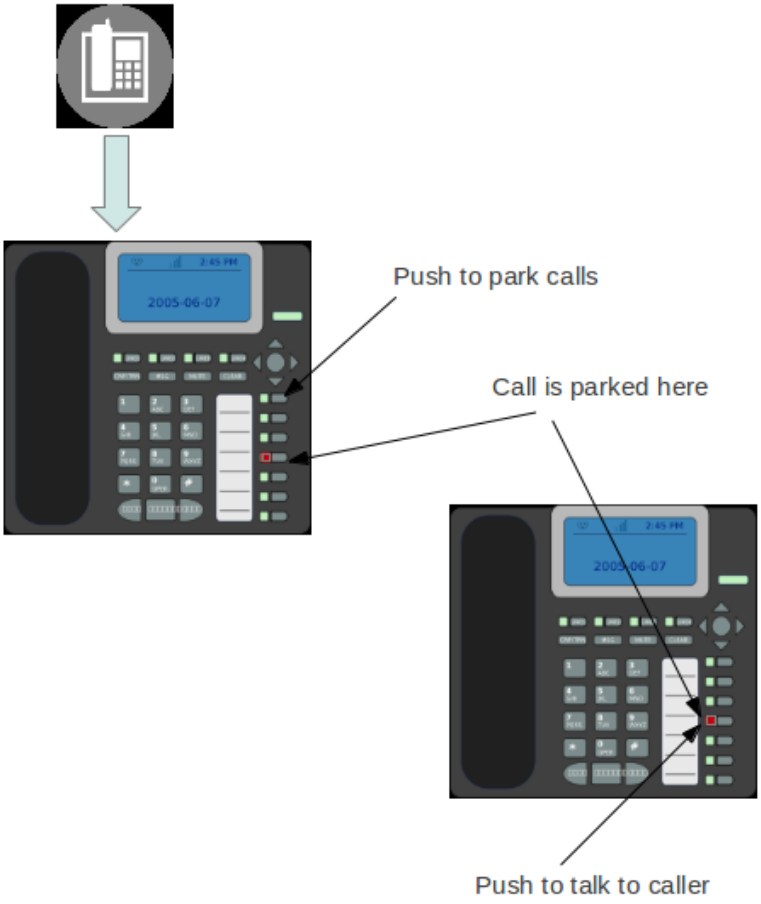
When adding a new paging extension, the number can be any numeric value; to call it, you just need to prefix the paging number with *11.

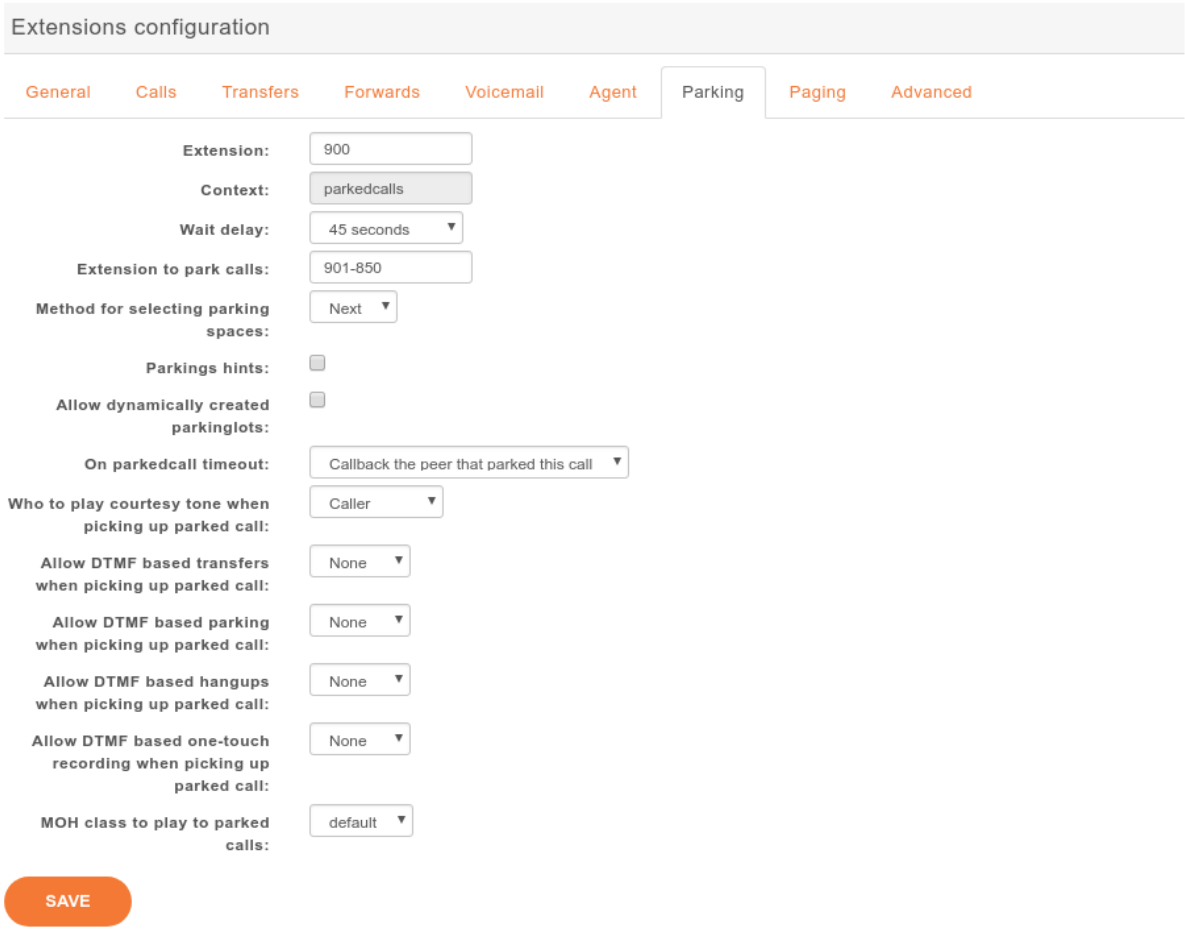
Parking

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.


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

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.

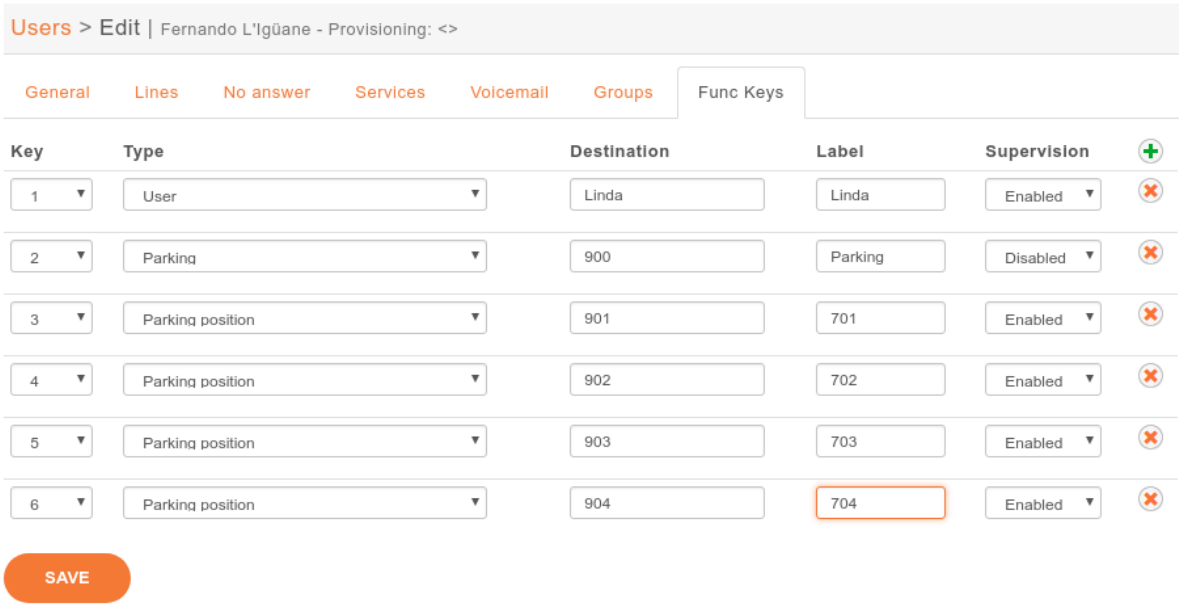
- Send park call to the dialplan

  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:

```
[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.



Phonebook

A global phone book can be defined in *Services → IPBX → IPBX Services → Phonebook*. The phone book can be used from XiVO assistants, 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*.

Meeting room contact

You can create a meeting room contact to been seen as an entry in your phonebook so that it can be searched and clickable from assistant.

To do so you need to fill at least: * displayname of the room * any number field with value following this format : *video:roomName* where roomName is the value you defined when you created a meeting room in XiVO webi.

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[Page 311, 1]
- faxnumber[1]
- officeaddress1
- officeaddress2
- officecity
- officestate
- officezipcode
- officecountry[2]
- homenumber[1]
- homeaddress1
- homeaddress2
- homecity
- homestate
- homezipcode
- homecountry[2]

[1] These fields must contain either numeric characters, or a video meeting number starting with "video:". For example you can write : "video: my wednesday meeting". The following characters are not allowed in the meeting room name : ? & ' " % : #

[2] These fields must contain ISO country codes. The complete list is described here.

- othernumber[1]
- otheraddress1
- otheraddress2
- othercity
- otherstate
- otherzipcode
- othercountry[Page 311, 2]

Provisioning

XiVO supports the auto-provisioning of a large number of telephony devices, including SIP phones, SIP ATAs, and even softphones.

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.

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

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:

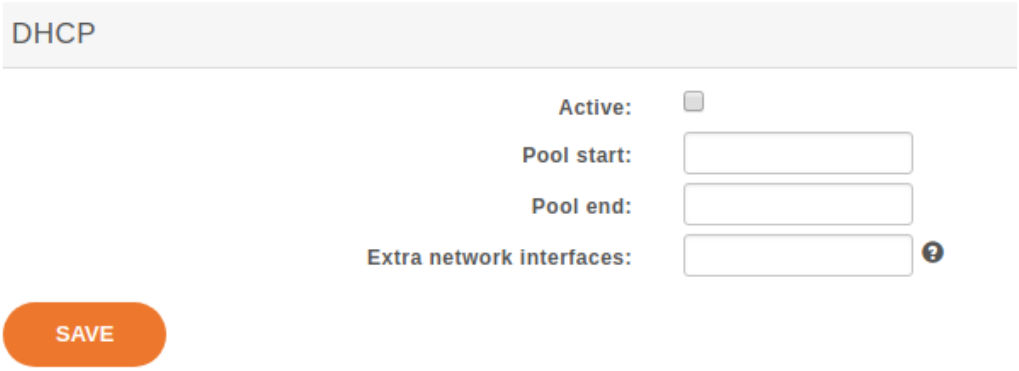


Fig. 42: *Configuration* → *Network* → *DHCP*

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.

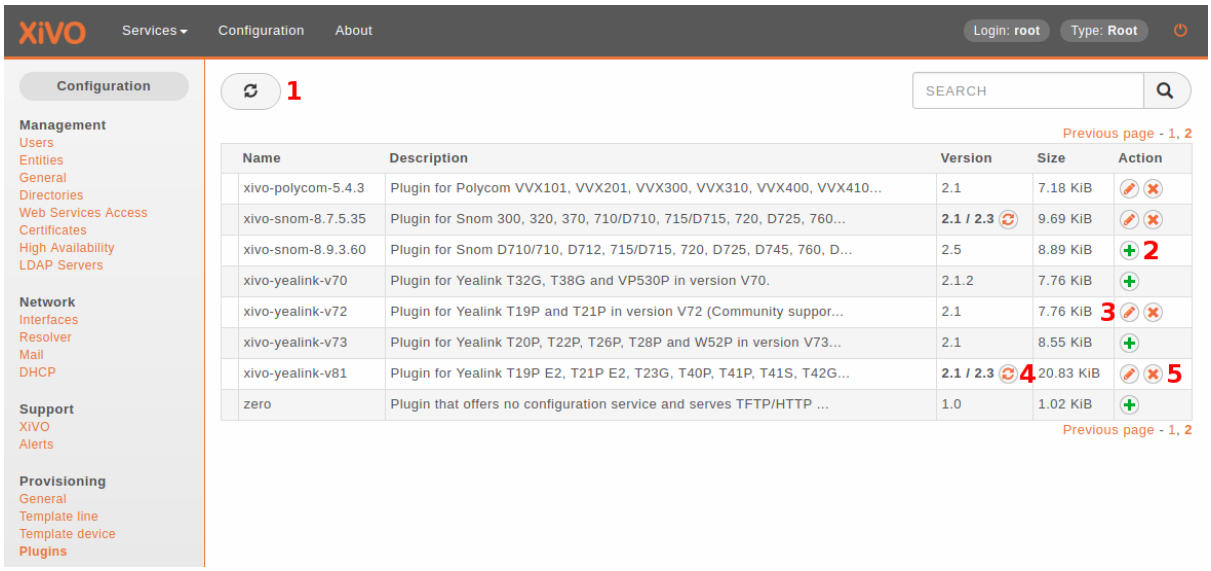


Fig. 43: *Configuration* → *Provisioning* → *Plugin*

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.

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.

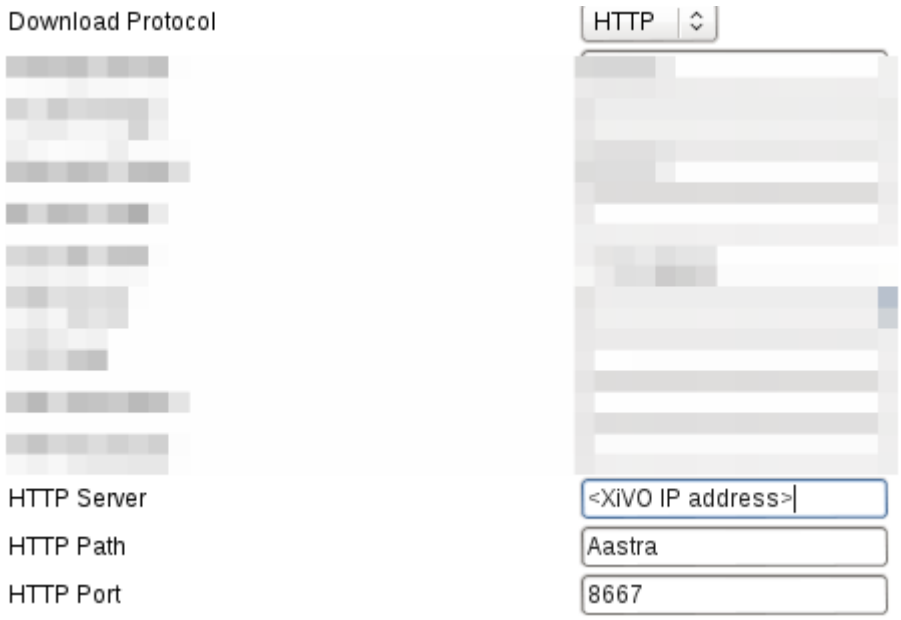
How to manually tell the phones to get their configuration

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.

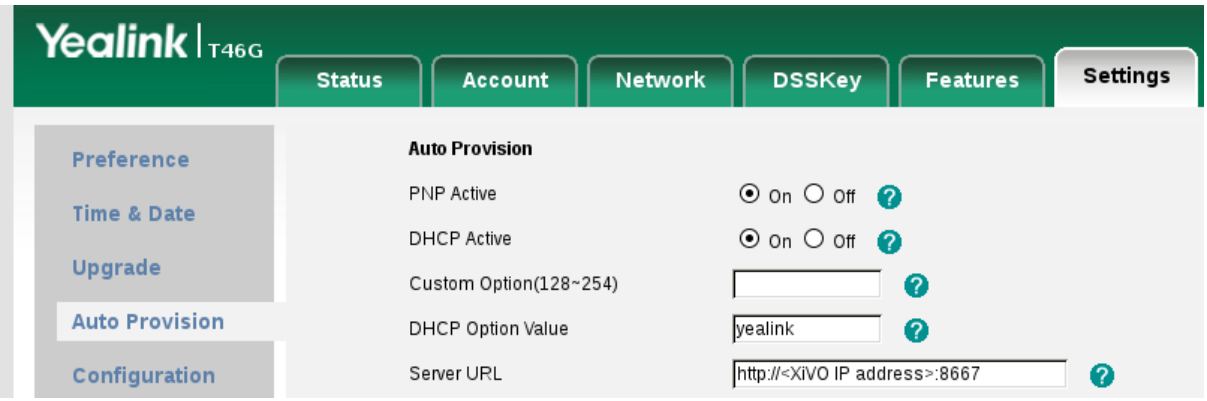
Snom

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`

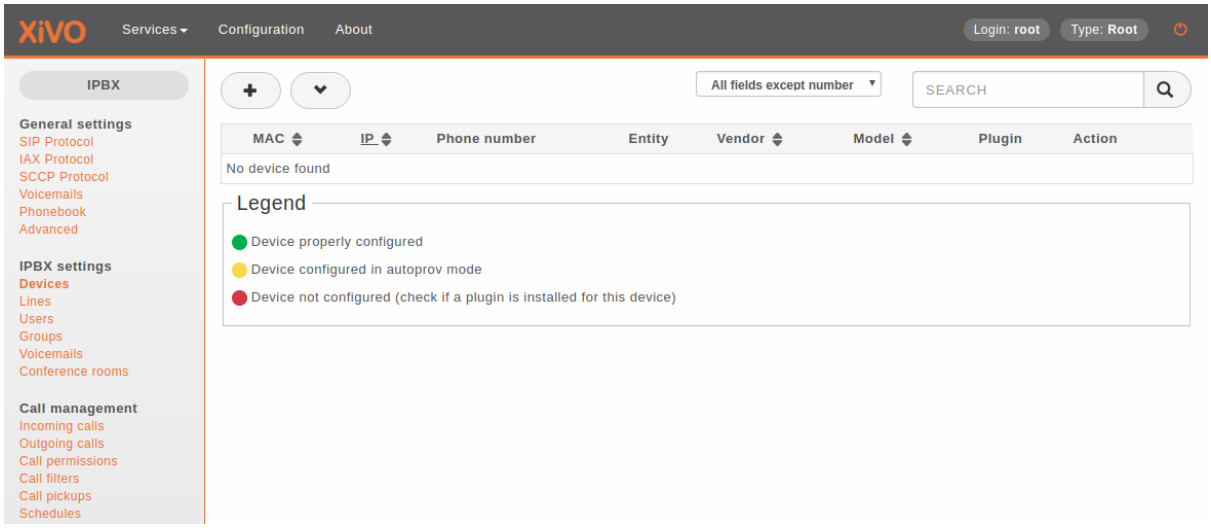


Save the changes by clicking on the *Confirm* button and then click on the *Autoprovision Now* button.

Autoprovisioning a Device

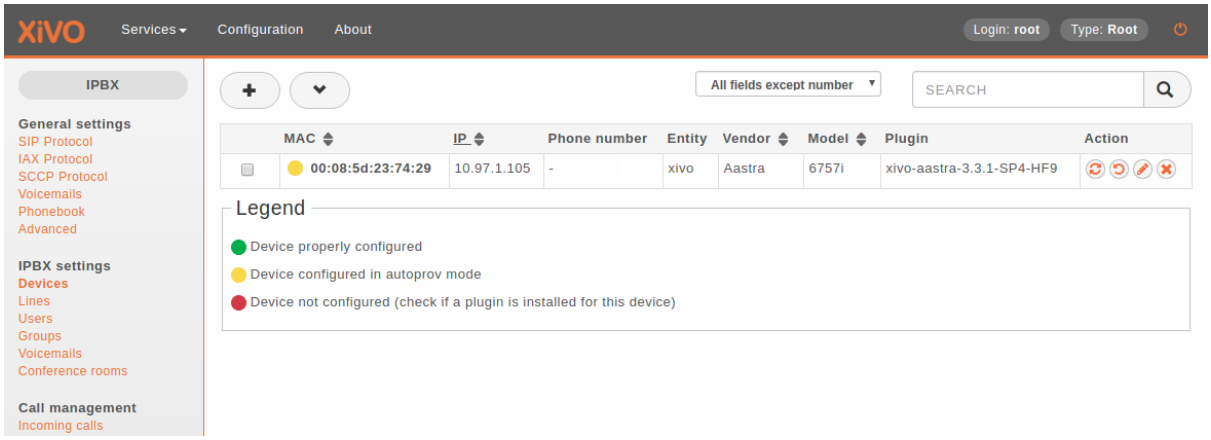
Once you have installed the proper provd plugins for your devices and setup correctly your DHCP server, you can then connect your devices to your network.

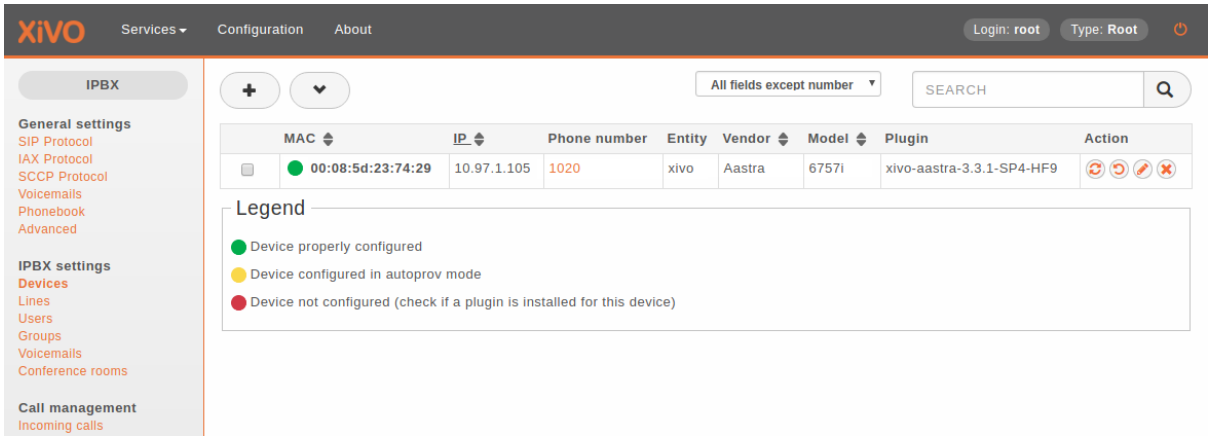
But first, go to *Services* → *IPBX* → *Devices* page. You will then see that no devices are currently known by your XiVO:



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:





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.

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-spa	>= 1.0
xivo-digium	>= 1.0
xivo-polycom	>= 1.7
xivo-snom	>= 1.6
xivo-yealink	>= 1.26

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.

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

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.

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.

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"`.

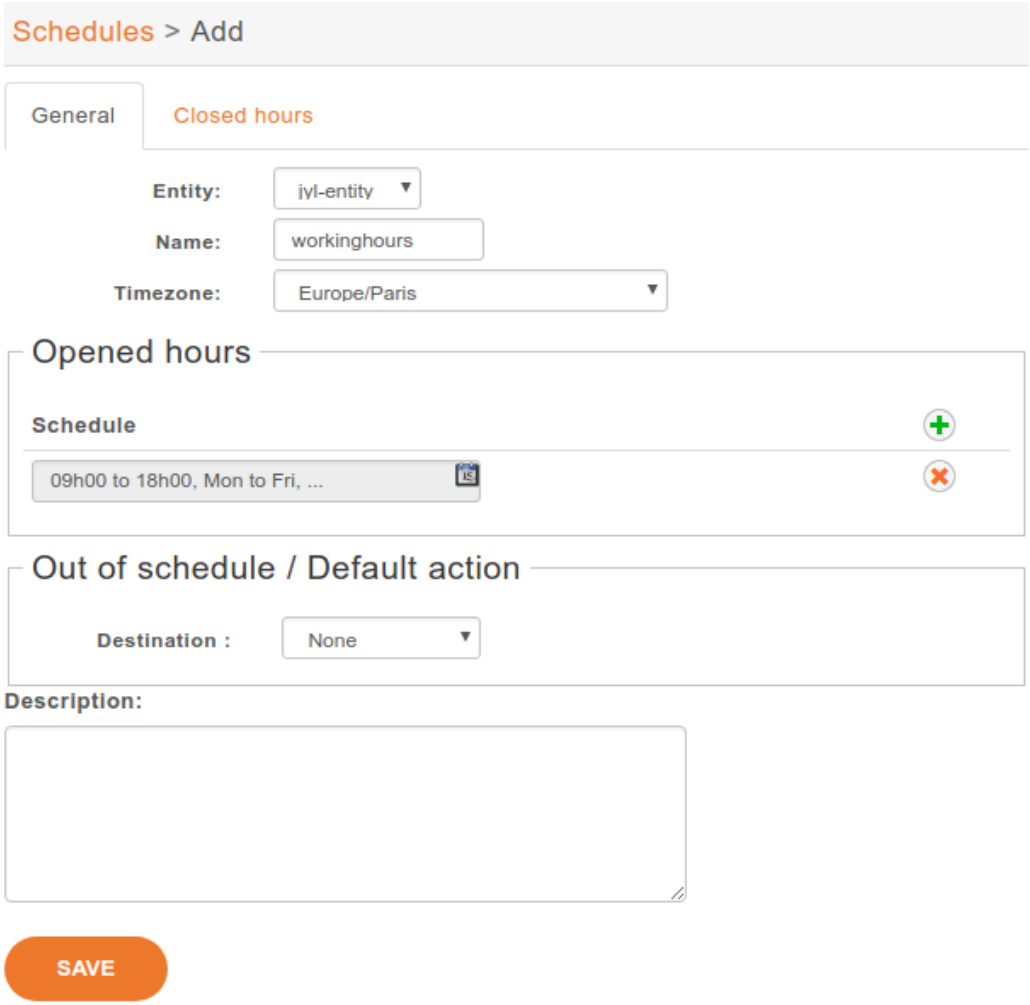


Fig. 44: Creating a schedule

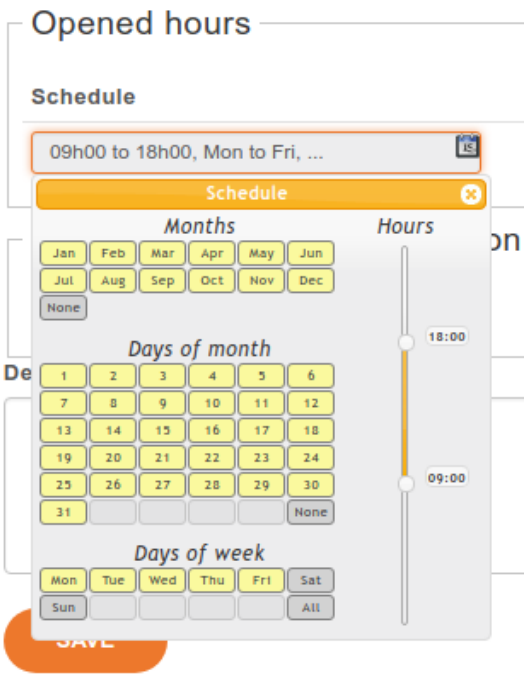


Fig. 45: Schedule calendar widget

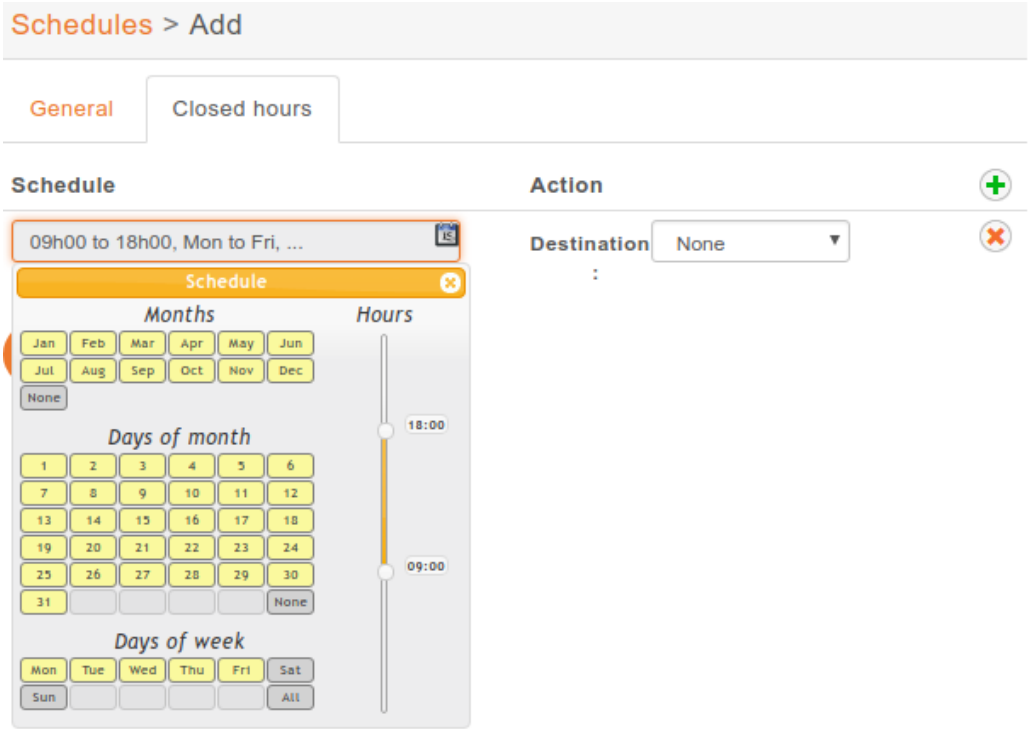


Fig. 46: Schedule closed hours

Sound Files

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.

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
```

Subroutine

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
- Intra MDS call
- Internal 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):

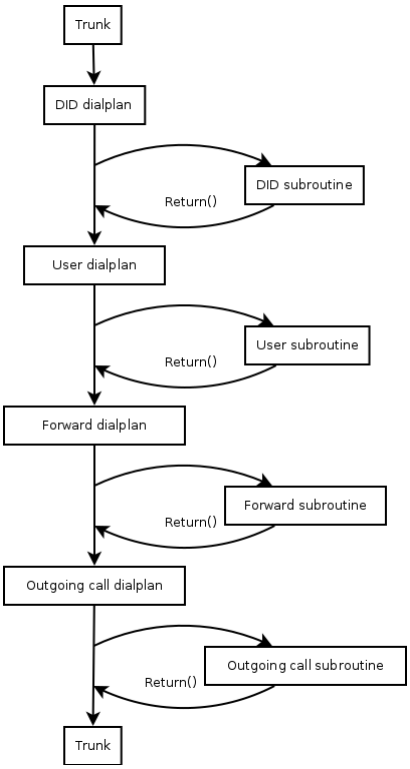


Fig. 47: Where subroutines are called in dialplan

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*.

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
XIVO_PRESUBR_GLOBAL_OUTCALL = xivo-subrgbl-outcall
XIVO_PRESUBR_GLOBAL_PAGING = xivo-subrgbl-paging
XIVO_PRESUBR_GLOBAL_XDS = xivo-subrgbl-xds
XIVO_PRESUBR_GLOBAL_INTERNAL = xivo-subrgbl-internal
```

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()
```

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
```

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.

Switchboard

This page describes the configuration needed to have a switchboard on your XiVO.

Table of Contents

- *Overview*
- *Configuration*
  - *Quick Summary*
  - *Supported Devices*
    - *Snom Phones*
    - *Yealink Phones*
  - *Create a Queue for Your Switchboard*
  - *Create Hold Queue*
    - *Create a General Hold Queue for Your Switchboard*
    - *Create a Personal Hold Queue for Your Switchboard*

  - *Create the Users that will be Operators*
  - *Create Agents for the Operator's Users*
- *Optional Configuration*
  - *Send Incoming Calls to the* Switchboard *Queue*
  - *Configure "No Answer" Destinations of the* Switchboard *Queue*
- *Statistics*
  - *Report*
  - *Usage*
- *Limitations*

Overview

Switchboard functionality is available in *XiVO CC* or *UC Embedded*.

The switchboard application allows an operator to view incoming calls, answer them, put calls on hold, view and retrieve the calls on hold. See the *Switchboard* page for detailed explaination on usage.

The goal of this page is to explain how to configure your switchboard.

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
- 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
Snom	D735	>= Electra	>= xivo-snom-10.1.46.16, v3.0.3
Yealink	T54W	>= Electra	>= xivo-yealink-v84, v1.4.2
WebRTC		>= Electra	N.A.

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.

If you changed the administrator username or administrator password for your phone, via *Configuration* → *Provisioning*→ *Template Device*, you need to follow the *Configuration Customization* procedure.

Yealink Phones

When using a Yealink switchboard, you must not configure a function key on position 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.

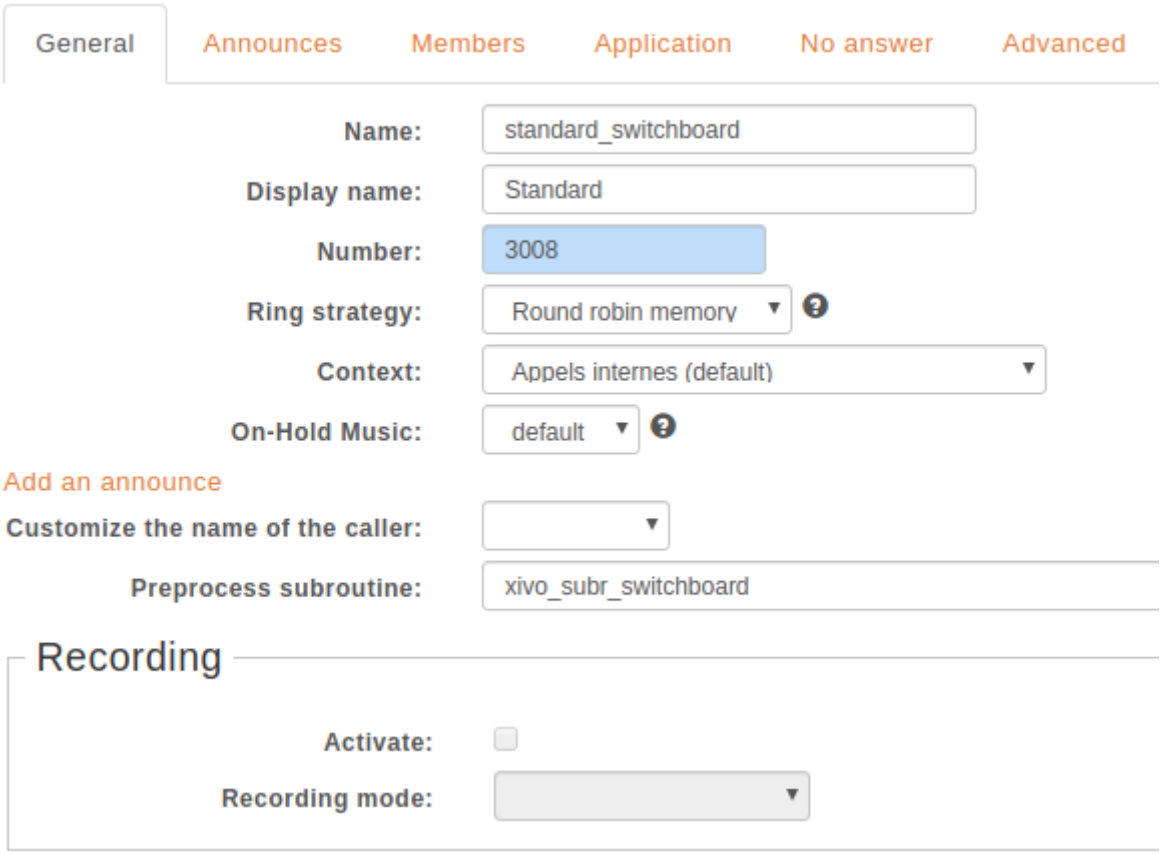
The following configuration is mandatory :

- *General* tab:
  - *Name* field must end with `_switchboard` suffix (e.g. *standard_switchboard*)
  - *Number* field must **valid number** in a context
  - *Preprocess subroutine* field has to be `xivo_subr_switchboard`
- *Advanced* tab:
  - *Member reachability timeout* option must be **disabled**
  - *Time before retrying a call to a member* option should be **5 second**
  - *Delay before reassigning a call* option must be **disabled**
  - *Call a member already on* option must be **disabled**
  - *Autopause agents* option has to be **No**

Create Hold Queue

Depending on your customer workflow you can create either:

- a single general hold queue that will be shared among all the operator of the switchboard
- or a personal hold queue per switchboard operator



Create a General Hold Queue for Your Switchboard

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 :

- *General* tab:
  - *Name* field **MUST**:
    - **start** with the name of your switchboard incoming call queue
    - and **end** with `_hold`

    Note: For example, if the switchboard incoming call queue is *standard_switchboard*, the name of the switchboard hold queue must be *standard_switchboard_hold*.

  - *Number* field must a **valid number** in a context reachable by the switchboard operator
- *Advanced* tab:
  - *Join an empty queue* option list has to be **empty**
  - *Remove callers if there are no agents* option list has to be **empty**

Create a Personal Hold Queue for Your Switchboard

A switchboard operator can have a personal hold queue instead of the general hold queue.

Note: A personal hold queue will always have the priority over the general one, meaning that the calls on hold for this operator will be sent by default to their personal hold queue.

To create this queue, go to *Services* → *Call center* → *Queues* and click the add button.

The following configuration is mandatory :

- *General* tab:
  - The *Name* field **MUST**:
    - **start** with the name of the switchboard incoming calls queue
    - and **end** with `_hold` and the operator's agent number

      Note: For example, if the switchboard incoming call queue is *standard_switchboard* and operator agent number is *8002*, the name of its personal switchboard hold queue must be *standard_switchboard_hold_8002*.
  - The *Number* field must a valid number in a context reachable by switchboard operator
- *Advanced* tab:
  - *Join an empty queue* option list has to be empty
  - *Remove callers if there are no agents* option list has to be empty.

Create the Users that will be Operators

Each operator needs to have a user configured with a line.

The following configuration is mandatory for switchboard users

- *General* tab:
  - *First name* field has to be set
  - *Enable CTI account* option has to be *enabled*
  - *Login* field has to be set
  - *Password* field has to be set
- *Lines* tab:
  - *Number* field has to have a valid extension
  - *Device* field has to be a *supported device*
- *Services* tab:
  - *Enable supervision* option has to be *enabled*

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

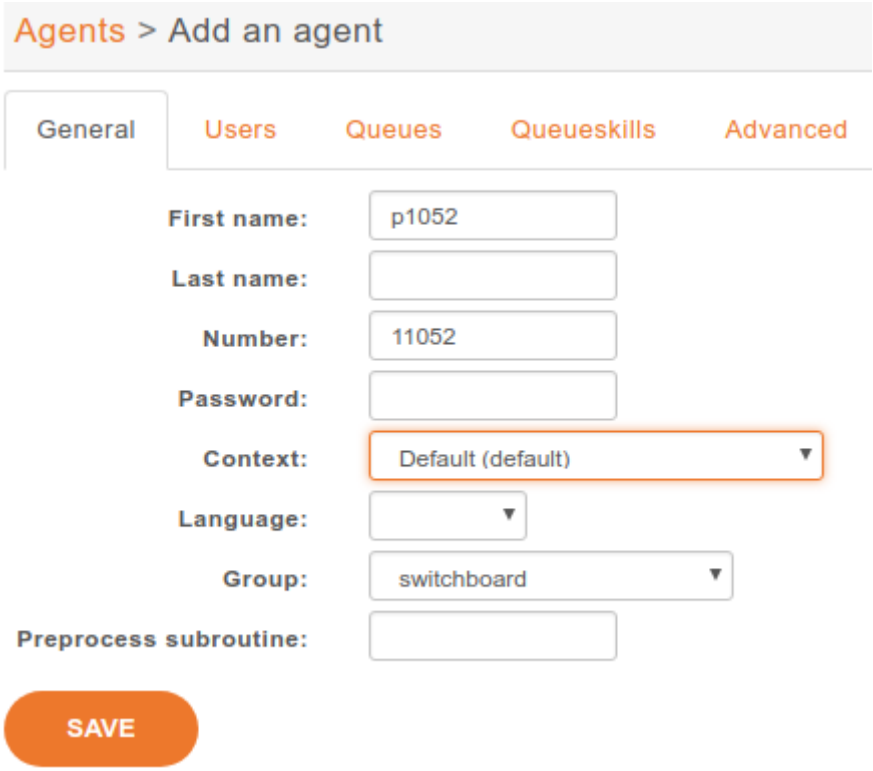
Create Agents for the Operator's Users

Each operator's user needs to have an associated agent.

> **Warning:** Those agents must be members of the *switchboard* queue only, they should not be a member of any other queue. The Hold Queue(s) (General Hold Queue/Personnal Hold Queues) must have NO members.

To create an agent:

- Go to *Services* → *Call center* → *Agents*
- Click on the group *default*
- Click on the *Add* button



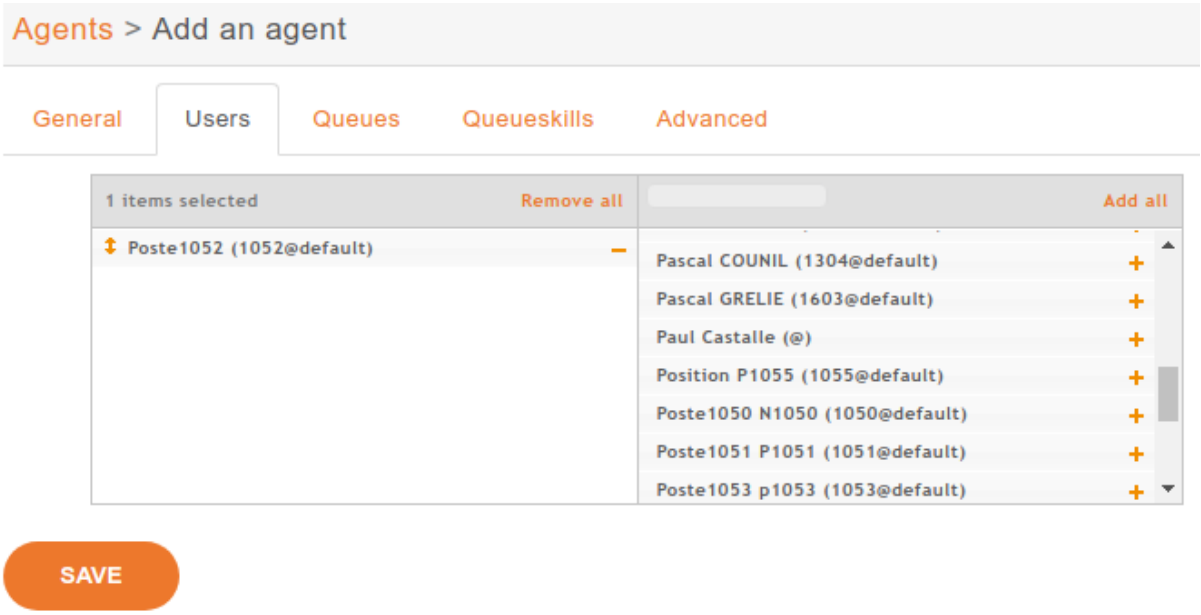
- Associate the user to the agent in the *Users* tab
- Assign the Agent to the *Switchboard* Queue.

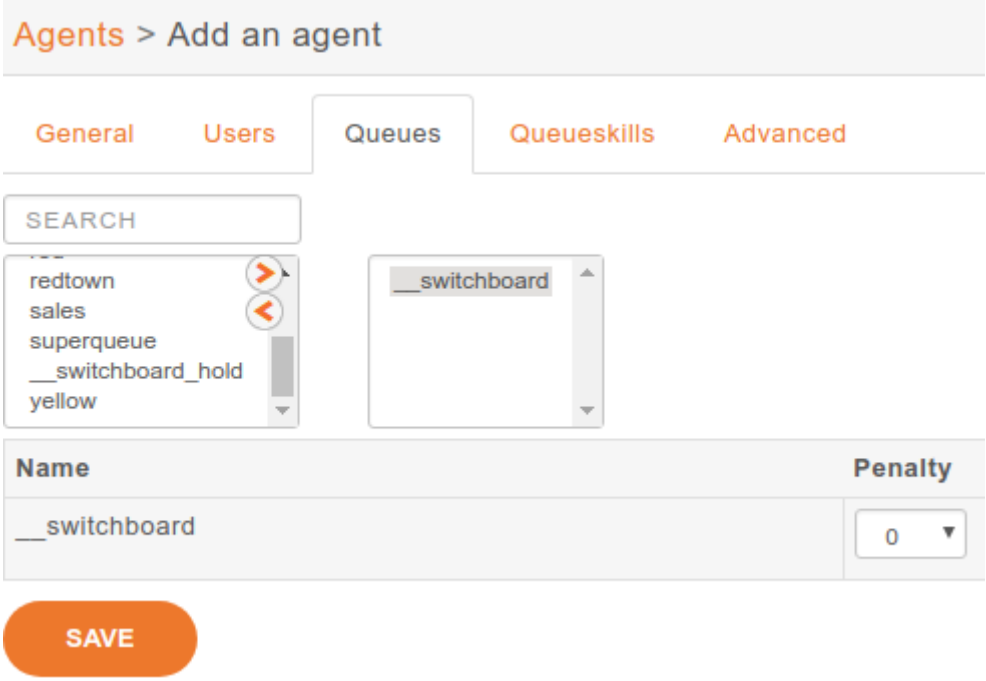
Optional Configuration

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:


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


Agents > Add an agent
General
Users
Queues
Queueskills
Advanced
SEARCH
redtown
sales
superqueue
__switchboard_hold
yellow
__switchboard
Name
Penalty
__switchboard
0
SAVE

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 :

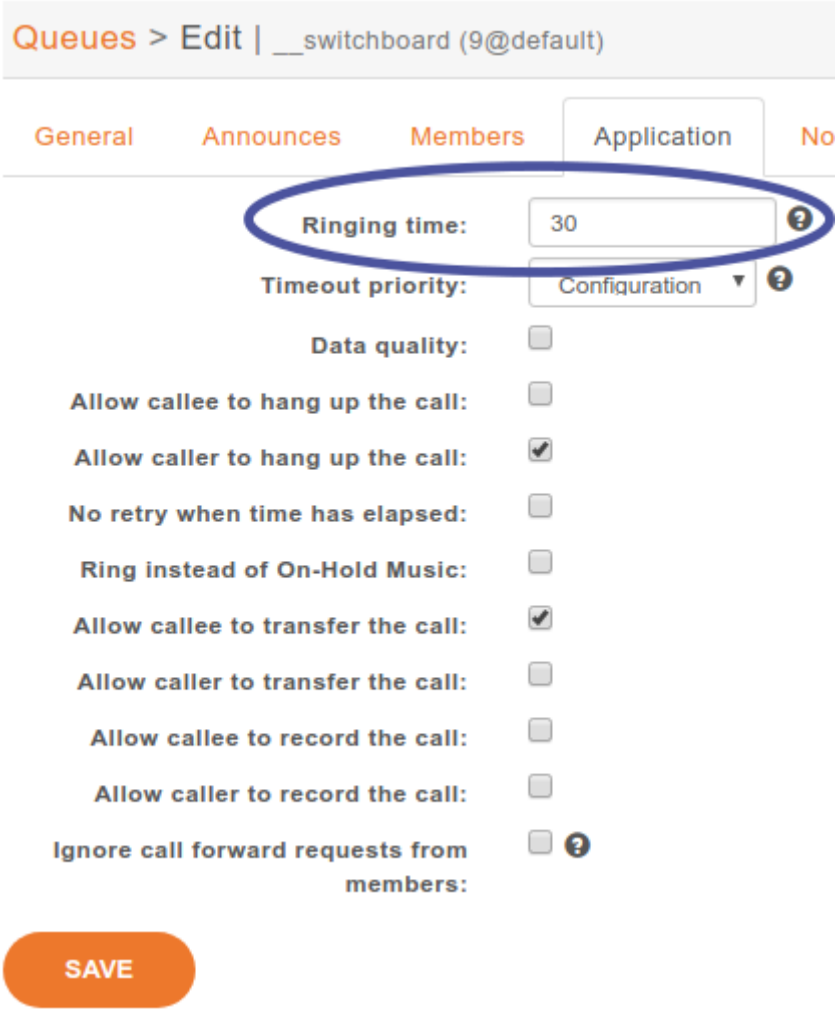
Description :

SAVE

Configure "No Answer" Destinations of the *Switchboard* Queue

You can configure the switchboard queue to redirect its calls to different destinations when none of its operator are available. The "No Answer" destinations defines where to redirect such calls.

To do so, you need to set the timeout of the Switchboard queue to know when calls will be redirected.

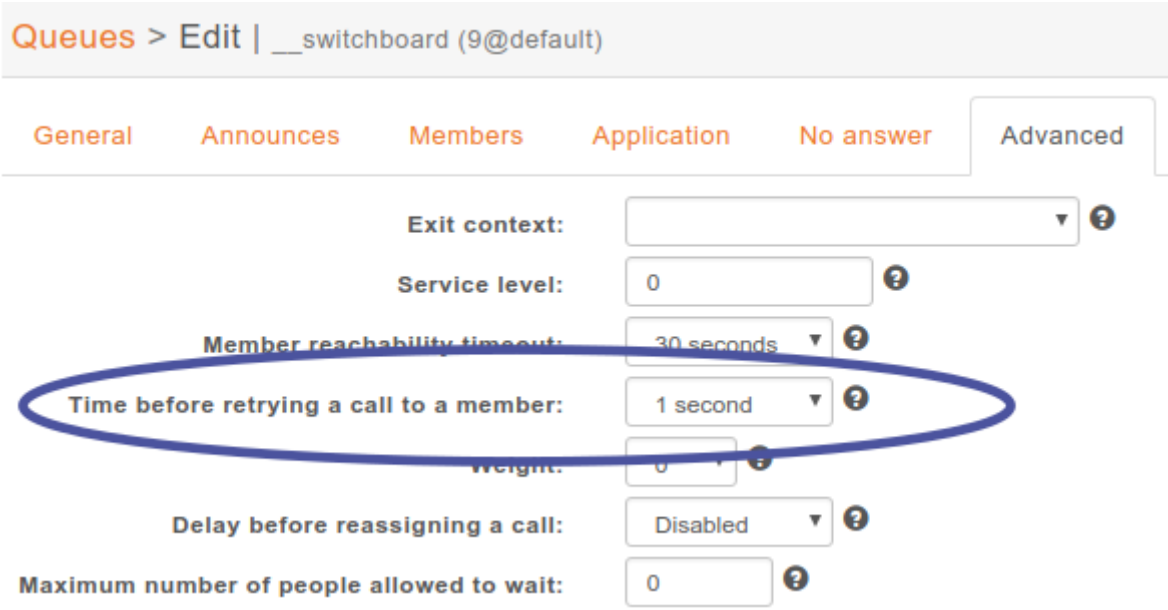


You must change the reachability timeout to not be disabled nor be too short.

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.



Statistics

Report

Switchboard report includes the following columns:

- date: The date range at which the calls were received
- offered: 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
- transferred: The number of calls that have been answered and then 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
- diverted: 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 either in switchboard or hold queue

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

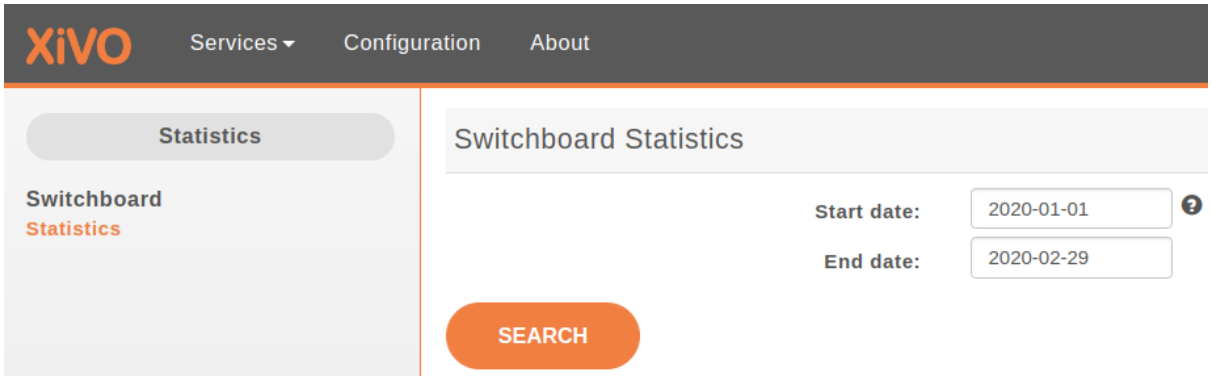
Redirect to : Poste1057 p1057

Ring time :

SAVE

Usage

Switchboard statistics can be retrieved in PDF format via the web interface in *Services → Statistics → Switchboard → Statistics*.



- Start date: required field
- End date: if empty, the result will contain statistics until the current date

Note: Switchboard statistics older than number of weeks defined by `WEEKS_TO_KEEP` environment variable in `/etc/docker/xivo/custom.env` are automatically removed.

Limitations

- Incoming calls can only be answered in order of arrival. It means the agent can't choose which call to answer first, they must answer the first one of the list, then the second, etc…
- When operator retrieves a call, it will cancel the current *ringing* call. Mainly it will cancel the call the the Incoming call queue was sending to him. It therefore puts back the Incoming call in the queue and gives the possibility to the operator to deal with the retrieved call. Note therefore that it will behaves the same for a *ringing* call on the operator phone which don't come from the queue: for example if someone inside the company called the operator without going through the queue. In this case the ringing call will be cancelled and lost. This is a limitation. To overcome this you must make sure that people inside the company should join the standard only via the Incoming call queue.

Unique Account

The *Unique Account* feature is a new feature introduced in Freya (2020.18) LTS version.

This feature enables a user to manage, with the same phone number, its Deskphone or UC Assistant (using WebRTC) or even Mobile App.

Description

Important: Prerequisites:

- have the *Requirements* configured
- and to use UA user on MDS follow the *Enable WebRTC on MDS* section

A *Unique Account* user can manage where to receive call though call management - check the *Features* section.

To configure a *Unique Account* user a XiVO Administrator must follow the *Configuration* section.

See also the *User's guide* for more details.

User's guide

Connection to UC Assistant

A *Unique Account* user can connect to the UC Assistant. When connected, the user will be connected as a WebRTC user.

1. Go to https://xivocc/ucassistant
2. Login with CTI credentials of UA user
3. The user is logged in with its WebRTC account
4. Then:

- All calls towards your user's extension will be sent to your UC Assistant
- You can also place internal our outgoing calls with your UC Assistant

Note: When connecting for the first time, the user will be using its WebRTC account. He can then chose the device he wants to use using the *Device selection* menu.

Device selection

The user can see which device he's using in the menu `Call Management` shown just above. In this menu he can chose which devices he want to receive calls. That means the calls received will be on the selected devices.

If you are only using the deskphone you can control it with the UC Assistant.

Note: If you have a mobile app paired for the user you will see it in the devices selection.

Note: The Device Selection is saved for a user.

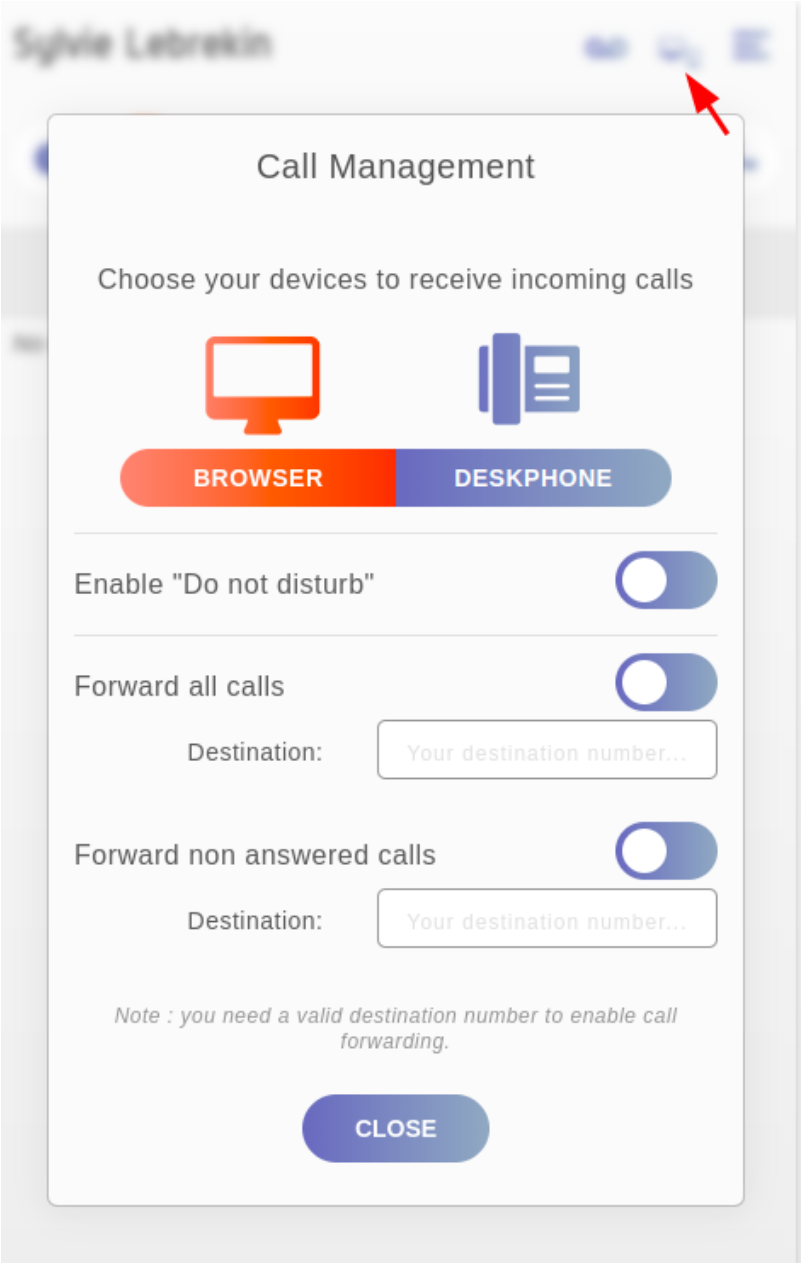
Device selection from deskphone

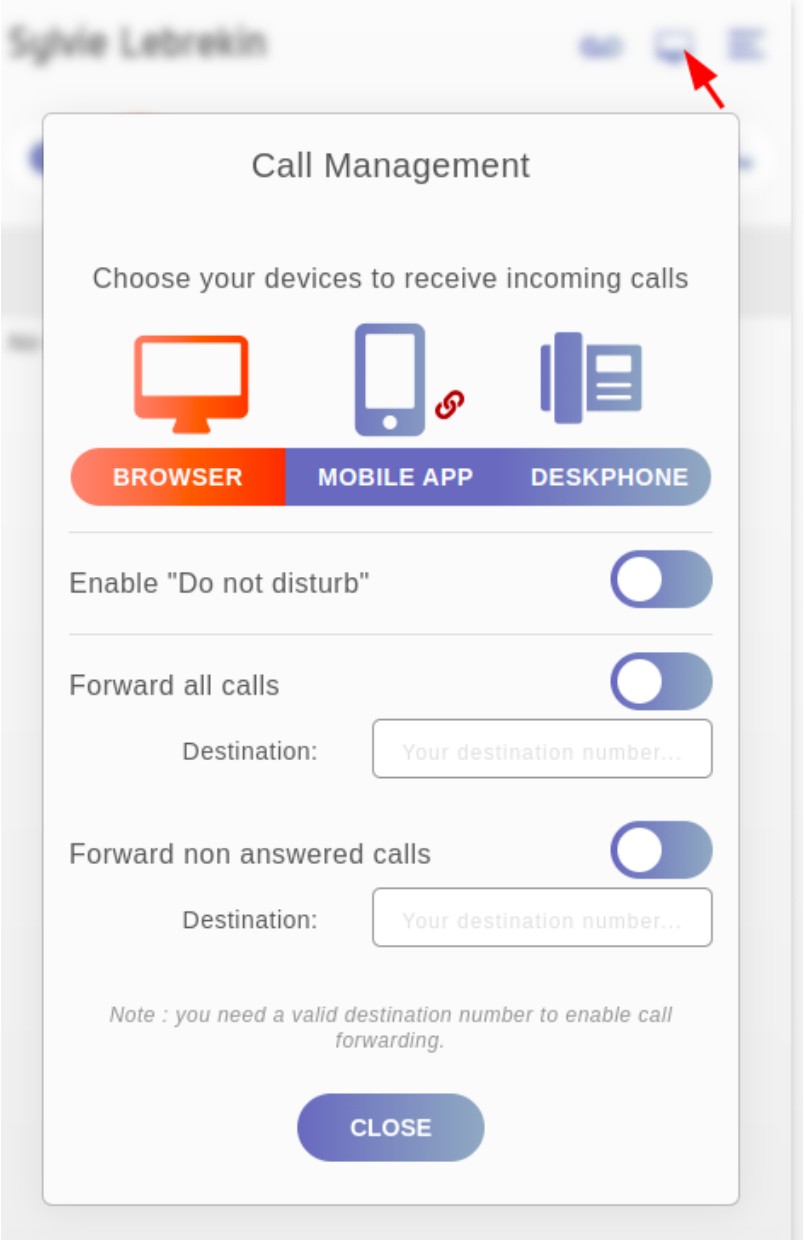
Deskphone can be added / removed from the selected devices by dialing the `*987` extension

Call history

Once you are connected to UC assistant you are able to see mixed history of your deskphone and your calls made through webrtc.

Note: If someone is making a call with deskphone while you are connected to UC assistant, the call will be also displayed in call history.


Sylvie Lebrekin
Call Management
Choose your devices to receive incoming calls
BROWSER
DESKPHONE
Enable "Do not disturb"
Forward all calls
Destination:
Your destination number...
Forward non answered calls
Destination:
Your destination number...
Note : you need a valid destination number to enable call forwarding.
CLOSE


Sylvie Lebrekin
Call Management
Choose your devices to receive incoming calls
BROWSER
MOBILE APP
DESKPHONE
Enable "Do not disturb"
Forward all calls
Destination:
Your destination number...
Forward non answered calls
Destination:
Your destination number...
Note : you need a valid destination number to enable call forwarding.
CLOSE

Features

Given a user Alice configured as a *Unique Account* user with an internal phone number. Given Alice's account is provisionned on a deskphone.

Use UC Assistant

If Alice **logs in** UC Assistant, then:

- Alice can use her UC Assistant
- Alice can call people from her UC Assistant
- When someone calls Alice (through its internal extension or DID number if she has one), her UC Assistant rings

Use Deskphone

If Alice *switches device* from the `Call Management` menu, nd chose the *Deskphone* then:

- Alice can call people from her Deskphone
- When someone calls Alice (through its internal extension or DID number if she has one), her Deskphone rings.

Use Mobile App

If Alice has a mobile app configured see *Mobile Application* and *switches device* from the `Call Management` menu, nd chose the *MobileApp* then:

- Alice can use her UC Assistant
- Alice can call people from her UC Assistant
- When someone calls Alice (through its internal extension or DID number if she has one), her Mobile App rings.

Use Multiple devices

If Alice wants to receive calls on several devices she can set multiples devices see *Device selection*

- Alice can call people (from her Deskphone or UC assistant depending on the scenario)
- When someone calls Alice (through its internal extension or DID number if she has one), her selected devices rings.

Note: For details on configuring devices and call management scenarios, refer to the relevant sections linked above.

Fallback

If Alice is unable to initialize the WebRTC stack in the UC Assistant, a popup notification will appear offering her the option to receive calls on her Deskphone instead.

Presence

Alice's presence is seen by her colleague whether she uses her deskphone or UC Assistant:

- When Alice is logged out of UC Assistant then
  - presence shown to other users is the status of the Deskphone
- When Alice is logged in the UC Assistant then
  - presence shown to other users is the status of the device Alice selected
  - status of the WebRTC line takes precedence over the Deskphone status
    - unless Alice emits a call with its Deskphone while logged in (then status will be In Use)

History

When Alice is logged into the UC Assistant, she can view the complete call history for all calls made or received through her deskphone, UC Assistant, and Mobile App.

Voicemail

Alice can check its voicemail with its Deskphone and UC Assistant.

No Answer and Forwarding

For a *Unique Account* user, the forwarding rules are applied to the device the user is using.

For example if the user is connected to the UC Assistant using its WebRTC line, and if he doesn't answer a call, the non answer rule is directly applied without making the deskphone ring.

Limitations

- To receive calls on the Mobile App, you must have an Edge server configured. For more information, see *XiVO Edge*
- Specific limitation for the users simultaneous count for group calls - see *Limitation*

Not supported features

These XiVO features **don't work** with a *Unique Account* user:

- **Paging**: a *Unique Account* user cannot be part of a paging
- **Contact Center**
  - CC Agent: a *Unique Account* user cannot be an agent.
- **Switchboard**: a *Unique Account* user can't be used for a Switchboard agent.

Configuration

To enable the *Unique Account* feature for a user:

1. Create a user
2. Go to the *Lines* tab and select the **Unique Account** line type
3. Save the form

Line type	Name	Context	Number
Unique Account		Appels internes	1021

The *Unique Account* feature is activated for this user.

Note: You can also create users via *User Import, Export and Update*.

Users

User Import, Export and Update

Contents

- *User Import, Export and Update*
  - *CSV Import*
    - *CSV file*
      - *User*
      - *CTI Profile*
      - *Phone*
      - *Incoming call*
      - *Voicemail*
      - *Call permissions*
    - *Importing a file*
    - *Examples*
  - *CSV Update*
  - *Import/Update Log*
  - *CSV 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

Additionally, these restrictions for specific fields must also be respected:

- Field *labels*: All user labels in this field must already exist before the import

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_numbe	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_enabl	bool			Enable/Disable call transfers by DTMF
dtmf_hangup_enat	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_pa	string			Overwrite all passwords set in call permissions associated to the user
labels	string		list separated by semicolons (;)	Names of the labels to assign to the user

CTI Profile

Field	Type	Required	Values	Description
cti_profile_ena	bool	No		Activates the CTI account for this user
username	string	Yes, if profile enabled		CTI Login username
password	string	Yes, if profile enabled	Must be 4 characters long[1].	CTI Login password
cti_profile_nan	string	Yes, if profile enabled		CTI profile (Defined in menu *Services → CTI server → Profiles*)

[1] If password is less than 8 characters long and does not contains both letters and digits a warning will be issued during the import/update but the user will still be imported.

Phone

Field	Type	Require	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_p	string	Yes	sip, webrtc, ua	Line protocol
line_s	string			Unique name of one of the *Template line* (defined in menu *Configuration* → *Provisioning* → *Template Line*)
sip_us	string			SIP username
sip_se	string			SIP secret
sip_op	string		list key=value separated by semicolons (;)	List of SIP options formatted as key=value pairs separated by semicolons (;). This field allows providing one or more SIP configuration options to be applied to the line. Each option is defined as a key and its associated value.

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_contex	string	Yes		context used for calls coming from outside of XiVO
incall_ring_s	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_permission	string		list separated by semi-colons (;)	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.

Fig. 48: Import Users

Examples

The following example defines 2 users who each have a phone number.

entity_id,firstname,lastname,exten,context,line_protocol 1,John,Doe,1000,default,sip
1,George,Clinton,1001,default,sip

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.
- An incall cannot be updated if the user has more than one incall associated.
- All user labels in the labels field must already exist before the update

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.

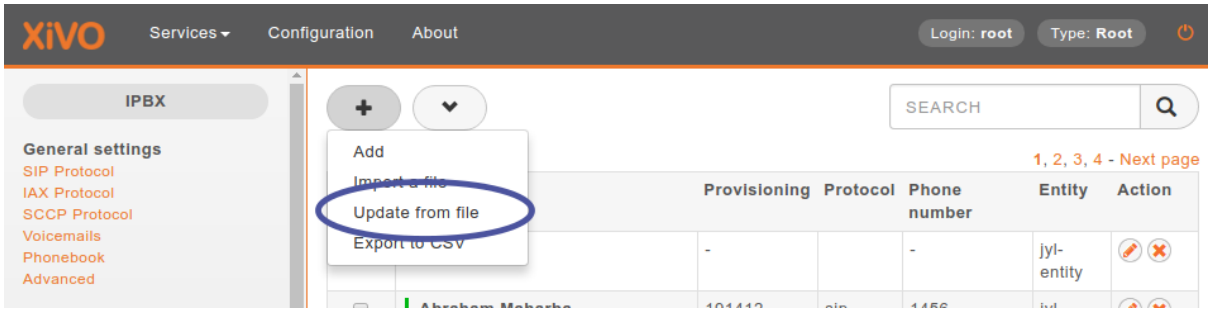


Fig. 49: *Services → IPBX → IPBX settings → Users → Update from file*

Import/Update Log

Beginning, end and all import/update errors and warnings are logged to `/var/log/xivo-confd.log`.

CSV import/update can take several minutes. If it reaches a timeout you'll see a message in the webi explaining that the import/update continues in the background and that you should check the log to verify that it was successfully finished.

If there are any invalid CSV rows, associated warnings or errors will be printed in the log:

- If there are warnings only the changes will be applied.
- If there is one error or more all changes will be rollbacked.

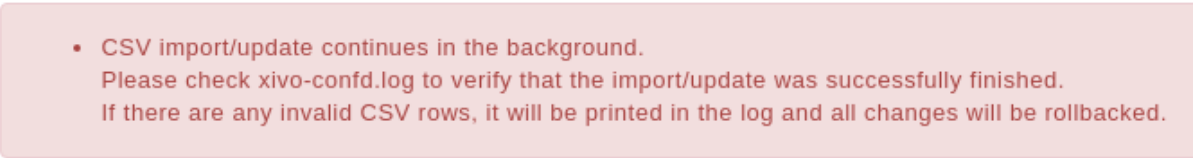


Fig. 50: Import or Update timeout message

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.

Users Configuration.

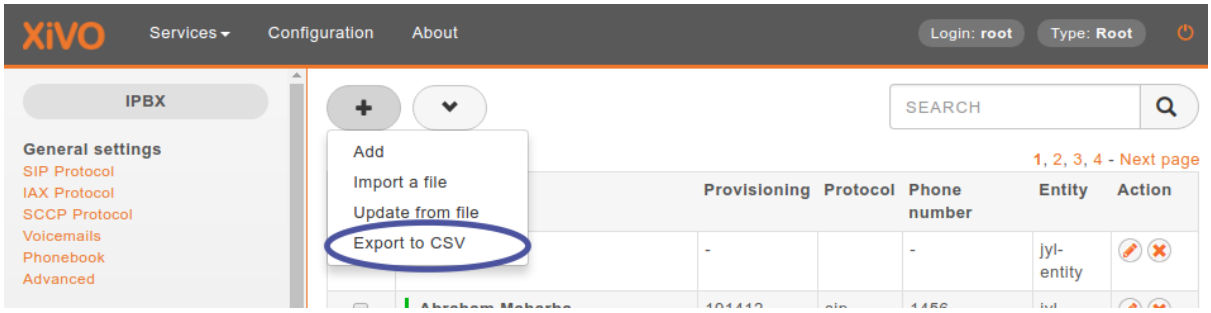
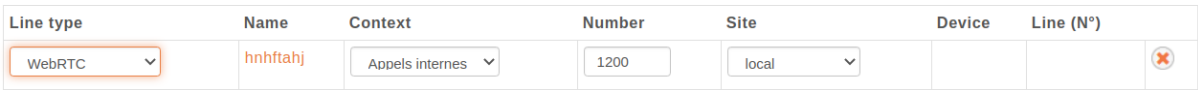


Fig. 51: *Services → IPBX → IPBX settings → Users → Export to CSV*

- *User lines*
- *Simultaneous Calls Limit*
  - *Overview*
  - *What Counts Toward the Limit*
  - *Behavior*
  - *Limitation*
- *Function keys*
- *Extensions*
  - **3 (online call recording)*
  - **26 (call recording)*
  - **55 (echo test)*

User lines

Lines can be configured from the Lines tab when creating a user. Users with line *Unique Account*, *Phone* or *WebRTC* can easily edit their line type from the same menu.



Users with line type *Custom* cannot change line type. In that case, the line should be deleted and created again with a new line type.

Simultaneous Calls Limit

Overview

The **simultcalls** parameter controls the maximum number of concurrent calls a user can handle. When this limit is reached, additional incoming calls will be sent to the busy no answer scenario instead of ringing the user's device.

What Counts Toward the Limit

The simultcalls counter tracks **all active calls**[1] involving the user, regardless of how the call was initiated or received:

- Calls[2] made by the user
- Calls towards the user either:
  - directly to the user (internally or via its incoming call (i.e. DID))
  - or coming via groups where the user is a member
  - or coming via queues where the user's agent is logged in
  - or coming via boss-secretary filter calls

Behavior

Given:

- User U1 has **simultcalls = 1**
- G1 is a group which U1 is a member of
- G2 is a group which U1 is a member of and which has the *Call a member already in line* activated
- Q1 is a queue which U1's agent is a member of
- Q2 is a queue which U1's agent is a member of and which has the *Call a member already in line* activated

U1 Current Call State	New Call Towards U1	Result
U1 calls[2]	U2 calls U1 directly	**Busy**
U1 calls[2]	U2 calls group G1	**U1 not called** (InUse)
U1 calls[2]	U2 calls group G2	**U1 not called**
U1 calls[2]	U2 calls queue Q1	**U1 agent not called** (InUse)
U1 calls[2]	U2 calls queue Q2	**U1 agent not called**
U1 on call (via a group)	U2 calls U1 directly	**Busy**
U1 on call (via a queue)	U2 calls U1 directly	**Busy**
U1 on call (via Boss/secretary filter)	U2 calls U1 directly	**Busy**

Note: Even when the limit is reached, the user can still **initiate** new outgoing calls.

Note: The behavior can be reverted to pre-Quantum by editing `/etc/xivo/asterisk/xivo_globals.conf` file and setting `XIVO_SIMULTCALLS_COUNT_ONLY_INCOMING = 1` and then reloading the dialplan. In this mode only direct incoming call to the user are counted in its simultcalls.

[1] Ringing calls are counted in the active calls.
[2] Call emitted by the user to other users, groups, queues, external numbers, conferences, meeting rooms.

Limitation

- Group calls for *Unique Account* users: for *unique account users* with multiple *ringing devices selected*, when a call arrives through a group, each ringing device is counted separately in the simultcalls counter during the ringing phase.

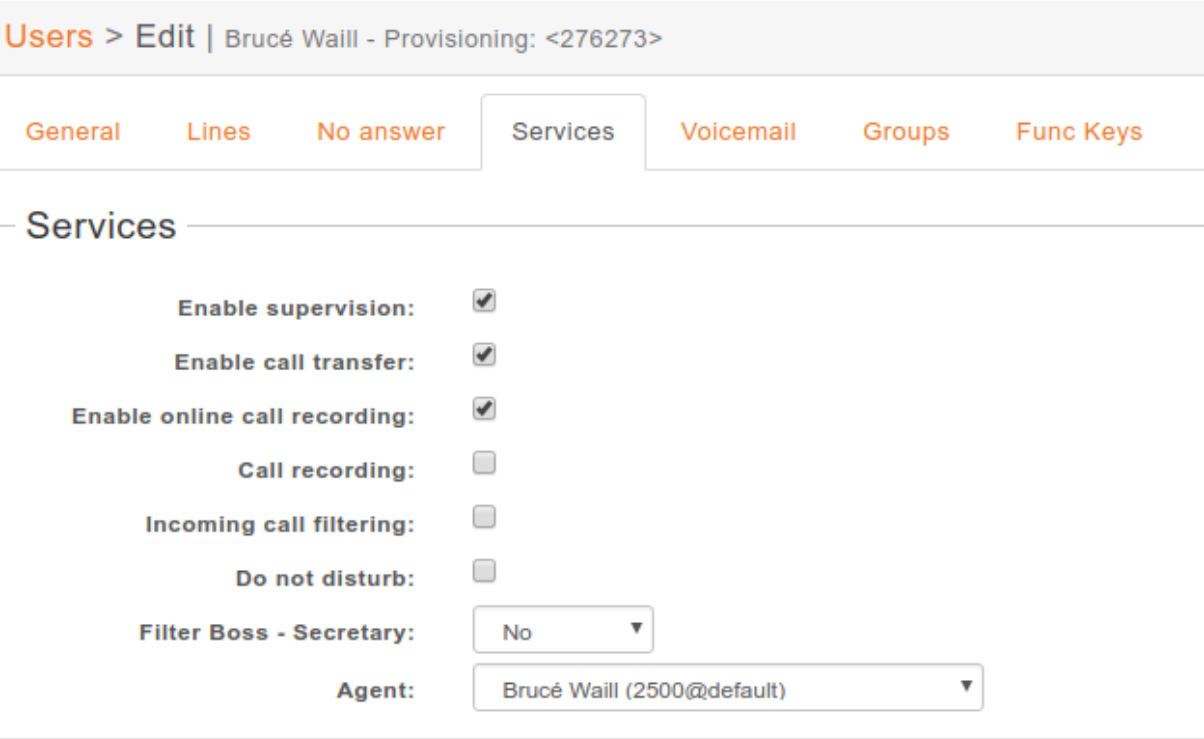
  Example: UA user U1 selected 2 ringing devices (WebApp and Desktop phone).

  - When a new call arrives to U1 via a group
  - During ringing, the simultcalls counter reaches 2 (2 ringing devices) - the call is ringing on WebApp and Desktop phone
  - Once U1 answers, it drops to 1

  Note therefore that, if U1 had a simultcalls of 2 and were to be called during this call group rung, the new call would be refused (as the limit would be reached). Once U1 would have answered this group call, a new call could be presented to U1.

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.



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.

Extensions

**3* (online call recording)

To enable online call recording, you must check the "Enable online call recording" box in the user form.

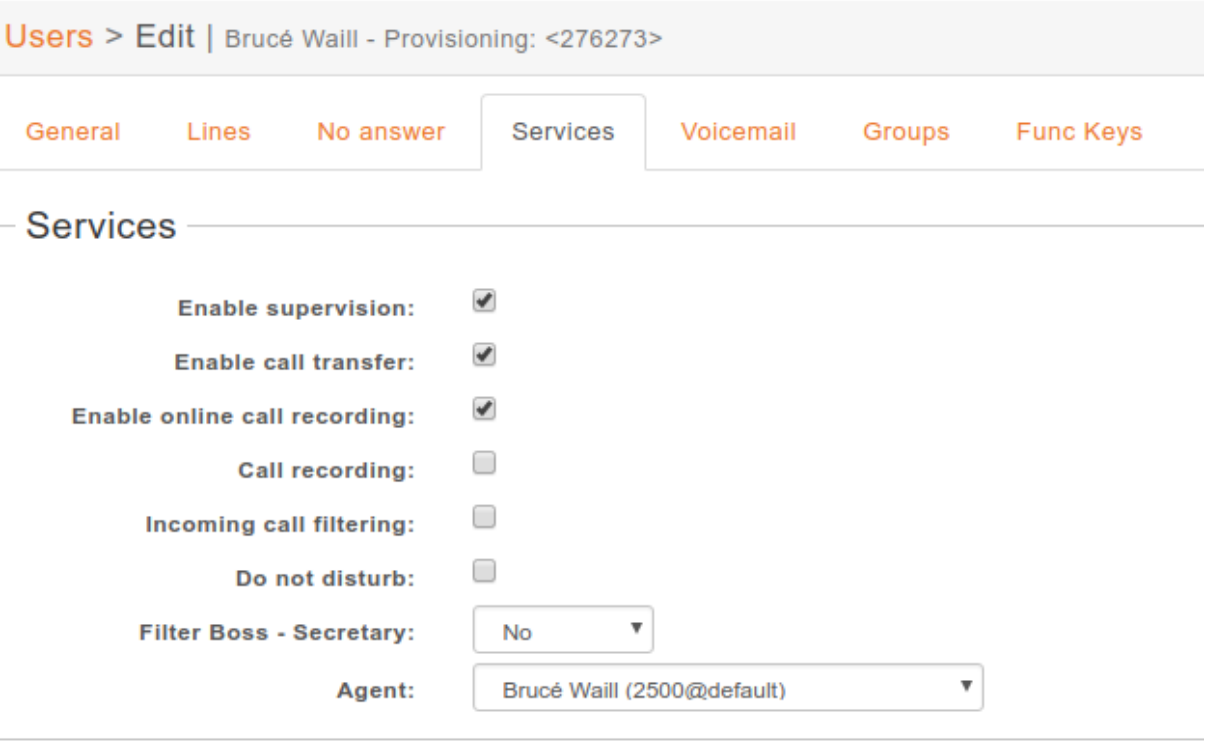


Fig. 52: Users Services

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.

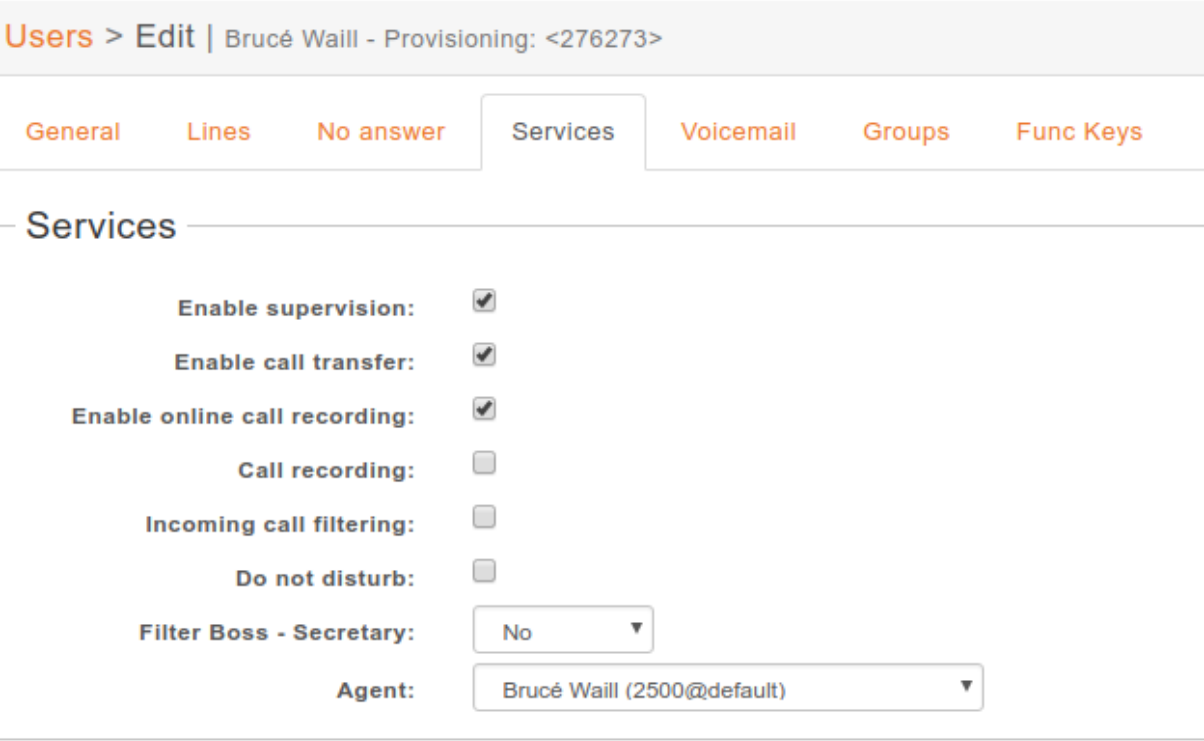


Fig. 53: Users Services

**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.

Voicemail

Voicemail Configuration.

General Configuration

The global voicemail configuration is located under *Services* → *IPBX* → *General Settings* → *Voicemails*.

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.

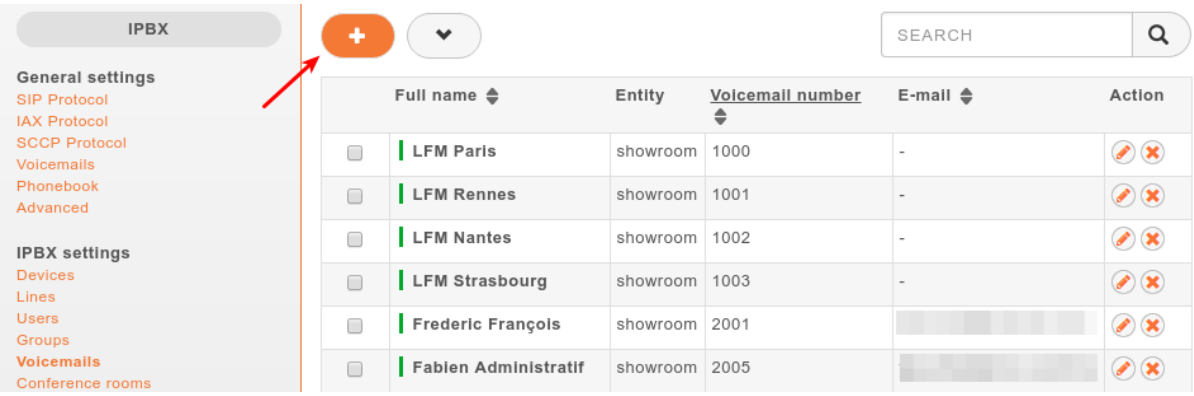


Fig. 54: Add voicemails from voicemail menu

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.

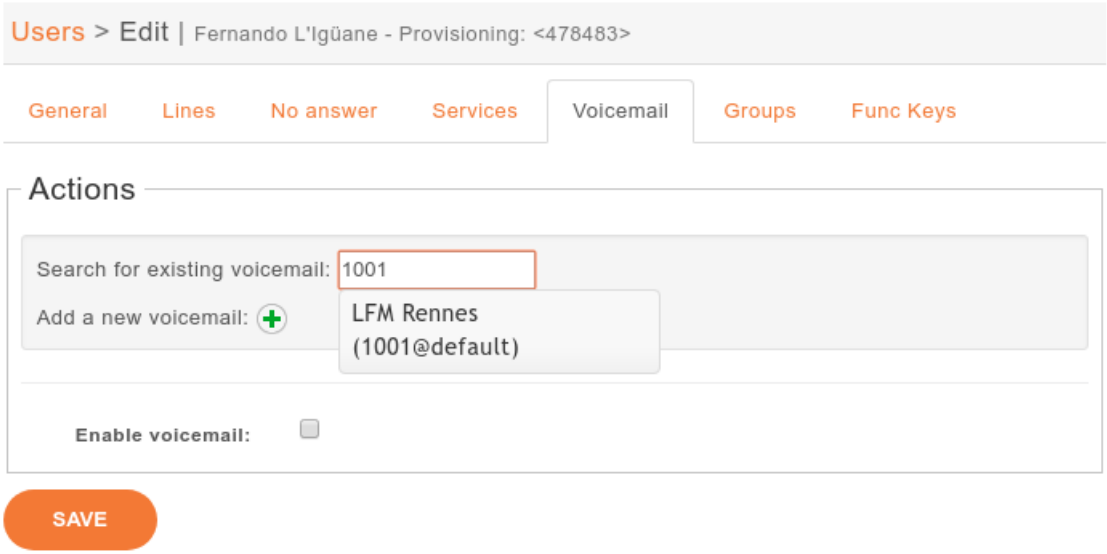


Fig. 55: Search for a voicemail in the user's configuration

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
3. Saving

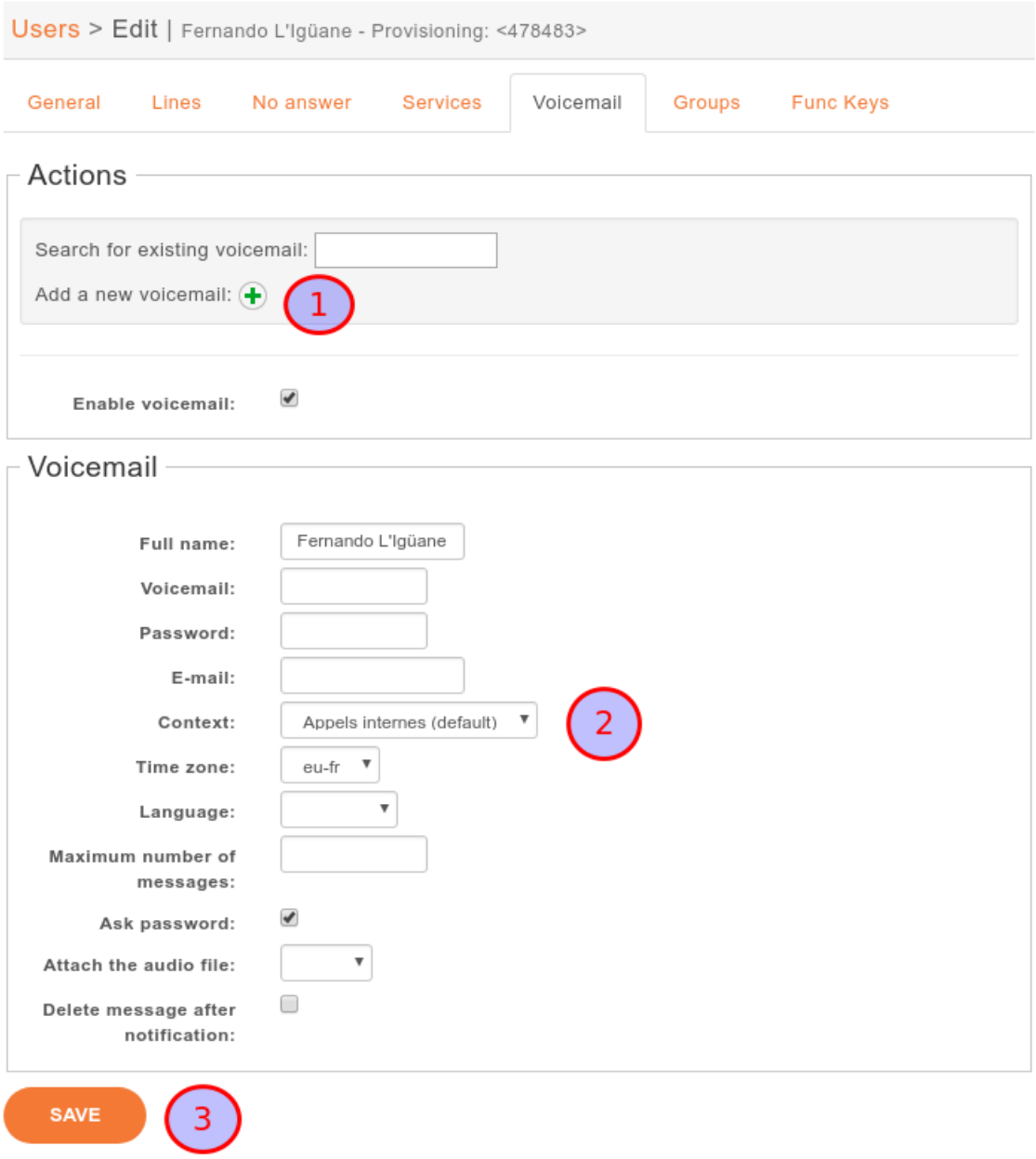


Fig. 56: Add a voicemail from the user's configuration

Note: The user's language *must* be set in the *general* tab

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.

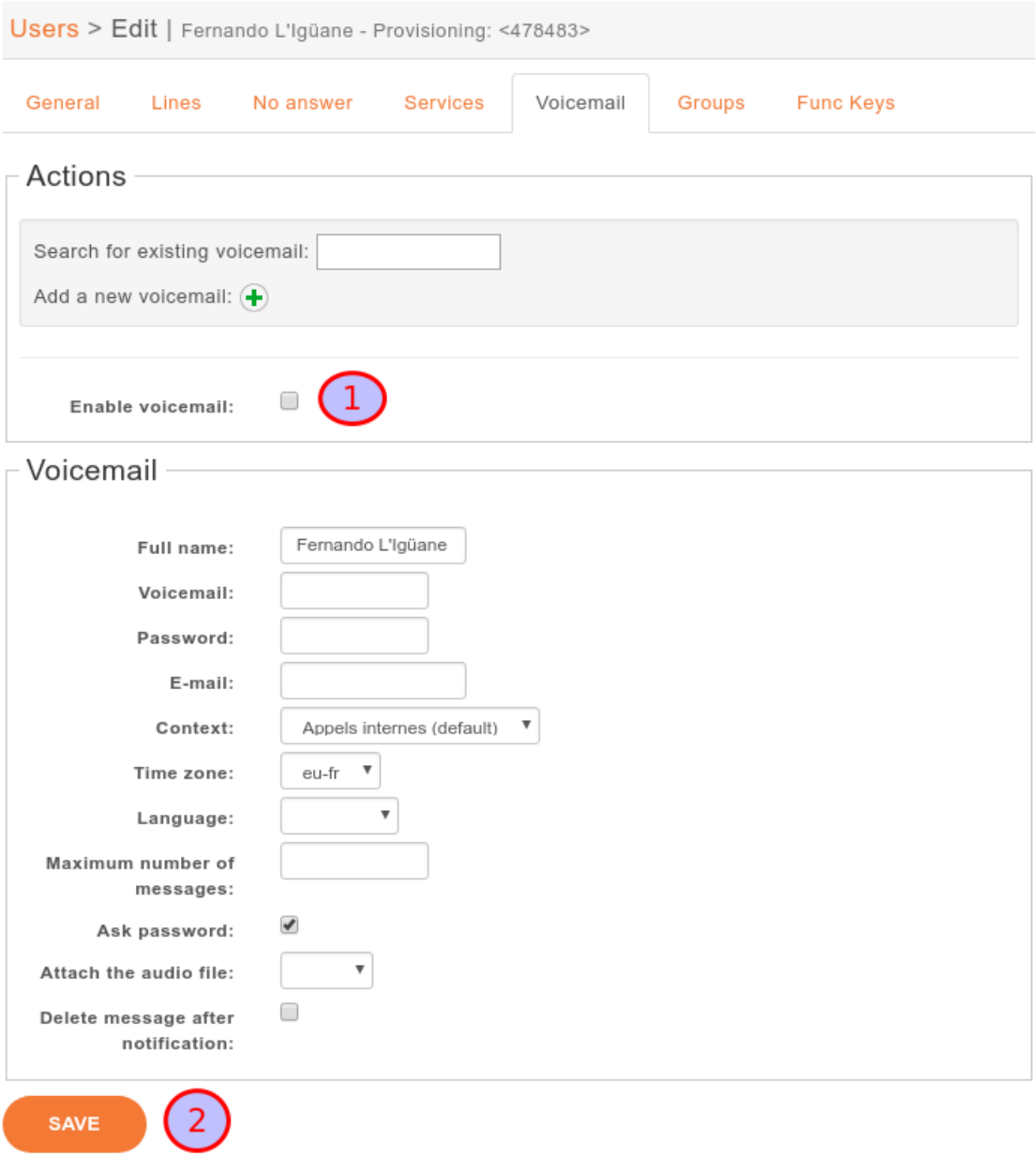


Fig. 57: Deactivate user's voicemail

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.

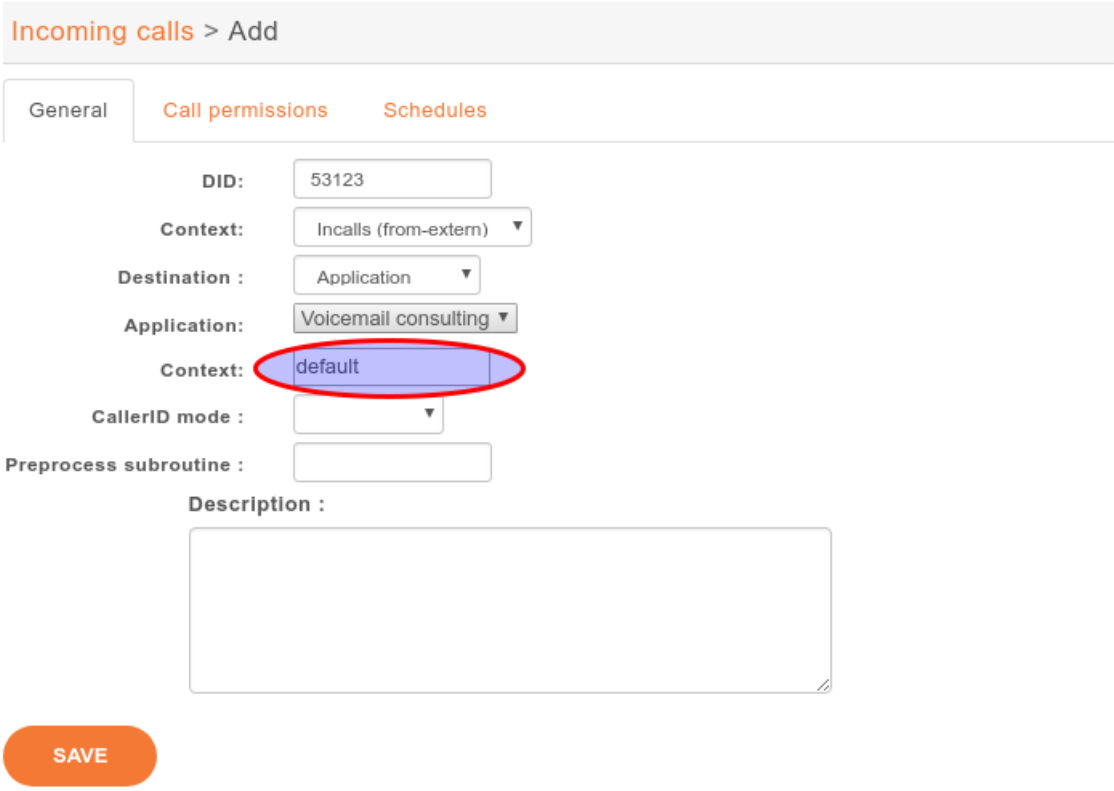
Disable password checking

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.

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.

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
```

WebRTC

General notes

Note: added in version 2016.04

Note: Current WebRTC implementation (since XiVO Freya 2020.18) does not require any of these manual configuration steps *Users*

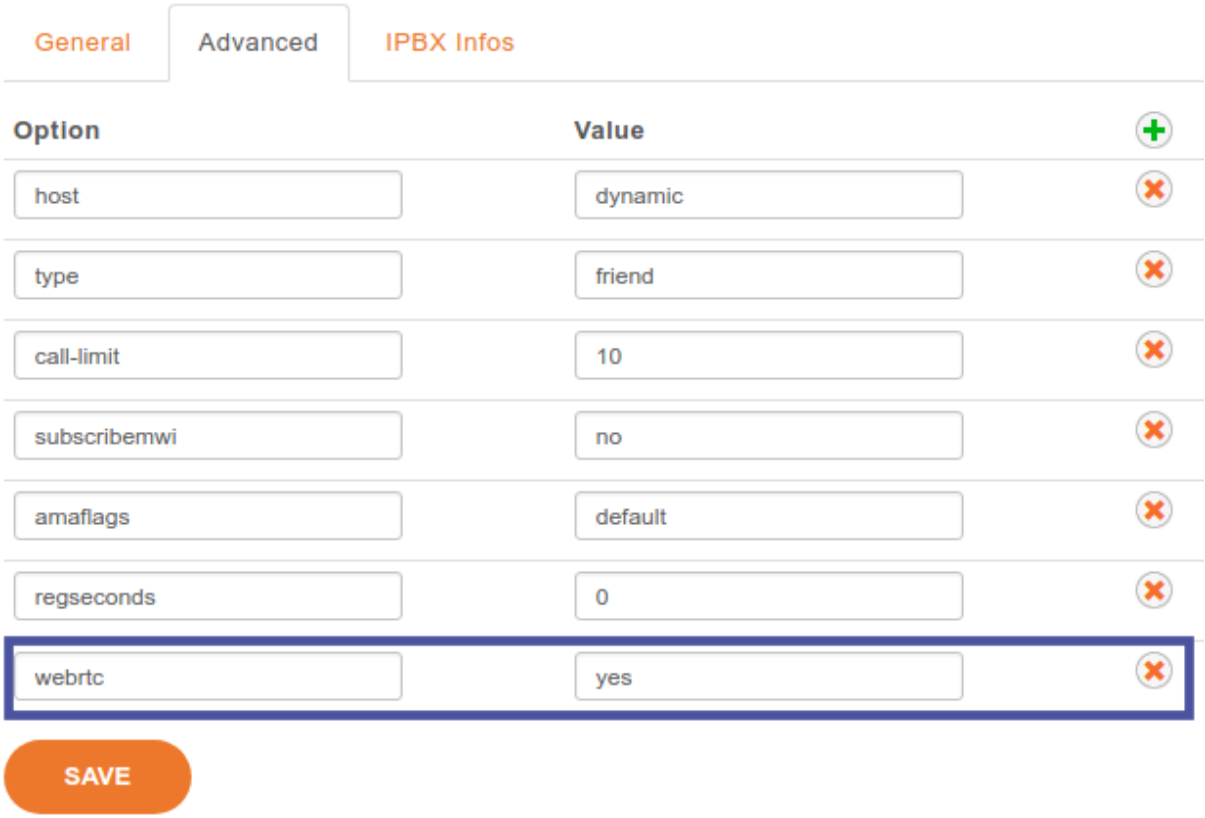
XiVO comes with a WebRTC lines support, you can use in with XiVO *UC Assistant* and *Desktop Assistant*. Before starting, please check the *Requirements*.

Former WebRTC implementation might require following configuration steps:

- configure *Asterisk HTTP server*,
- and create user *with one line configured for WebRTC*. To have user with both SIP and WebRTC line is not supported.

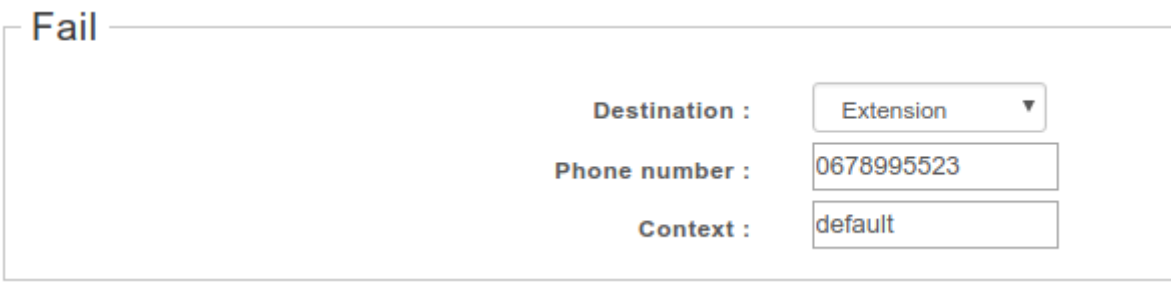
Configuration of user with WebRTC line

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:



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.



Experimental video call feature

Important: Removed since 2021.08

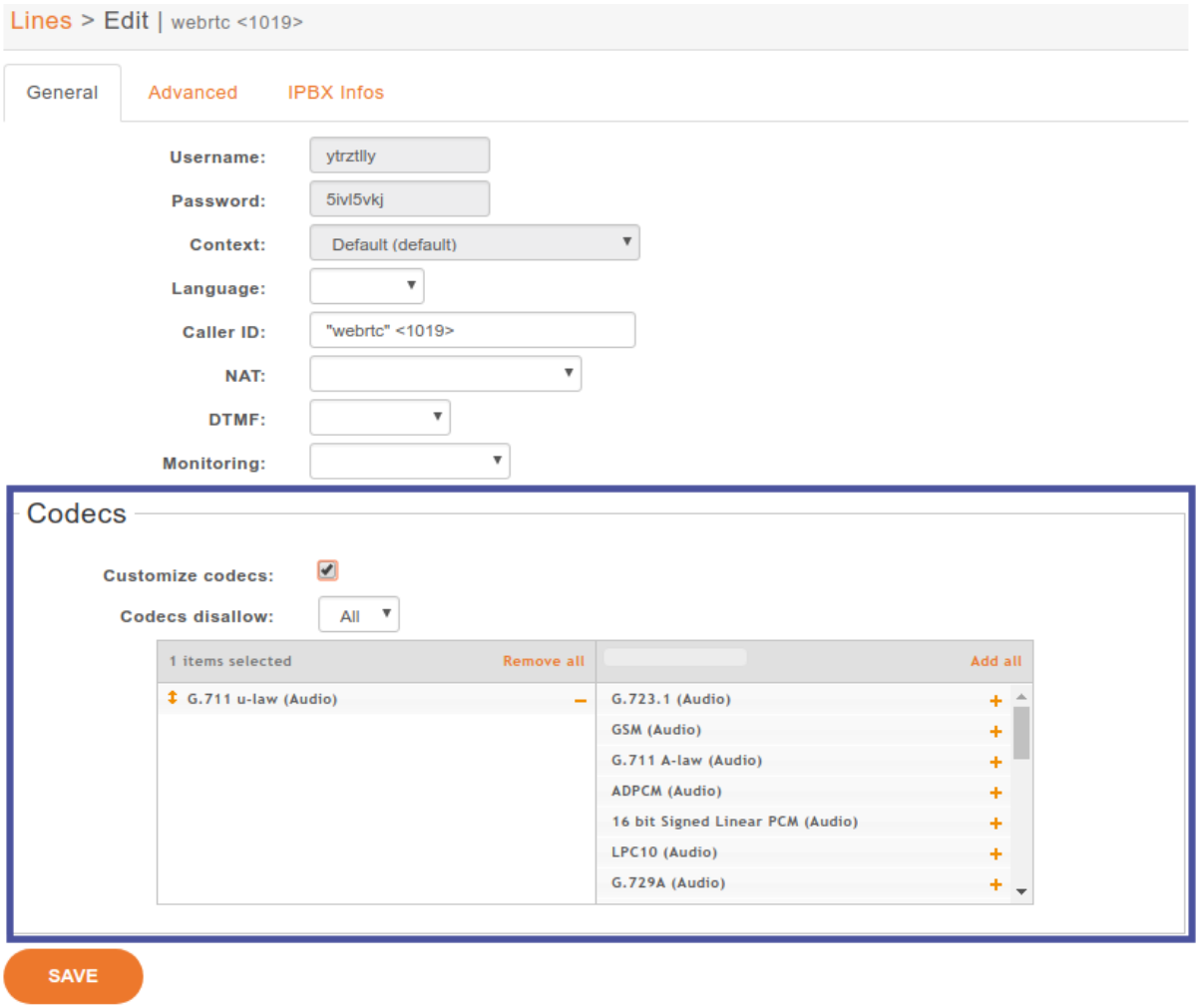
Manual configuration of user with WebRTC line

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
```

Updating LineConfig Sip response

Since Hellios 17, we added a new field *sipPort* inside LineConfig Sip response to avoid DNS requests.

Before :

```
{
    "msgType": "LineConfig",
    "ctiMessage": {
        "hasDevice": false,
        "id": "1",
        "isUa": false,
        "mobileApp": true,
        "name": "l1ze3zd8",
        "number": "4000",
        "password": "e2xyubvn",
        "sipProxyName": "default",
        "vendor": null,
        "webRtc": true,
        "xivoIp": "192.168.56.3"
    }
}
```

After :

```
{
    "msgType": "LineConfig",
    "ctiMessage": {
        "hasDevice": false,
        "id": "1",
        "isUa": false,
        "mobileApp": true,
        "name": "l1ze3zd8",
        "number": "4000",
        "password": "e2xyubvn",
        "sipProxyName": "default",
        "sipPort": "5060",
        "vendor": null,
        "webRtc": true,
        "xivoIp": "192.168.56.3"
    }
}
```

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

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:

- 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/*`?

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.

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.

xivo-confd

Warning: DEPRECATED

There is also a special case for authentication with xivo-confd. See *IPBX API* for more details.

4.1.3 XDS Administration

File Synchronization

When using XDS architecture one should activate the *File Synchronization* between *XiVO Main* and *MDS*.

Installation

1. One must initialize the configuration on *XiVO Main*: refer to *XDS File Synchronization* section in Installation page.

   Note: What does the initialization do ? The `xivo-xds-sync -i` is to be run on XiVO Main and will:

   1. generate a ssh key pair: `~/.ssh/rsync_xds` and `~/.ssh/rsync_xds.pub`
   2. copy the public key to `/usr/share/xivo-certs/`
   3. make the public key available at `https://XIVO_HOST/ssh-key/rsync_xds.pub`
   4. create cron job `/etc/cron.d/xivo-xds-sync` to schedule the synchronization

2. The configuration on MDS is automatic when installing a new MDS.

   Note: What does the MDS install script do ? When running the `mds_install.sh` script it will:

   1. get the *XiVO Main* pub key from `https://XIVO_HOST/ssh-key/rsync_xds.pub`
   2. and put it in the ssh *authorized_keys*

What is synchronized ?

Synchronized dirs

- /etc/asterisk/extensions_extra.d
- /var/lib/xivo/moh
- /var/lib/xivo/sounds
- /var/lib/xivo/sounds/recordings

Important: Specifically this `/var/lib/xivo/sounds/recordings` directory is synchronized **both ways**:

- from Main to each MDS
- and then from each MDS to Main

Warning: Because of the both ways sync, it means that the deletion of files in this folder won't be taken into account. Deleting a file will delete it on (for example) the Main, but during next sync, it will be retrieved back from the MDS. If you want to delete/clean this folder you currently need to:

1. Delete it on Main:
   ```
   rm /var/lib/xivo/sounds/recordings/*.wav
   ```
2. And then delete it also on **each** MDS:
   ```
   ssh -i /root/.ssh/rsync_xds root@MDS_IP "rm /var/lib/xivo/sounds/
   ↪recordings/*.wav"
   ```

Synchronization period

Every **15 min**

Exclusion

- Files in the above listed folders prefixed with `xds_override` **won't** be synchronized.

  Note: For example if you create on the *XiVO Main* the file `/etc/asterisk/extensions_extra.d/xds_override_mydialplan.conf`, it won't be synchronized to the other MDS.

- Files in `/etc/asterisk/extensions_extra.d/` are not overriden on MDS. It means that if you create on **MDS1** a file in `/etc/asterisk/extensions_extra.d/` it will remain on MDS1 (as long as the filename is not the same as one on the *XiVO Main*).

Destination

All files are synchronized to every configured MDS (MDS as defined in *Configuration → Management → Media Servers*).

Note: And therefore uses the MDS so-called VoIP IP address.

Reload

Dialplan (*dialplan reload*) and MOH (*moh reload*) are reloaded on each MDS after each synchronization.

Logging and Troubleshooting

All the script logs are logged in `/var/log/syslog`. All logs are prefixed with `xivo-xds-sync` prefix. You can run the following command to get the logs:

```
grep xivo-xds-sync /var/log/syslog
```

To see what the script is doing/will do you can run it in *dry* mode:

```
xivo-xds-sync -n
```

4.2 XiVOcc Administration

4.2.1 System Configuration

Docker 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
- `reload` - reload container (**only applicable to ``pgxivocc`` and ``nginx``**)

If you don't enter container name, the command applies on all containers. Use container names without the `xivocc_` prefix and `_1` suffix.

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               2019.10.00
xivocc_spagobi_1             2019.10.00
xivocc_xuc_1                 2019.10.00
xivocc_xucmgt_1              2019.10.01
xivocc_xivo_stats_1          2019.10.00
xivocc_recording_server_1    2019.10.00
xivocc_pack_reporting_1      2019.10.00
xivocc_pgxivocc_1
xivocc_recording_rsync_1
```

This only applies to the following images:

- nginx
- pack_reporting
- recording
- spagobi
- xivo_stats
- xuc
- xucmgt

Note: You can also use docker commands to display information about containers or images:

- 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_fingerboard_1         xivoxc/fingerboard:latest                         ␣
↪             Up 32 hours
xivocc_recording_rsync_1     xivoxc/recording-rsync: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:
├── purge-reporting-database.log
├── recording-server
│   ├── downloads.log
│   ├── recording-server.log
│   └── access.csv
├── spagobi
│   ├── Quartz.log
│   ├── SpagoBIBirtReportEngine.log
│   ├── SpagoBIChartEngine.log
│   ├── SpagoBIJasperReports.log
│   ├── SpagoBI.log
│   ├── SpagoBIQbeEngineAudit.log
│   ├── SpagoBIQbeEngine.log
│   └── SpagoBITalendEngine.log
├── specific-stats.log
├── xivo-full-stats
│   └── xivo-full-stats.log
├── xuc
│   ├── xuc_ami.log
│   └── xuc.log
└── xucmgt
    └── xucmgt.log
```

Backup

Database 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

pg_dump -h localhost -p 5443 -U postgres --format=c -f spagobi_dump spagobi
pg_dump -h localhost -p 5443 -U postgres --format=c -f xivo_stats_dump xivo_stats
pg_dump -h localhost -p 5443 -U postgres --format=c -f recording_dump recording
```

Restore

Database Restore

You may restore a backup using a similar command (to be adapted)

```
#Restore

cd /var/backups/xivocc

pg_restore -j 10 -h localhost -p 5443 -c -U postgres -d xivo_stats xivo_stats_dump
pg_restore -j 10 -h localhost -p 5443 -c -U postgres -d spagobi spagobi_dump
pg_restore -j 10 -h localhost -p 5443 -c -U postgres -d recording recording_dump
```


- *Configuration Override*
  - *XucMgt Configuration Override*
  - *Xuc Server Configuration Override*
- *Disabling XiVO based authentication*
- *Nginx Customization*
  - *Extend the Configuration*
- *Nginx path distribution*
- *ACD Outgoing Calls For Call Blending*
  - *How Does it Work ?*
  - *Configuration Steps*
  - *How to check correct configuration*
- *Allowing xuc user to login to API or Applications*
- *Install trusted certificate for nginx*
  - *What is a complete certificate chain*
- *Mobile Assistant*
- *Additional Configuration*
- *AMI Keepalive*

Configuration Override

Our dockerized scala applications working with an `application.conf` file can have some of their variables overriden by providing the adequate configuration. You have to :

1. define a custom `.conf` file with the extra configuration you want
2. define a `.override.yml` in `/etc/docker/compose` file to mount the conf file within the docker container, having specifically:

1. the **CONFIG_FILE** environment variable with the location of the new conf file within the container
2. a volume mounting your local .conf file to the appropriate location in the container

XucMgt Configuration Override

1. Define or complete an .override.yml file like `/etc/docker/compose/50-xucmgt.override.yml`:

```
services:
  xucmgt:
    environment:
    - CONFIG_FILE=/opt/docker/conf/custom/xucmgt.conf
    volumes:
    - /etc/docker/xucmgt:/opt/docker/conf/custom
```

2. Create the mounted override conf file, `/etc/docker/xucmgt/xucmgt.conf` in the example, and add your custom configuration:

```
include "application.conf"

# Put here the custom configuration
```

Xuc Server Configuration Override

1. Define or complete an .override.yml file like `/etc/docker/compose/66-xuc.override.yml`:

```
services:
  xuc:
    environment:
    - CONFIG_FILE=/opt/docker/conf/custom/xuc.conf
    volumes:
    - /etc/docker/xuc:/opt/docker/conf/custom
```

2. Create the mounted override conf file, `/etc/docker/xuc/xuc.conf` in the example, and add your custom configuration:

```
include "application.conf"

# Put here the custom configuration
```

Disabling XiVO based authentication

If you configured another kind of authentication, like *Login Timeout Configuration* or *Kerberos Authentication* for example, you can enforce this authentication mechanism by preventing fall back on the XiVO authentication mechanism. In order to achieve that, you need to perform the following configuration.

1. Override the configuration as explained in *Configuration Override* for xuc
2. In the custom `application.conf` file created, add this:

```
include "application.conf"

authentication {
   xivo = ""
}
```

Recreate the containers according to the new configuration : `xivocc-dcomp up -d`

Nginx Customization

Extend the Configuration

(*Since Naos.03*)

It is possible to extend the Nginx configuration. Our Nginx is split under 3 server:

- one on port 80,
- one on port 443 with SSL,
- one on port 8443 with SSL

It is possible to extend each of these server section. It is mainly useful to add some location.

1. Create a directory on your UC server:

```
mkdir -p /etc/docker/nginx/extra-conf.d/{http,https,8443}/
```

2. Create an override file `/etc/docker/compose/15-nginx.override.yml` for Nginx with the following content:

```yaml
services:
  nginx:
    volumes:
    - /etc/docker/nginx/extra-conf.d/:/etc/nginx/templates/extra-conf.d/
```

3. Now add you can extend one of the server configuration section. For example:
   - create a file `/etc/docker/nginx/extra-conf.d/https/xivo-api.conf.template`
   - with the folowing content:

```
location /xivo-api {
  proxy_pass http://${NGINX_XIVO_HOST}/api;
}
```

   - this will:
     - add a location `/xivo-api` in the https section (so it will be available when accessing the UC Nginx on port 443)
     - this location is "proxy passed" to the XiVO `/api` location on XiVO

Nginx path distribution

XiVO CC services can be installed on separate servers, but they can be opened through nginx as if they were running on the same server:

Service	URL	Target URL (should not be used)
Finger-board	https://XUC_HOST/fingerb	http://XUC_HOST/
UC Assis-tant	https://XUC_HOST/	http://XUC_HOST/
CC Agent	https://XUC_HOST/ccagen	http://XUC_HOST:8070/ccagent
CC Man-ager	https://XUC_HOST/ccman	http://XUC_HOST:8070/ccmanager
Config Mgt	https://XUC_HOST/configr	http://XIVO_HOST:9100/configmgt
Kibana	https://XUC_HOST/kibana	http://XUC_HOST/kibana
Recording	https://XUC_HOST/recordi	http://RECORDING_SERVER_HOST:RECORDING_SERVER_PORT/recording
SpagoBI	https://XUC_HOST/SpagoI	http://REPORTING_HOST:9500/SpagoBI

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.

How Does it Work ?

1. Pre-condition:
   - create a queue whose name starts with **out**
   - agent uses CC Agent application
2. Then, as soon as an agent (using CC Agent application) dials a number with more digits than `xuc.outboundLength`, then
   - the call is routed to the agent **via** the outbound queue
   - therefore its queue statistics will be updated

Note: The `xuc.outboundLength` default value is 6. You can customize this value by creating a custom `application.conf` file and changing the `xuc.outboundLength` value (see *Configuration Override* for xuc).

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 holtime, 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 holtime, 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.

Allowing xuc user to login to API or Applications

By default, the 'xuc' user can only be used internaly by the xucserver to connect to the xivo. We strongly advise you shouldn't change this behavior but if you need to enable it, you can do it by setting the following variable in the `custom.env` file of your xivocc server.

```
AUTH_PREVENT_XUC_LOGIN=false
```

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`,
  - `xivoxc.key` while keeping filenames unchanged
- reload nginx container by command:

  ```
  xivocc-dcomp reload nginx
  ```

- you should then check that the certificate chain is complete (especially for the XiVO Mobile Assistant) - see *What is a complete certificate chain*.

What is a complete certificate chain

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`).

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 Assistant

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*.

Additional Configuration

If you want to expose custom variable by key for your config you can follow these steps:

1. Override the configuration as explained in *Configuration Override* for xucmgt
2. In this custom `application.conf`
   - add your custom variables:

```
include "application.conf"

client {
  custom_variable1 = "your_custom_value_1"
  custom_variable2 = "your_custom_value_2"
  # Add more custom variables as needed
}
```

   - xucmgt supports nested path for you conf like :

```
include "application.conf"

client {
  custom_nested {
    variable1 = "your_custom_value_1"
    variable2 = "your_custom_value_2"
  }
  custom_variable3 = "your_custom_value_3"
  # Add more custom variables as needed
}
```

You can acces this value with curl request like this :

```
curl https://XIVOCC_IP/conf/custom_nested/variable1
```

AMI Keepalive

The Xuc server monitors the Asterisk AMI connection using periodic pings. These pings ensure regular traffic to keep the session active and help to detect connection instabilities.

The default ping interval is 10 seconds. You can adjust this interval by adding the following variable to the `custom.env` file of your XiVOCC server:

```
XIVO_AMI_KEEPALIVE=60 # Set the KeepAlive interval to 60 seconds
```

By default, missed pings are logged in the Xuc logs. You can also configure the AMI connection to restart automatically if 3 consecutive pings are missed. To enable this, add the following variable to the `custom.env` file:

```
AMI_RECONNECT_ON_MISSED_PINGS=true # Enable automatic AMI connection restart on 3␣
↪missed pings
```

4.2.2 XiVOcc Applications Configuration

This section covers specific configuration parameters for the different application of *XiVO CC*.

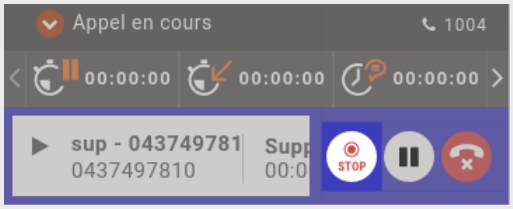
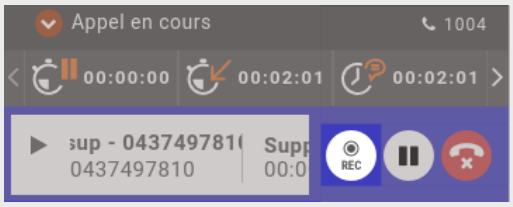
CC Agent configuration

Contents

- *CC Agent configuration*
  - *Recording*
  - *Activity (Queue) control*
  - *Activity's Failed Destination Configuration*
    - *1. Import sound files in the* XiVO PBX
    - *2. Configure the default queue for dissuasion in the* XiVO PBX
    - *3. Give the rights to the XiVO User*
  - *On hold notification*
  - *Pause Cause and Status*
  - *Screen Popup*
  - *Run executable*
  - *Login or Pause management using function keys*
    - *Behavior*
    - *Configuration*
  - *Listened advertisement using function keys*
    - *Behavior*
    - *Configuration*

Recording

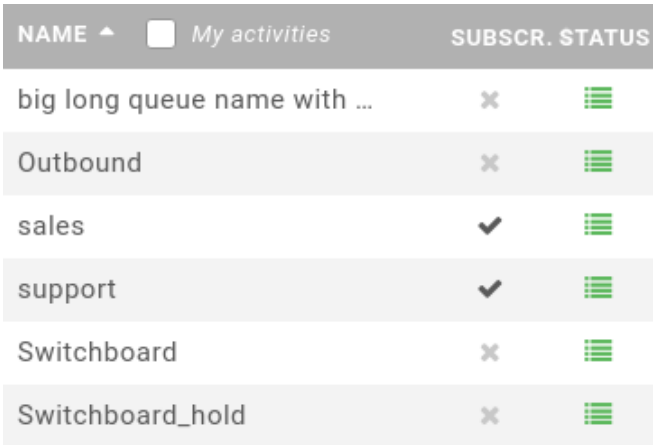
Recording can be paused or started by an agent.

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.



Activity's Failed Destination Configuration

An agent can see and change the sound file played to the caller when the Activity is temporarily closed[1].

For this to work the XiVO Administrator needs to:

1. *import sound files in the XiVO PBX with the queue name prefix*,
2. *configure the default queue*,
3. *give the rights to the XiVO User*.

1. Import sound files in the *XiVO PBX*

To see the sound files, a XiVO Administrator must import them with the queue name prefix as described below:

- Given the queue *Sales* with its *Name* being `sales`,
- Given I want to have two messages available for this queue:
  - `closed_during_afternoon` (*'We are closed this afternoon ...'*)
  - `in_a_meeting` (*'Currently in a meeting ...'*)
- Then, as a XiVO Administrator I **MUST** import these sound files **with the prefix** `sales_`. For each file, do:
  - Go to *Services* → *IPBX* → *IPBX Services* → *Audio files*
  - Click to **Add** a file
    - *File name*: select the file on your disk
    - *Directory*: select **playback** directory (this is mandatory)
  - When file is added, edit it and *prefix the file name* **with the queue name followed by an underscore**, in our exampe `sales_`

Said differently the agent will be able to see and set all sound files prefixed with the queue name.

[1] Temporarily closed, here, means when no agent is logged in. More precisely when the call is sent to the *Fail* destination (see *Services* → *Call Center* → *Queues* in queue configuration, tab *No Answer* and *Fail* section). A call is sent to the *Fail* destination according to the *Join an empty queue* and *Remove callers if there are no agents* parameters in the *Advanced* tab of the queue.

2. Configure the default queue for dissuasion in the *XiVO PBX*

You can configure a default queue for dissuasion by setting the queue name in `defaultQueueDissuasion` in application.conf. This option once set, will show the queue in the dissuasion dropdown. You can also use `QUEUEDISSUASION_DEFAULT_QUEUE` environment variable in `/etc/docker/xivo/custom.env` file on the XiVO PBX and restart the services with `xivo-dcomp up -d`.

3. Give the rights to the XiVO User

To see/change the dissuasion destination via the CC Agent or CC Manager, a XiVO User must have a supervisor right with *Update dissuasion* access. To do this a XiVO Administrator must connect to the configmgt (https://XIVO_IP/configmgt) and configure the user with:

- *Profile*: Supervisor
- *Access*: check the box *Update dissuasions*
- *Queues*: select the queues the user can see

See *Profile Management*.

On hold notification

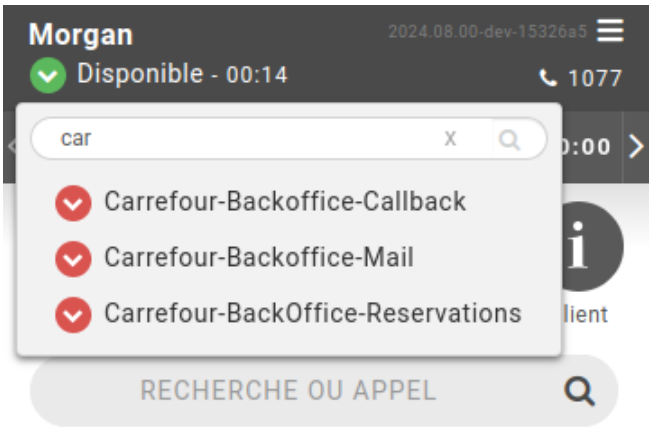
You can configure `notifyOnHold` (in seconds)) in application.conf, this option once set, will trigger a popup and a system notification to user if he has a call on hold for a long time. You can also use `NOTIFY_ON_HOLD` environment variable in `/etc/docker/compose/custom.env` file.

Pause Cause and Status

Custom pause status can be configured in XiVO.

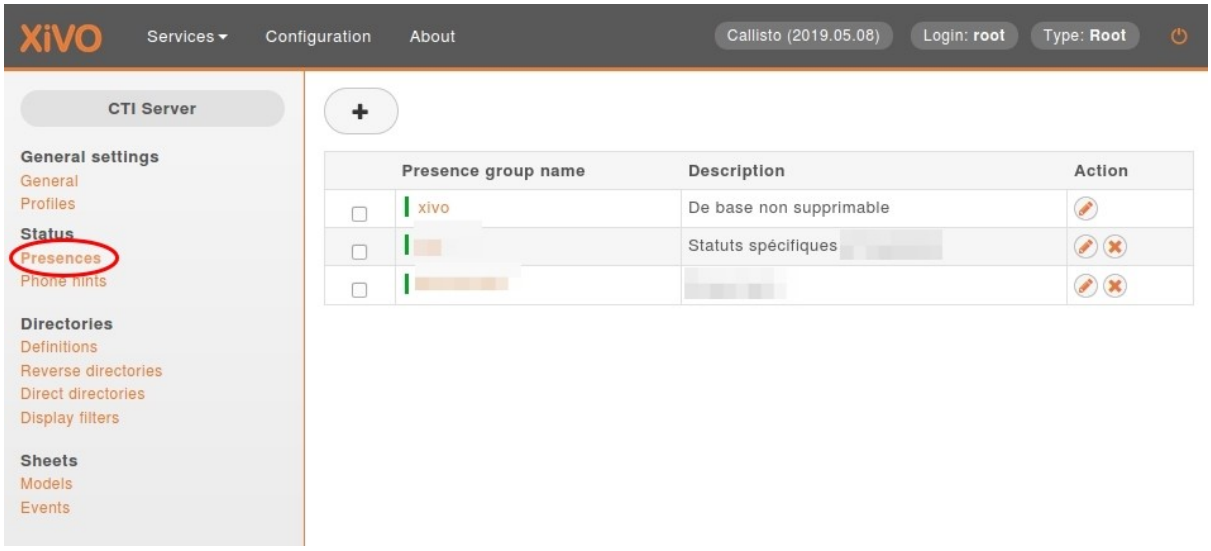
With this feature the agent can specify his pause type ("Out to lunch", "Backoffice - Mail" ...) This pause type will be seen:

- by the other agents,
- and also the manager in the CC Manager application



To customize the pause status you need to:

- go in *Services -> CTI Server -> Status -> Presences*.
- then go in the **xivo** presence group



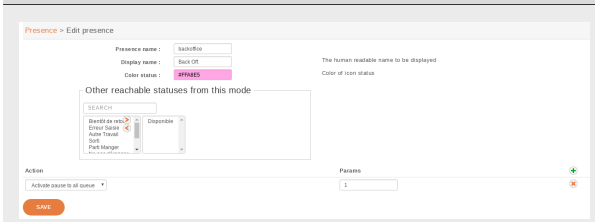
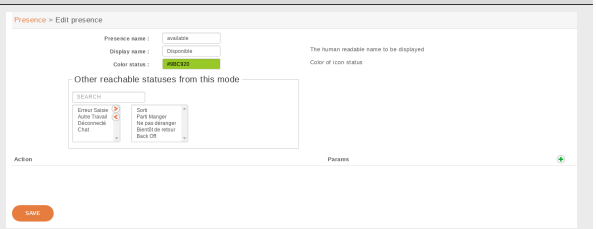
Note that when you open the pause menu you can also use the search bar to filter out pause statuses.

Inside the presence group you need to add a new presence with:

- a name
- a display name
- color and other status available are currently not used
- and in *Action* you need to add the action **Activate pause to all queue** with the value **true**

Note: **Optional** You can edit *ready* presence defined with an action **Disable pause to all queue** to fallback to be available whenever the agent reconnects back to CCAgent. For example, you closed the webpage, or did a F5 refresh. Once you reconnect you will be put in *ready* state. By default, if you don't add option to *ready* state, you will reconnect with last presence state known.

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.

Presence from ready to pause	Presence from pause to ready
	

Note: Presence color are not taken into account in the CC Agent (or CC Manager) application. They all will be displayed in *Red*.

Screen Popup

This section has been moved to *configuration*

Run executable

This section has been moved to *configuration*

Login or 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

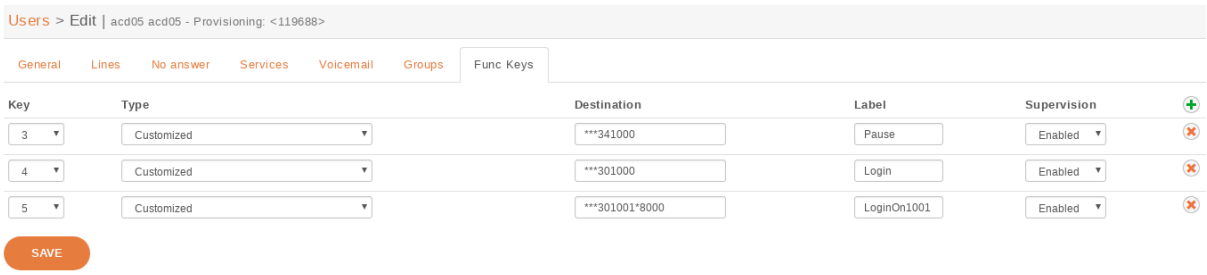
There are two types of customizable *function keys* that can be used

- 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. On XiVO CC (XuC) check that the *XiVO PBX* IP is in the list of host allowed to use the *Deprecated APIs* (see `DEPRECATED_API_HOST`)
3. 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`



Listened advertisement using function keys

You can configure a *Listened* key on agent phone to warn him when his conversation is listened by a supervisor.

Behavior

- **Listen key**:
  - LED of phone key will be lit if supervisor starts to listen to agent's conversation (see *Agents actions*)

Configuration

To configure add a customizable *function key*:

- Listened: the phone key will be lit when the agent's conversation is listened to by the supervisor. Configuration pattern (see also below):
  - `***35<PHONE NUMBER>` : it will toggle listened status for agent loged on phone `<PHONE NUMBER>`

1. Configure function key on user (example with 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*: `***351000`,
     - *Label*: `Listened`,
     - *Supervision*: `Enabled`
2. Activate option for xucserver: see *Warn agent when spied* section.

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.

Warn agent when spied

By default when a supervisor spies on an agent - see *Agents actions* - the agent is only warned via its *CC Agent Call control* listen icon. You can change the behavior to warn the agent when spied swith a small beep and/or a function key lit on its deskphone.

To do this, set the `ENABLE_CHANSPY_BEEP` environment variable to `true` in the `/etc/docker/compose/custom.env` file:

```
ENABLE_CHANSPY_BEEP=true
```

See also *Listened advertisement using function keys*.

Configure queue statistics displayed

You can change the statistics displayed in the set of squares. Note that any configuration will be applied to all queues, those statistics cannot be configured per queue.

To configure it, those keys are to be set in the custom env of your xivocc (you are not forced to set them all) :

- CCM_QINDICATOR_TL for the top left square
- CCM_QINDICATOR_TR for the top right square
- CCM_QINDICATOR_BL for the bottom left square
- CCM_QINDICATOR_BR for the bottom right square

The available statistics are documented in the API and SDK part of the documentation - see *Real time calculated Queue statistic* and *Other queue statistics*

Example of custom.env configuration :

```
...
CCM_QINDICATOR_TL="TotalNumberCallsAnswered"
CCM_QINDICATOR_TR="TotalNumberCallsAbandonned"
CCM_QINDICATOR_BL="AvailableAgents"
CCM_QINDICATOR_BR="WaitingCalls"
...
```

Recording switch for queues

> **Warning:** The recording switch is currently not compatible with the percentage of recording feature (legal obligation, introduced in pollux).

By default, the button for enabling/disabling queue recording from the CCManager interface is disabled.

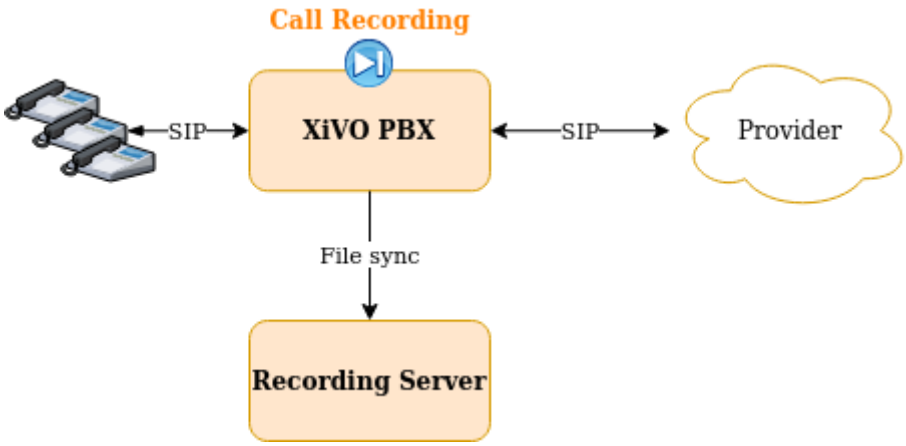
Setting the `SHOW_RECORDING_SWITCH` environment variable in `/etc/docker/compose/custom.env` will retrieve the switch button to enable / disable recording for all queues.

```
SHOW_RECORDING_SWITCH=true
```

Then you need to recreate the xucmgt container with `xivocc-dcomp up -d`.

Recording

This page describes how to configure the recording feature on the XiVO PBX.



- *Recording Configuration*
  - *1. Add link towards Recording Server*
  - *2. Enable recording*
    - *Enable recording in the Queue configuration*
    - *Enable recording via subroutines*
- *Recording Features*
  - *Automatic Stop/Start Recording On Queues*
  - *Stop recording upon external transfer*
  - *Manual purge of recording*
  - *Recording filtering configuration*

Important: Recording can also be activated on the *XiVO Gateway*. For this, you need to follow the *Recording on Gateway* guide. **Beware that you cannot use both.**

Recording Configuration

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 <RECORDING_SERVER_HOST> <IPBX_NAME>
```

The parameters are:

Note: if one parameter is missing the script will ask them.

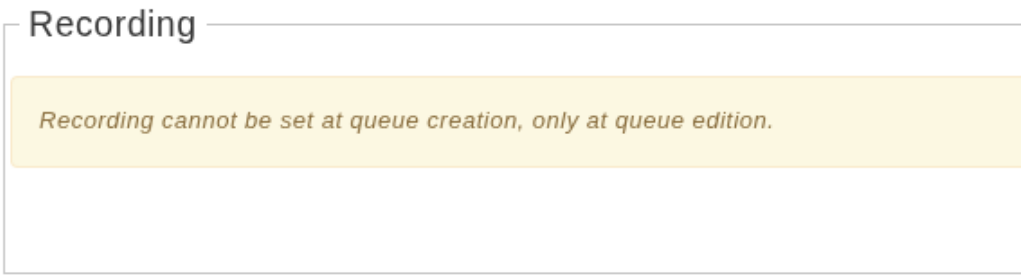
- the Recording Server FQDN or IP (e.g. 192.168.0.2)
- the *XiVO PBX* name (any space or dash (-) character will be removed)

  Important: When configuring more than one *XiVO PBX* on the same Recording Server, give a different name to each of them.

After having configured the recording, you have to enable it in the Queue's configuration or via sub-routines. See below.

2. Enable recording

Warning: At this time recording cannot be set at queue creation, only at queue edition.



Enable recording in the Queue configuration

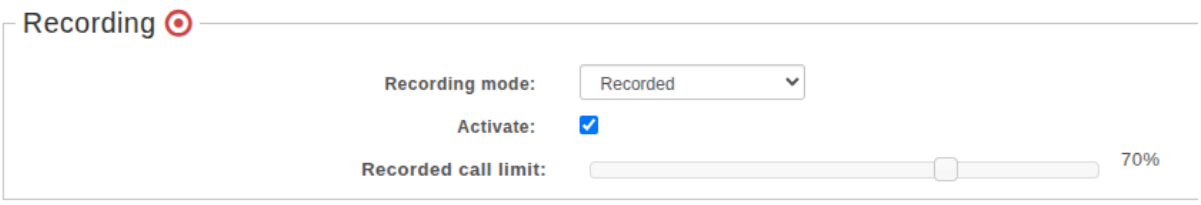
Note: Steps to be done on **XiVO PBX**

To enable recording on a queue, go to *Services* → *Contact Center* → *Queues* and edit the queue. Then, in the recording section, select your recording mode:

- *Recorded*:

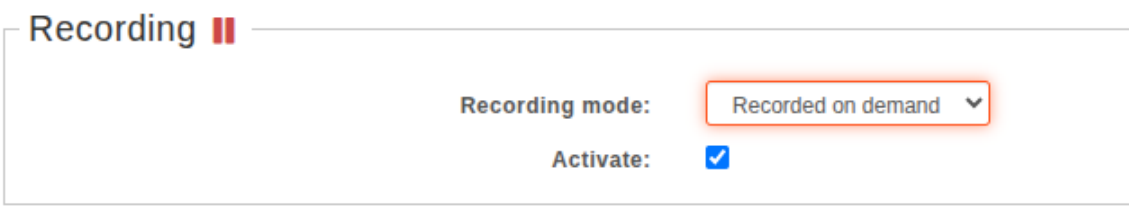
  The calls will start recorded.

  To enhance GDPR compliance, recorded queues can be set to avoid recording more than a specific percentage of calls. If recording a call would go beyond this limit, the call will start its recording paused (the agent can enable it manually if it's important). Set the slider to 100% to record every calls.

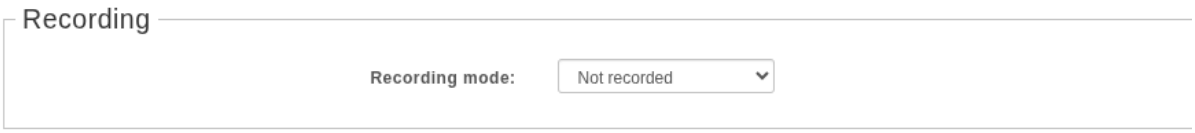


- *Recorded on demand*: The call always starts with its recording in paused state and can be started by the agent (see *agent recording configuration*)

  Activate toggles this mode on and off (allowing to move to *Not recorded* temporarily).



- *Not recorded*: no call will be recorded and the recording option wont be visible for the agent.



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-recordi`	Records incoming calls
`xivocc-incall-recordi`	Records incoming calls, but record starts in paused state and can be activated by the agent (see *agent recording configuration*)
`xivocc-outcall-record`	Records outgoing calls
`xivocc-outcall-record`	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`

Recording Features

Automatic Stop/Start Recording On Queues

By default recording is stopped when a call is transferred to a queues in *Not Recorded* mode (see the feature description: *Automatic Stop/Start Recording On Queues*).

This can be deactivated by adding `ENABLE_RECORDING_RULES` environment variable to the xuc section of your `docker-xivocc.yml` file:

```
xuc:
  image: ...

  environment:
  - ...
  - SECURED_KRB5_PRINCIPAL
  - ENABLE_RECORDING_RULES
```

and `ENABLE_RECORDING_RULES=false` value to your `custom.env`.

```
XIVO_HOST=192.168.1.1
XUC_HOST=192.168.1.2
XUC_PORT=8090
...
ENABLE_RECORDING_RULES=false
```

and then relaunch the xivocc services with `xivocc-dcomp up -d` command.

Stop recording upon external transfer

By default recording is stopped when both parties of the call are external (see the feature description: *Automatic Stop Recording Upon External Transfer*).

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.

Manual purge of recording

Recording files are automatically deleted after the configured time (set during installation, see *Installation*).

However you can mark a recording to be deleted before the configured expiration time by attaching a specific data to the call you want to remove the recording. To do this, use the *Attach call data* API with the following body:

- key: `xivo_recording_expiration`
- value: a datetime formatted as `yyyy-MM-dd HH:mm:ss`

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>/configmgt (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>/recording
   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:

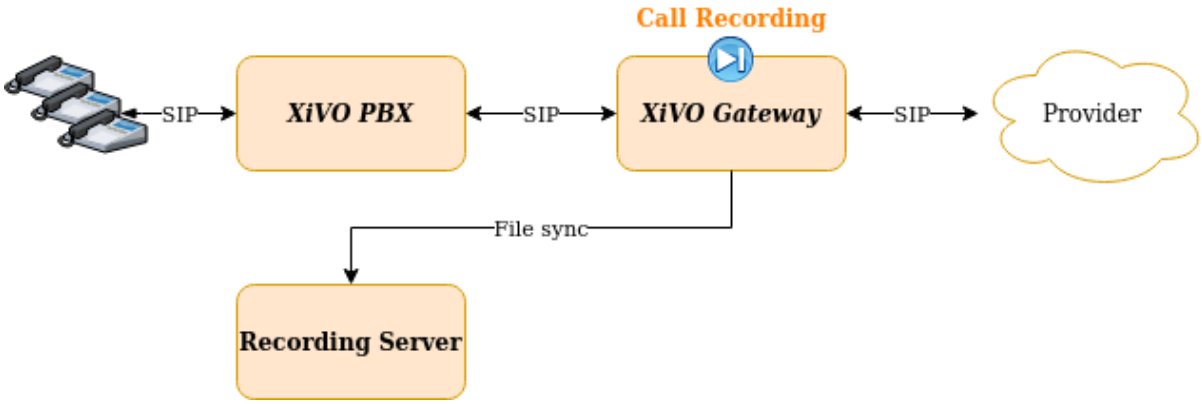
- XiVO Users internal numbers

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.

Recording on Gateway

Recording can be enabled on a gateway instead of the XiVO PBX where the users, queues and other objects are configured. This architecture will allow to off-load the recording process to a gateway server. Note that in this architecture not all the recording features are available - see *Recording On Gateway Features* section.



- *Activate Recording on Gateway*
  - *1. Add link towards Recording Server*
  - *2. Enable recording*
- *Activate Recording on XiVO PBX*
- *Recording On Gateway Features*
  - *Available Features*
  - *Unavailable Features*
    - *Activate the Recording Control on CC Agent*

Important: This page describes the case when recording is activated on the **XiVO Gateway**. If you want to configure it on the *XiVO PBX* you MUST follow the *Recording* guide. **Beware that you cannot use both.**

Activate Recording on Gateway

To configure recording there are two steps to follow on **XiVO Gateway**:

1. Add link towards Recording Server,
2. and then enable recording via subroutines

1. Add link towards Recording Server

Note: Steps to be done on **XiVO Gateway**

The first step is to configure the link towards the Recording Server by running the configuration script:

```
xivocc-recording-config <RECORDING_SERVER_HOST> <IPBX_NAME>
```

The parameters are:

Note: if one parameter is missing the script will ask them.

- the Recording Server FQDN or IP (e.g. 192.168.0.2)
- the *XiVO PBX* name (any space or dash (-) character will be removed)

  Important: When configuring more than one *XiVO PBX* on the same Recording Server, give a different name to each of them.

After having configured the recording, you have to enable via sub-routines. See below.

2. Enable recording

Note: Steps to be done on **XiVO Gateway**

You must activate the recording by adding subroutines:

- Create the subroutines in the file `/etc/asterisk/extensions_extra.d/gateway-recording.conf`:

```
[gateway-recording-outcall]
exten = s,1,NoOp(=== Recording calls from Gateway to XiVO PBX ===)
same = n,Set(gateway_name=xivogw1)
same = n,Set(_recordingid=${gateway_name}-${UNIQUEID})
same = n,Set(XIVO_CALLOPTIONS=${XIVO_CALLOPTIONS}b(gateway-recording-outcall-add-
↪header^s^1))
same = n,MixMonitor(/var/spool/xivocc-recording/audio/${recordingid}.wav,,/usr/
↪bin/xivocc-synchronize-file "^{MIXMONITOR_FILENAME}")
same = n,Return()

[gateway-recording-incall]
exten = s,1,NoOp(=== Recording calls from Gateway to Provider ===)
same = n,GoSub(xivo-generic-sip-get-header,s,1(X-Xivo-Recordingid,recordingid))
same = n,GotoIf($["${recordingid}" = ""]?norecord:)
same = n,MixMonitor(/var/spool/xivocc-recording/audio/${recordingid}.wav,,/usr/
```

(continues on next page)

(continued from previous page)

```
↪bin/xivocc-synchronize-file "^{MIXMONITOR_FILENAME}")
same = n(norecord),Return()

[gateway-recording-outcall-add-header]
exten = s,1,NoOp(Adding SIP header on outgoing channel)
same = n,GoSub(xivo-generic-sip-add-header,s,1(X-Xivo-Recordingid,${recordingid}
↪))
same = n,Return()
```

- Activate subroutines:
  - Configure the `gateway-recording-outcall` on the Outgoing call rule towards the XiVO PBX
  - Configure the `gateway-recording-incall` on the Outgoing call rule towards the Provider

Activate Recording on XiVO PBX

Note: Step to be done on the *XiVO PBX*

On the XiVO PBX you need to use specific subroutines instead of the default ones.

- Create the subroutines in the file `/etc/asterisk/extensions_extra.d/xivo-gateway-recording.conf`:

```
; This subroutine is to be used only when recording is set up on gateway
[xivocc-incall-recording-viagw]
exten = s,1,NoOp(=== Recording incoming calls (XiVO PBX) ===)
same = n,GoSub(xivo-generic-sip-get-header,s,1(X-Xivo-Recordingid,XIVO_
↪RECORDINGID))
same = n,Set(DO_RECORD=true)
same = n,System(test -f /usr/share/asterisk/agi-bin/xivocc-determinate-record-
↪incall)
same = n,GotoIf($["${SYSTEMSTATUS}" = "SUCCESS"]?:noagi)
same = n,AGI(xivocc-determinate-record-incall)
same = n(noagi),GotoIf($["${DO_RECORD}" = "true"]?:norecord)
same = n,CelGenUserEvent(ATTACHED_DATA,recording=${XIVO_RECORDINGID})
same = n(norecord),Return()

[xivocc-outcall-recording-viagw]
exten = s,1,NoOp(=== Outgoing Call Recording (XiVO PBX) ===)
same = n,Set(DO_RECORD=true)
same = n,System(test -f /usr/share/asterisk/agi-bin/xivocc-determinate-record-
↪outcall)
same = n,GotoIf($["${SYSTEMSTATUS}" = "SUCCESS"]?:noagi)
same = n,AGI(xivocc-determinate-record-outcall)
same = n(noagi),GotoIf($["${DO_RECORD}" = "true"]?:norecord)
same = n,Set(ipbx_name=xivo)
same = n,Set(_XIVO_RECORDINGID=${ipbx_name}-${UNIQUEID})
same = n,Set(XIVO_CALLOPTIONS=${XIVO_CALLOPTIONS}b(gateway-recording-outcall-add-
↪header^s^1))
same = n,CelGenUserEvent(ATTACHED_DATA,recording=${XIVO_RECORDINGID})
same = n(norecord),Return()

[gateway-recording-outcall-add-header]
exten = s,1,NoOp(Adding SIP header on outgoing channel)
```

(continues on next page)

(continued from previous page)

```
same = n,GoSub(xivo-generic-sip-add-header,s,1(X-Xivo-Recordingid,${XIVO_
↪RECORDINGID}))
same = n,Return()
```

- Activate the subroutines:
  - Configure the `xivocc-incall-recording-viagw` on the Incoming calls/Queues in your XiVO PBX
  - Configure the `xivocc-outcall-recording-viagw` on the Outgoing call rule towards the Provider

Recording On Gateway Features

Available Features

When you activate the recording on a XiVO Gateway all features of the recording server are available:

- *Search, Download, Listen*
- *Disk Space*
- *Access logs*
- *Recording filtering*

Unavailable Features

When you activate the recording on a XiVO Gateway these features **are not** available:

- *Automatic Stop/Start Recording On Queues*
- *Automatic Stop Recording Upon External Transfer*
- *Recording Control on CC Agent* *except if you follow the* *Activate the Recording Control on CC Agent* *procedure*

Activate the Recording Control on CC Agent

When the recording is activated on the XiVO Gateway you can activate the Recording Control feature by following this specific procedure :

Allow the xucserver to connect to your XiVO Gateways:

1. Copy the file `/etc/asterisk/manager.d/02-xivocc.conf` from the XiVO PBX to the same folder on the gateway.

Configure the XiVO Gateway as a Media Server

1. On your XiVO PBX, add your *XiVO Gateway* as a Media Server in *Configuration* → *Media servers*

   Important: After this step the xucserver will try to connect to the *XiVO Gateway* asterisk AMI

2. On your **XiVO PBX**, configure the provider trunks: you need to add on your *XiVO PBX* the **same** provider trunk that is already created on your *XiVO Gateway* BUT with selecting, for the field *Media Server* the *XiVO Gateway* you added.

Configure the xuc

1. Disable the feature Stop recording upon external transfer feature: see *Stop recording upon external transfer*
2. Disable the Automatic Stop/Start Recording On Queues feature : see *Automatic Stop/Start Recording On Queues*

3. Restart the xuc:

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

Web / Desktop Application

Contents

- *Web / Desktop Application*
  - *Common features*
    - *Disabling call group disconnection functionality*
    - *Disabling WebRTC*
    - *Call History Number of Days*
    - *Disabling chat in UC Assistant and Switchboard*
    - *Disabling mail in UC Assistant*
    - *Disabling download desktop applications*
    - *Change default fallback language*
    - *External Directory*
    - *Credentials validity period*
    - *Email Template*
    - *Visioconference Email Template*
    - *Screen Popup*
      - *Screen popup on UC Assistant*
  - *Desktop Assistant Specific Features*
    - *Run executable*

Common features

Disabling call group disconnection functionality

To prevent users from leaving a group in UC Assistant, set the `UC_SHOW_GROUP_CONTROL` environment variable to `false` in the `/etc/docker/compose/custom.env` file.

Disabling WebRTC

WebRTC can be disabled globally by setting the `DISABLE_WEBRTC` environment varibale to `true` in `/etc/docker/compose/custom.env` file.

Call History Number of Days

Call history by default shows the last 7 days. You can change it by setting the `CALL_HISTORY_NB_OF_DAYS` environment variable to a specific number of days in the `/etc/docker/compose/custom.env` file.

> **Warning:** Note that setting this to a large number of days may slow down the solution.

Disabling chat in UC Assistant and Switchboard

The chat feature can be disabled globally by setting the `DISABLE_CHAT` environment varibale to `true` in `/etc/docker/compose/custom.env` file.

Disabling mail in UC Assistant

The mail action can be disabled globally by setting the `DISABLE_MAIL` environment varibale to `true` in `/etc/docker/compose/custom.env` file.

Disabling download desktop applications

To disable download buttons from the login page, you need to set `SHOW_APP_DOWNLOAD` to `false` in XiVO CC `/etc/docker/compose/custom.env` file.

Change default fallback language

Currently all web applications are translated in French if the locale of the browser (or the OS lang for desktop application) is not found. You can change, for example to fallback to english if the locale is not known by XiVO.

To do so, you need to set `APP_LANG_FALLBACK` in XiVO CC `/etc/docker/compose/custom.env` file with one of the following value:

- `fr` : French (default)
- `en` : English
- `de` : German

External Directory

To enable External Directory feature you need to configure it in the `/etc/docker/compose/custom.env` file:

```
EXTERNAL_VIEW_URL=https://myxivocc/externaldir
```

> **Note:** When `EXTERNAL_VIEW_URL` is set, it will be displayed in both UC Assistant and CC Agent

> **Warning:** Take care of the following restrictions:
>
> - The `EXTERNAL_VIEW_URL` must be seen as hosted by XiVO CC platform (otherwise it won't open because of CORS restriction).

- You MUST use the same HTTP protocol to access the CC application (UC Assistant or CC Agent) AND to access the external view. For example, if you access the application over HTTPS, you must access the external view over HTTPS too (to the avoid *Mixed Content* errors).
- The external URL *MUST NOT* have the 'X-Frame-Options' to 'sameorigin', else the feature will not work (e.g. you can't use google.com as directory...).

Credentials validity period

Warning: For security reasons, we strongly recommend **to not change** these options (i.e. to not **raise** the credential lifetime). This warning applies even more if you are connecting UC application through the *XiVO Edge* solutions (i.e. when the UC applications are opened on the Internet).

The default validity period for authentification credentials is one hour for a CTI user and one day for a web service user. This credential is automatically renewed while using the assistants. This is also the period during which you do not have to enter your password when reconnecting to the assistants.

This duration in seconds can be configured in the `/etc/docker/compose/custom.env` file:

```
AUTH_CTI_EXPIRES=3600
AUTH_WEBSERVICE_EXPIRES=86400
```

Email Template

To pre-fill email when clicking on the email button of XiVO users search results (see UC Assistant *Search* section) you need to create the `/etc/docker/xucmgt/email.tpl` file on XiVO CC server and add it as a volume in the yml file:

```yaml
services:
  xucmgt:
    volumes:
    - /etc/docker/xucmgt/email.tpl:/opt/docker/conf/email.tpl
```

This template must be in a *Json* valid format and **must contain** the following fields:

field	Description
subject	Pre-filled text for email subject
body	Pre-filled text for email body

The template text can contain parameters that will be automatically replaced by actual value when clicking on email link. Format to include in text is `{parameter}`. If the replacement is not possible the parameter will have to be changed by hand.

Parameter	Description
callernum	The ongoing call caller phone number
dstname	The full name of search result associated to this email

Here is a sample of a valid email template:

```
{
  "subject": "Information about missed call",
  "body": "Hello {dstname}, someone tried to reach you. He wants to be called back on
↪{callernum}."
}
```

Warning: Take care of the following limitations:

- Email display must be configured for directory search (see *Directories* and *Personal contacts*)
- This features relies on the html *mailto:* tag therefore:
  - an email client must be configured for the email to be sent
  - beware of the URI maximum length for browser: the *subject* and *body* will be appended in a *mailto:* tag to the email address. Beware that there may be some limitations on the URI size (2000 characters might be good maximum as you can see in this stackoverflow post)
- Template will apply to all users of the system on all applications (UC Assistant, CC Agent, Switchboard)
- No carriage return can be put in the text template (email body will be only on 1 line)

Visioconference Email Template

To pre-fill email when clicking on the email button of a visioconference search results, you need to create the `/etc/docker/xucmgt/emailInvite.tpl` file on XiVO CC server and add it as a volume in the yml file:

```
services:
  xucmgt:
    volumes:
    - /etc/docker/xucmgt/emailInvite.tpl:/opt/docker/conf/emailInvite.tpl
```

This template must be in a *Json* valid format and **must contain** the following fields:

field	Description
recipient	Pre-filled text for email to be sent to
subject	Pre-filled text for email subject
body	Pre-filled text for email body

The template text can contain parameters that will be automatically replaced by actual value when clicking on email link. Format to include in text is `{parameter}`. If the replacement is not possible the parameter will have to be changed by hand.

Parameter	Description
meetingroom_name	The display name of the meetingroom
meetingroom_link	The link of the meetingroom for external participants
meetingroom_number	The meetingroom phone number for internal users

You can also use "user data" parameters to enrich the email template ; it will be resolved if you have an ongoing call when clicking the email button.

Here is an example:

- Given the dialplan sets the "user data" DATA1 with email and DATA2 with case id:

```
same = n,Set(USR_TITLE1=Email)
same = n,Set(USR_DATA1=user@email.com)
same = n,Set(USR_TITLE2=Case ID)
same = n,Set(USR_DATA2=11223344)
```

- Then I can use the following template so that it is used if a call is ongoing with this caller:

```
{
  "recipient": "{attached_data_usr_data1}",
  "subject": "Invite to {meetingroom_name}",
  "body": "Hello, you can join me here {meetingroom_link} or at number
↪{meetingroom_number} to discuss case {attached_data_data2}"
}
```

Note that the keys retrieved from user data will be matched in lowercase with the template. If the replacement is not possible in this case, the parameter will be removed automatically. This allows non-mandatory user data parameters.

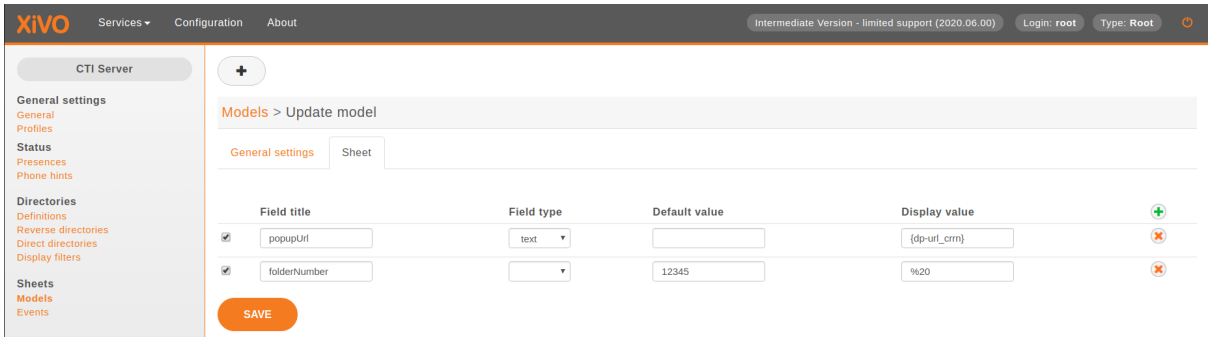
Screen Popup

It is possible to display customer information in an external web application using Xivo *sheet* mecanism.

- Go to *Services* > *CTI Server* > *Sheets* > *Models* to configure a sheet:
  - 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 / symbol or 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.
- Then go to *Services* > *CTI Server* > *Sheets* > *Events* and 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



Screen popup on UC Assistant

By default the sheet:

- on *CC Agent* application the sheet is opened by default,
- on *UC Assistant* application the sheet is **not** opened by default.

You can change the behavior with the following sheet variables:

- `popupUCActivated`:
  - if set to `true` the sheet will be opened on *UC Assistant* application
  - if set to `false` (default) the sheet won't be opened on *UC Assistant* application
- `popupAgentActivated`:
  - if set to `true` (default) the sheet will be opened on *CC Agent* application
  - if set to `false` the sheet won't be opened on *CC Agent* application

For example, if you want the sheet to **only** open on UC Assistant application you should add in your sheet configuration:

- in *Services* > *CTI Server* > *Sheets* > *Models*:
  - Tab *Sheet* add the following definition:
    - `popupAgentActivated`
      - field type = `text`
      - display value = `false`
    - `popupUCActivated`
      - field type = `text`
      - display value = `true`

Note: These variables can also be filled via a dialplan variable value with the `UserEvent` application and the `{dp-...}` syntax mechanism. See the XiVO PBX *sheet description*.

Desktop Assistant Specific Features

Run executable

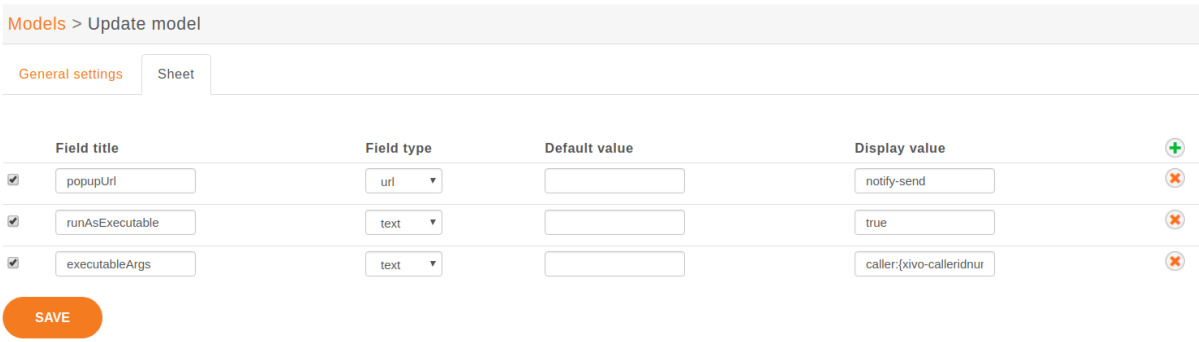
It is also possible to run an executable using Xivo *sheet* mecanism. This is only available in the desktop agent and desktop assistant.

> **Warning:** For the executable to be run on a Desktop Assistant in **UC mode**, you need to activate the *Screen popup on UC Assistant* (in **CC Agent mode** the screen popup doesn't need to be activated and therefore the executable will be run out-of-the-box).

- Go to *Services* > *CTI Server* > *Sheets* > *Models*:
  - Tab *General Settings*: Give a name
  - Tab *Sheet*: You must define a sheet with at least `runAsExecutable` and `popupUrl` fields set:
    - `popupUrl` (MANDATORY)
      - field type = `text`
      - It should contain an executable name accessible by the client user (where the desktop application is) or a full executable path.
    - `runAsExecutable` (MANDATORY)
      - field type = `text`
      - Display value `true`
    - `executableArgs` (OPTIONAL)
      - field type = `text`
      - set the argument for the executable.
- Then go to *Services* > *CTI Server* > *Sheets* > *Events* and choose the right events for starting the application.

Example : Run the `notify-send` command on linux:

- Define `popupUrl` with a display value of `notify-send`
- Define `runAsExecutable` with a display value of `true`
- Define `executableArgs` with a display value of `caller:{xivo-calleridnum}` where the variable `xivo-calleridnum` will be replaced by the caller phone number.



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 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

Agents

> *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.

- *Creating agents*
  - *Agent with external line*
    - *Creating agents with external line*
    - *Usage*
- *Managing agents in groups*
- *Assigning agents to queues*
- *Global settings for all agents*

Creating agents

- Create a user with a SIP line and a provisioned device
- Select agent group in *Services* → *Call Center* → *Agents*
- *Add an agent* from the dropdown menu

General tab:

- *Context* must be the same as the associated user's line context. If you change context of an existing agent, the agent must relog to apply the change.

Users tab:

- Associate the agent with the user (with SIP line and a provisioned device)

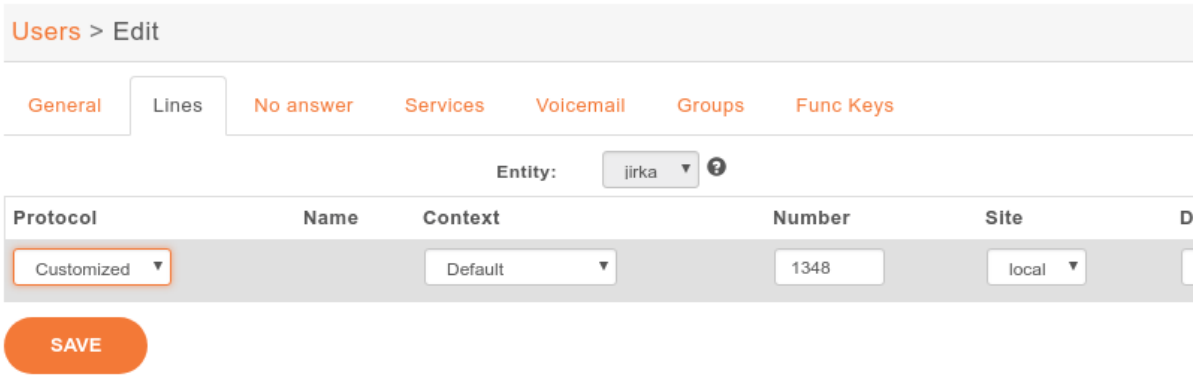
Agent with external line

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 *0xxxxxxxxx@default*.

Usage

The 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.

Managing agents in groups

Every agent must be in exactly one group. Groups can be used to:

- Assign group of agents to a queue in *CCmanager* Group View
- Filter agents by group in *CCmanager* Agent View

Groups can be created from *Services* → *Call Center* → *Agents* → *Add a group*. There is a limit of 120 groups.

Assigning agents to queues

There are three ways to assign agents to queues:

- Assign agents one by one in *CCmanager* Global View
- Assign one agent to any selection of queues: *Services* → *Call Center* → *Agents* → *Edit* → *Queues*
- Assign any selection of agents to a queue: *Services* → *Call Center* → *Queues* → *Edit* → *Members*

Global settings for all agents

This option can be set in *Services* → *IPBX* → *General settings* → *Advanced* → *Agent*:

- Autodisconnect if channel unavailable

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.

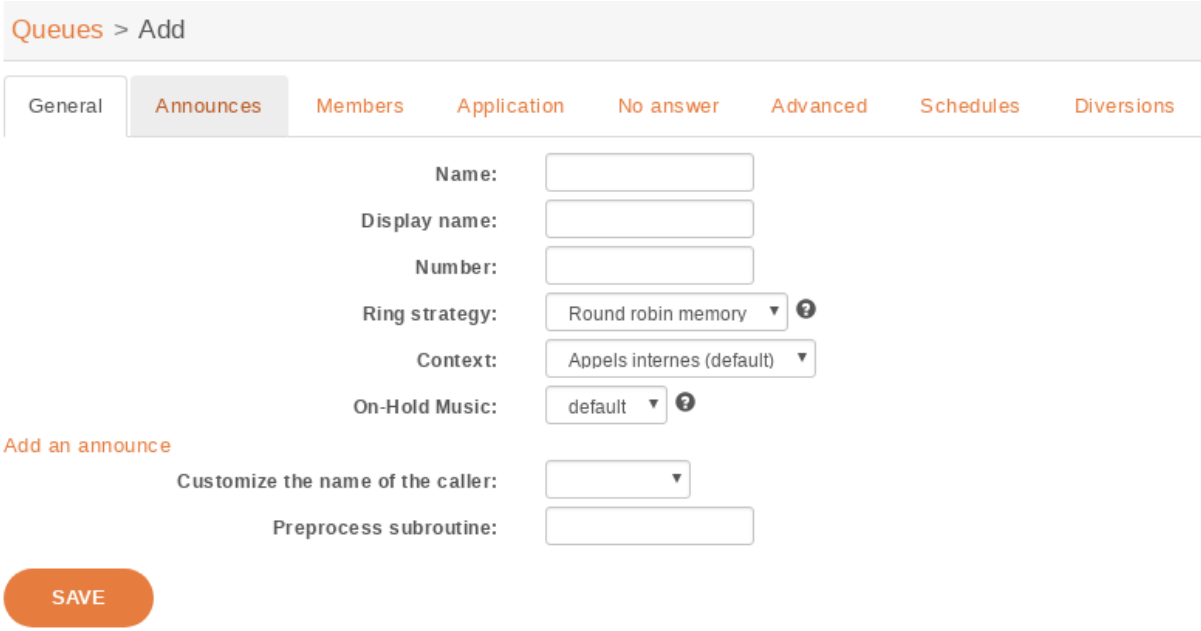


Fig. 58: *Services* → *Call Center* → *Queues* → *Add*

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
- 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.

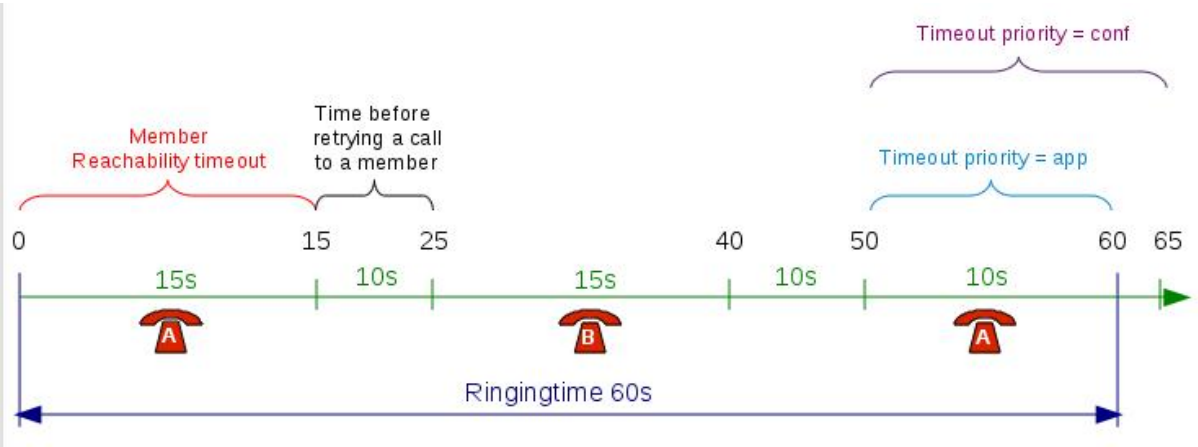
In order to have a deterministic behavior, you MUST *configure* different weight on each queues.

Note: When creating a new queue, this queue will not appear immediately in *CCAgent* and *CCManager* until the agent or the manager is not relogged to these applications accordingly.

Timers

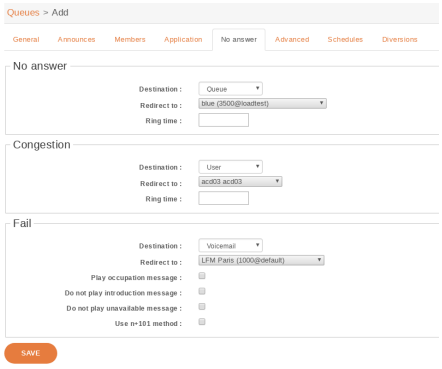
You may control how long a call will stay in a queue using different timers:

- 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".



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)

Advanced

- **Weight: Give the queue a priority to others queues (if agents belong to two or more queues). Check *weight warning* for explanation.**

> **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.

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.

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:

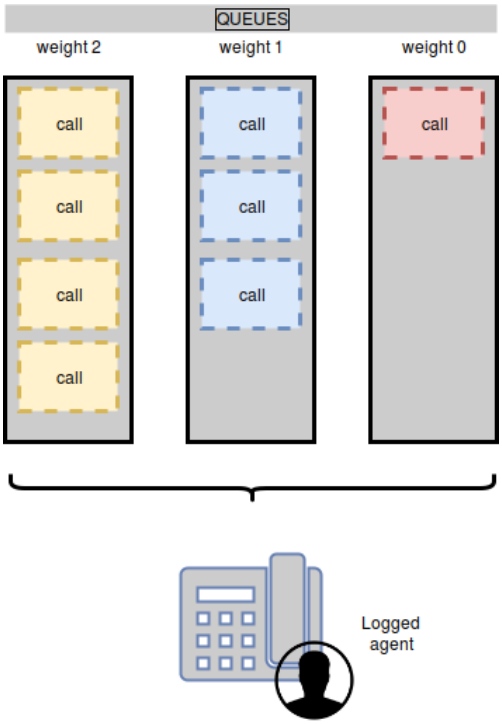
3 items selected	Remove all		Add all
unavailable	–	paused	+
invalid	–	penalty	+
unknown	–	inuse	+
		ringing	+
		wrapup	+

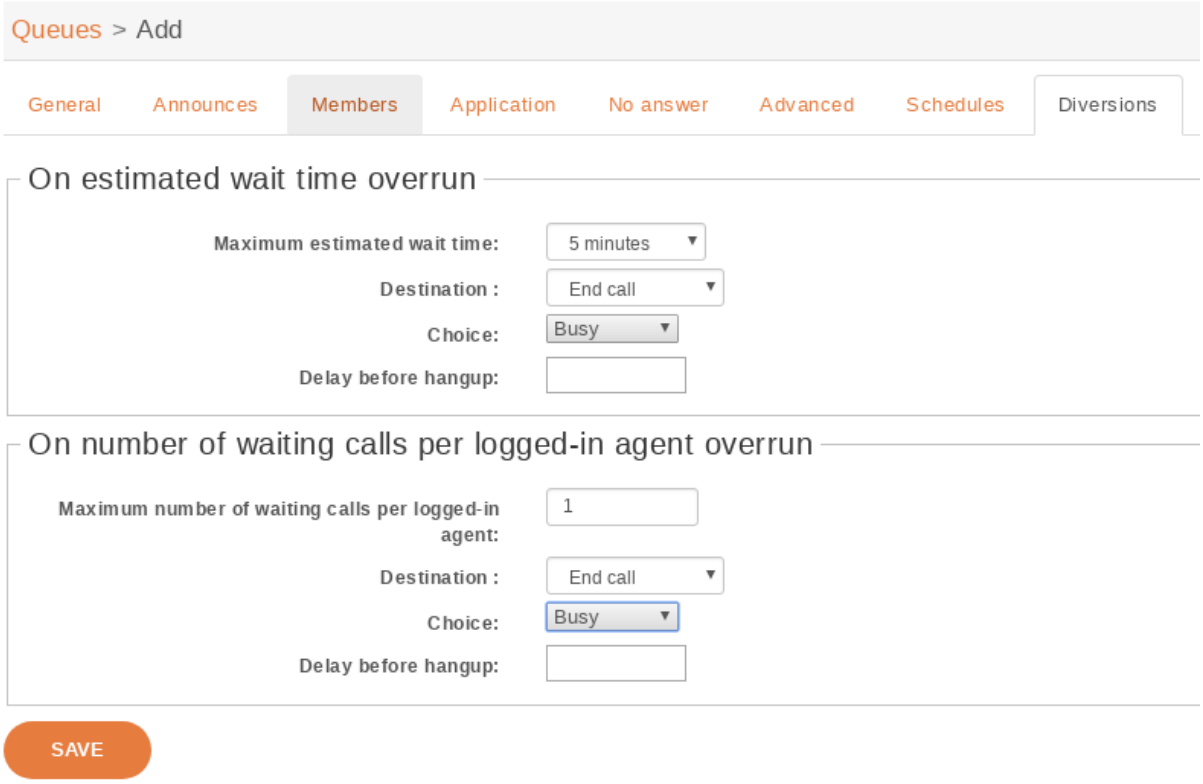
Remove callers if there are no agents:

3 items selected	Remove all		Add all
unavailable	–	paused	+
invalid	–	penalty	+
unknown	–	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





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.

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
```

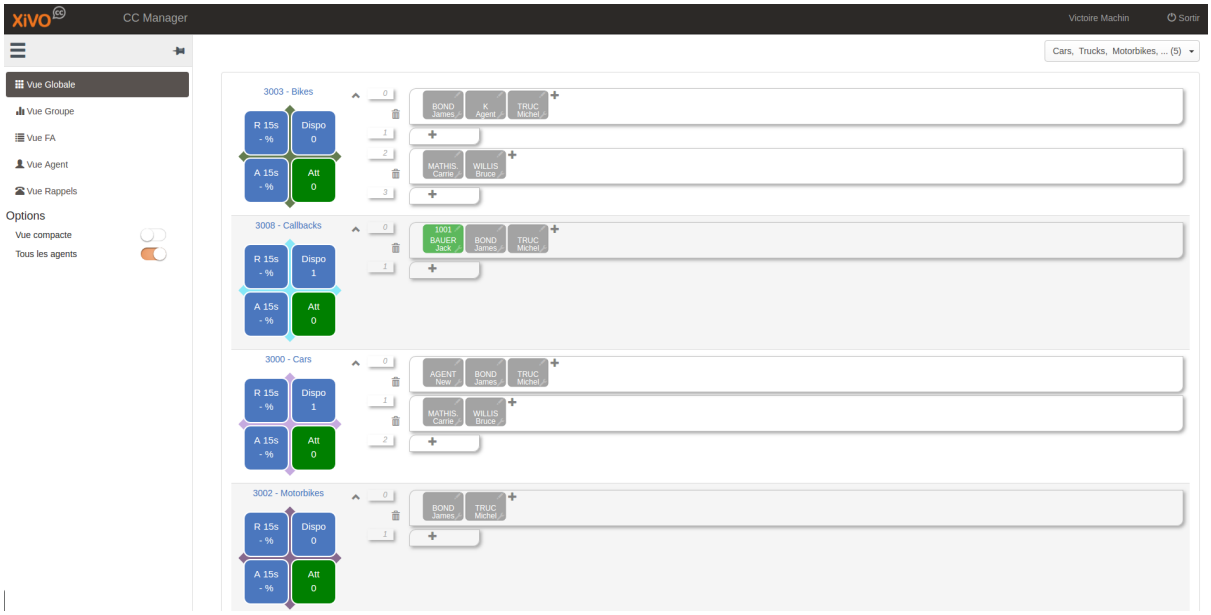
Note that if a new call arrives when there are no waiting calls in the queue, the call will **always** be allowed to enter the queue. For example, in the following scenario:

```
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.

Contact Center Management

Introduction



CCmanager is a web application to manage and supervise a contact center, different menus are available from hamburger icon with following features:

Start the application : http://<xucmgt:port>/ccmanager

Global view

Queues and penalties with real time activity of each agent, possible options are

- Enable Compact view (remove queue statistics)
- Show/Hide agents that are not logged in
- Manage agents in queues:
  - Add/Remove agents from queues,
  - Drag&Drop agents from queue to queue,
  - Change agent penalties
  - Login/Logout, Pause/Unpause, Call[2] or Listen to[Page 413, 2] an agent

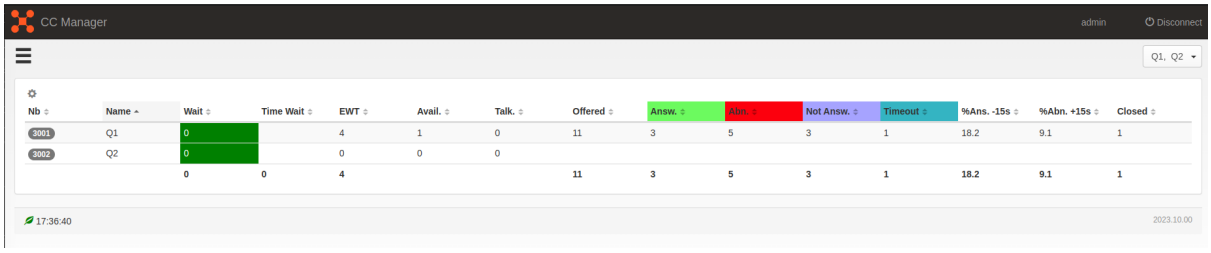
Group view

Distribution of agents per queues

Queue view

Activity per queue

Since Luna, some color and placeholders were added. They intend to better fit the queue statistics schema description, available here *Queue statistics*



Agent view

Activity per agent, possible *Agents actions* are[1]

- Login / Logout
- Pause / Available
- Listen[2]
- Call[2]

[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

[1] Available actions depend on the state of the agent

Callback view

List of callbacks. See *Callbacks*.

Qualification view

To export calls qualification. See *Exporting qualifications*.

Authorizations and Access Control

Access to the application is restricted to authorized users(see *Access authorizations in CCManager*).

Agents actions

As a supervisor you have some actions available on agent[1]:

- Login / Logout
- Pause / Available
- Listen[Page 413, 2]
- Call[Page 413, 2]

Listen to agent's conversation

While an agent is on call, as a supervisor you can listen to it's call. To do this:

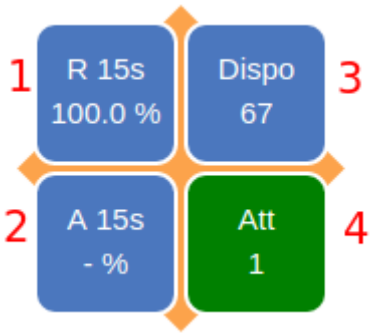
- use the *Listen* action on a agent,
- then you'll receive a call on your phone[Page 413, 2]
- when answered you'll be listening to the agent conversation
- use DTMF to change listening mode:
  - 4 spy mode: the supervisor listens to the conversation
  - 5 whisper mode: as *spy mode* plus the supervisor can speak to the agent without the other participant to hear
  - 6 barge mode: as *spy mode* plus the supervisor can speak to both the agent and the other participant (e.g. the customer)

Note: By default when a supervisor spies on an agent the agent is only warned via the *CC Agent Call control* listen icon.

XiVO CC can be configured to warn the agent via a beep and/or a lit key on its deskphone - see *Warn agent when spied*.

Queue statistics

This diagram shows some aggregation about statistics collected from queue activity



Those are the statistics available by default without configuration.

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

You can configure which statistic is displayed in those squares - see *Configure queue statistics displayed*.

Queue details and configuration

By clicking on the name of the queue, a modal will appear. Supervisors with no access to dissuasion will see some details about the queue. Administrators and supervisors who have access to the dissuasion will see different tabs :

- One tab showing some information about the queue
- One tab to configure the dissuasion in case of an exceptional closing for this queue.

Configuring the dissuasion for a queue

In the configuration tab, click on the title "Failed Destination". A dropdown appears, showing the sound files and/or the queue that can be set up as a failed destination. The failed destination can be changed from this dropdown.

To change the failed destination options for this queue, see *Activity's Failed Destination Configuration* section.

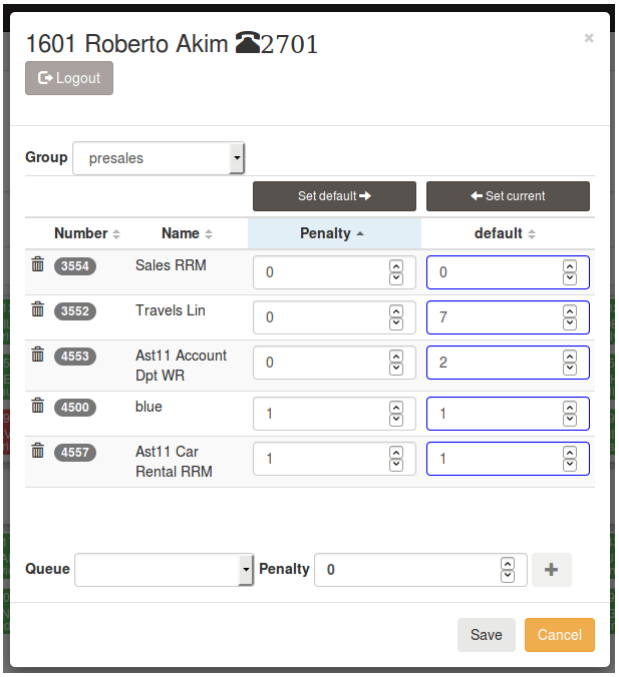
Editing Agent Configuration

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".

Sales - 2002
Queue details
Queue configuration
Failed destination
queue1
Lunch
Meeting
Close

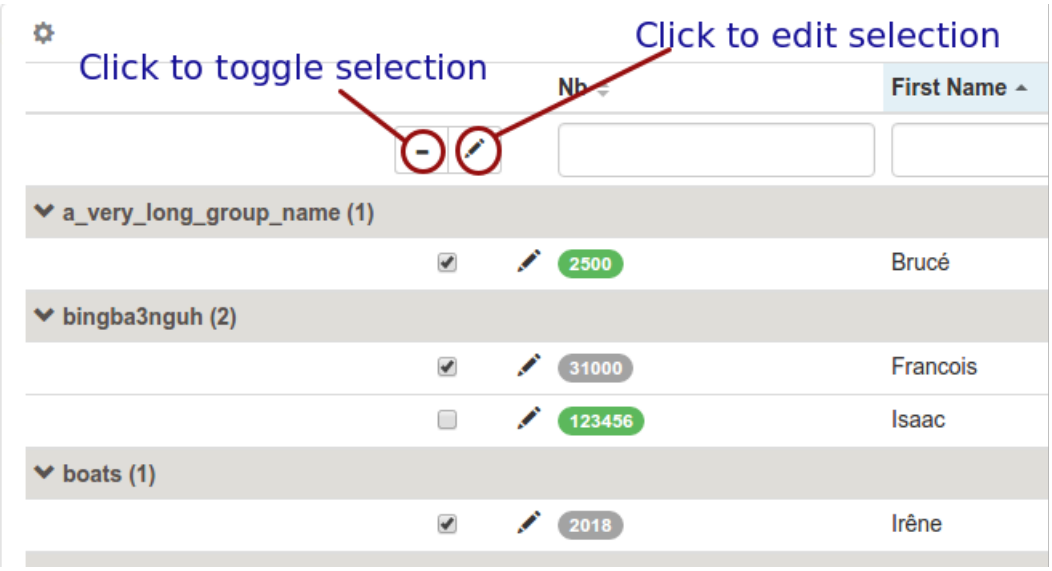

1601 Roberto Akim 2701
Logout
Group
presales
Set default
Set current
Number
Name
Penalty
default
3554
Sales RRM
0
0
3552
Travels Lin
0
7
4553
Ast11 Account Dpt WR
0
2
4500
blue
1
1
4557
Ast11 Car Rental RRM
1
1
Queue
Penalty
0
Save
Cancel

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.

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.

Agent Base Configuration

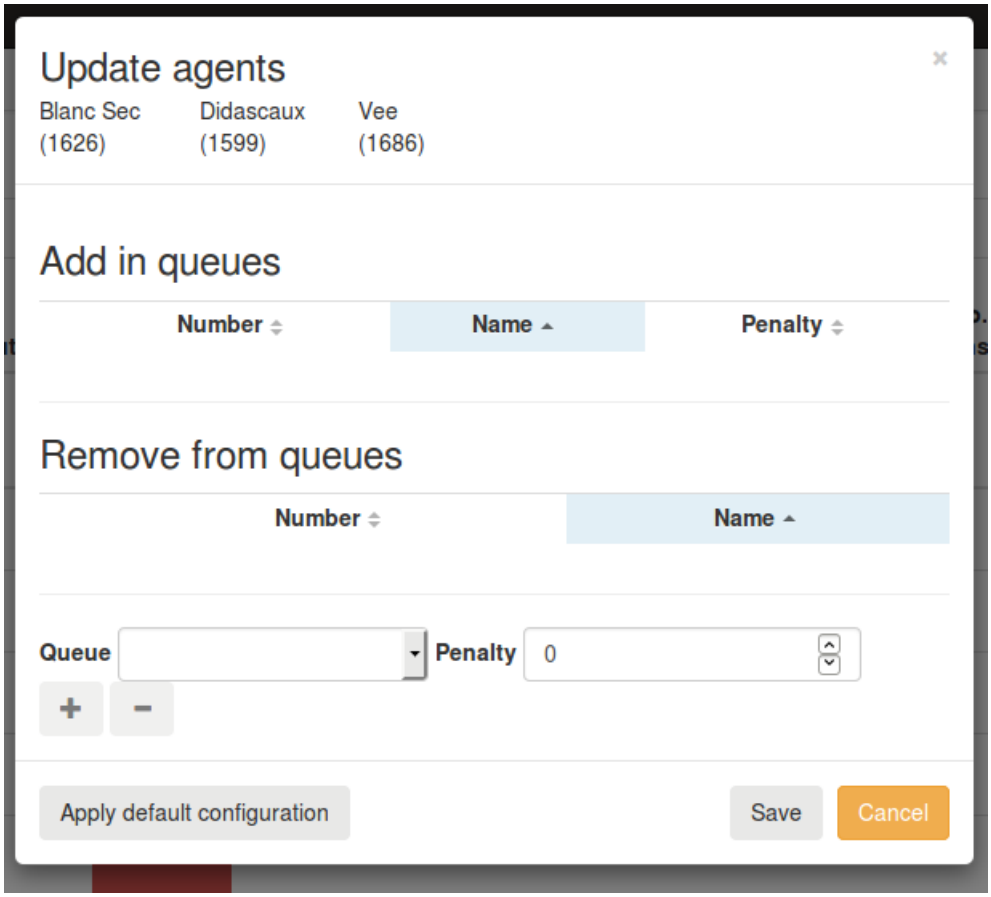
From the agent view, after selecting one or more agents, you can create a base configuration by clicking on one of the menu item in the following drop down:

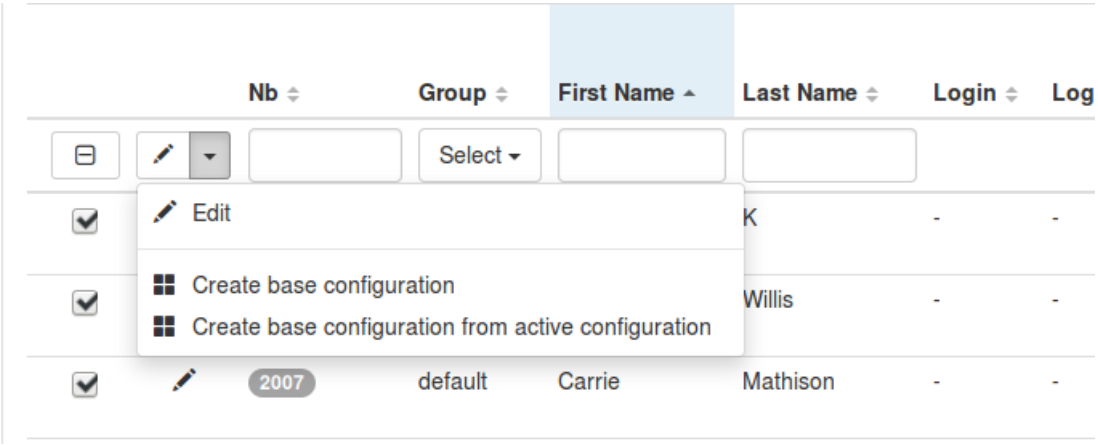
- 'Create base configuration' will allow you to create a base configuration from scratch for all the selected agents.
- 'Create base configuration from active configuration' will allow you to create a base configuration using the selected agents active configuration. The queue membership and penalty populated will be built based on the merged membership of all the selected agents. In case of conflict, the lowest penalty will be used.

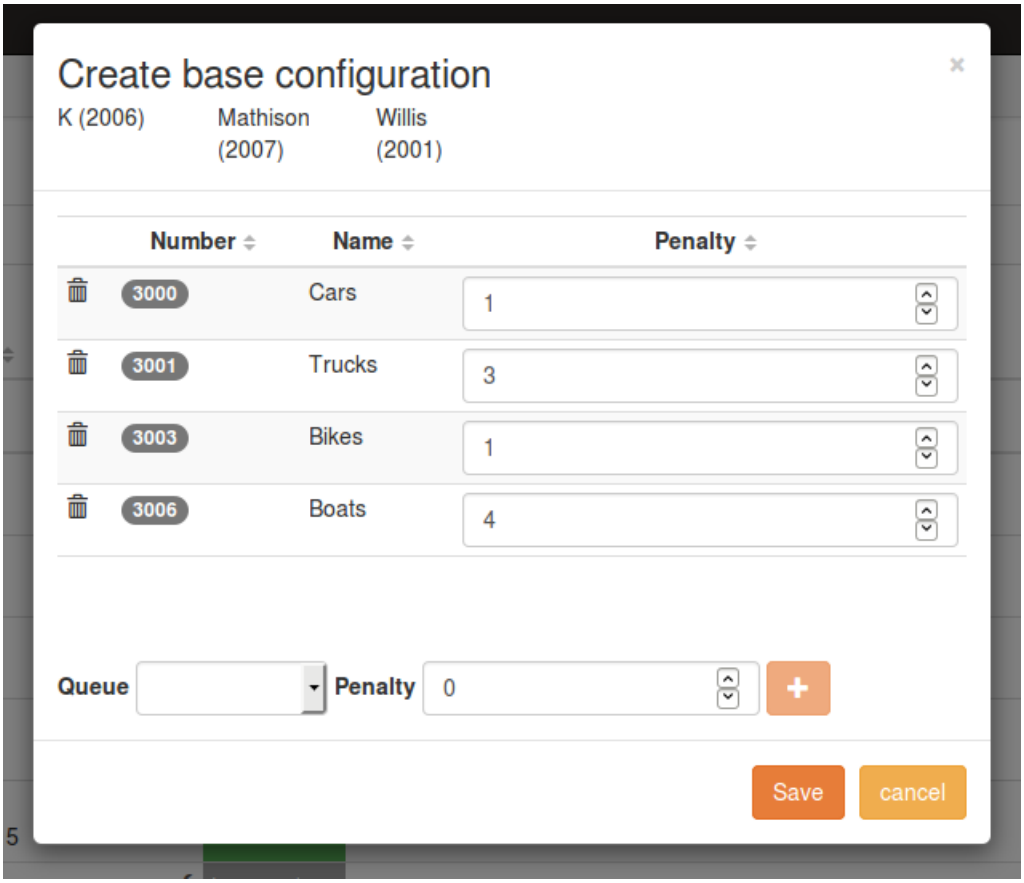
In both cases, you will be able to review your changes before applying them. The 'Create base configuration' popup is similar to the single agent edition popup:

The queue table display the following columns:

- "Number": The queue number
- "Name": The queue name
- "Penalty": The active penalty for the corresponding queue


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


Nb
Group
First Name
Last Name
Login
Log
Select
Edit
Create base configuration
Create base configuration from active configuration
K
Willis
2007
default
Carrie
Mathison

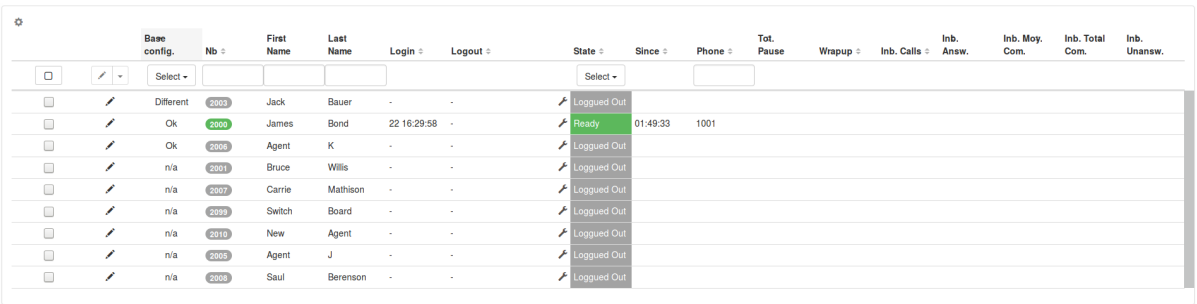


Click on the plus button to add a queue for selection. Once your configuration is complete, click on save button to apply your configuration change.

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:



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	-	-	Logged Out									
Ok	2000	James	Bond	22 16:29:58	-	Ready	01:49:33	1001							
Ok	2006	Agent	K	-	-	Logged Out									
n/a	2001	Bruce	Willis	-	-	Logged Out									
n/a	2007	Carrie	Mathison	-	-	Logged Out									
n/a	2099	Switch	Board	-	-	Logged Out									
n/a	2010	New	Agent	-	-	Logged Out									
n/a	2005	Agent	J	-	-	Logged Out									
n/a	2008	Saul	Berenson	-	-	Logged Out									

The possible values for this field are:

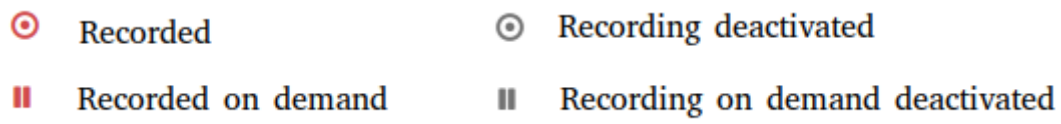
- "n/a": The base configuration is not available for this agent
- "Ok" : The base configuration match the active configuration
- "Different": The base configuration **does not** match the active configuration

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.

Queue Recording

Description

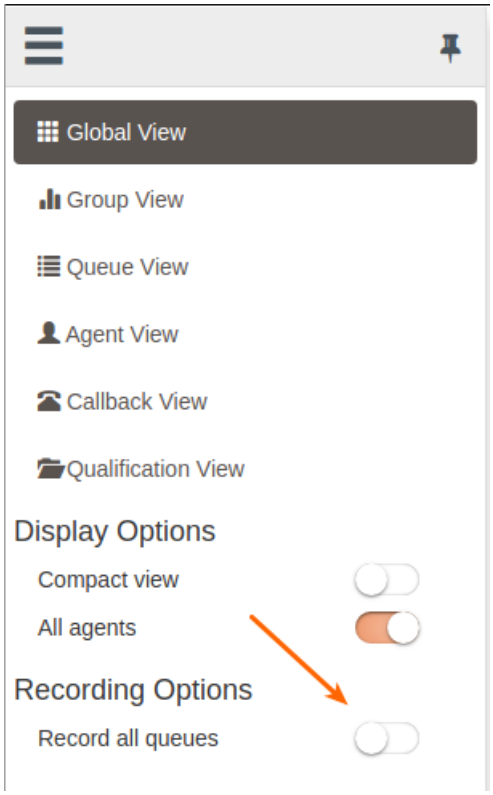
Once you setup queue recording in XiVO (see *Enable recording in the Queue configuration*), visual indicators are displayed next to queue name in **Global view**. Respectively following icons represents **recording mode** set on the queue.



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.

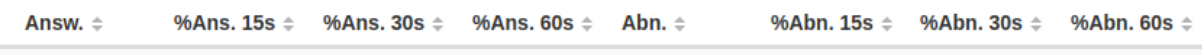


Note: Action is applied only for next calls. Ongoing call recording is not started nor stopped when switch is triggered.

Thresholds

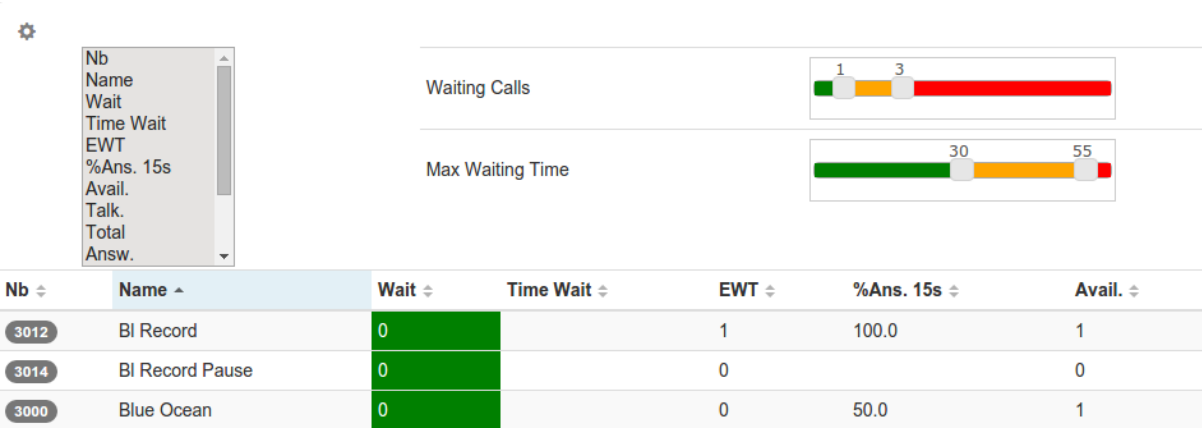
Additional Queue statistics columns

It is possible to add columns to reflect more performance indicators, such as **Percentage of calls answered/abandoned within XX seconds**. See *Configuration* to add such stats thresholds.



Colors

Color thresholds can be define for the waiting calls counter and the maximum waiting time counter



Once done, it is applied to the queue view and the global view

Callbacks

This view allows to manage callback request see *Managing Callbacks Using CCManager* for details.

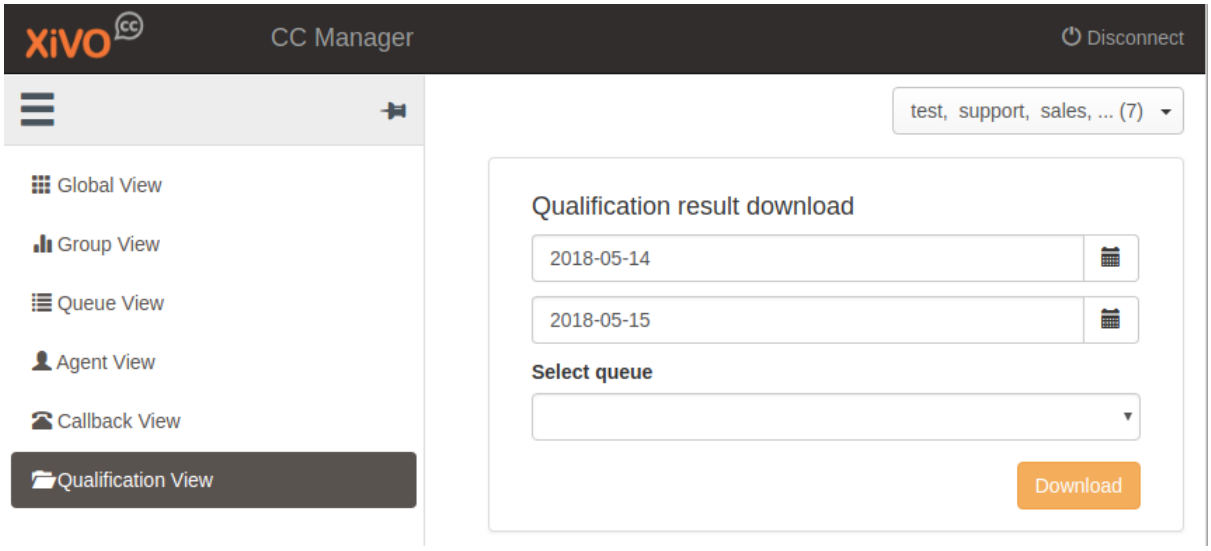
Exporting qualifications

This view allows to export calls qualification see *Call Qualifications*.

To export the qualification answers, open **Qualifification 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**.



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 ?

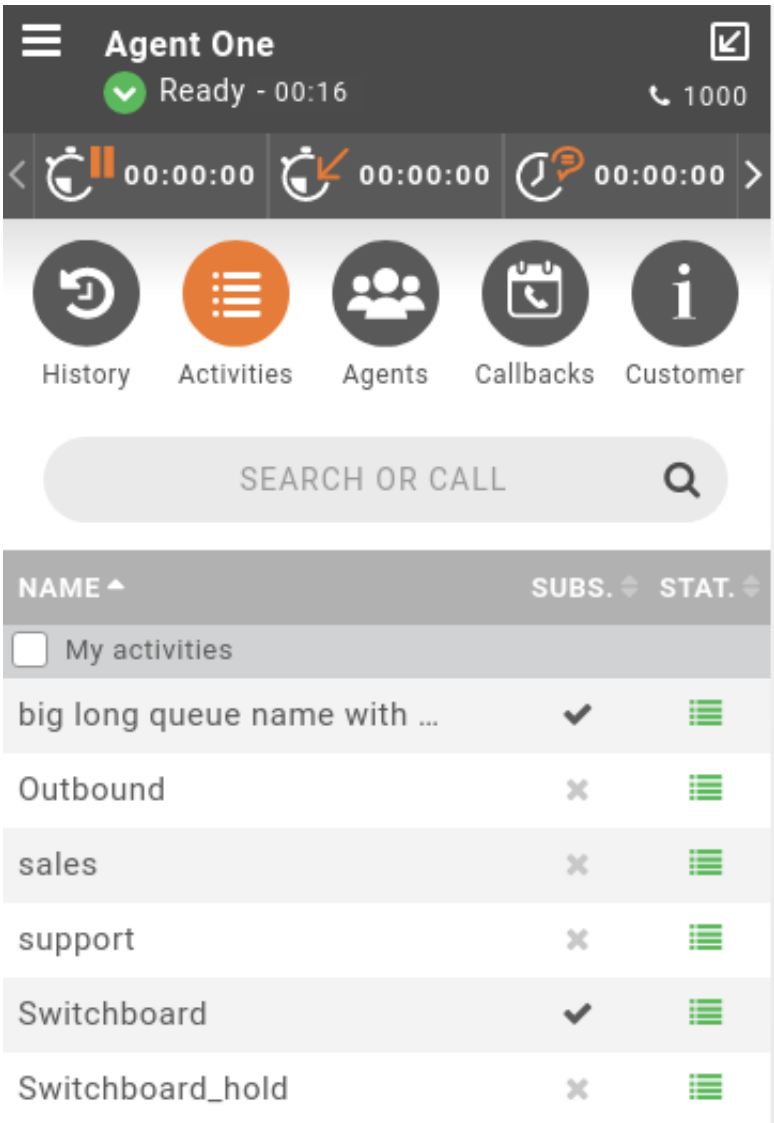
CC Agent is a Web application for contact center operators. Some parameters for Recording, Callbacks, Queue control, Pause statuses and Sheet popup may be configured. Instructions can be found in the *configuration section*.

From the interface you will be able to :

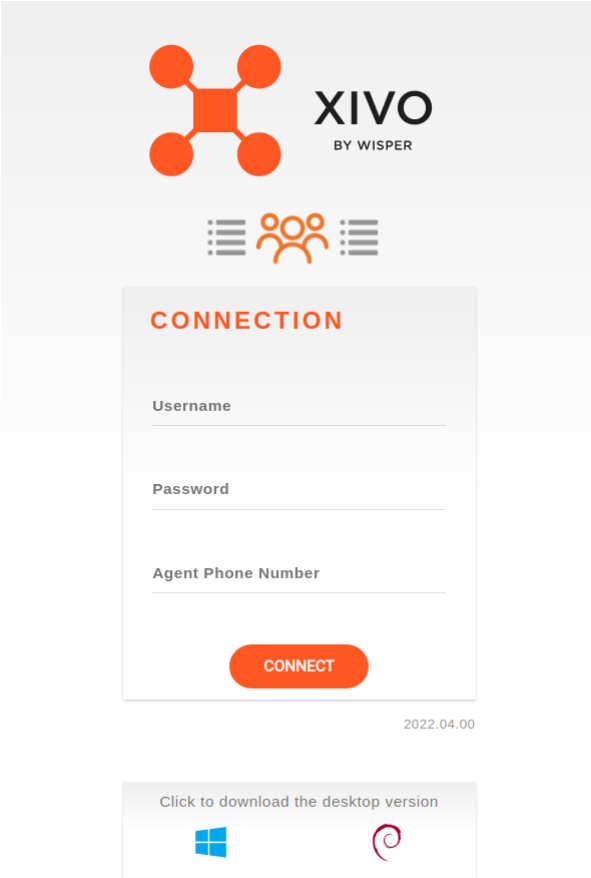
- Manage your activities you are subscribed to receive calls.
- See the customer history inside your organization when a call is coming
- Interact with your phone from the call control panel
- Get some real time statistics of your session

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 the second call which is only partially supported.


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

Login



Enter your CTI 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:

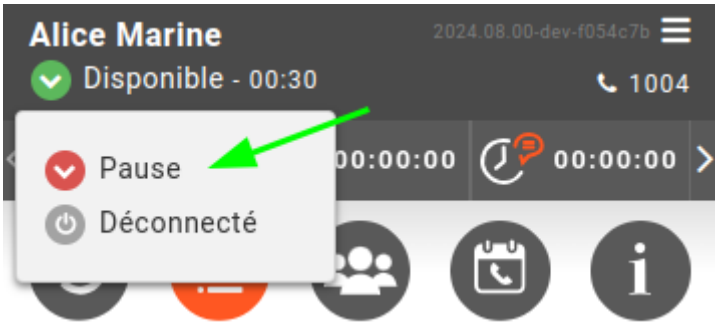
Note: Automatic login keeps you signed in until you log out.

Pause

When connected the agent is *Available*. In this status the agent receives calls from the activities it is logged in.

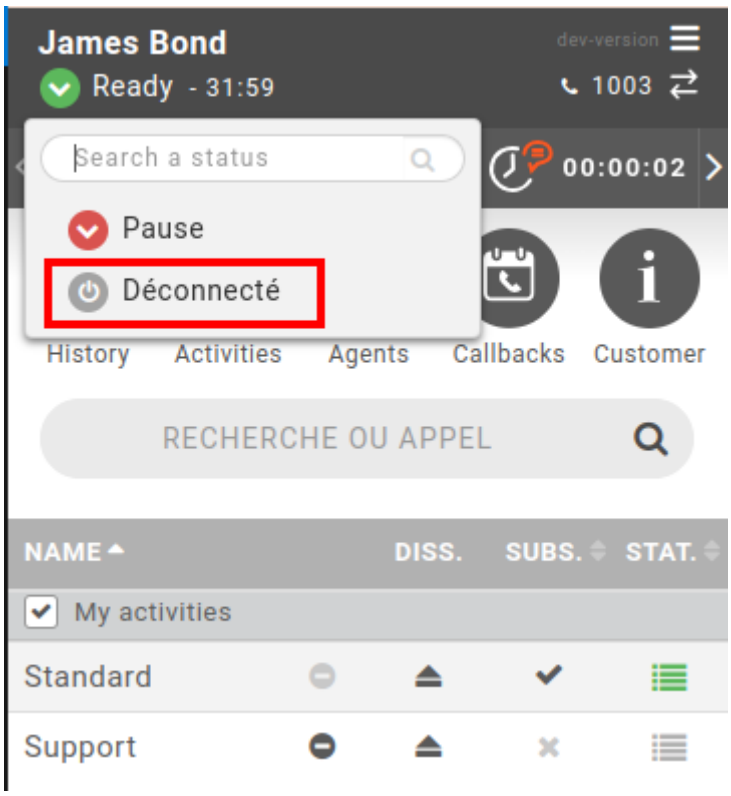
If he needs, the agent can select the **Pause** status. In this mode the agent is seen in *Pause* for all activities and won't receive any call from these activities. But he can still receive direct calls (internal calls from his colleagues for example).

Note: By default there is only one *pause* status named **Pause**. This can be configured - see *Pause Cause and Status*.



Logout

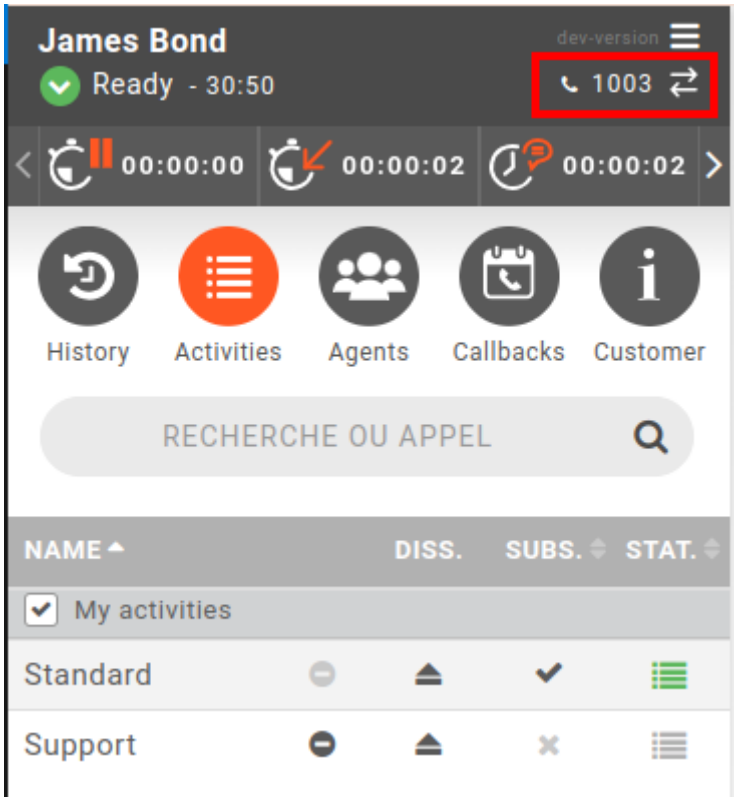
To logout from the CC Agent, you have to select the status *Disconnected* from the pause status list.



Switch phone number

The agent can see the line number he's logged on in the CC Agent application.

If the agent needs to change the line number, there is a dedicated button next to his line number.



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.

Those statistics are clickable and display dynamic charts on top of the counters. The goal is to offer a global view to agents on their activity during the day.

There are two differents charts displayed.

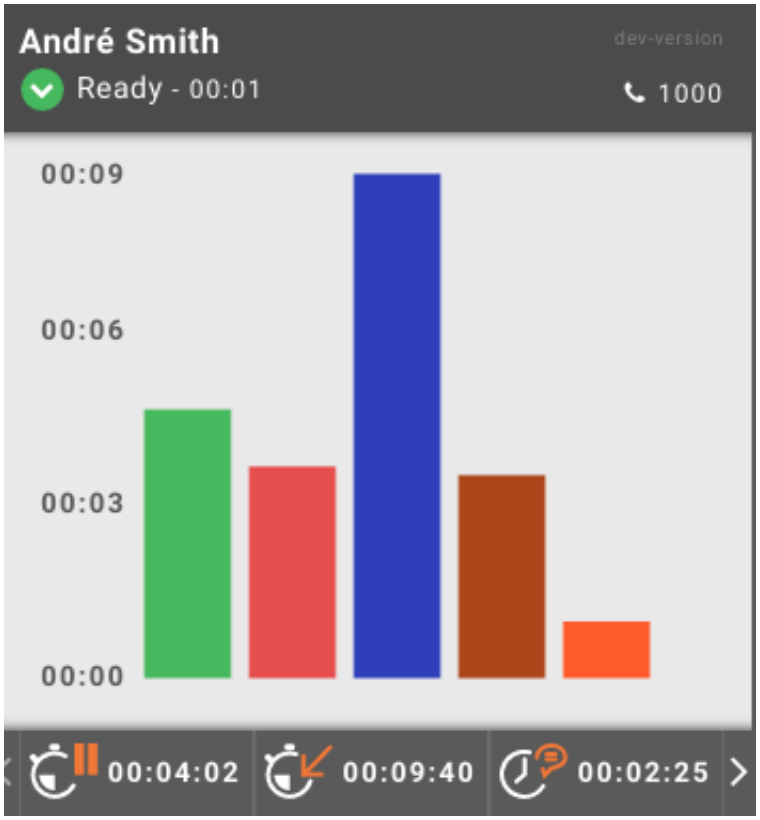
The first one is a bar chart focusing on the time unit :

- Total Available Time
- Total Pause Time
- Inbound ACD Calls Total Time
- Outbound Calls Total Time
- Total Wrapup Time

On click on each one of those buttons, the bar chart will be displayed. You can hover each bar to have the detailed time spent for a specific indicator (HH:MM:SS).

The second one is a donut chart focusing on the number of calls :

- Number of Inbound ACD calls
- Number of Inbound Answered ACD Calls
- Number of Outbound Calls



On click on each one of those buttons, the donut chart will be displayed. You can hover each part of the donut to have the number of calls for one indicator.

The charts are updated once current action is over.

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.

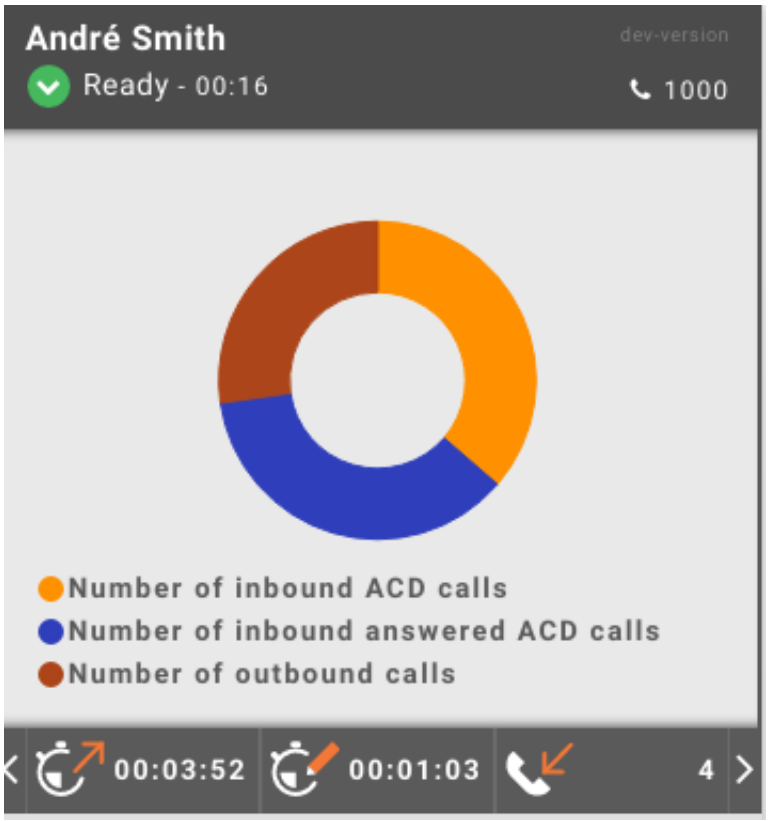
Activity Management

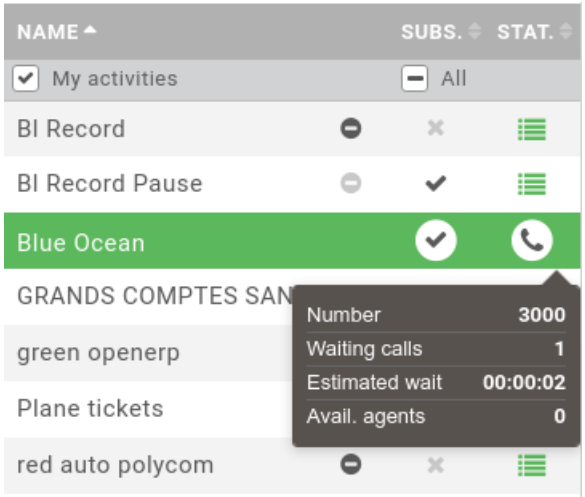
You can manage `subscription` if allowed globally for the application (see *CC Agent configuration*). If enabled you will be able to enter/quit an activity just by clicking on checkbox associated to it.

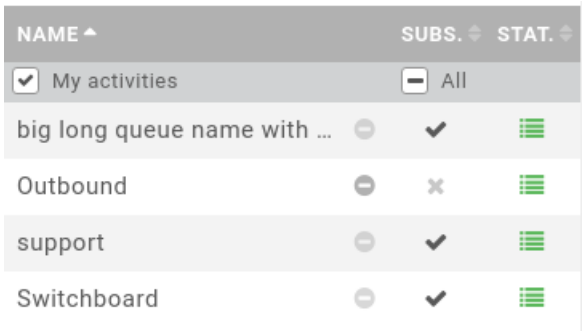
Each time you enter an activity, it is automatically added to your favorites. At any time you can remove it by clicking on *minus* sign next to the name in `My activities` view.

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.


André Smith
dev-version
Ready - 00:16
1000
Number of inbound ACD calls
Number of inbound answered ACD calls
Number of outbound calls
00:03:52
00:01:03
4


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


NAME
SUBS.
STAT.
My activities
All
big long queue name with ...
Outbound
support
Switchboard

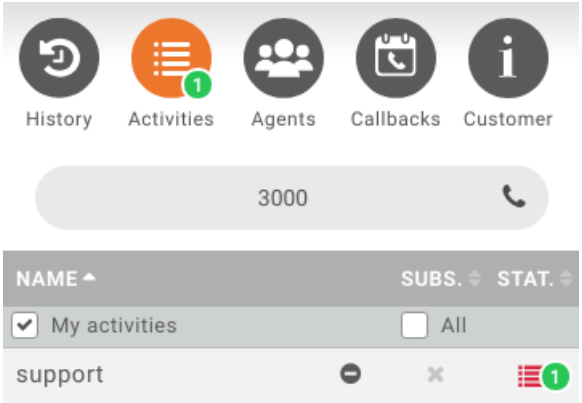
Activity Colors

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.



Activity's Failed Destination

In XiVO, when an activity is exceptionnally closed, a sound file containing a message can be played to the caller. From its *CC Agent* application, an agent who has access to the dissuasion can:

- see which sound file or default queue is currently configured
- select another sound file to be played
- select a default queue as a failed destination, instead of a sound file.

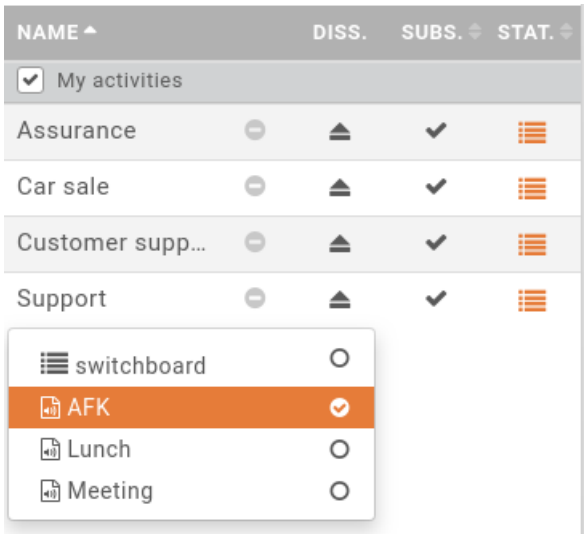
An agent can change the dissuasion if they have been granted access, it means if they have an administrator profile or a supervisor profile with the permission to change the dissuasion. The agent profile can be set from the configmgt interface. See *Profile Management* section.

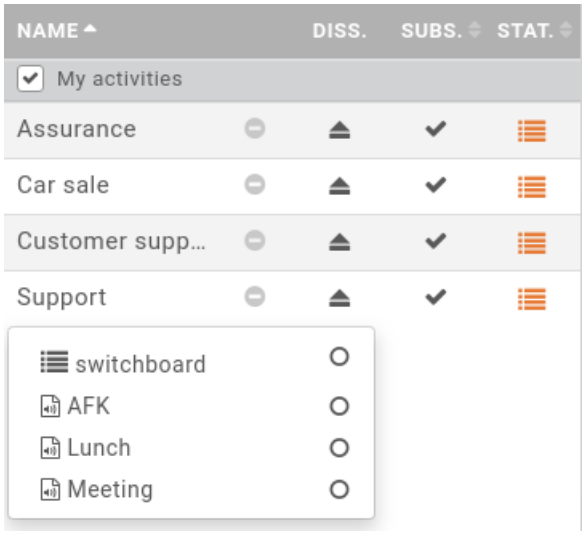
Note: The sound files available for the activity and displayed to the agent and the default queue are to be set by an administrator of the XiVO. See *Activity's Failed Destination Configuration* section.

In the Activity view, when clicking the *Dissuasion* button, a menu appears with the list of the sound files or default queues available for a queue. If one of the sound file or default queue is selected for this queue, it will be highlighted in orange and the circle icon will be checked.

If no sound file or default queue are selected for the queue, the circle icon won't be checked.

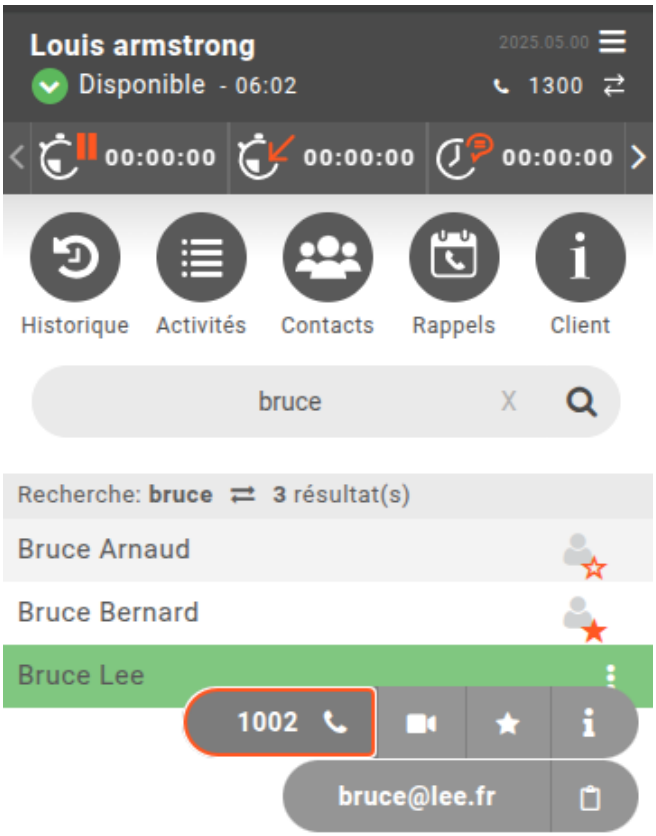
The agent can change the selected sound file or default queue from the dropdown menu.


NAME
DISS.
SUBS.
STAT.
My activities
Assurance
Car sale
Customer supp...
Support
switchboard
AFK
Lunch
Meeting


NAME
DISS.
SUBS.
STAT.
My activities
Assurance
Car sale
Customer supp...
Support
switchboard
AFK
Lunch
Meeting

Search

When using the Agent interface, you can at any time search for a user existing in your directory:



The DND (Do Not Disturb) status of each contact is displayed in the search results. When a contact has DND enabled, a red icon appears next to their avatar.

When clicking on the a line of a search result a sub-menu appears with the following actions:

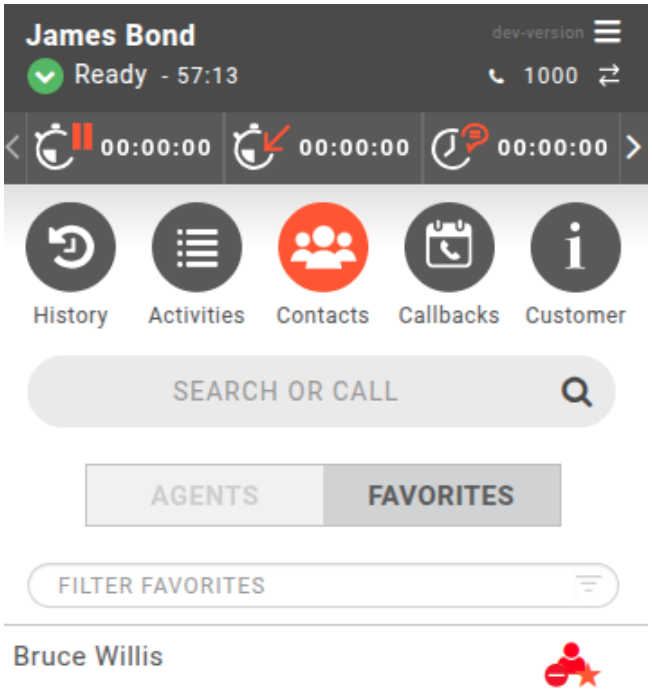
Information icon which open a contact sheet:

For phone numbers entries:

- start call (if you click on the number)
- copy phone number into your clipboard (to paste it elsewhere)

For email entry:

- start writing an email (it will open the user's configured email client)



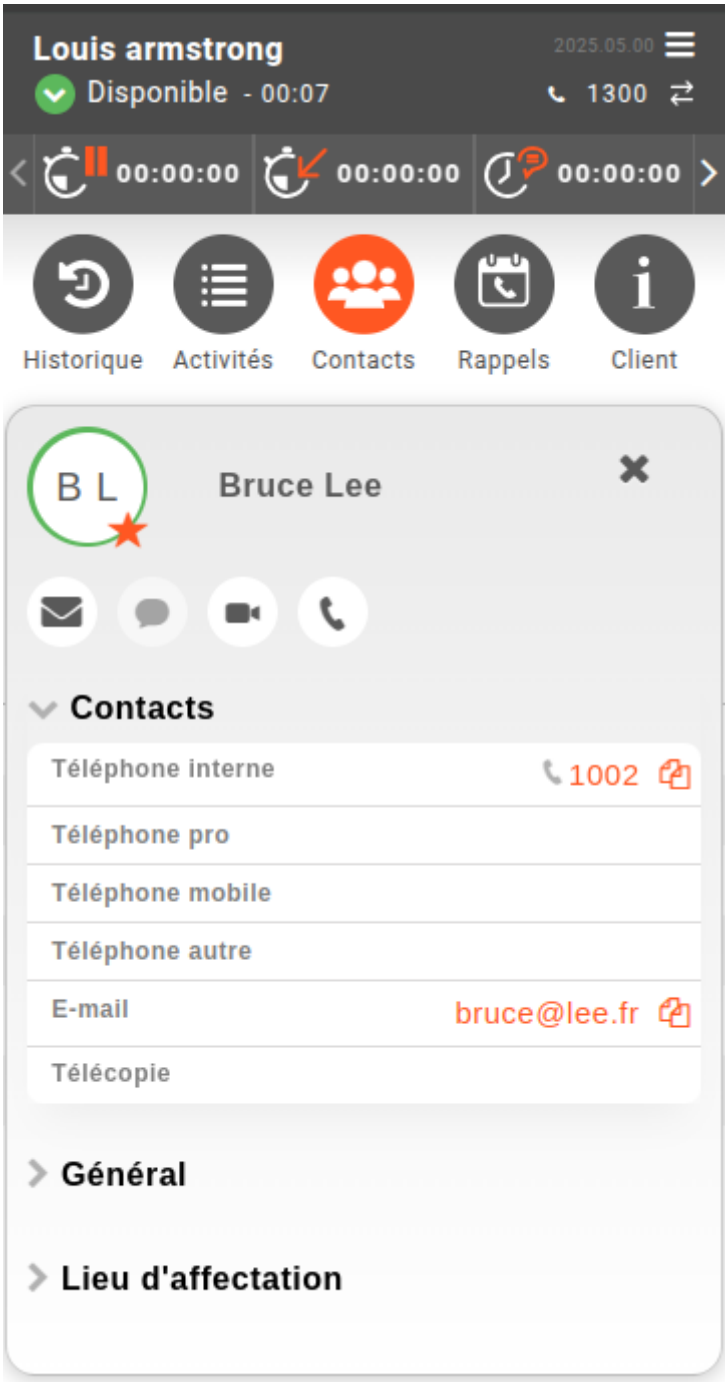
Note: by default only the email destination is pre-filled but you can also pre-fill the email *subject* and *body* via a template - see *Email Template*

- copy email into your clipboard (to paste it elsewhere)

Important: Integration: to enable this feature, you must configure the directories in the *XiVO PBX* as described in *Directories* and *Personal contacts*.

Though the *CC Agent* only supports 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

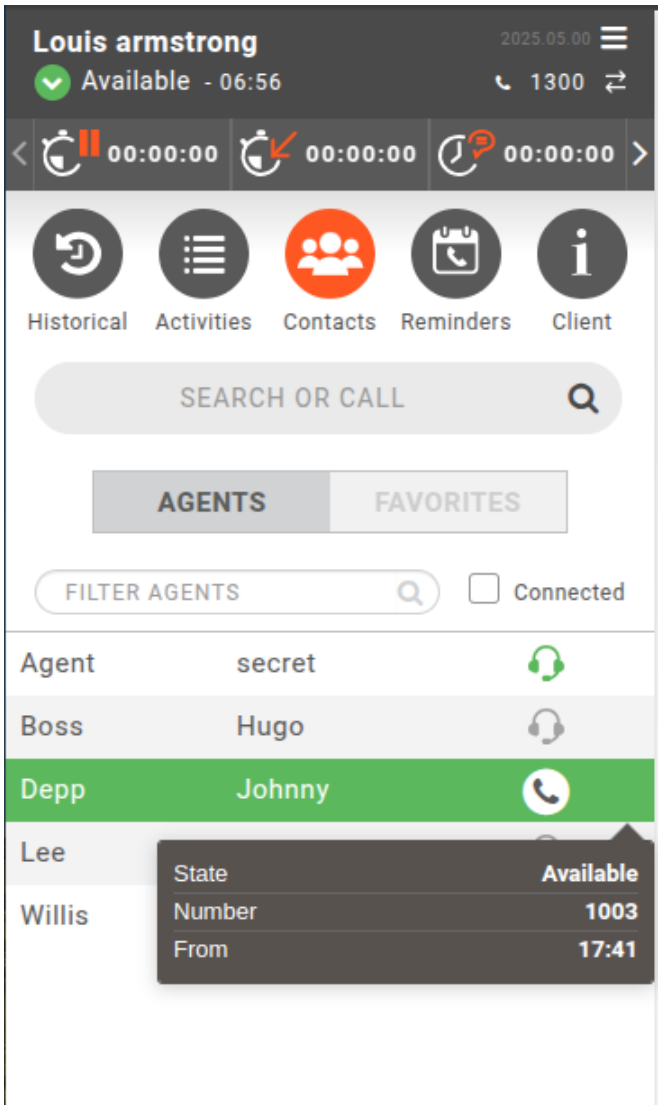

Louis armstrong
2025.05.00
Disponible - 00:07
1300
00:00:00
00:00:00
00:00:00
Historique
Activités
Contacts
Rappels
Client
B L
Bruce Lee
Contacts
Téléphone interne
1002
Téléphone pro
Téléphone mobile
Téléphone autre
E-mail
bruce@lee.fr
Télécopie
Général
Lieu d'affectation

Contacts

When clicking on the `Contacts` menu, you will have access to two sections: the list of agents of your group and the list of your favorite contacts.

Agents 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.

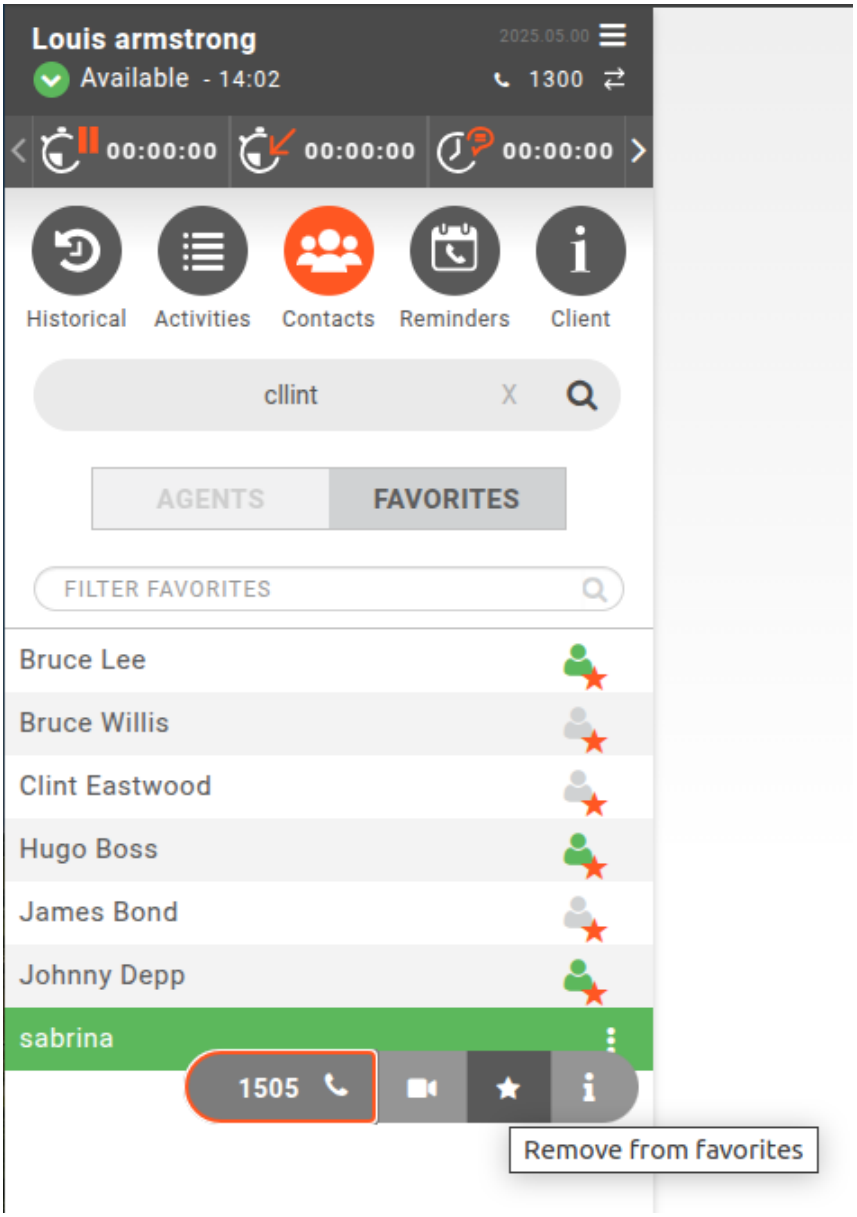
Displayed agents can be filtered using the filter bar. Start typing the name of the agent you are looking for, the list will dynamically filter and display matching results.

Favorites

When clicking on the `Favorites` menu, you will see all your favorite contacts list.

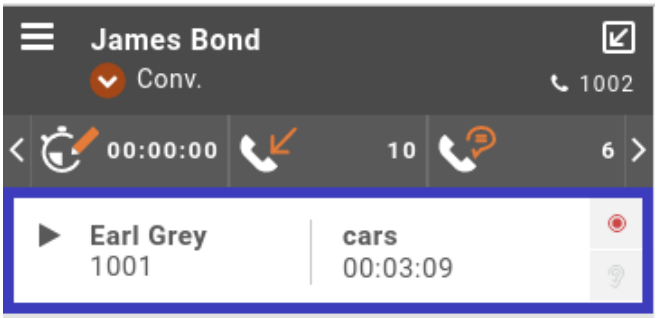
Displayed favorites contacts can be filtered using the filter bar, start typing the name of the contact you are looking for, the list will dynamically filter and display matching results.

To add a favorite, you first need to search it via the search bar. Then click on the line of the contact you want to add as favorite. A sub-menu will appear with different available actions. Click the star icon to add the contact to your favorites. If the contact is already in your favorites, clicking the star icon will remove it from the list.



Call tracking & control

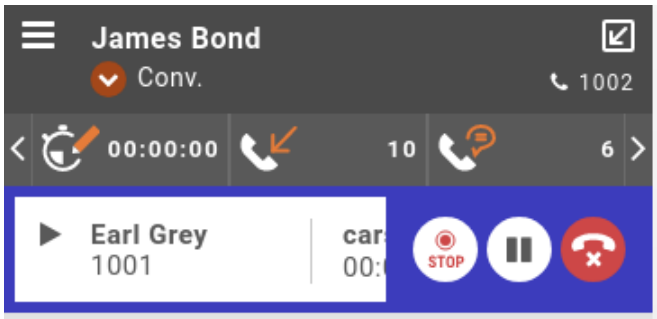
When using the Agent interface, you will see your current calls at the top of the screen:

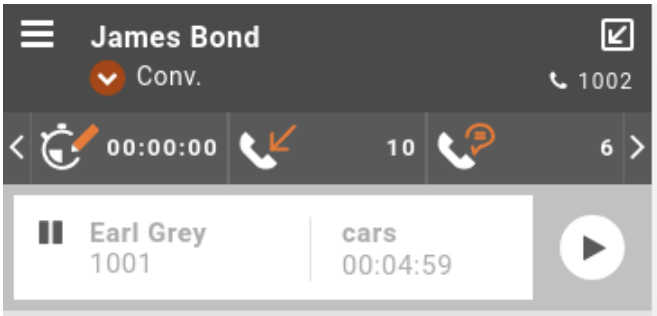


This panel will display the current caller name & number and also the associated activity if the call came from one. You also have two indicator on the right side letting you know:

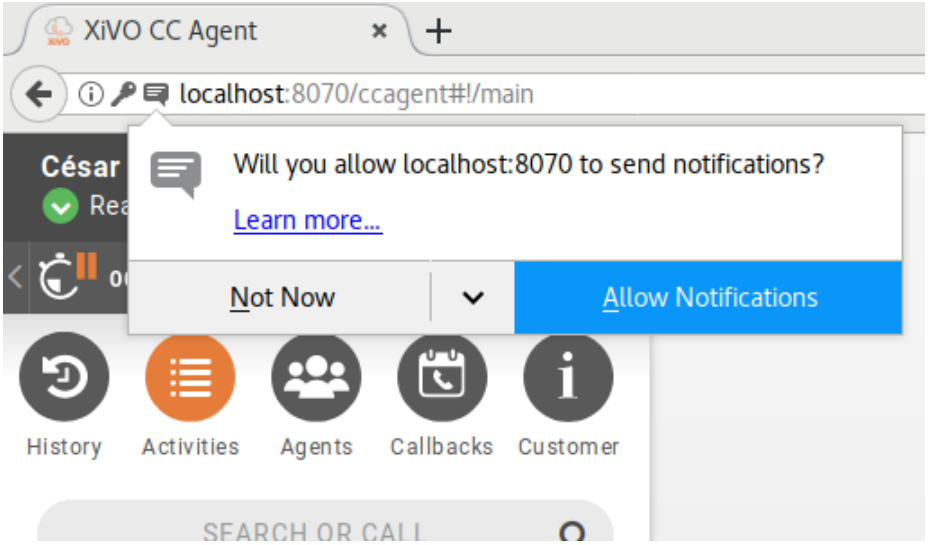
- if the call is currently recorded
- and if the call is currently listened by a supervisor (on this subject see also *Warn agent when spied*).

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:



Keyboard shortcuts for call control

Simple keyboard navigation (with **UP** and **DOWN** directional keys) is allowed to browse search results. Once a contact has been focused, **ENTER** key can be pressed to open drop-down menu and so choose, still with directional keys, the action to achieve.

The Agent can also use the basic keyboard shortcuts available to perform basic call control tasks :

- **F3** to **answer** a call
- **F4** to **hangup** the current ongoing call
- **F7** to **complete an attended transfer**
- **F10** to put the *focus* in the **search bar**

Note: The list of usable keybindings is available by clicking on the menu at the top-right corner of the ccagent and the switchboard.

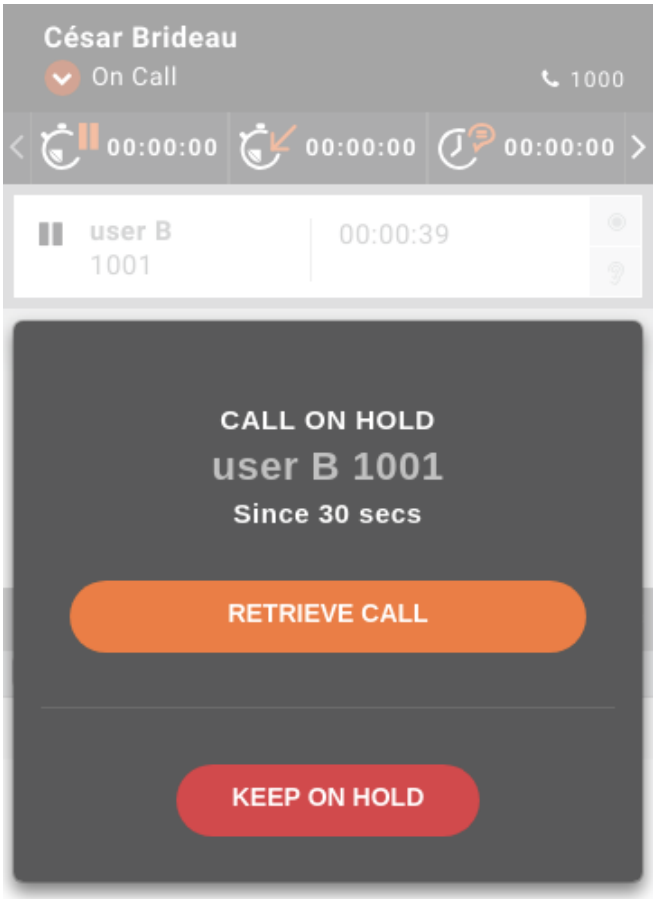
On hold notifications

You can be notified if you forget a call in hold for a long time, see *configuration section*.

Known limitations

- For agents with default WebRTC lines that connects to another webrtc line (aka "roaming agent") may randomly take the wrong line configuration (especially when you refresh the page). Only workaround so far is to logout/login the agent to force the retrieval of the correct line. For roaming agent, it is recommended to not set default line to agent and use free sitting to wanted line.

Also see the phone integration *Known limitations*.



Agent Call history

First menu `History` tab is displaying the call history of connected agent. It displays the last 20 calls for the last 7 days period.

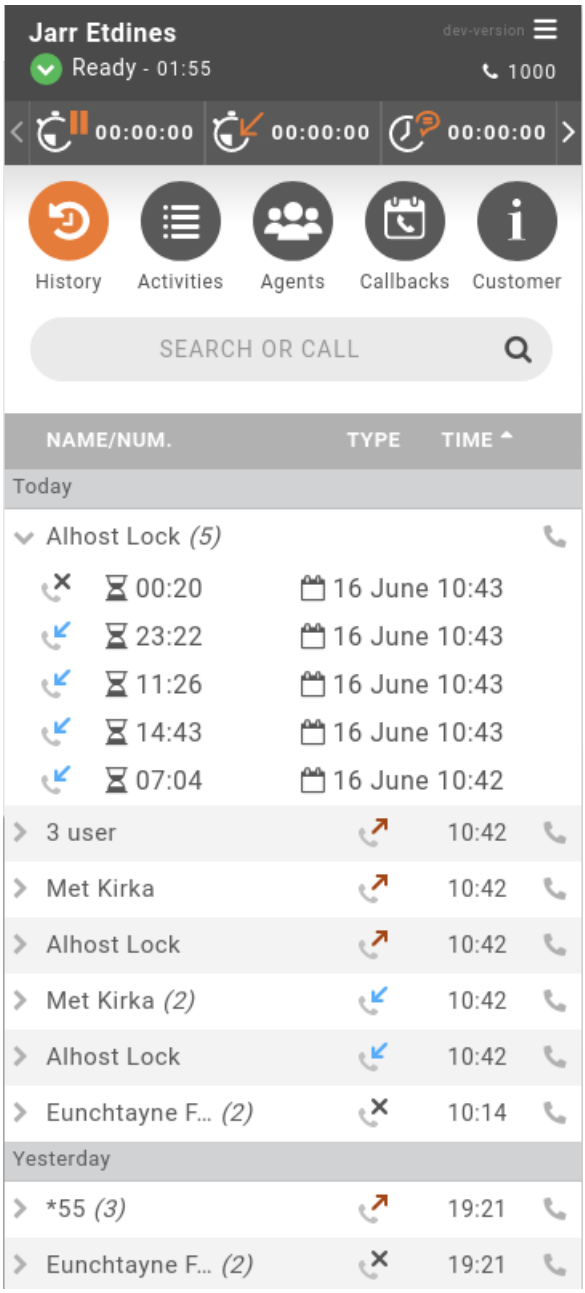
Information shown in the call list are :

- *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.


Jarr Etdines
dev-version
Ready - 01:55
1000
00:00:00
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
SEARCH OR CALL
NAME/NUM.
TYPE
TIME
Today
Alhost Lock (5)
00:20
16 June 10:43
23:22
16 June 10:43
11:26
16 June 10:43
14:43
16 June 10:43
07:04
16 June 10:42
3 user
10:42
Met Kirka
10:42
Alhost Lock
10:42
Met Kirka (2)
10:42
Alhost Lock
10:42
Eunchtayne F... (2)
10:14
Yesterday
*55 (3)
19:21
Eunchtayne F... (2)
19:21

Customer Call History

While 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.

Important: *Caller Number Normalization* rules breaks the Customer Call History

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	

Customer Call Context

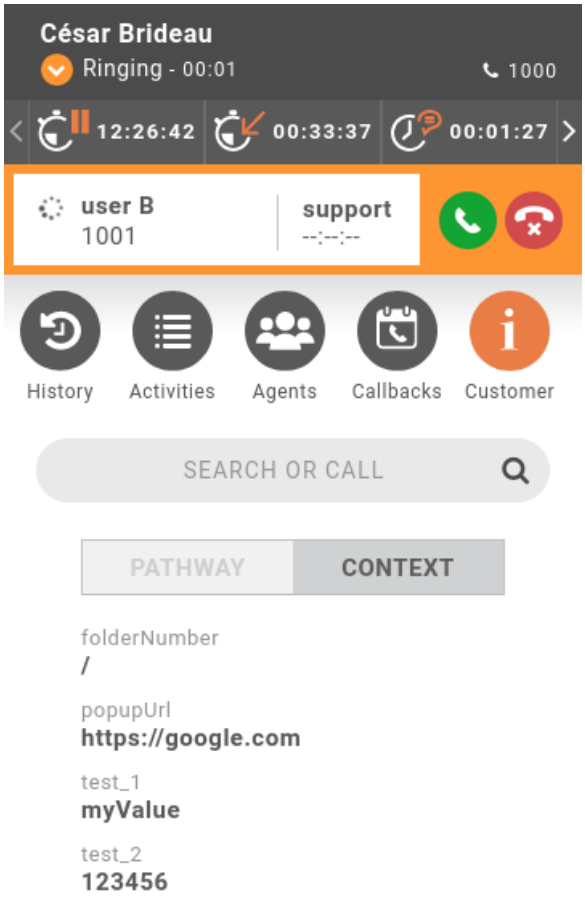
Second tab of `information` menu displays all attached data enriched to the ongoing call or display Sheet fields if you are using *Sheet Configuration*.

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*.

Callbacks

This view allows to manage callback request see *Processing Callbacks with CCAgent* for details.



External directory

This feature will add an additional book button on the left side of the search bar. It allows to open and close an external directory and is integrated the same way as any other tab content.

To enable it, see *configuration section*.

Meeting room

You can join an audio-video conference, called meeting room, which will open on the side of your application. See *Meeting Rooms* for more information.

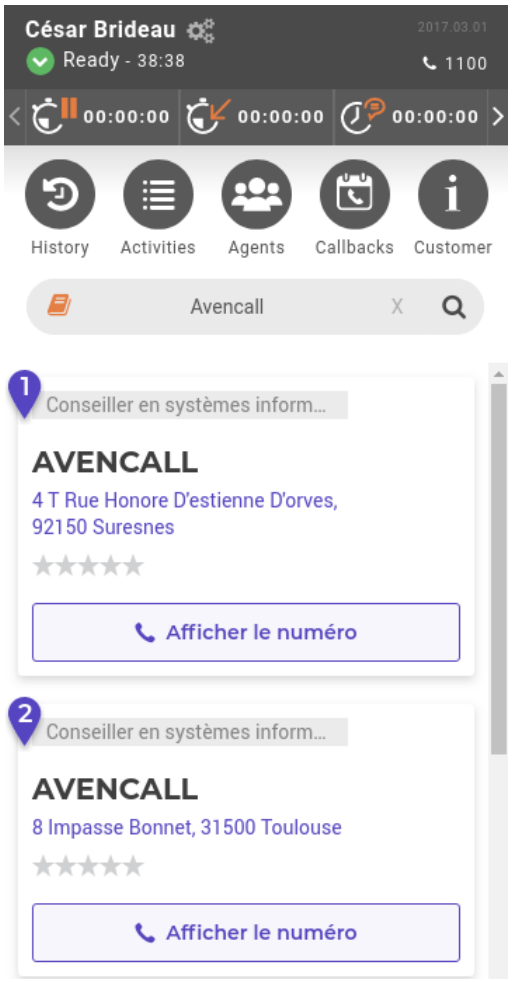
Recording

XiVO CC includes a call recording feature:

- recording is done on *XiVO PBX* (or the XiVO Gateway)
- and recorded files are sent to the *XiVO CC* Recording server.

It's then possible to search for recordings and download recorded files from *Recording server*.

If connected user is an administrator, he will also be able to download an access log file that tracks all actions made on files by all users having access to recording server.


César Brideau
2017.03.01
Ready - 38:38
1100
00:00:00
00:00:00
00:00:00
History
Activities
Agents
Callbacks
Customer
Avencall
X
1
Conseiller en systèmes inform...
AVENCALL
4 T Rue Honore D'estienne D'orves,
92150 Suresnes
Afficher le numéro
2
Conseiller en systèmes inform...
AVENCALL
8 Impasse Bonnet, 31500 Toulouse
Afficher le numéro

- *Recording Server*
  - *Description*
  - *Search, Download, Listen*
  - *Disk Space*
  - *Access logs*
  - *File Sync*
- *Recording and Transfers*
- *Automatic Stop/Start Recording On Queues*
- *Automatic Stop Recording Upon External Transfer*
- *Recording filtering*
  - *Description*
  - *Limitations*

Recording Server

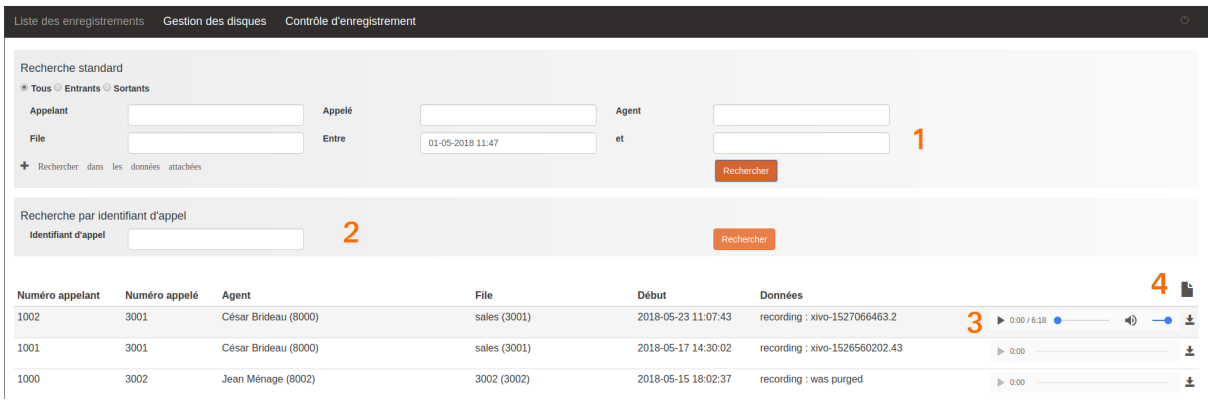
Recording must first be enabled on the *XiVO PBX*, see *Recording* (or on the *XiVO Gateway*, see *Recording on Gateway*).

Description

When recording is enabled the whole conversation between caller and callee is recorded and then sent on dedicated recording server. All the files are then available for download only for predefined *granted* users (see *Profiles Definition*).

Search, Download, Listen

Here's the recording search page:



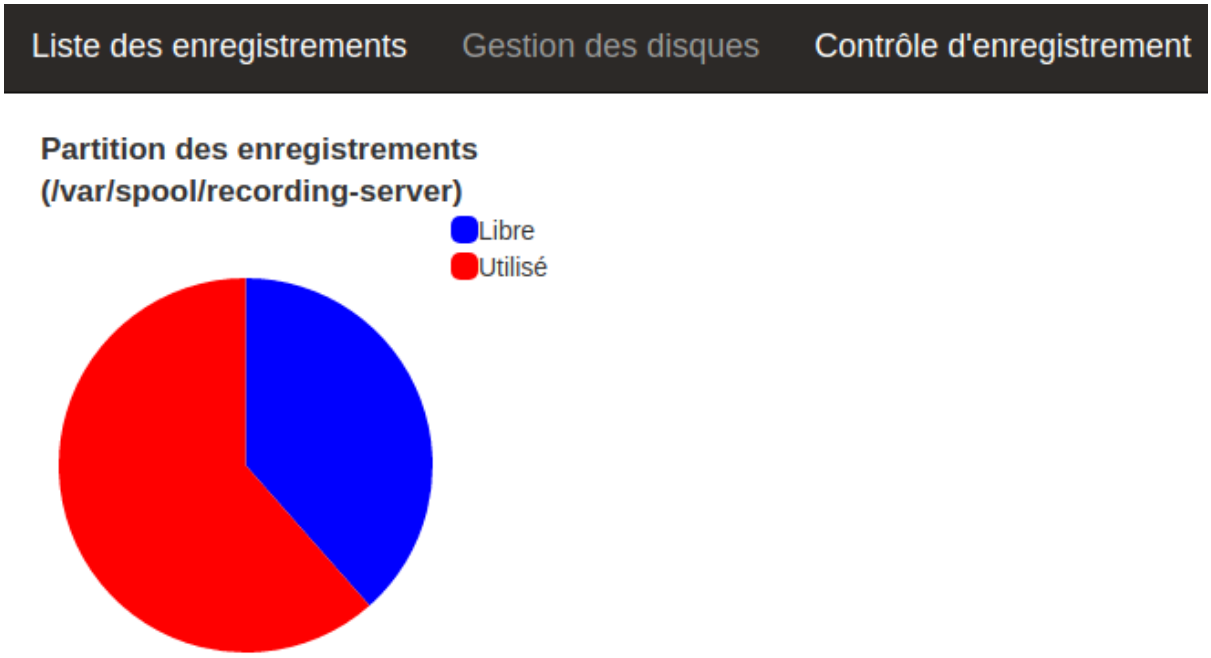
You can then do following actions:

- Find **answered** recorded calls
  - By caller, callee by using exact match pattern or using % character as a wildcard. (e.g. 10% to search for all numbers starting with 10)
  - By agent or queue, either part of the name or full name, also possible to search by its number
- Find **any** recorded call by id

- Listen or download the recording
- Download access log to recording files

Disk Space

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.

File Sync

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.

Recording and Transfers

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

Automatic Stop/Start Recording On Queues

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	

This behavior can be deactivcated, see *configuration section*

Automatic 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*.

Recording filtering

To configure this feature, see *Recording filtering configuration*.

Description

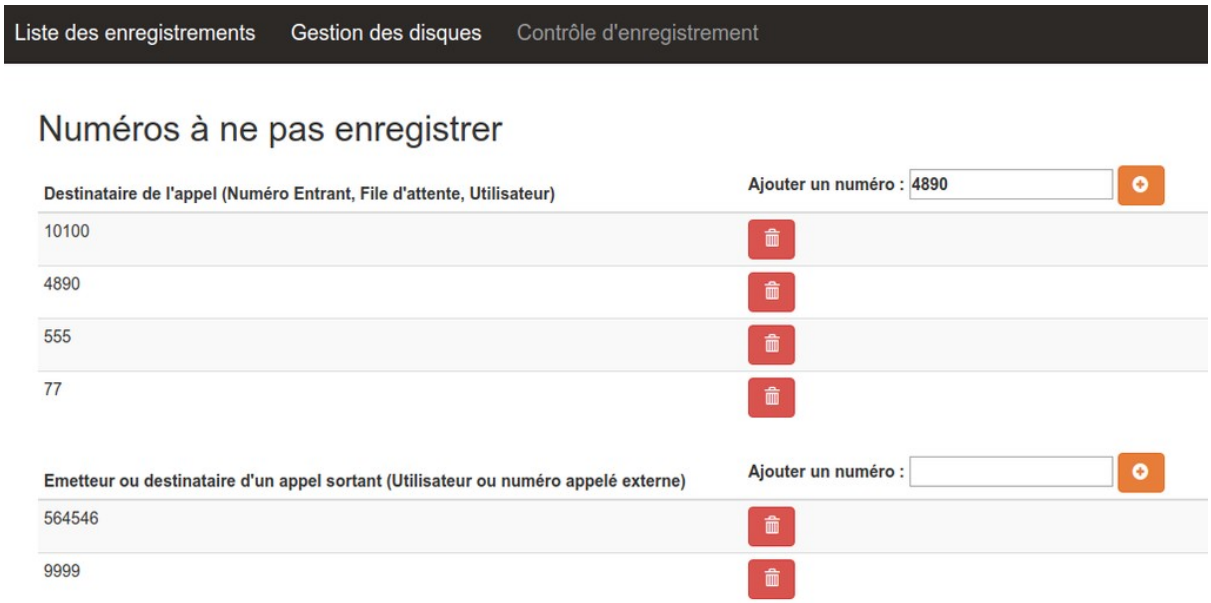
Recording server allows to prevent some numbers not to be recorded.

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).

Callbacks

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

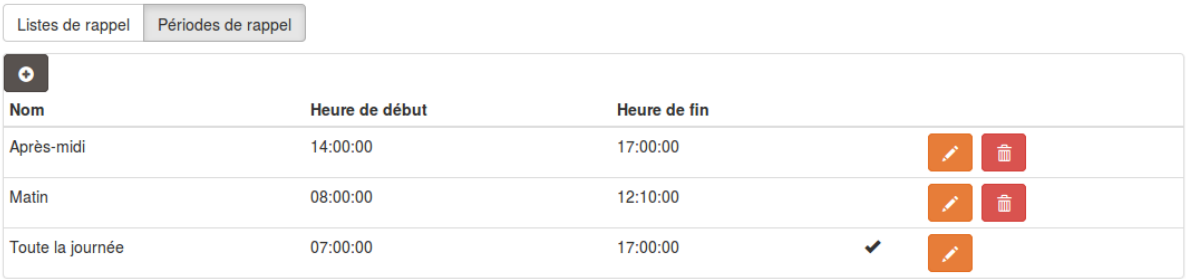
Callback Lists

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**, also known as **configmgt**

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.

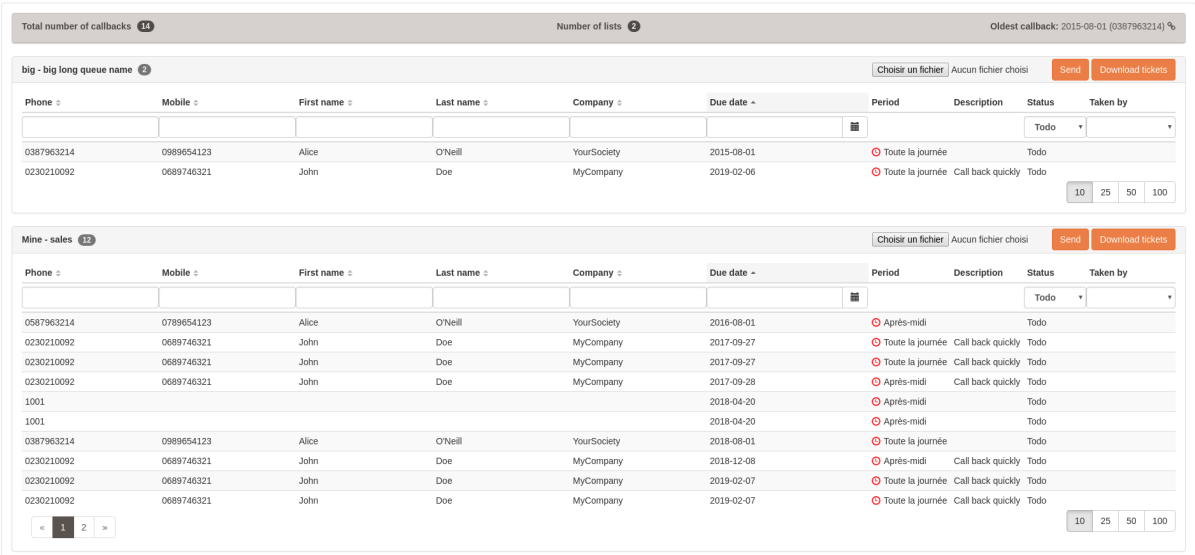
Callback Periods

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.



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.

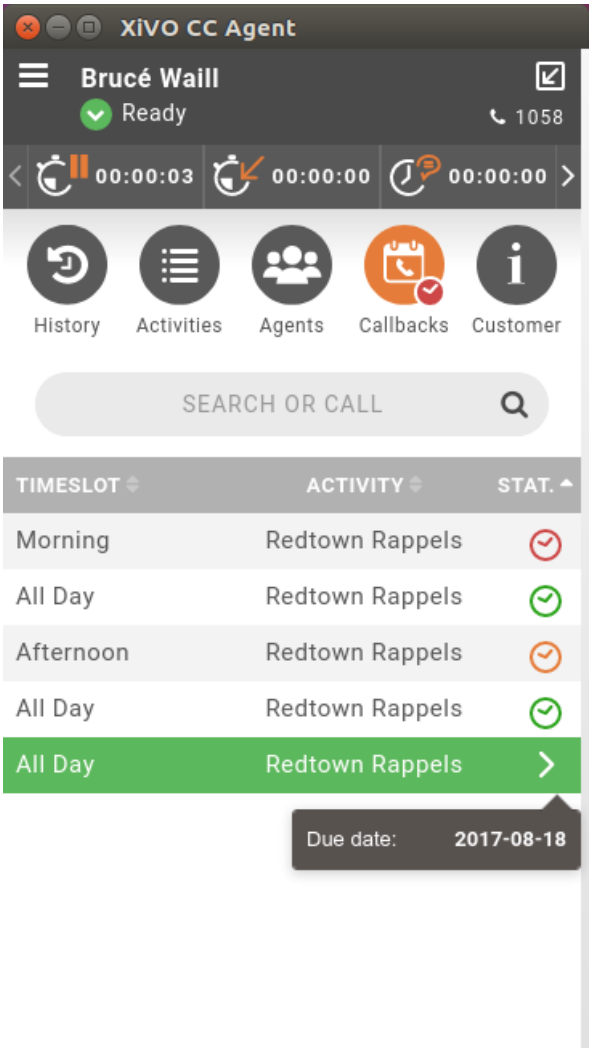
The downloaded file is a csv file with the comma ',' as delimiter.

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:

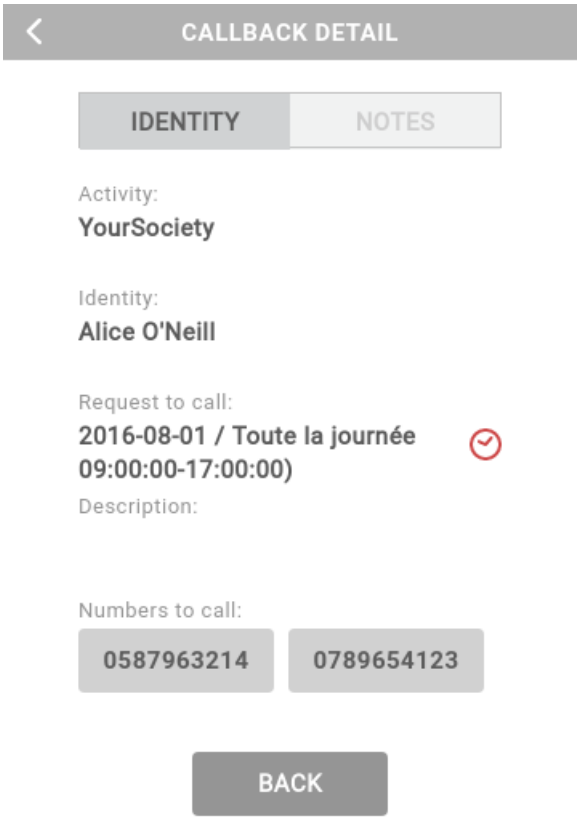
- *blue* if the callback is to be processed later
- *green* if we are currently inside the callback period
- *red* if the callback period is over



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:

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.

Profile Management

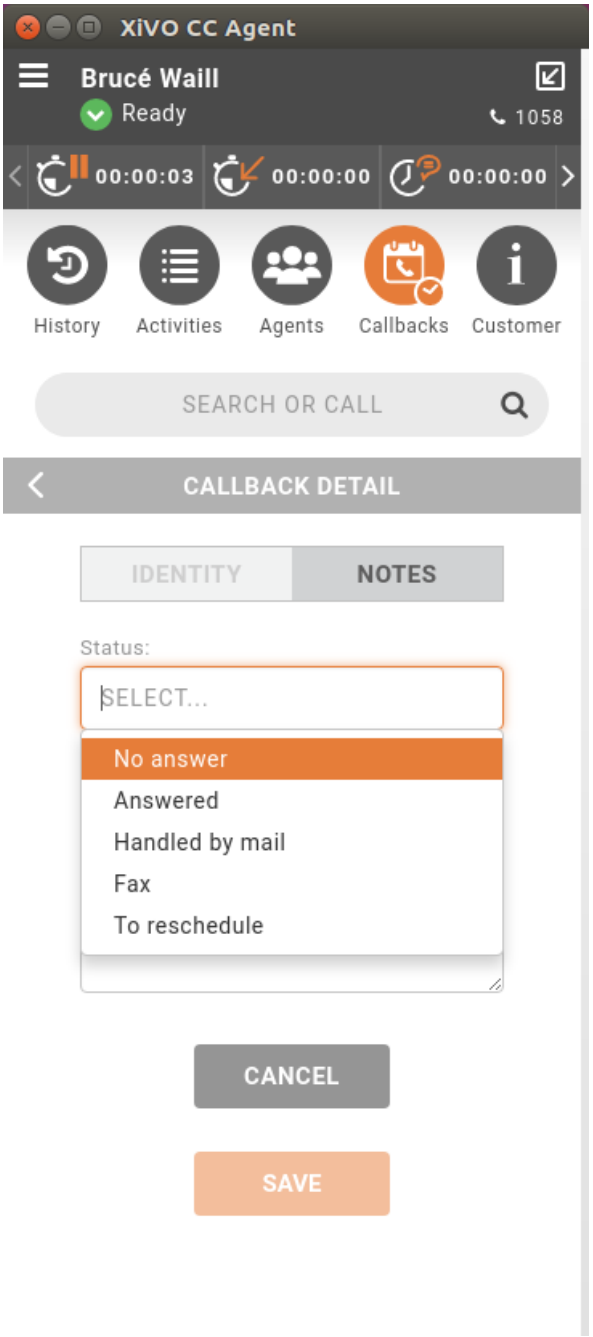
With what is called the Configuration Management Server one can specify the profile of a user.

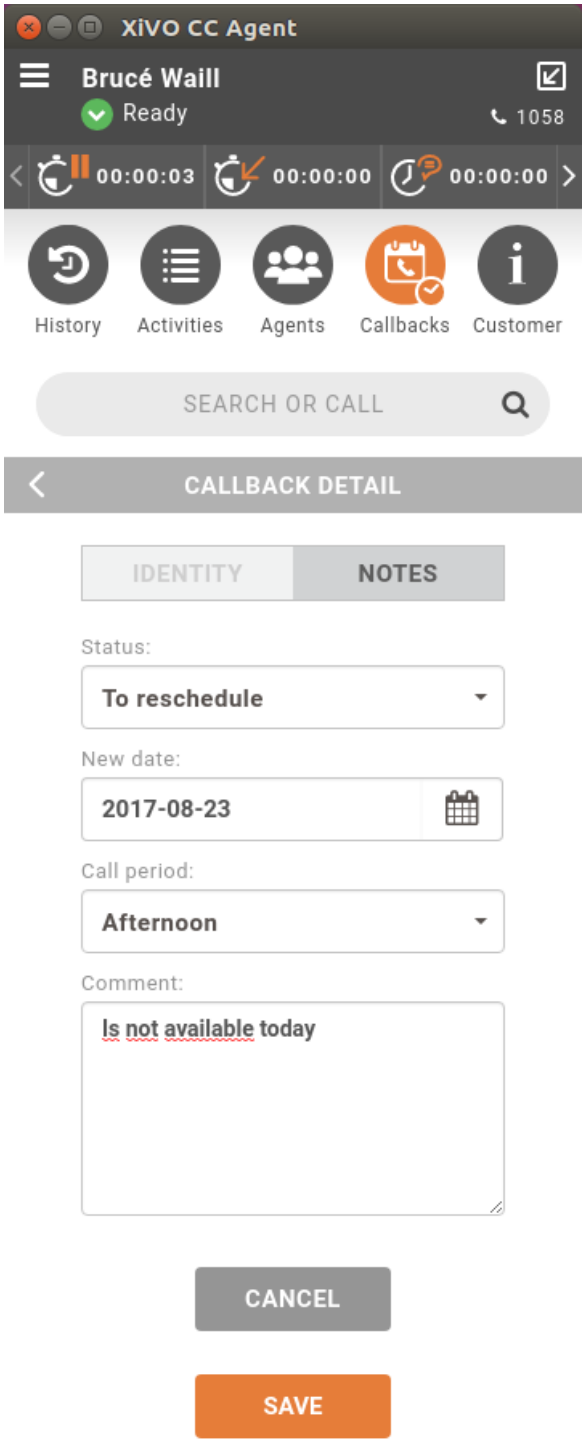
To do that one has to:

- 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*)


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

Profiles Definition

Profiles and their rights are summed up in the following table:

Profile	Application **CC Manager** *Access*	*Actions*	**Recording** *Access*	*Actions*
Administrator	Yes	All	Yes	• all recording • recordings filtering
Supervisor[Page 453, 4] • w/ *Recording*	Yes	• All on its queues[1] • Recording switch • Create/delete teachers	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.

Also administrators and supervisors with "dissuasion" access right can change the exceptional closing dissuasion destination when logged in webagent or in the CCManager. see *Activity's Failed Destination Configuration*.

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)

[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

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.

Skills-Based Routing

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.

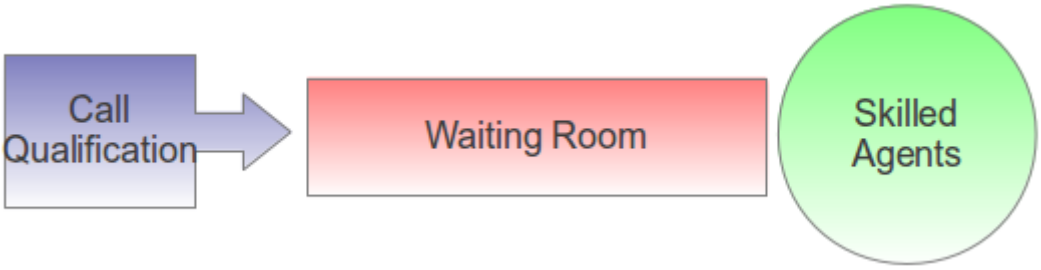


Fig. 59: Skills-Based Routing

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.

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).

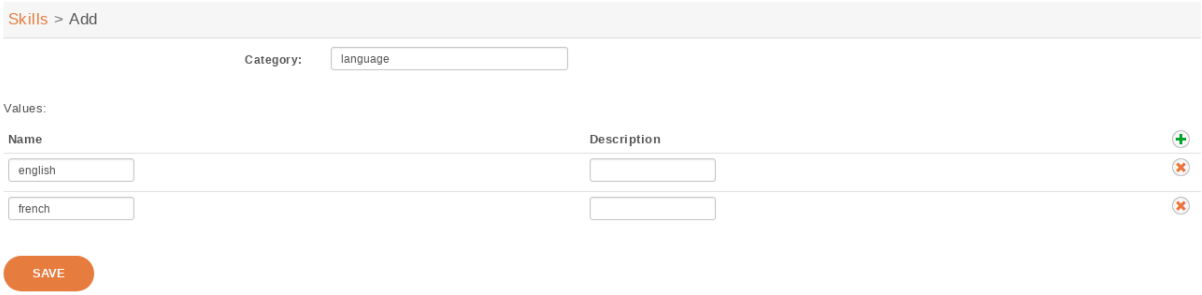


Fig. 60: Skills Creation

Once all the skills are created you may apply them to agents. Agents may have one or more skills from different categories.

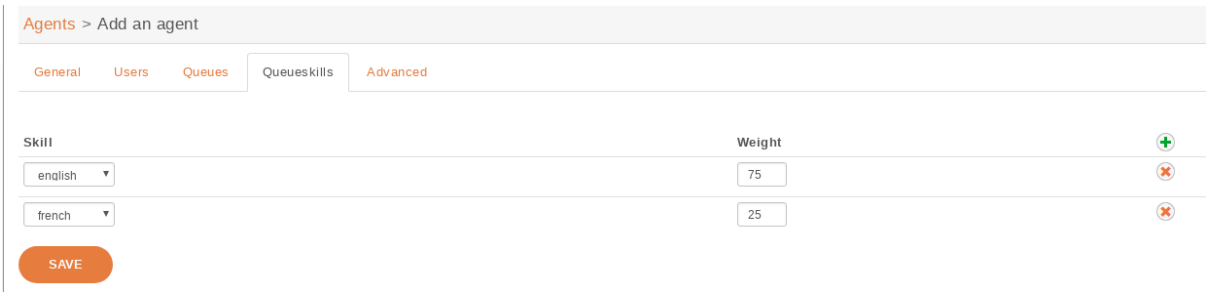


Fig. 61: Apply Skills to Agents

It is typical to use a value between 0 and 100 inclusively as the weight of a skill, although any integer is accepted.

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'.

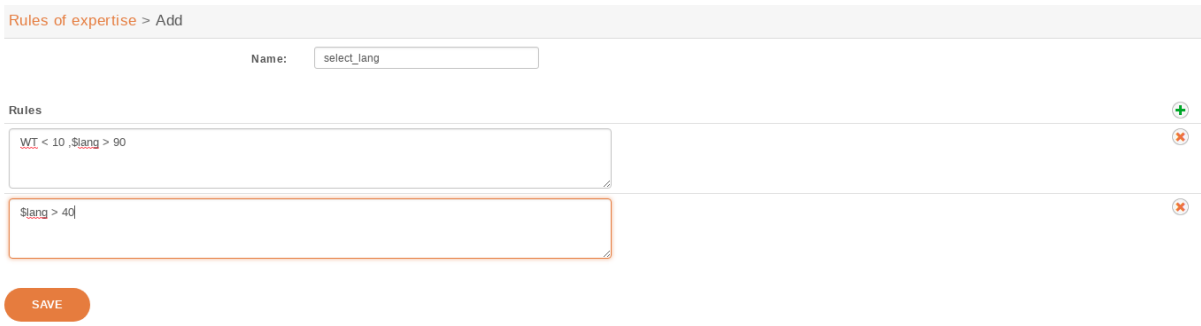


Fig. 62: Create Skill Rule Sets

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)

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*.

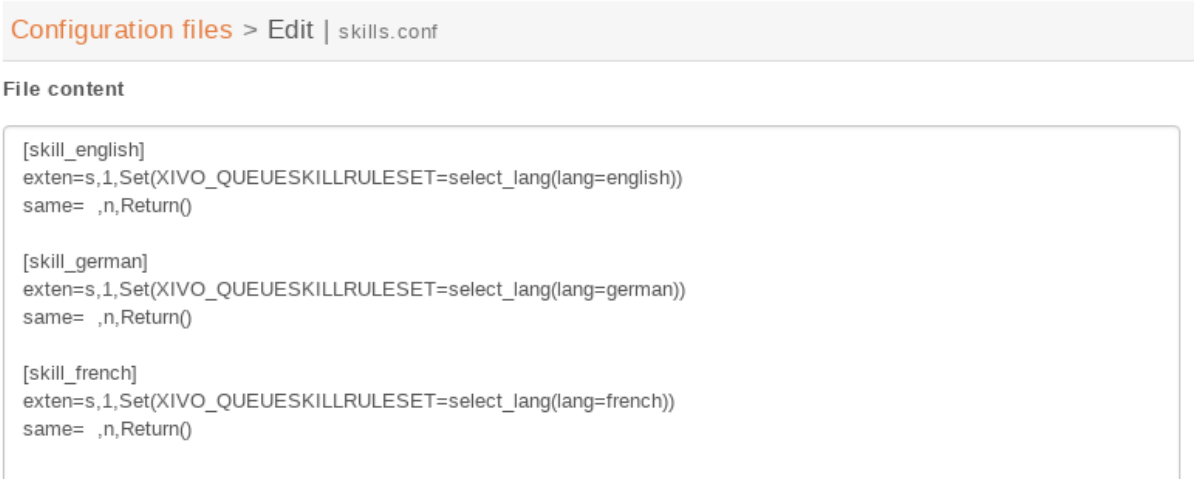


Fig. 63: Use Rule Set In Dialplan

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 :

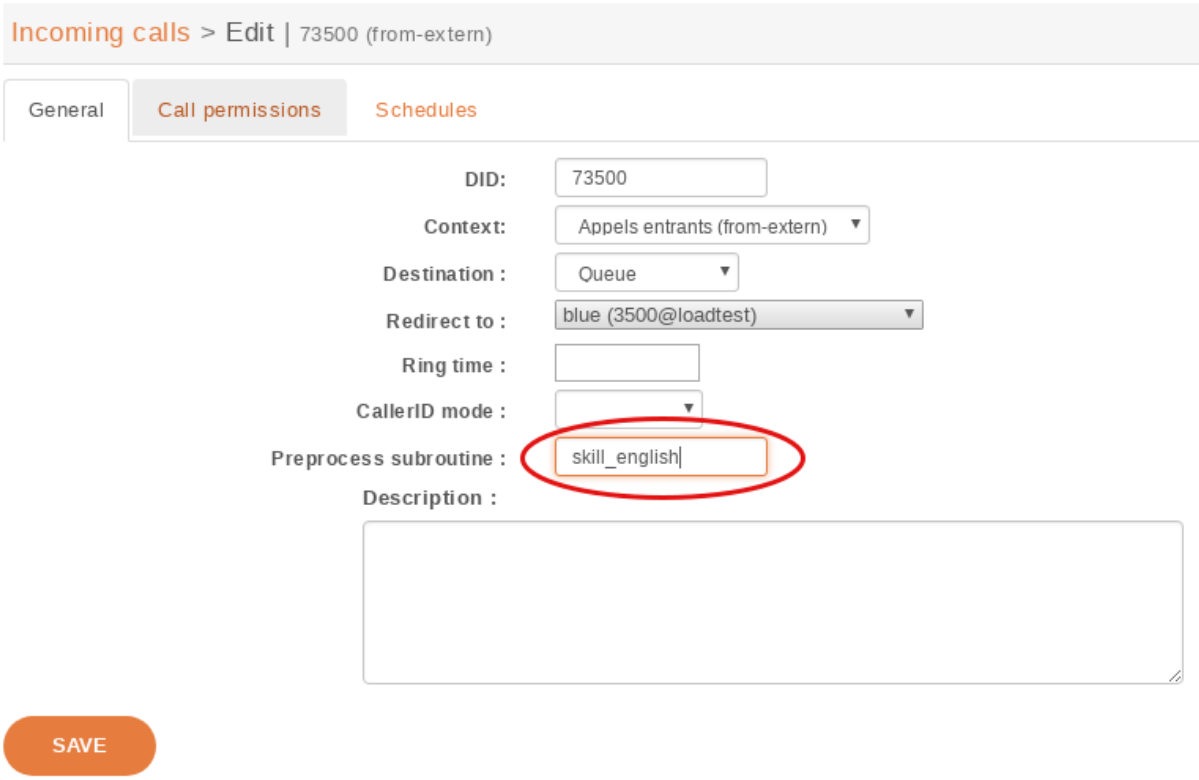


Fig. 64: Apply Rule Set to Incoming Call

```
[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()
```

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':
  => english>90
```

Reporting and statistics

Introduction

Pack reporting is a part of the XivoCC. It aims at computing historical statistics, which are stored in the **xivo_stats** database. Interactive reports and dashboards are accessible through **Metabase**.

Important: For information on installing and configuring Metabase for reporting, see *Metabase*.

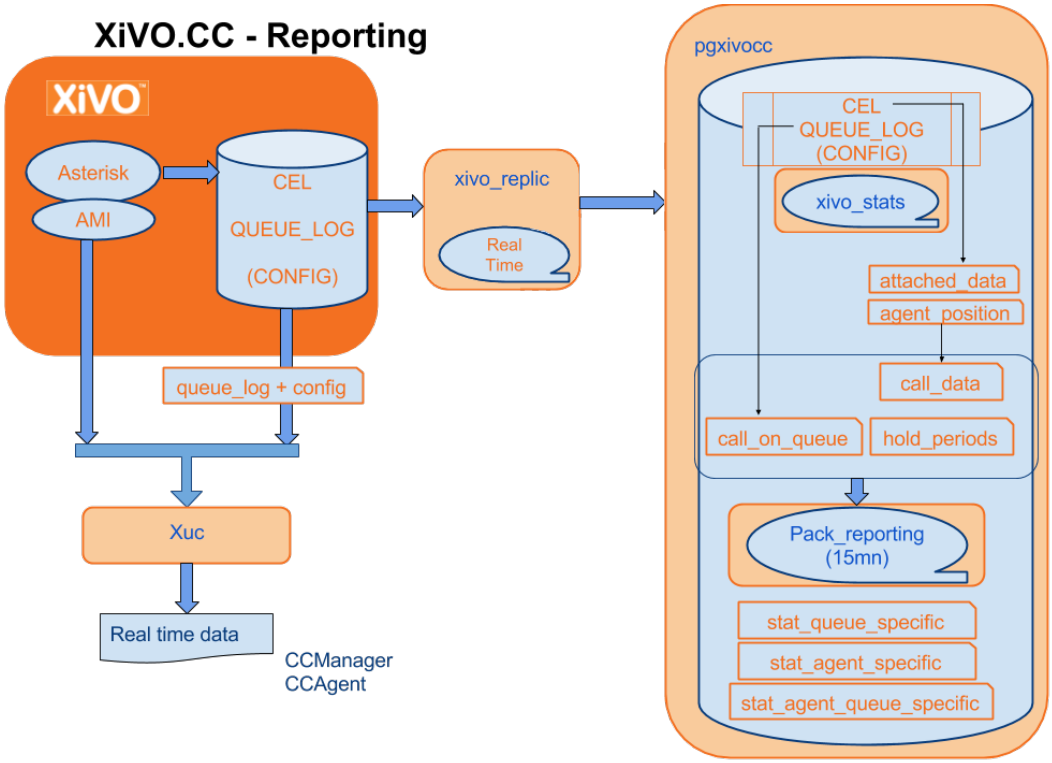
Important: New tables in **xivo_stats** schema are available (prefixed with **xc_**). These tables are (so far) for internal use and can be changed without notice. Documentation will be available once this new statistic model will replace the existing one and fix the existing limitations of previous model described above.

Reporting Architecture

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 the 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



Queue statistics

It summarizes how we sort and name every call to a queue, reaching different states.

These states are based on asterisk queue logs event types, you should find more infos about them here : https://docs.asterisk.org/Operation/Logging/Queue-Logs/?h=queue+log
The CCM icon means the value is readable on the CC Manager (see *Queue view*).

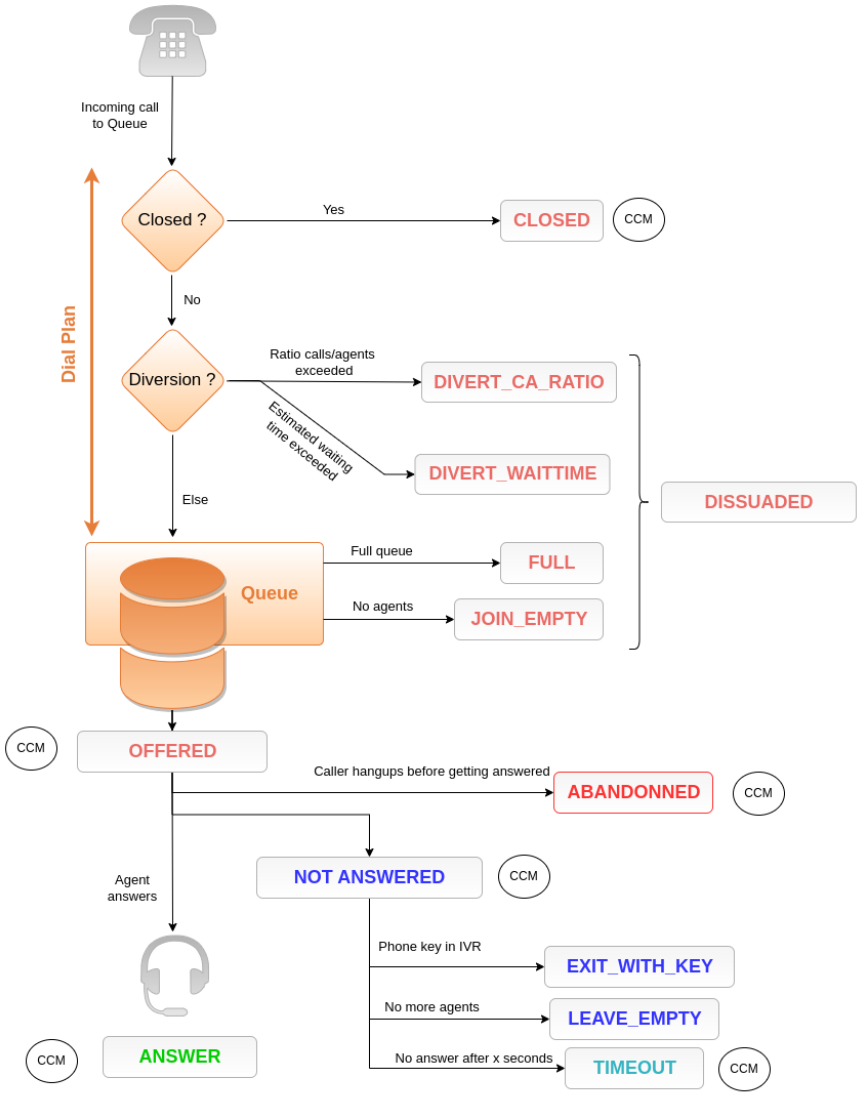
Agents Statistics

A call may be submitted to an agent several times. Thus a call 'not answered' by an agent may still be answered by another. This may be more explicit through an example below.

An offered call can be submitted several times.

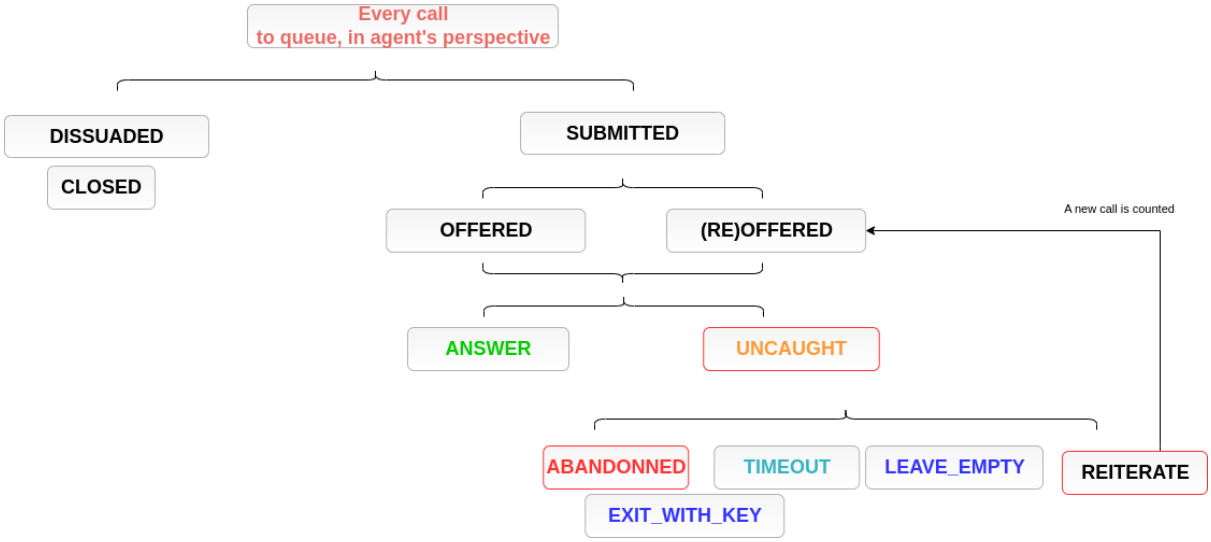
Known limitations

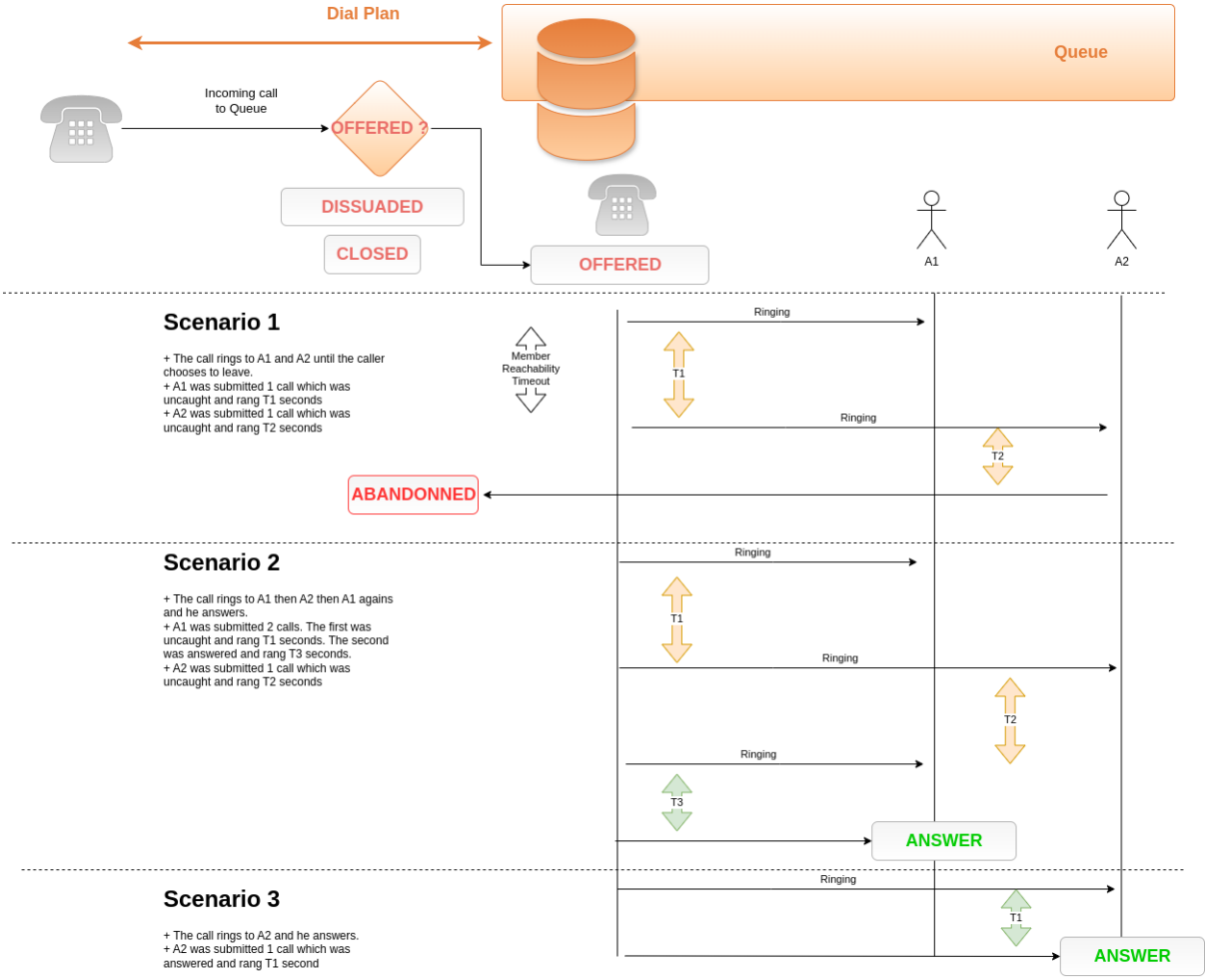
- Queue members should only be agents. If users are members of a queue, their statistics will be incomplete.
- Configuration modifications on the XiVO (such as an agent deletion) are replicated on the statistics server, and their previous value is not kept. However, statistics history is preserved.
- 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 in *call_data* table:
  - Given two queues Q1 and Q2, two agents A1 and A2, and an external caller C.



Every calls to queue

- CLOSED
- DISSUADED = DIVERT_CA_RATIO + DIVERT_WAITTIME + FULL + JOIN_EMPTY
- OFFERED = ANSWER + ABANDONNED + NOT ANSWERED = TIMEOUT + LEAVE_EMPTY + EXIT_WITH_KEY


Every call
to queue, in agent's perspective
DISSUADED
CLOSED
SUBMITTED
OFFERED
(RE)OFFERED
A new call is counted
ANSWER
UNCAUGHT
ABANDONNED
TIMEOUT
LEAVE_EMPTY
REITERATE
EXIT_WITH_KEY


Dial Plan
Queue
Incoming call
to Queue
OFFERED ?
DISSUADED
CLOSED
OFFERED
A1
A2
Scenario 1
+ The call rings to A1 and A2 until the caller chooses to leave.
+ A1 was submitted 1 call which was uncaught and rang T1 seconds
+ A2 was submitted 1 call which was uncaught and rang T2 seconds
Member Reachability Timeout
Ringing
T1
Ringing
T2
ABANDONNED
Scenario 2
+ The call rings to A1 then A2 then A1 agains and he answers.
+ A1 was submitted 2 calls. The first was uncaught and rang T1 seconds. The second was answered and rang T3 seconds.
+ A2 was submitted 1 call which was uncaught and rang T2 seconds
Ringing
T1
Ringing
T2
Ringing
T3
ANSWER
Scenario 3
+ The call rings to A2 and he answers.
+ A2 was submitted 1 call which was answered and rang T1 second
Ringing
T1
ANSWER

    * 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.

Using Metabase for Reporting

Metabase provides interactive business intelligence and reporting capabilities for XiVOCC contact center statistics.

Important: SpagoBI has been deprecated starting with Maia (2024.05) and replaced by Metabase. For installations using SpagoBI, migration to Metabase is recommended.

Statistics computed in real time are available in the ccmanager (see *Queue view*).
For historical statistics and custom reports, use Metabase (see *Metabase*).

Metabase Features

- **Interactive Dashboards**: Pre-configured dashboards for queue and agent statistics
- **Custom Reports**: Build custom queries and visualizations
- **Scheduled Reports**: Email delivery of reports on a schedule
- **LDAP Authentication**: Enterprise authentication support
- **Multi-language**: Interface available in multiple languages

Installation and Configuration

For complete installation and configuration instructions, see *Metabase*.

Quick reference:

1. Install the package: `apt-get install xivocc-metabase`
2. Start the service: `xivocc-dcomp up -d`
3. Access Metabase at `http://XIVOCC_IP:3000`
4. Configure the admin user and connect to the **xivo_stats** database
5. Import pre-configured reports (see *Re-running the Import*)

Available Reports

The pre-configured report collections include:

- **Queue Statistics**: Call metrics, wait times, service levels, abandonment rates
- **Agent Reports**: Activity tracking, login/pause/wrapup time, call handling statistics
- **Performance Dashboards**: Real-time monitoring and historical trends

For the complete database schema and statistics model, see the sections below.

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 :

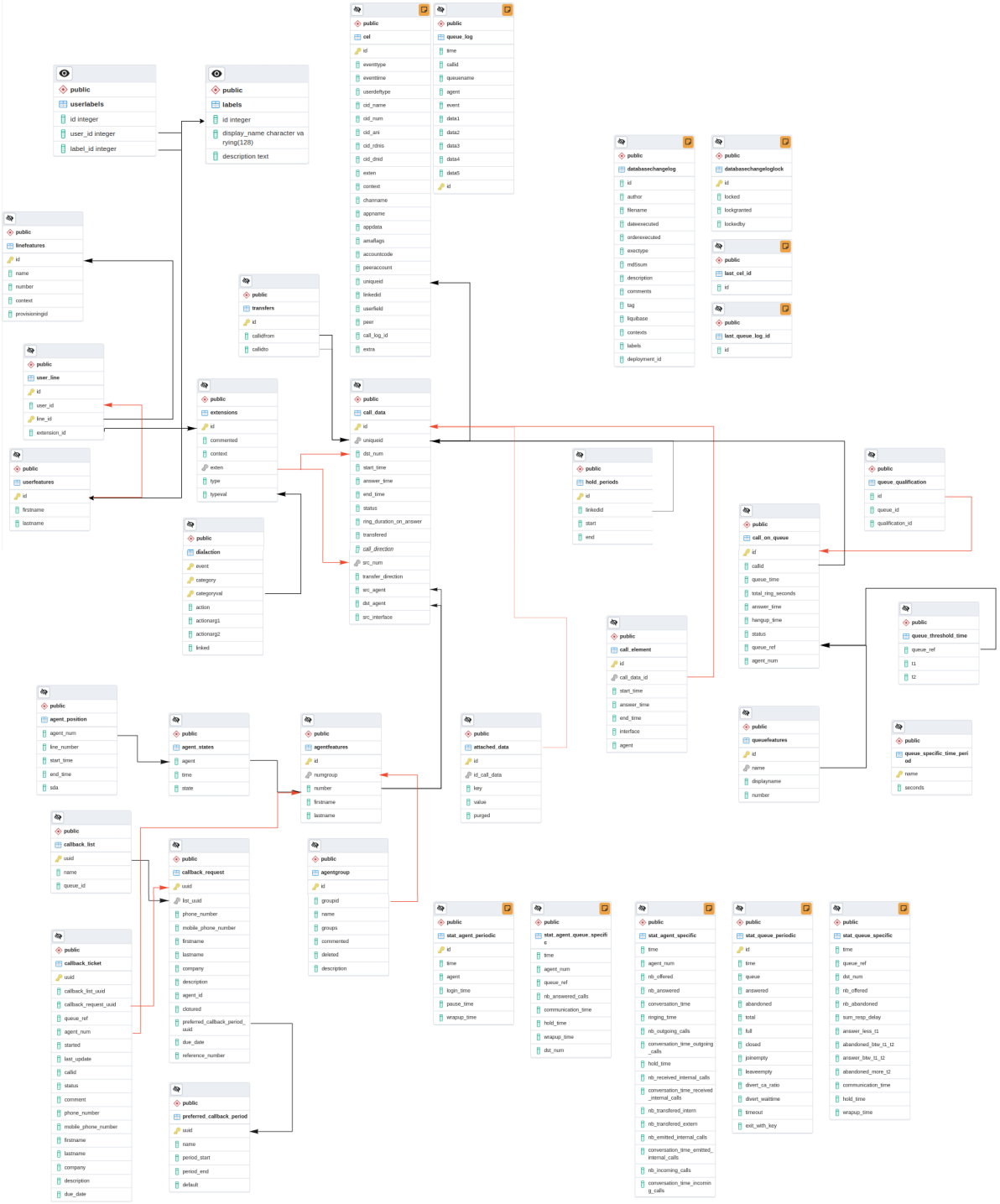
- the column *linkedId* in *cel* Asterisk table contains a uniqueId which links the two legs of the call.
- The leg which initiates the call (**A** leg) will have the same value for both *uniqueId* and *linkedId* columns.
- The other leg (**B** leg) will have the uniqueId of **B** in *uniqueId* column and will have the uniqueId of **A** in *linkedId* one.

Overview

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.


public
cel
id
eventtype
eventtime
userdefType
cid_name
cid_num
cid_ani
cid_rdnis
cid_dnid
exten
context
channame
appname
appdata
amaflags
accountcode
peeraccount
uniqueid
linkedid
userfield
peer
call_log_id
extra
public
queue_log
time
callid
queuename
agent
event
data1
data2
data3
data4
data5
id
public
labels
id integer
display_name character varying(128)
description text
public
userlabels
id integer
user_id integer
label_id integer
public
linefeatures
id
name
number
context
provisioningid
public
transfers
id
callidfrom
callidto
public
user_line
id
user_id
line_id
extension_id
public
extensions
id
commented
context
exten
type
typeval
public
call_data
id
uniqueid
dst_num
start_time
answer_time
end_time
status
ring_duration_on_answer
transfered
call_direction
src_num
transfer_direction
src_agent
dst_agent
src_interface
public
userfeatures
id
firstname
lastname
public
dialaction
event
category
categoryval
action
actionarg1
actionarg2
linked
public
hold_periods
id
linkedid
start
end
public
queue_qualification
id
queue_id
qualification_id
public
call_on_queue
id
callid
queue_time
total_ring_seconds
answer_time
hangup_time
status
queue_ref
agent_num
public
call_element
id
call_data_id
start_time
answer_time
end_time
interface
agent
public
queue_threshold_time
queue_ref
t1
t2
public
agent_position
agent_num
line_number
start_time
end_time
sda
public
agent_states
agent
time
state
public
agentfeatures
id
numgroup
number
firstname
lastname
public
attached_data
id
id_call_data
key
value
purged
public
queuefeatures
id
name
displayname
number
public
queue_specific_time_period
name
seconds
public
callback_list
uuid
name
queue_id
public
callback_request
uuid
list_uuid
phone_number
mobile_phone_number
firstname
lastname
company
description
agent_id
clotured
preferred_callback_period_uuid
due_date
reference_number
public
agentgroup
id
groupid
name
groups
commented
deleted
description
public
callback_ticket
uuid
callback_list_uuid
callback_request_uuid
queue_ref
agent_num
started
last_update
callid
status
comment
phone_number
mobile_phone_number
firstname
lastname
company
description
due_date
public
preferred_callback_period
uuid
name
period_start
period_end
default
public
stat_agent_periodic
id
time
agent
login_time
pause_time
wrapup_time
public
stat_agent_queue_specific
time
agent_num
queue_ref
nb_answered_calls
communication_time
hold_time
wrapup_time
dst_num
public
stat_agent_specific
time
agent_num
nb_offered
nb_answered
conversation_time
ringing_time
nb_outgoing_calls
conversation_time_outgoing_calls
hold_time
nb_received_internal_calls
conversation_time_received_internal_calls
nb_transfered_intern
nb_transfered_extern
nb_emitted_internal_calls
conversation_time_emitted_internal_calls
nb_incoming_calls
conversation_time_incoming_calls
public
stat_queue_periodic
id
time
queue
answered
abandoned
total
full
closed
joinempty
leaveempty
divert_ca_ratio
divert_waittime
timeout
exit_with_key
public
stat_queue_specific
time
queue_ref
dst_num
nb_offered
nb_abandoned
sum_resp_delay
answer_less_t1
abandoned_btw_t1_t2
answer_btw_t1_t2
abandoned_more_t2
communication_time
hold_time
wrapup_time

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_duratio	INTEGER	Ring time of the endpoint answering the call, in seconds
transfered	BOOLEA	True if the call has been transfered
call_directio	call_direc	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_directio	call_direc	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_interfac	VARCHAR	Interface originating the call (in the Asterisk sense, ex : SIP/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

The reporting allows to attach as much 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

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	VARCHA	Call unique reference, generated by Asterisk
queue	TIMESTAM	Time of entrance in the queue
total_rin	INTEGER	Total ring time, in seconds (includes ringing of non-answered calls)
answer_	TIMESTAM	Answer time
hangu	TIMESTAM	Hangup time
status	call_	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	VARCHA	Technical queue name
agent_	VARCHA	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

transfers

Table that follow transfers between calls

Column	Type	Description
id	INTEGER	id of the transfer
callidfrom	VARCHAR	initial call
callidto	VARCHAR	destination call

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_waittim	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

queuefeatures

Gather information about queue

Column	Type	Description
id	INTEGER	Queue id
name	VARCHAR	Technical name
displayname	VARCHAR	Display name
number	VARCHAR	Number to reach the queue

userfeatures

Gather information about user profile

Column	Type	Description
id	INTEGER	User id
firstname	VARCHAR	User first name
lastname	VARCHAR	User last name

linefeatures

Gather information about line used by a user

Column	Type	Description
id	INTEGER	Line id
name	VARCHAR	SIP interface name
number	VARCHAR	Phone number
context	VARCHAR	Line context
provisioningid	INTEGER	Number to provision a device

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

agent_states

Last state known for an agent

Column	Type	Description
agent	VARCHAR	Agent number
time	TIMESTAMP	Last state time
state	agent_state_type	logged_on, logged_off, paused, wrapup

agentgroup

Agent group ownership

Column	Type	Description
id	INTEGER	Agent group Id
groupid	INTEGER	*Deprecated* use id
name	VARCHAR	Name of the group
groups	VARCHAR	*Deprecated*
commented	INTEGER	know if group is disabled
deleted	INTEGER	know if group is deleted
description	TEXT	description set in XiVO

dialaction

Action to trigger when a number (extension) is matching some criteria (congestion, busy, no answer…)

Column	Type	Description
event	VARCHAR	Type of criteria (answer, noanswer, busy, congestion, chanunavail, qwaittime, qwaitratio)
category	VARCHAR	user, meetme, queue, incall, group, callfilter…
categoryval	VARCHAR	Id applying to the category (user id if category is user)
action	VARCHAR	What to trigger (can be **custom** or any existing **category** defined above)
actionarg1	VARCHAR	Value associated to action. e.g. if queue must be called, arg1 will contain the queue id
actionarg2	VARCHAR	Rarely used to define a second value if category chosen need it
linked	INTEGER	*Deprecated*

extensions

List of all numbers that you can call within XiVO

Column	Type	Description
id	INTEGER	Extension Id
commented	INTEGER	Define if extension is disabled
context	VARCHAR	Context of your extension (xivo-features are hardcoded ones)
exten	VARCHAR	Number to reach the extension
type	VARCHAR	user, meetme, queue, incall, group, extenfeatures…

user_line

Mapping table between a user, its line and associated extension

Column	Type	Description
id	INTEGER	User line association id
user_id	INTEGER	User id
line_id	INTEGER	Line id
extension_id	INTEGER	Extension id

labels

Defines the list of labels that can be set to users

Column	Type	Description
id	INTEGER	Label id
display_name	VARCHAR	Name displayed for the label
Description	TEXT	description set in XiVO

userlabels

Defines the association between labels and users

Column	Type	Description
id	INTEGER	id of the association
user_id	INTEGER	User Id
label_id	INTEGER	Label Id

cel & queue_log

These tables are generated by Asterisk and cloned in stats repository, for more information about it see https://wiki.asterisk.org/wiki/pages/viewpage.action?pageId=5242932

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

—

Tables **agentgroups** and **agentfeatures** can be linked together by doing a join on a column holding the agent group id reference. The columns are the following:

Table	Reference column
agentfeatures	numgroup
agentgroups	id

—

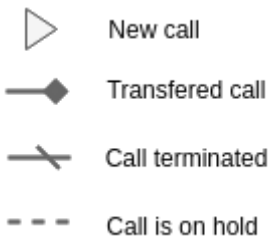
Tables **transfers** and **call_data** can be linked together by doing a join on a column holding the call uniqueid reference. The columns are the following:

Table	Reference column
transfers	callidfrom / callidto
call_data	uniqueid

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:

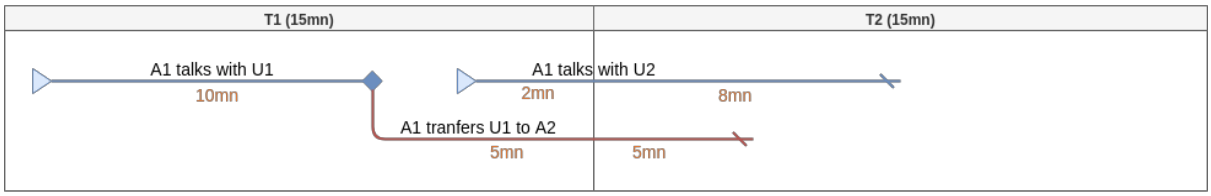
id	linkedId	start	end
1	12345678.9	2018-12-01 15:13:00	2018-12-01 15:17:00

Statistics (done by pack-reporting for spago reports) generates then *stat_agent_specific*:

time	agent_num	nb_emitted_internal_c	conversa-tion_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_nun	answer_time	end_time	src_agei	dst_agei	trans-fered	trans-fer_direction
1	12345678	1001	2018-12-01 15:02:00	2018-12-01 15:12:00	8000	8002	true	internal
2	12345679	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*:

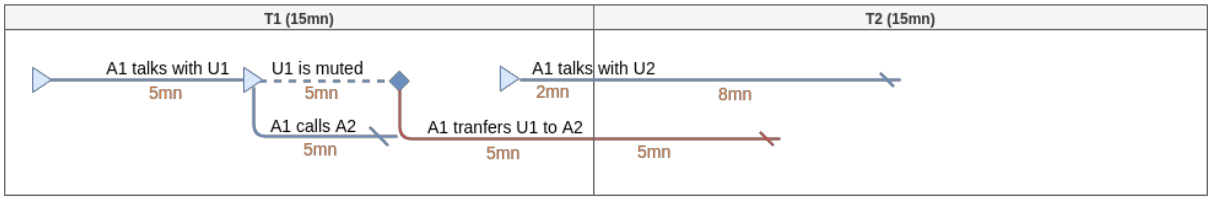
time	agent_nun	nb_emitted_internal_c	conversa-tion_time_emitted_internal_calls	hold_time
2018-11-29 15:15:00	8000	1	720	0
2018-11-29 15:30:00	8000	1	480	0

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_nun	answer_time	end_time	src_agen	dst_agen	trans-fered	trans-fer_direction
1	12345678	1001	2018-12-01 15:02:00	2018-12-01 15:12:00	8000	8002	true	internal
2	12345679	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_ca	conversa-tion_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.

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,
       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 for a queue

This query aggregates all received call by month and by queue 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')␣
```

(continues on next page)

(continued from previous page)

```
↪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	Mont	DID	Pre-sente	An-swere	Average Call Time	Dis-suade	Hang	Re-fusec	Aban-donnec	Average Waiting Time	Trans-fered	An-swered 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

Number of direct calls to user through its DID only

This query aggregates all received call through direct inward dial number. We are not counting here the internal calls of a user.

```
SELECT
              CASE
                      WHEN call_direction='outgoing'
                              THEN (select ext_a.typeval::integer from extensions ext_
↪a where ext_a.type = 'user' and ext_a.exten=src_num)
                      WHEN call_direction='incoming'
                              THEN (select dl_a.actionarg1::integer from dialaction␣
↪dl_a,extensions ext_b where  dl_a.category = 'incall'
                                      AND dl_a.action = 'user' and dl_a.categoryval =␣
↪ext_b.typeval AND ext_b.type = 'incall' AND dst_num IN (ext_b.exten, '+' || ext_b.
↪exten) )
                      ELSE 0
              END as user_id,
        sum(CASE WHEN call_direction= 'incoming' THEN 1 ELSE 0 END) AS nb_incoming_
↪offered,
        sum(CASE WHEN call_direction= 'incoming' and status = 'answer' THEN 1 ELSE 0␣
↪END) AS nb_incoming_answered,
        sum(CASE WHEN call_direction= 'incoming' THEN 1 ELSE 0 END)-sum(CASE WHEN␣
↪call_direction= 'incoming' and status = 'answer' THEN 1 ELSE 0 END) as nb_incoming_
↪missed
FROM
        call_data
GROUP BY user_id
```

This query will result to something like:

user_id	nb_incoming_offered	nb_incoming_answered	nb_incoming_missed
1	24	18	6

4.3 XiVO Connect Administration

4.3.1 XiVO Connect Administration

This document describes how to administer **XiVO Connect** (a sorft fork of Element Web) using the **Matrix Synapse** backend and the **Synapse-Admin** interface.

It focuses exclusively on:

- User Management
- Roles & Permission Model
- Moderation & Room Management

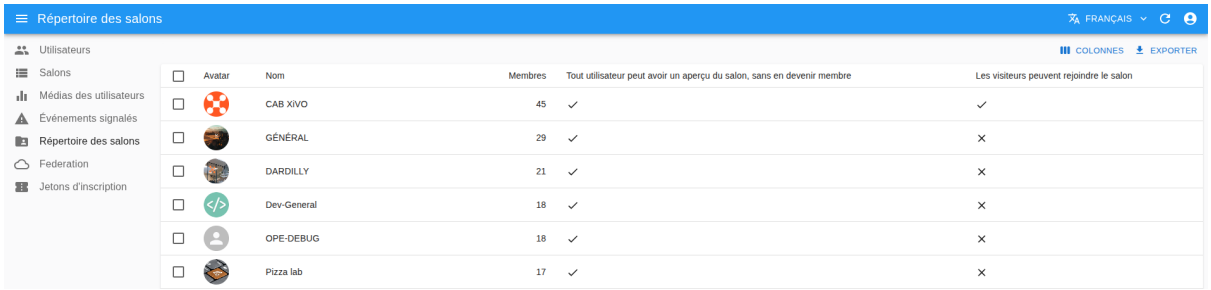
Contents:

- *1. Application Administration*
  - *1.1 User Management*
  - *1.2 Role & Permission Model*
  - *1.3 Moderation & Room Management*

1. Application Administration

1.1 User Management

User accounts are managed on the **Synapse homeserver**. Administrators can use **Synapse-Admin** (web UI) or the **Synapse Admin API** to manage user lifecycle and permissions that can be found at: https://admin.your-homeserver

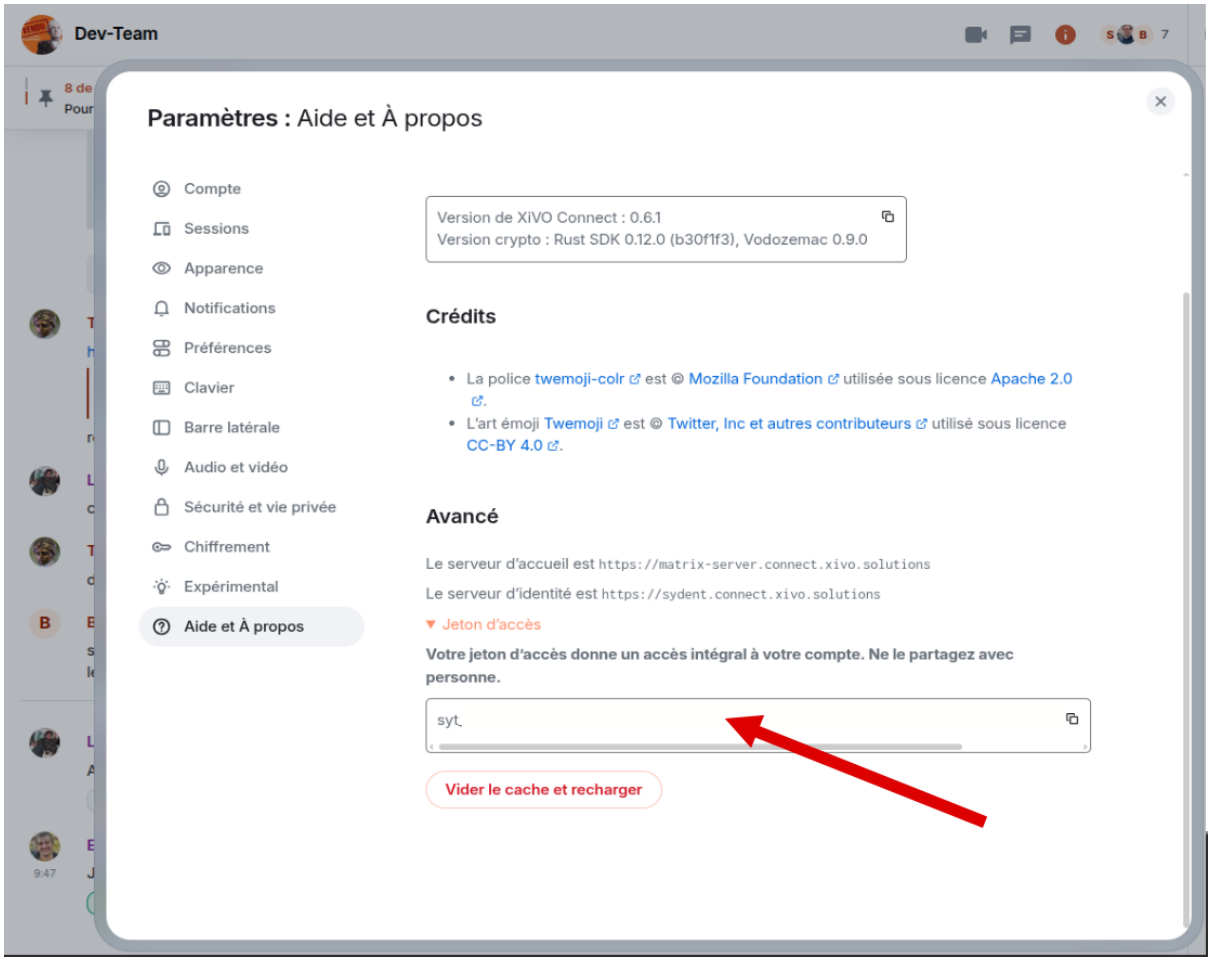


Available management capabilities include:

- Listing all registered accounts
- Creating new user accounts
- Resetting or updating passwords
- Deactivating or reactivating users
- Managing user devices and access tokens
- Editing user attributes (admin flag, display name)

- Forcing logout from all devices

If you don't want UI, you can use the Synapse Admin REST API directly through CLI. To be able to use it, you need to be an admin user and have an access token to be used as **<ADMIN_TOKEN>** that can be found in your Connect profile.



Example: **Reset a user's password via API**

```
curl -XPOST -H "Authorization: Bearer <ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{"password": "NewPassword123"}' \
  "https://your-homeserver/_synapse/admin/v1/reset_password/@user:domain"
```

Example: **Force logout of all user's devices**

```
curl -XDELETE -H "Authorization: Bearer <ADMIN_TOKEN>" \
  "https://your-homeserver/_synapse/admin/v2/users/@user:domain/devices"
```

1.2 Role & Permission Model

Matrix uses a **Power Level (PL)** system to control permissions in rooms. These numeric levels define what a user can or cannot do.

Common values:

- **100** – Room Administrator (Owner)
- **50** – Moderator
- **0** – Standard user
- **Custom** – Optional intermediate permissions

Power levels control:

- Ability to kick, ban, mute other users
- Editing room settings (name, avatar, topic)
- Managing membership (invites, joins)
- Sending restricted event types
- Changing history visibility
- Managing encryption settings

1.3 Moderation & Room Management

Moderation tasks in XiVO Connect involve both user-level and room-level controls.

Typical moderation actions:

- Reviewing and resolving moderation reports
- Kicking or banning users from specific rooms
- Globally banning abusive accounts
- Muting users (prevent sending messages)
- Taking ownership of abandoned rooms
- Editing room visibility (public, private, restricted)
- Configuring join rules and membership policies

Example: **Global user ban**

```
curl -XPOST -H "Authorization: Bearer <ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{ "user_id": "@user:domain" }' \
  "https://your-homeserver/_synapse/admin/v2/banned"
```

Example: **Take ownership of a room with no active administrator**

```
curl -XPOST -H "Authorization: Bearer <ADMIN_TOKEN>" \
  "https://your-homeserver/_synapse/admin/v1/rooms/!roomid:domain/make_room_admin"
```

Room-level administration includes:

- Updating room name, topic, and avatar
- Configuring invites, access control, and membership rules
- Enabling or disabling encryption
- Viewing and exporting membership lists

- Cleaning up old or inactive rooms

4.4 Troubleshooting

Important:

- When reading this section, keep in mind the *XiVO CC Architecture* diagram.
- If you want to troubleshoot your installation see *XiVOcc Installation Troubleshooting*
- For desktop application see *Troubleshoot Application*

4.4.1 IPBX 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'
```

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]
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
```

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_ /var/lib/postgresql/17/data/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-17-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 "/var/lib/postgresql/17/data/postgresql.conf" contains␣
↪errors
```

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 `/var/lib/postgresql/17/data/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 `/var/lib/postgresql/17/data/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.

Replication issues on a PostgreSQL instance

If your XiVO MDS replication is broken or stuck, you can recreate the replication slot from the main XiVO server and force a full resynchronization of the MDS database.

> **Warning:** This procedure will completely remove the PostgreSQL data directory on the MDS node and trigger a full database resynchronization from the main XiVO server.
>
> Before starting, you should:
>
> make sure you have enough free disk space on both the main XiVO server and the MDS node. prepare for a temporary service interruption on the affected MDS node. verify that the PostgreSQL replication slot name matches the affected MDS instance.

On the XiVO main server

List the current replication slots:

```
SELECT * FROM pg_replication_slots ;
```

Remove the replication slot corresponding to the affected MDS:

```
SELECT pg_drop_replication_slot('main_mds01');
```

Replace main_mds01 with the actual replication slot name of your MDS node.

On the XiVO MDS server

Stop the PostgreSQL container:

```
xivo-dcomp stop db
```

Remove the PostgreSQL container:

```
xivo-dcomp rm db
```

Delete the PostgreSQL data directory:

```
rm -rf /var/lib/postgresql/17
```

Recreate and restart the PostgreSQL container:

```
xivo-dcomp up -d
```

The MDS node should now automatically perform a full PostgreSQL resynchronization from the main XiVO server.

Verification

You can verify that the replication is working again by checking the PostgreSQL logs on both servers and ensuring that the replication slot is recreated automatically on the main XiVO server.

On the XiVO main server, you can also verify the replication status with:

```
SELECT * FROM pg_replication_slots;
```

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 (it is by default)

Check *rtp show settings* in the asterisk console if it says otherwise.

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`

Call Permission and Transfers

Some Call Permission issues may occur in case of call transfers. For example:

- Given user U1 with call permissions C1,
- Given user U2 with another call permissions set C2,
- When U1 calls U2 and transfers it somewhere
- Then, depending on the type of transfer it will take the call permissions C1 or C2.

Current behavior is descrbided in bug 1944.

4.4.2 Xuc & Xucmgt (CC & UC embedded)

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
  [...]

  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.
```

(continues on next page)

(continued from previous page)

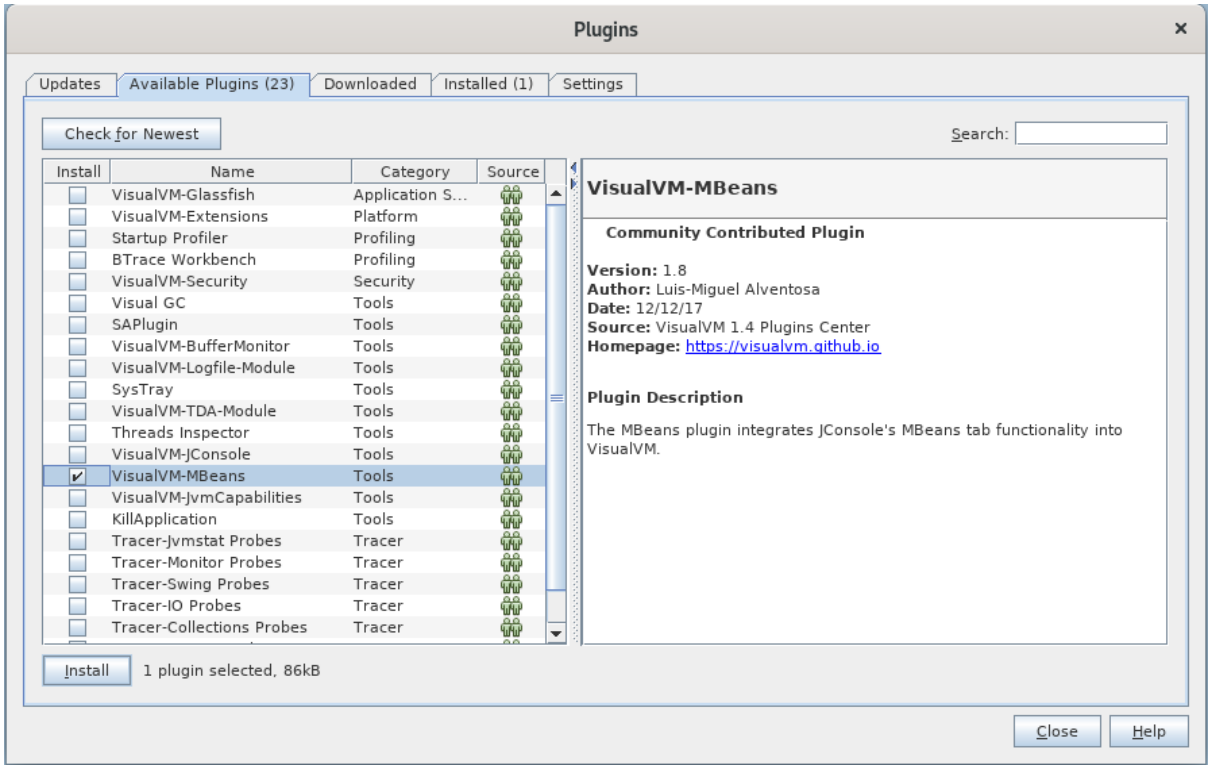
```
↪management.jmxremote.rmi.port=15701
  [...]
```

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

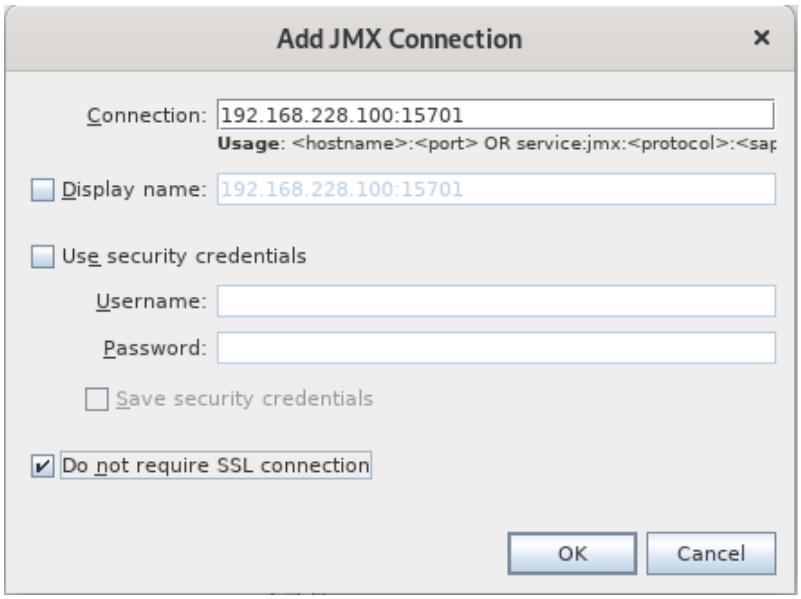


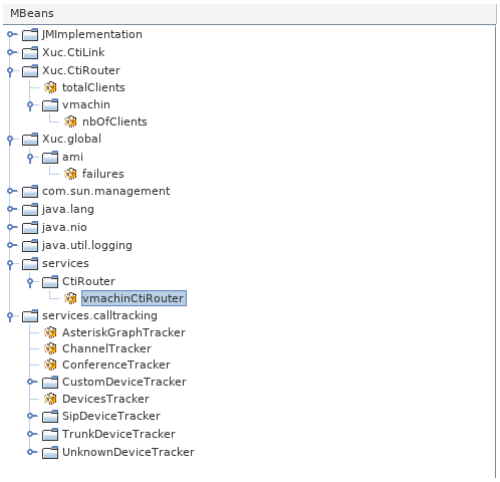
- Add remote host
- Double click on process under the newly added host
- Click on the MBeans tab
- Explore tree

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 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`:


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

```
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 and are now present in xuc application logs.

- `Xuc.CtiLink.<username>`: Information on the link per user between XUC and ctid on the XiVO PBX
  - `startConnection`: Represents the total number of WebSocket opening attempts (whether successful or not, and it only ever increments until the XUC is restarted)
  - `stopConnection`: Indicates the number of failed TCP connection attempts during these WebSocket openings
- `Xuc.CtiRouter.<username>.nbOfClients`: Shows the number of currently active WebSocket sessions for a user (normally increasing and decreasing as tabs are opened or closed)
- `Xuc.CtiRouter.totalClients`: Total number of websocket connected to the XUC (sum from previous counter)
- `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.

Queue created

New queue will not appear immediately in *CCAgent* and *CCManager* until the agent or the manager is not relogged to these applications accordingly. The Xuc server restart **is not required**.

Queue deleted

Deleted queue will still appear in *CCAgent* and *CCManager* until the Xuc server **is restarted**.

Audio quality issues

If something goes wrong with the quality of an ongoing call, it will be reflected in live, every 20 seconds, inside the xuc server logs, with the following format:

```
WARN  s.ClientLogger- [userw] 1590da900aba2a761573c0493c04a0a1@192.168.56.2:5060␣
↪Audio quality issues ->  RTT 186ms - Jitter upstream 1ms / downstream 0ms - Packet␣
↪loss upstream 21% / downstream 6%
```

At the end of each call, a call quality report will also be sent in the xuc server logs, with the following format:

```
INFO  s.ClientLogger- [userw] 1590da900aba2a761573c0493c04a0a1@192.168.56.2:5060 Call␣
↪quality report -> Highest RTT 186ms - Highest Jitter upstream 3ms / downstream 3ms -
↪ Highest Packet loss upstream 21% / downstream 12%
```

Both those format include the user as well as the sip call id, so it can be grepped. It can also be grepped by the type of message, "Audio quality issues" and "Call quality report".

Audio quality visualisation

When needed, a dedicated module can be setup to allow graphical representation of those data. See the following section for more details.

Activation

On XiVO PBX:

1. In `/var/lib/postgresql/17/data/pg_hba.conf` add rights for following users on specific tables (replace the <XUC_IP> with the IP of VM where xuc is running):

   ```
   host  call_quality_stats    rografana   <XUC_IP>/32    md5
   host  call_quality_stats    rwfluent    <XUC_IP>/32    md5
   ```

2. Add the service in file `/etc/docker/xivo/docker-xivo.grafana.yml` to `/etc/docker/xivo/docker-xivo.override.yml`.

3. And finally run:

   ```
   xivo-dcomp reload db
   xivo-dcomp up -d
   ```

On XiVO CC / UC :

1. Add the service in file `/etc/docker/compose/docker-xivocc.fluentd.yml` to `/etc/docker/xivo/docker-xivocc.override.yml`.

2. And then run:

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

Usage

When *audio quality visualisation activation* step is done.

You can access grafana web page via:

```
https://<XIVO_IP>:3000/
```

Replace the <XIVO_IP> with the IP of your Xivo PBX.

Dashboard view :

On grafana dashboard you can:

1. Select user that you want to display thank to the `users_list` parameter
2. Zoom out by pressing CRTL+Z or by clicking on the magnifier

> **Warning:** All the added dashboards will not be saved in case of update.

Xuc CLI

The xuc CLI is a small terminal tool that allows you to retrieve a bit of informations on the state of xuc internal tracking mechanics. You can compare them with your Asterisk state and clean them up if needed, which can troubleshoot some issues in production without the need to restart xuc.

The main command is the following : `xivocc-dcomp exec xuc bin/xuc-cli`

This command allows you to list the available command submodules and typing the command with one of those submodule allows you to list the available actions or run the command directly.

Here is a sum up of everything available (you need the xivocc-dcomp part mentionned above in front of each command) :

- `getversion`: Prints xuc version
- `starttime`: Prints xuc start time
- `info`: Prints start time, version, and agent / users available in xuc internal repository

- `mds YOURMDS`: Prints AMI infos and connectivity of YOURMDS (replace with the technical name of your mds)
- `siptracker YOURCONTACTID info`: Prints infos of YOURCONTACTID (replace with the contact name of your choice, retrieved from asterisk)
- `siptracker YOURCONTACTID clearcalls`: Clear the list of calls that is tracked internally by xuc. Can be used to fix ghost call and stuck transfert issues
- `siptracker YOURCONTACTID stop`: Stops the xuc internal call tracker for this specific contact. The user will need to refresh his assistant to initialize a clean one

4.4.3 NGINX - proxy web

Basic check

On the standard HTTP port of the machine (80) you are redirected to UC Assistant over HTTPS (443).

You can open the fingerboard page https://xivocc/fingerboard

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.

4.4.4 Reporting

DB 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. The following error can be found only in docker logs of DB Replic:

```
xivo-dcomp logs -tf --tail=10 db_replic
```

Error:

```
db_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
```

This 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. Stop *db_replic* (XiVO), *xivo_stats* (XiVO CC) and *pack_reporting* (XiVO CC)
2. On XiVOCC, enter inside the pgxivocc docker container:

   ```
   docker exec -it xivocc_pgxivocc_1 /bin/bash
   ```

3. Log on the database as postgres:

   ```
   su - postgres
   psql xivo_stats
   ```

4. 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)
```

5. If so, delete the lock:

```
xivo_stats=# truncate databasechangeloglock;
xivo_stats=# select * from databasechangeloglock;
 id | locked | lockgranted | lockedby
----+--------+-------------+----------
  1 | f      |             |
```

6. Restart the containers:

```
xivo-dcomp start db_replic (XiVO)
xivocc-dcomp start pack_reporting (XiVO CC)
xivocc-dcomp start xivo_stats (XiVO CC)
```

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

FIVE

USER'S GUIDE

End user help and documentation.

5.1 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 OS integrated notifications when receiving call
- get keyboard shortcut to answer/hangup and make call using *Select2Call feature*
- *handle callto: and tel: links*
- open when machine startups
- close in tray

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
- chat between XiVO users also using UC assistant

5.1.1 WebRTC

Requirements

You can use WebRTC with *XiVO UC* or *XiVO CC* in the following environment:

- LAN network (currently no support for WAN environment without using *XiVO Edge*),
- with the:
  - *UC Assistant* or *CC Agent* with Chrome browser version **85** or later
  - or *Desktop Application*

- PC Configured with a microphone and headphones/speakers.

WebRTC Features

The *UC Assistant*, *CC Agent* and *Switchboard* can be used by users with WebRTC configuration, without physical phone.

For configuration and requirements, see *Requirements*.

**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.
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.

Headset call control (WebHID)

Important:

- This feature works only with **Poly, Jabra and Yealink headsets** supporting **WebHID** on the constructor website for now. You can contact the company to talk about additional brands support.
- Your headset **must be up to date**, and the update can be done through their respective brand software.
- The Linux support is not guaranteed and is discussed at the end of this section

Warning:

- **Plantronics** headset: *Plantronics Hub* application **MUST** be uninstalled for this feature to work (it can be replaced by the *Poly Lens* application).
- **Jabra** headset: in *Jabra Direct* application, the option `Call control with softphone` in *Device settings –> Softphone* menu **MUST** be enabled.

To enable it, go in the application menu, then click on the headset call control button. A popup will open so you can pick your headset in the list. Note that **only devices from the supported brands will be displayed in the list**.

Enabling on UC Assistant	Enabling on CCAgent & POPC

When enabled, the headset call control allows an user to answer and hangup using his headset, either in the web assistant or in the desktop application. You need to have your headset **connected using it's dongle** (or through an usb cable). **The feature does not work through direct bluetooth**.

When the headset is correctly connected to your application, you will have a visual hint displayed at the top of your application (green blinking / headset).

Headset control hint on UC Assistant	Headset control hint on CC Agent & POPC

Depending on the case, pushing the phone action button or the play/pause button of your headset (often placed at the center of an earpiece) should do the followings actions :

- if your softphone is ringing, you will answer
- if you are in a call or a conference, you will hangup

Note: If the last headset you used is still connected the next time you use your application, then headset call control will be **re-enabled automatically**.

To try this feature on Linux, you will need to have at least read/write permission over your hidraw driver. Here is a way to do it:

```
echo 'KERNEL=="hidraw*", SUBSYSTEM=="hidraw", MODE="0666"' | sudo tee /etc/udev/rules.
↪d/99-hidraw-readonly.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

Please note that, although there are generic drivers for Plantronics and Jabra devices in the Linux kernel (3.19 for Plantronics and 4.16 for Jabra), these devices appear to function quite differently when used on Linux compared to Windows. Therefore, we cannot guarantee that the headset will behave as expected on Linux. You could try some device-specific drivers if made available by these brands, provided they support call control on Linux.

Input/Output Audio Device

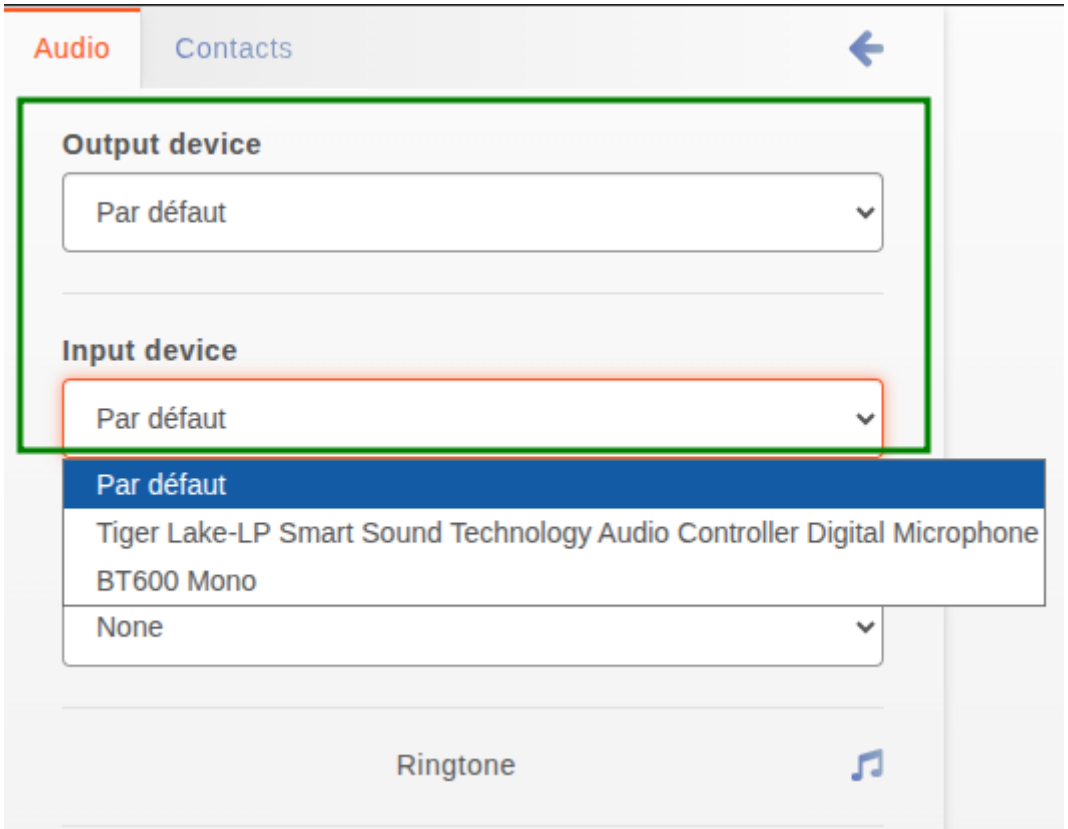
Audio device for input (microphone) or output (speaker) can be chosen within the application configuration:

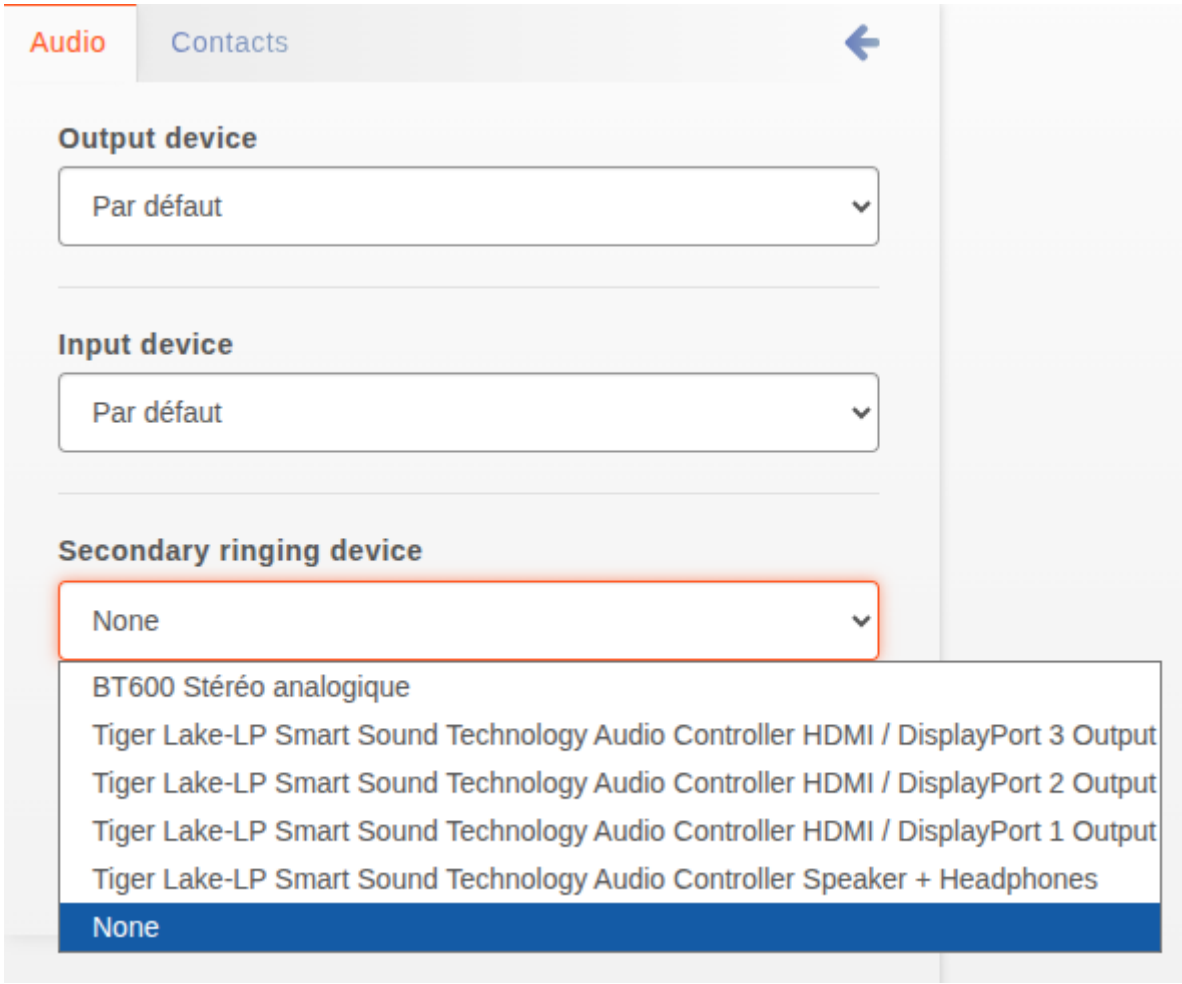
Secondary Ringing Device

When receiving a call, your computer will play a ringing sound to notify you. By default, this sound is played on the audio device selected in the audio configuration menu. In order to play the ringing sound on another audio device, you can configure it in the **Secondary ringing device** section.

For example, if your headset is set as the default audio device for calls, you can configure your speakers as the *Secondary ringing device* to ensure you hear incoming call notifications even when you're not wearing your headset.

- Ensures you never miss an incoming call notification.
- Allows flexibility in choosing where the ringing sound is played.
- Ideal for setups with multiple audio devices (e.g., headset and computer speakers).


Audio
Contacts
Output device
Par défaut
Input device
Par défaut
Par défaut
Tiger Lake-LP Smart Sound Technology Audio Controller Digital Microphone
BT600 Mono
None
Ringtone


Audio
Contacts
Output device
Par défaut
Input device
Par défaut
Secondary ringing device
None
BT600 Stéréo analogique
Tiger Lake-LP Smart Sound Technology Audio Controller HDMI / DisplayPort 3 Output
Tiger Lake-LP Smart Sound Technology Audio Controller HDMI / DisplayPort 2 Output
Tiger Lake-LP Smart Sound Technology Audio Controller HDMI / DisplayPort 1 Output
Tiger Lake-LP Smart Sound Technology Audio Controller Speaker + Headphones
None

Testing Audio Devices

Audio devices can directly be tested from the configuration menu to ensure they are working properly:

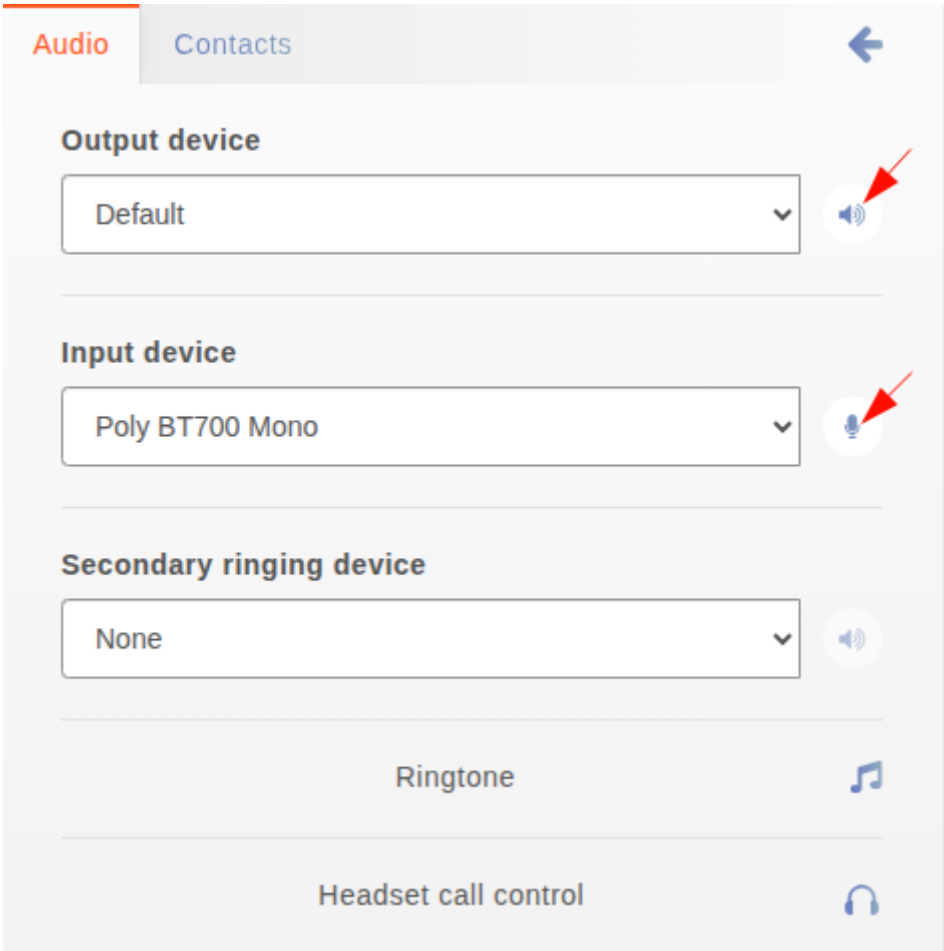
Testing Output Devices

For both the main output device and the secondary ringing device, test the audio by clicking the test button. This will play the configured ringtone on the selected device to verify it is working correctly.

Testing Input Device (Microphone)

For the microphone, test it by clicking the test button. This will:

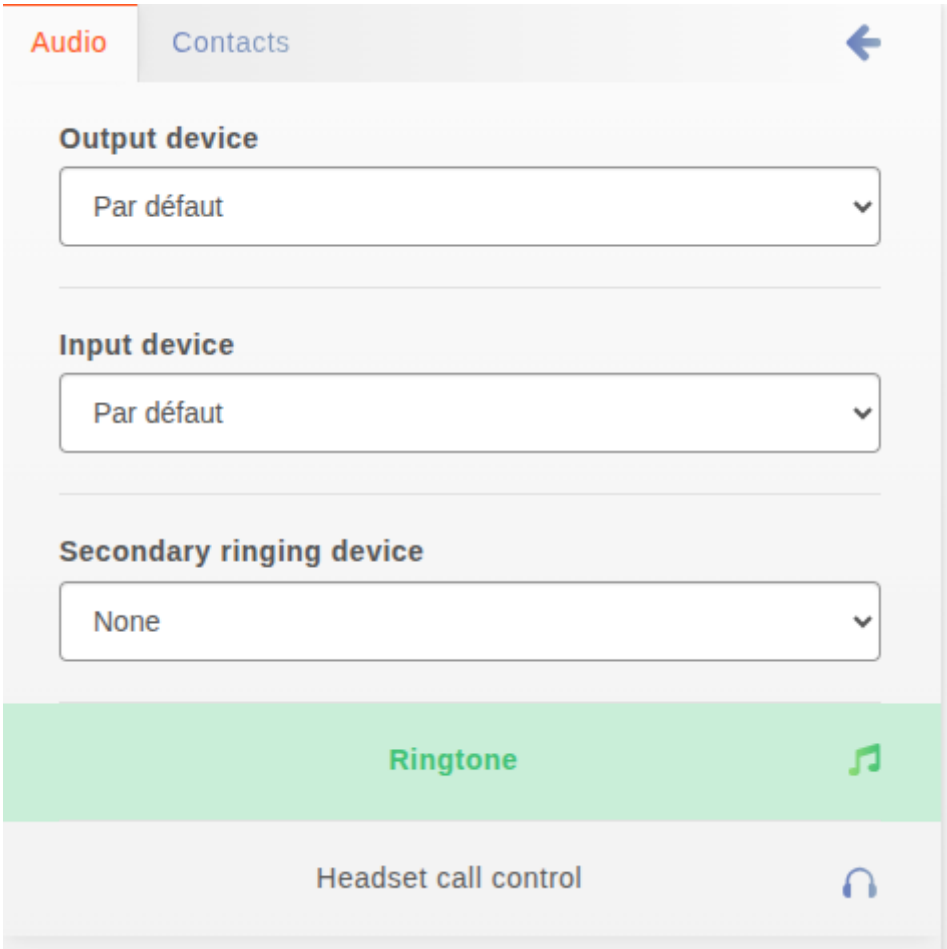
- Display a volume meter showing the microphone input level
- Play back your voice so you can hear yourself speaking
- Allow you to verify that your microphone is working and properly configured



This is particularly useful before making calls to ensure your audio setup is functioning as expected.

Ringing Sound

You can also choose to change your ringtone sound, both those options are available in the respective menu of the UCAssistant, CCAgent, and Switchboard, on the top right corner.

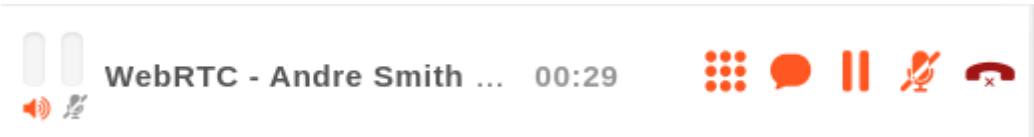


This feature is only available when using a WebRTC line.

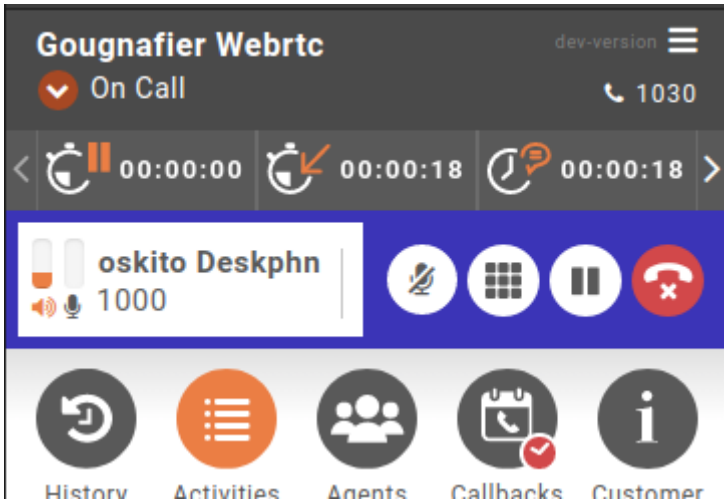
Mute your microphone

Being a webrtc user allows you to mute your microphone while being in a call with somebody. Clicking this button again will unmute the microphone. Clicking directly on the volume meter will have the same effect as clicking on the mute button. The little microphone icon in the volume meter will change according to your mute state.

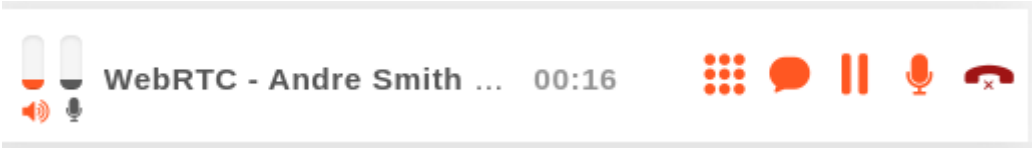
On UC Assistant :



On CC Agent and Switchboard :

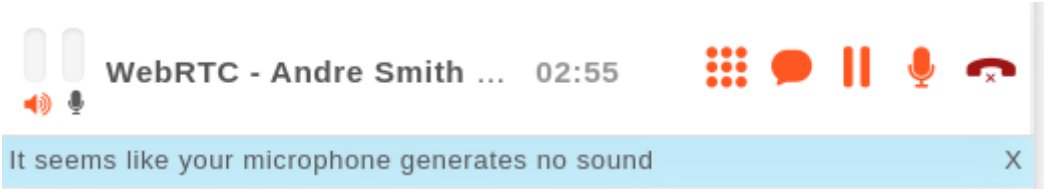


Volume indication



Two progress bars show the volume level of the speaker and the microphone. It certifies that the audio flow has been sent.

Sound detection



During the first five seconds of your call, if the application detect that there is no sound coming from your microphone, a small message will appear.

Call quality detection

If the call quality can be impacted by some network issues or server configuration issues, a small message will appear.

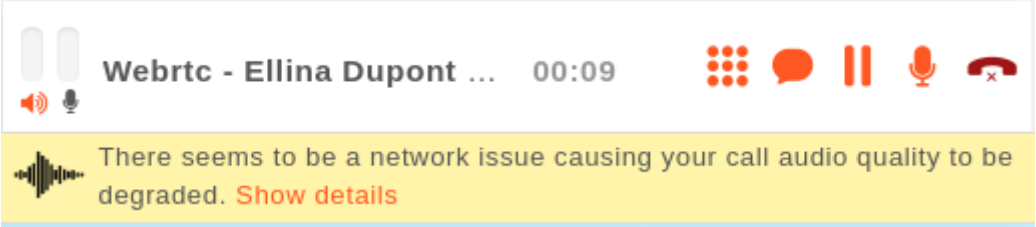
The message will stay up to 10 seconds after the audio quality is back to normal. The advanced statistics in the "show details" section and the play icon colors are updated live.

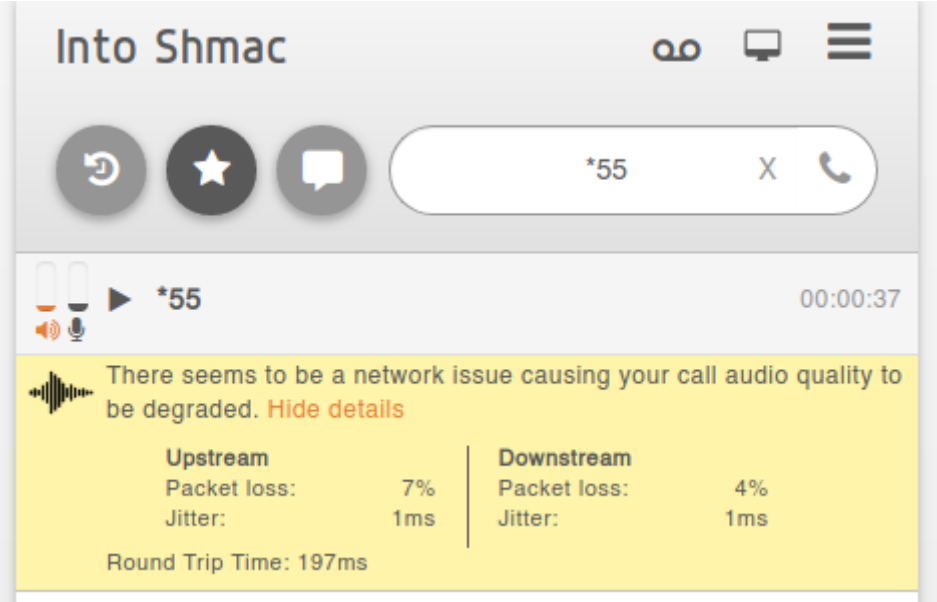
The current treshold are as following :

Quality goes to medium when something is higher than :

- Jitter: 50 ms
- Packet losts: 10%
- Round trip time: 150 ms

Quality goes to bad when something is higher than :





- Jitter: 100 ms
- Packet losts: 20%
- Round trip time: 300 ms

Call quality server-side feedbacks

If the quality of a call goes wrong, a live statistics feedback will be sent to the server. There will also be an automatic feedback at the end of each call, giving informations on the point where the call quality was at it's worst state. For more informations on those logs - see *Audio quality issues*.

Live feedbacks

Live audio quality measurement is taken every 2 seconds. If the quality is bad enough compared to the treshold, they are visible on the user interface as a small yellow message, and are also visible in the browser console. The statistics that are calculated using those measurements are done using the segment of time inbetween the last measurement and the new one, so it will be the last 2 seconds of the call for each measurement. The quality feedbacks are sent in xuc server logs only when it deteriorate enough to match the medium or bad quality treshold, and they respect a pause of at least 20 seconds inbetween each one to prevent flooding logs. For more informations on those logs - see *Audio quality issues*.

End of call report

The quality report at the end of each call is always sent to the xuc server logs even if the call went well. The statistics that are present in this end of call report are equals to the worst numbers found during the call. For example, if at some point the user had a spike of 20% packet loss during a few seconds but the rest of the call went well, the report will state that the packet loss spike was 20%. It is the same with jitter and RTT. It's not an average, it's always the highest detected statistic. For more informations on those logs - see *Audio quality issues*.

Upload and download directions

The upload direction is the direction from the user point of view (local outbound flow) sending data to the server (remote inbound flow). The download direction is the direction from the server point of view (remote outbound flow) sending data to the user (local inbound flow)

Opus Codec

By default WebRTC line uses the Opus codec.

It enhances the audio quality of calls with a lower bitrate (around 20kbps with the current configuration compared to 64kbps for alaw codec).

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`

Then just use:
`ifdown <ifname>`

This will switch off network interface not required to make your call.

Additional chrome WebRTC-specific options

There are various additional settings used in the code. They are used to improve audio quality by enabling or disabling chrome WebRTC-specific flags.

Note:

These options are not customisable. They are set in the code.

Chrome currently supports these audio quality options :

- **Automatic gain control** : Adjust voice sound level to make it linear, lowering sound level when the user speaks too loudly.
- **Echo cancellation** : Detect and delete echo coming from the playback of the user's own voice.
- **Noise suppression** : Cancel background noises coming from the user's environment.
- **Highpass filter** : Filters out low frequencies noises (like microphone background buzzing permanent sound).
- **Audio mirroring** : Reflect sound coming from different directions into a focus point (similar to a parabola).
- **Typing noise detection** : Detect and delete keypress sounds.

The current production code is set as follows :

- googAutoGainControl is set to *false*
- googAutoGainControl2 is set to *false*
- googEchoCancellation is set to *true*
- googEchoCancellation2 is set to *true*
- googNoiseSuppression is set to *false*
- googNoiseSuppression2 is set to *false*
- googHighpassFilter is set to *false*
- googAudioMirroring is set to *false*
- googTypingNoiseDetection is set to *true*

Note:

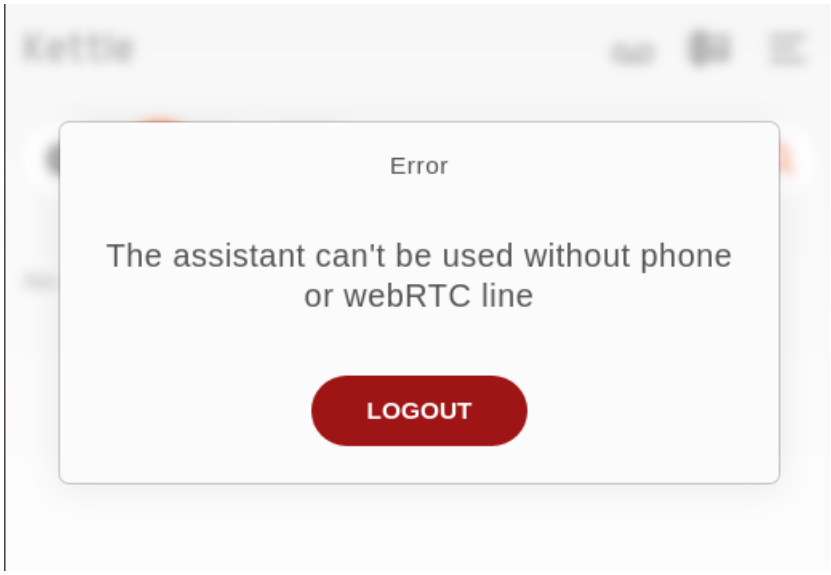
This flag used to be valid but is now deprecated :
- Ducking : Reduce an audio signal by the presence of another signal (multiple people talking at the same time).

5.1.2 Login

To login, you must have a user configured on the *XiVO PBX* with:

- CTI Login enabled,
- Login, password and profile configured
- A configured line with a number

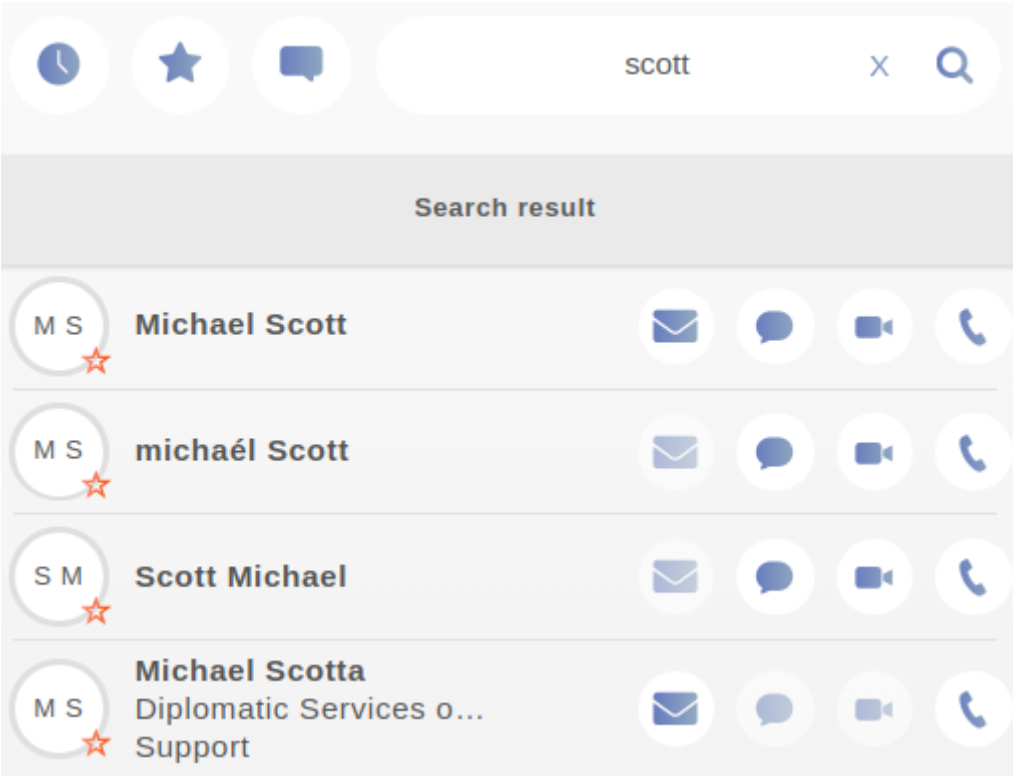
Warning: If a user tries to login without a line, an error message is displayed and user is redirected to the login page (this applies also to *Desktop Applications*)



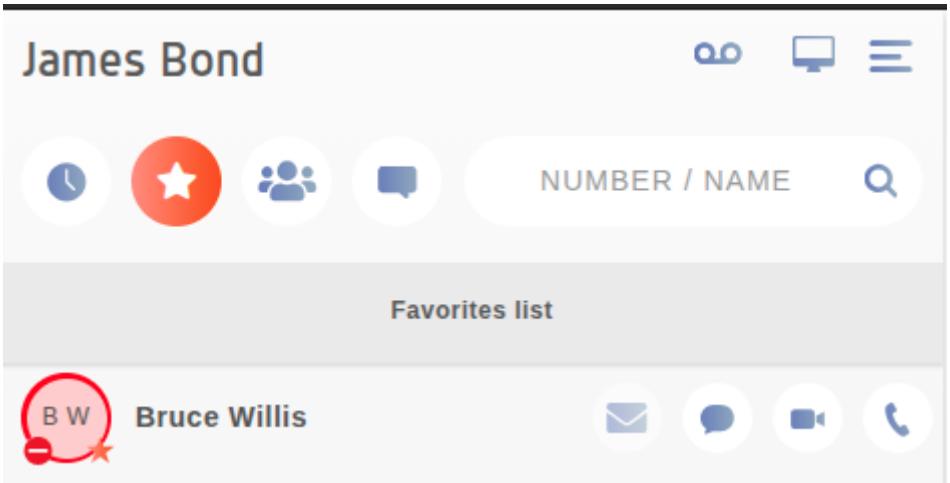
Note: Automatic login keeps you signed in until you log out.

5.1.3 Search

You can use the search section to lookup for people in the company, results will display all information known for the user.



The DND (Do Not Disturb) status of each user is displayed in the search results. When a user has DND enabled, a red icon appears next to their avatar.



On the line of the user, you will see the following actions :

- start call : will call the default phone number
- start chat conversation (if it is a XiVO user)
- start video call (if the video calls are available for this user)
- send an email (if user has an email)

When clicking on the line, a contact modal opens displaying detailed information about the selected user.

Contact modal

This modal is composed of a *Header* and three main sections *Contacts*, *General* and *Location*

Their content is filled with what is defined in the selected *Display Filter* and the selected *Directories*.

Header

Displays the name and up to two optionnal text fields (user's role/function…)

Contacts

Contains up to 5 configurable numbers/callables and supports to display only 1 email.

You will be able to:

- display all the phone numbers available for this user
- start call by clicking on the phone icon
- copy the phone number into your clipboard by clicking on the copy icon.
- display the user's email address if he has one.
- copy the user email into your clipboard by clicking on the copy icon
- start writing an email by clicking on the email (it will open the user's configured email client)

Note: By default only the email destination is pre-filled but you can also pre-fill the email *subject* and *body* via a template - see *Email Template*

General

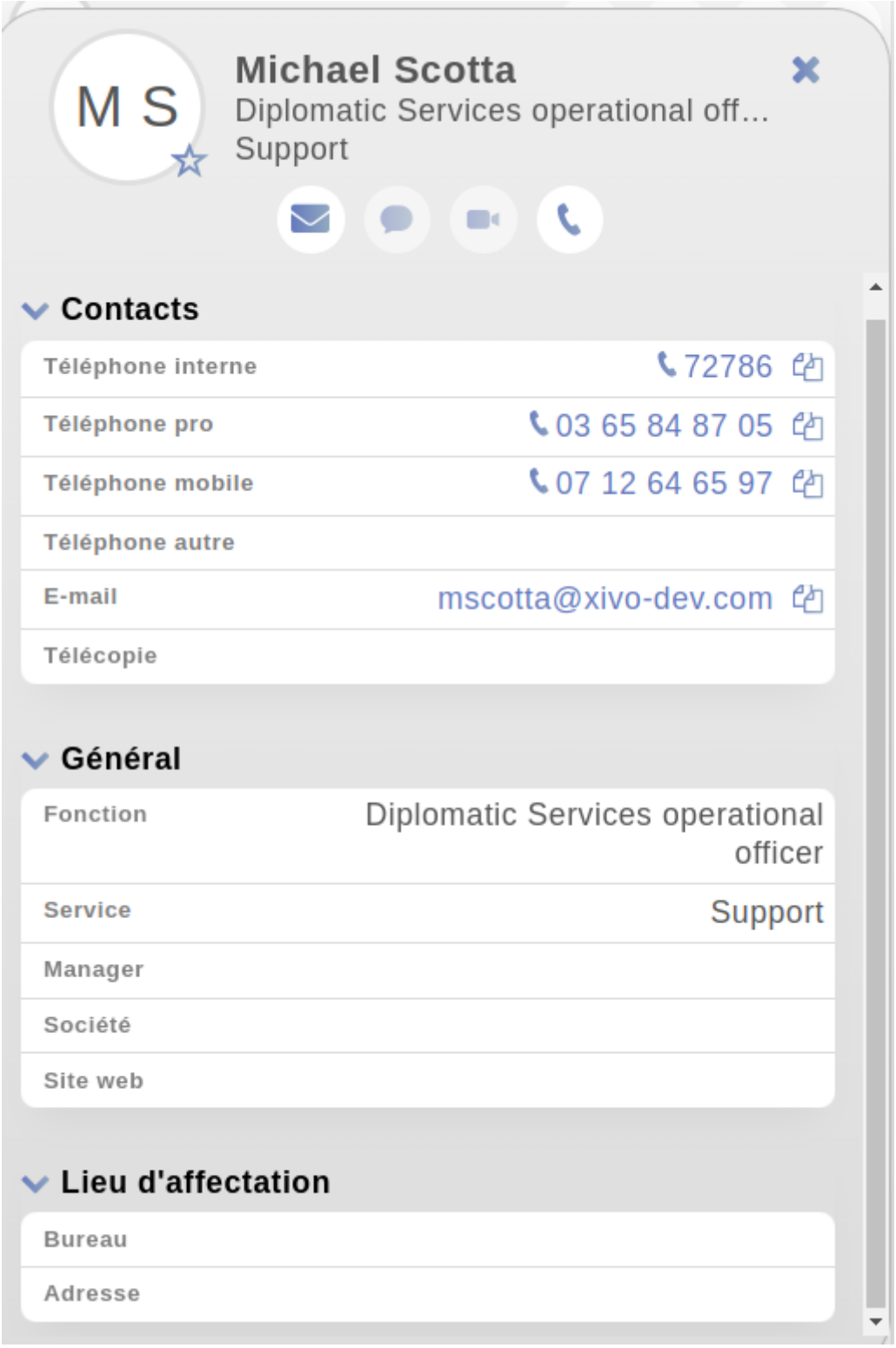
Contains up to 5 configurable text fields to display further information

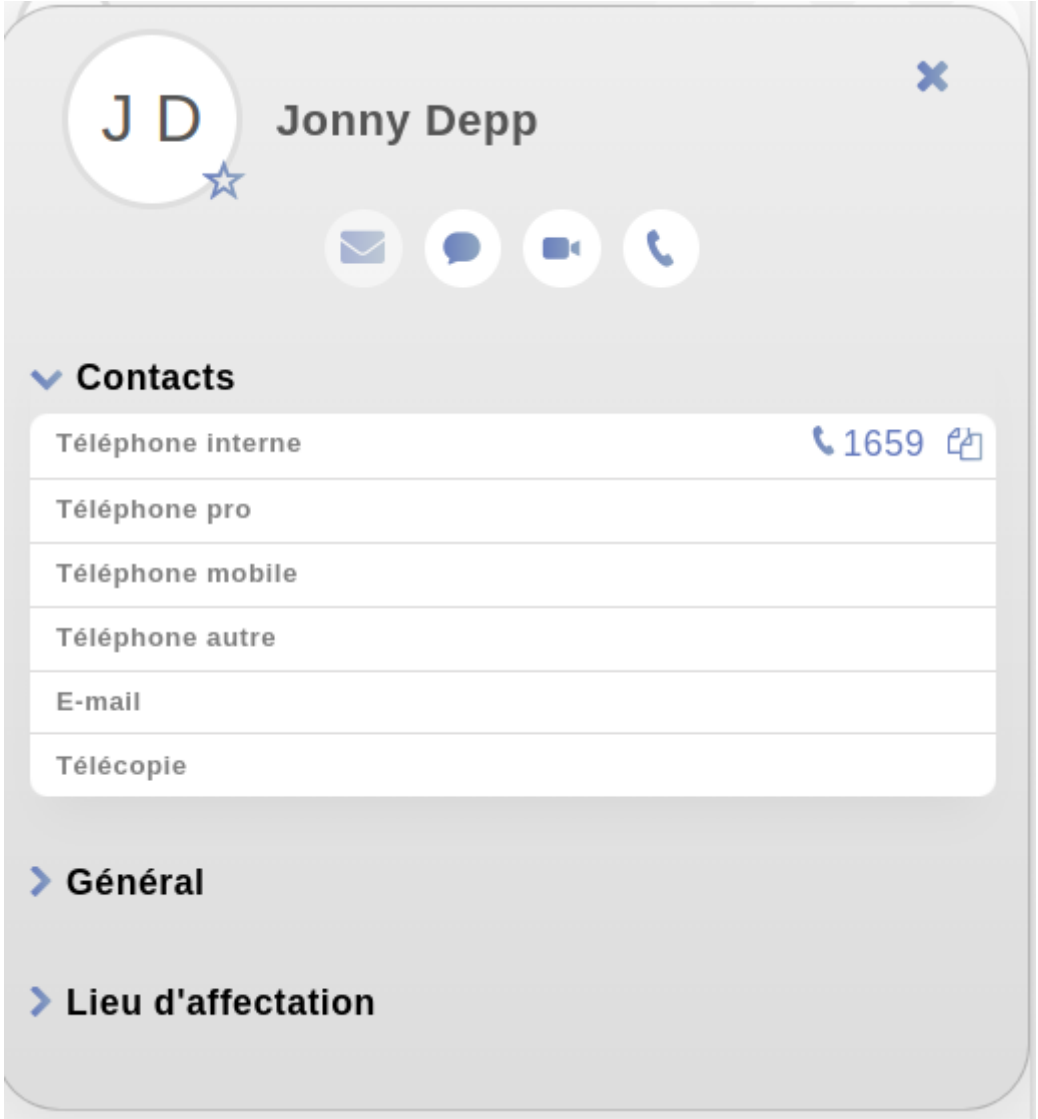
Location

Contains up to 2 configurable text fields to display further information

Note: Note that *General* and *Location* sections will be folded if no fields are filled in.

Important: Integration: to enable this feature, you must configure the directories in the *XiVO PBX* as described in *Directories* and *Personal contacts*.

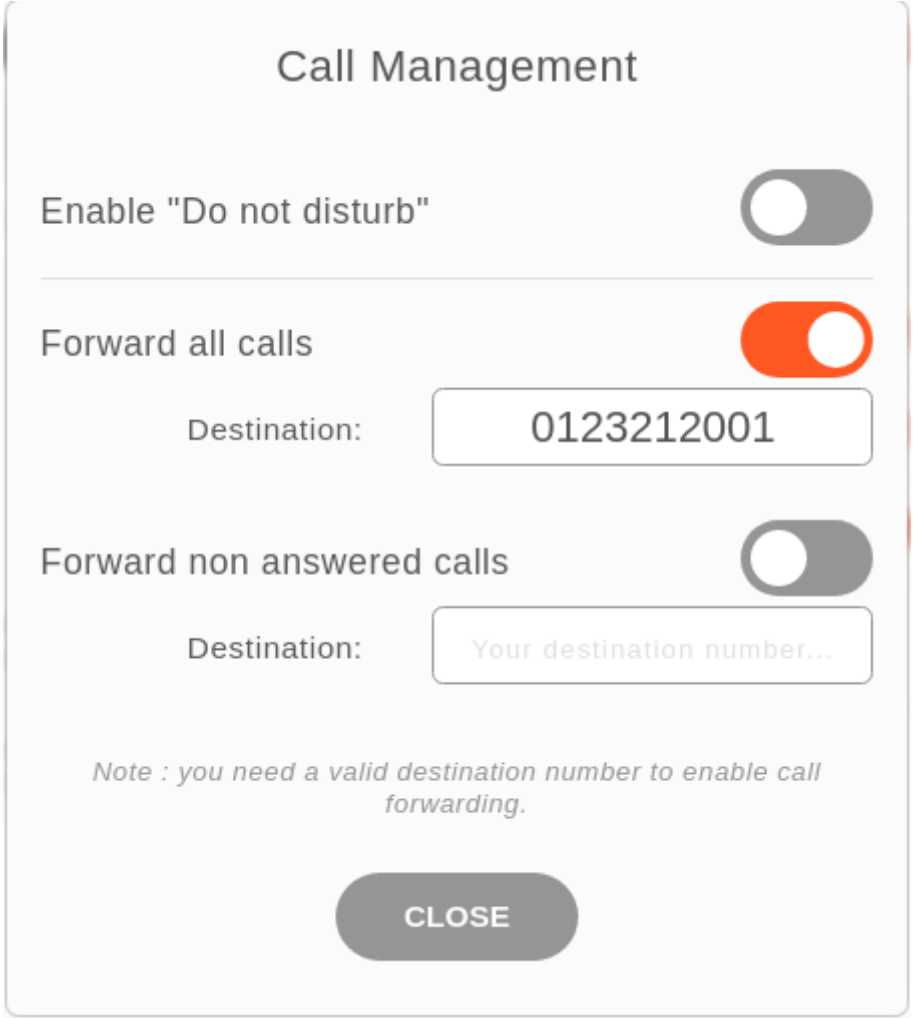

M S
Michael Scotta
Diplomatic Services operational off...
Support
Contacts
Téléphone interne
72786
Téléphone pro
03 65 84 87 05
Téléphone mobile
07 12 64 65 97
Téléphone autre
E-mail
mscotta@xivo-dev.com
Télécopie
Général
Fonction
Diplomatic Services operational officer
Service
Support
Manager
Société
Site web
Lieu d'affectation
Bureau
Adresse


J D
Jonny Depp
Contacts
Téléphone interne
1659
Téléphone pro
Téléphone mobile
Téléphone autre
E-mail
Télécopie
Général
Lieu d'affectation

5.1.4 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:

You can then change your settings and enable them.



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.

Besides, there is a precedence, if DND mode is enabled and also call forwarding, calls will be rejected.

5.1.5 Call history

The call history tab lists all the recent calls you were part of. For each call, it displays the status (received, emitted…), the duration, as well as the time when the call happened. You also see the phone status of internal users. You can hover your mouse cursor on a call to add this phone to your contact, or call it. You can also click on it to unfold it and see the call(s) details.

5.1.6 Groups

Clik on the groups button to see the groups you belongs to. Groups must be configured in the *XiVO PBX* as described in *Groups*

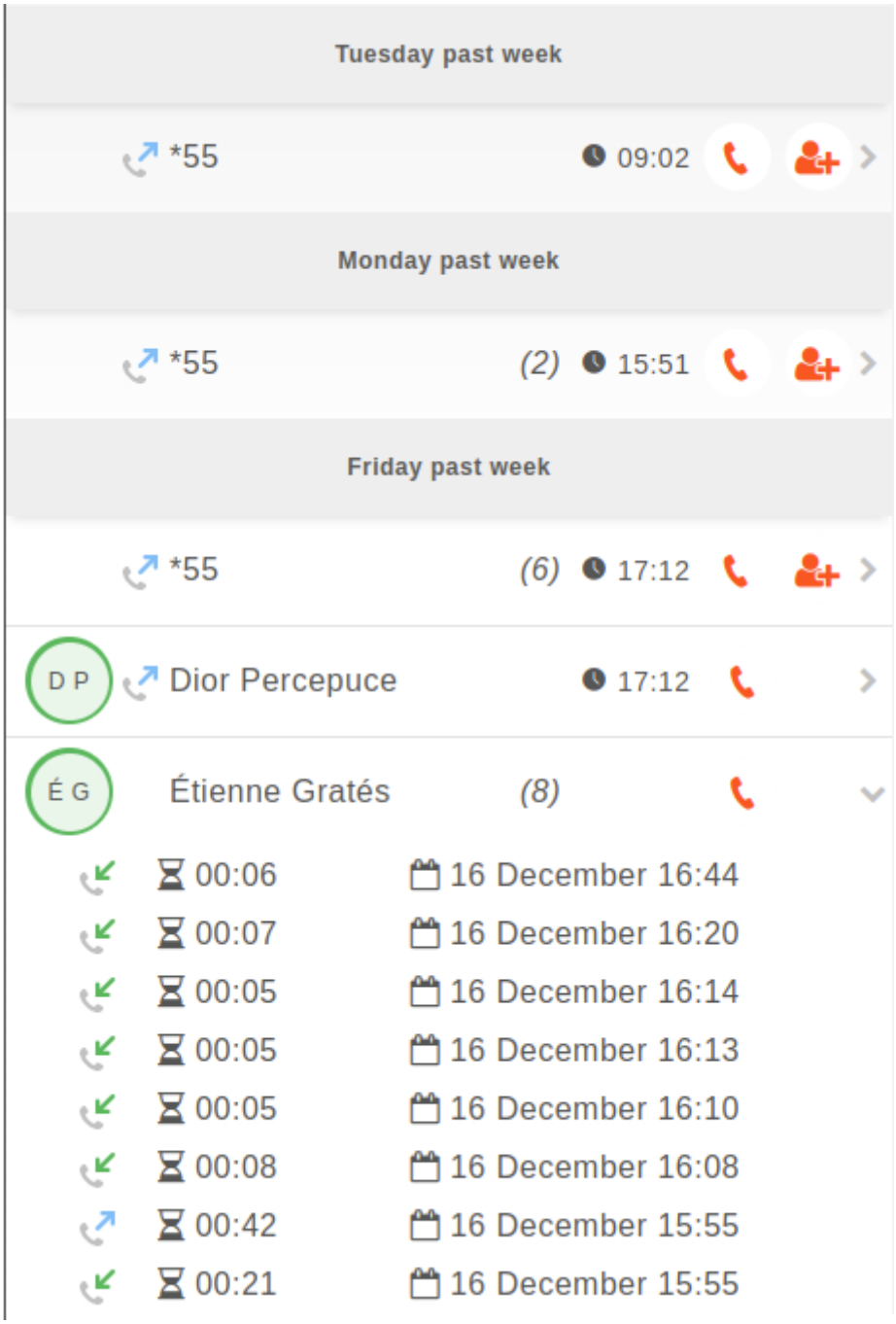
Note: Users can join or leave a group to start or stop receiving calls. If you want to prevent users from joining or leaving groups, you can disable this functionality. For more information, see *Disabling call group disconnection functionality*

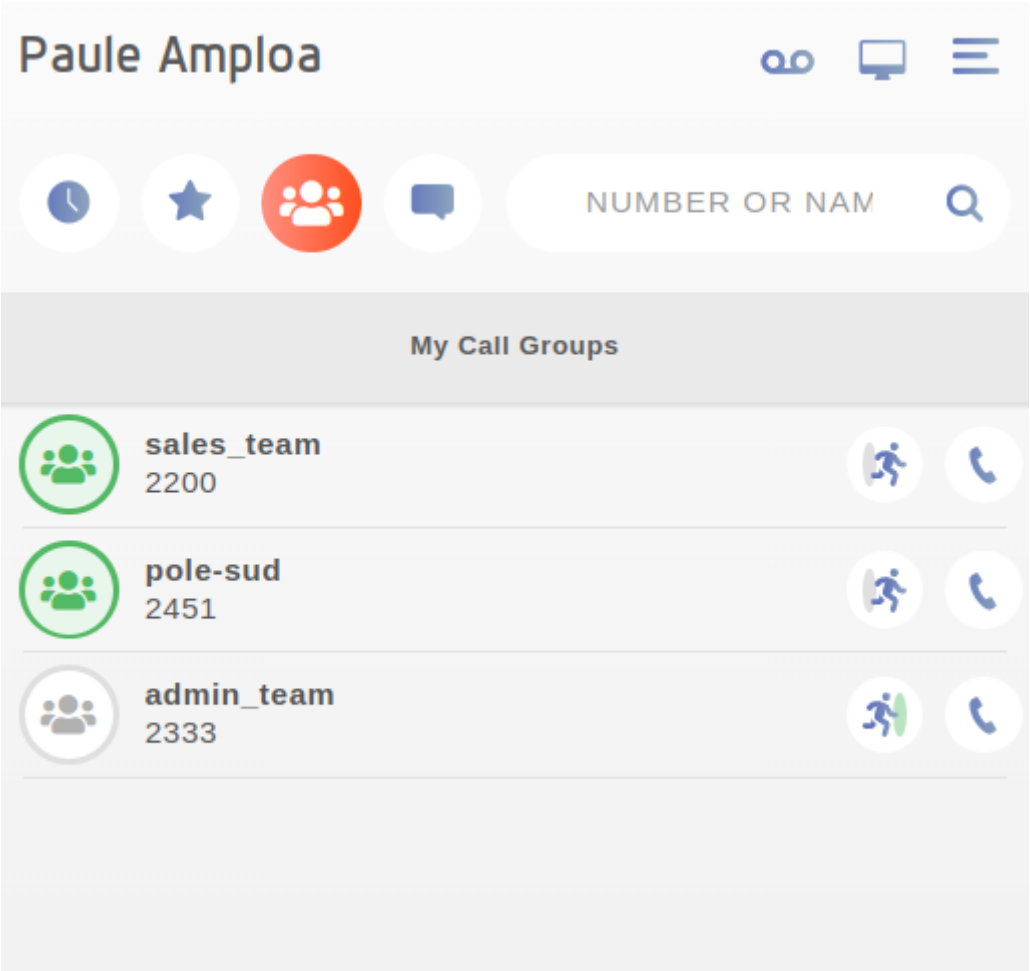
5.1.7 Favorites

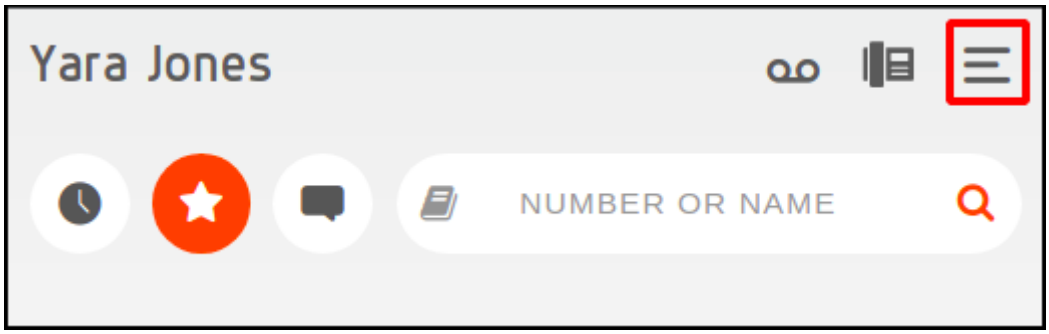
Click on the star to add/remove a search result to your favorites.

5.1.8 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 the toolbar or find them in *favorites* panel if starred.


Tuesday past week
*55
09:02
Monday past week
*55
(2)
15:51
Friday past week
*55
(6)
17:12
D P
Dior Percepuce
17:12
É G
Étienne Gratés
(8)
00:06
16 December 16:44
00:07
16 December 16:20
00:05
16 December 16:14
00:05
16 December 16:13
00:05
16 December 16:10
00:08
16 December 16:08
00:42
16 December 15:55
00:21
16 December 15:55


Paule Amploa
NUMBER OR NAM
My Call Groups
sales_team
2200
pole-sud
2451
admin_team
2333

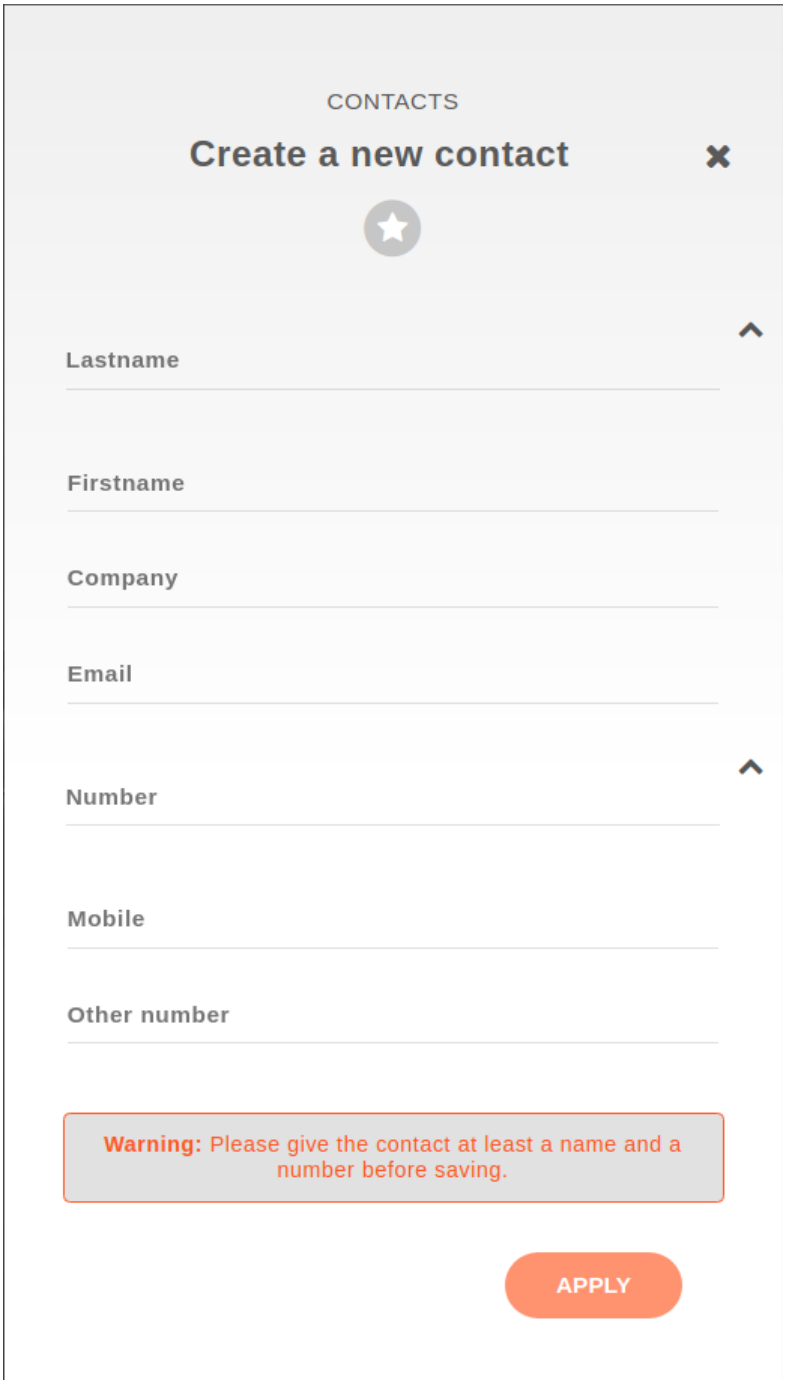

Yara Jones
NUMBER OR NAME

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.

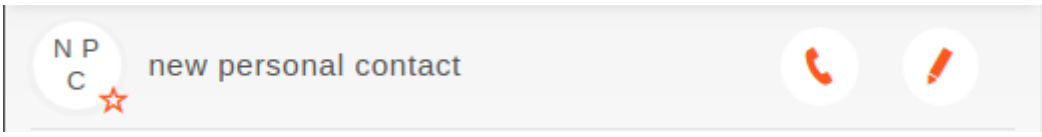
Warning: Every contact must have at least a name (either firstname or lastname) and a number (either number, mobile or other number).



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

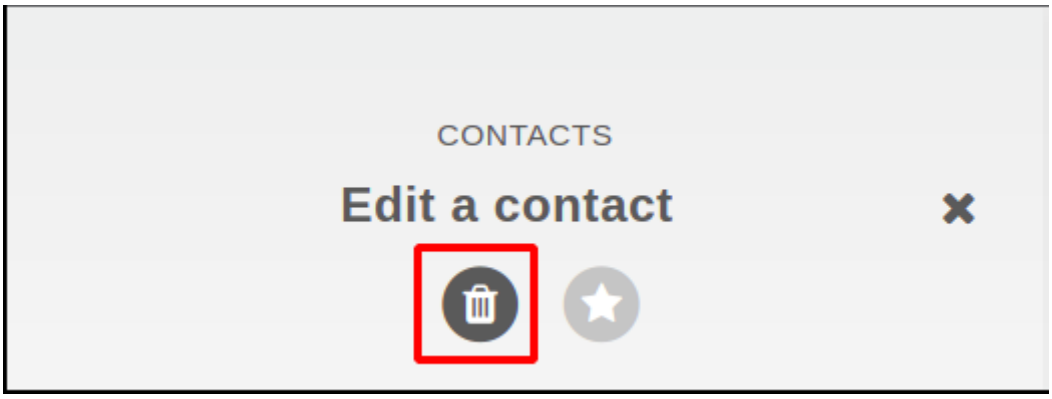
To edit a personal contact, you should search it first, and a pencil icon will be displayed on the user line as in the following screen:



Once clicked, you are redirected to edition pane where you just fill wanted fields.

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
```

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.

5.1.9 Phone integration

The *UC Assistant* can integrate with the phone to :

- Call / Hangup
- Put on hold
- Do direct or attended transfers - check *Known limitations*
- Initiate 3-party conference - check *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 :

5.1.10 On hold notifications

You can be notified if you forget a call in hold for a long time, see *configuration section*.

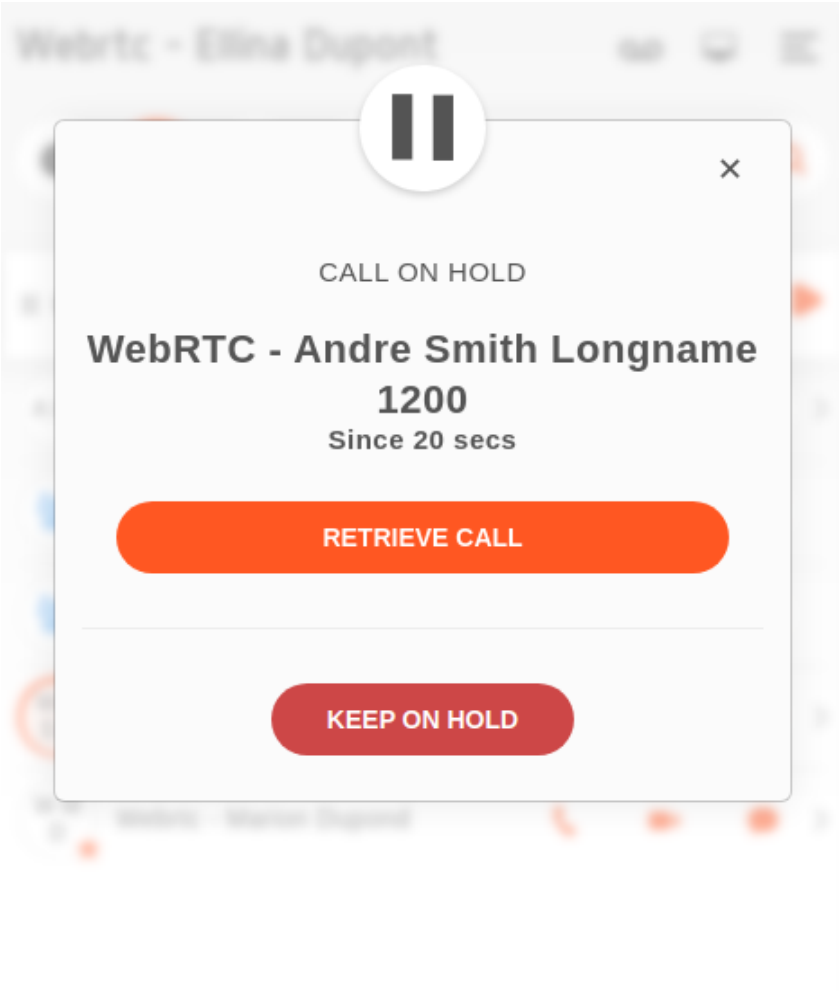
5.1.11 Conferences

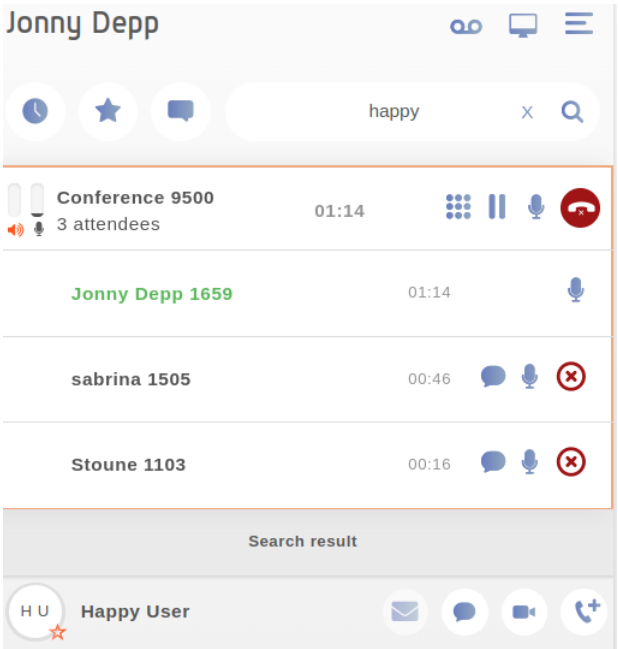
When joining a conference, either as an attendee or an organizer, the *UC Assistant* will display specific informations about the conference you are joining.

Note: On WebRTC and Snom phones you can also do a device hosted three-party conference, in this section we describe features of the XiVO hosted conferences.

Conference information:

- The timer next to the conference name displays how long the conference has been running.
- The first line displays the number of attendee.


CALL ON HOLD
WebRTC - Andre Smith Longname 1200
Since 20 secs
RETRIEVE CALL
KEEP ON HOLD


Jonny Depp
happy
Conference 9500
3 attendees
01:14
Jonny Depp 1659
01:14
sabrina 1505
00:46
Stoune 1103
00:16
Search result
H U
Happy User

Conference actions:

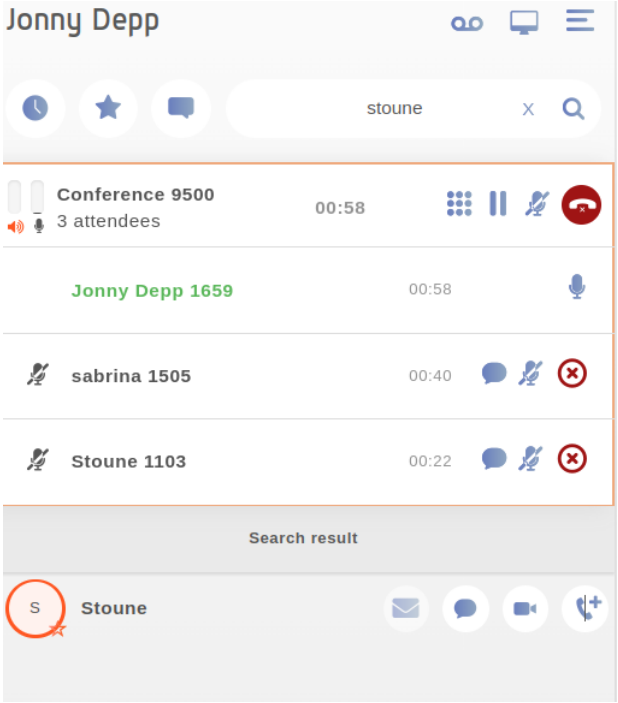
As an attendee, you can only:

- Exit a conference room by clicking the hangup button
- Put the conference on hold. Other attendees will not hear any hold music (if you're on the same XiVO) but will not be able to hear you neither you will be able to hear the conference room.

As an organizer, you will also be able to:

- Mute/Unmute all other attendee in the conference room

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 green
- When an attendee is muted, a slashed microphone icon will be displayed next to its name
- When an attendee is talking, a green speaker will be displayed next to its name

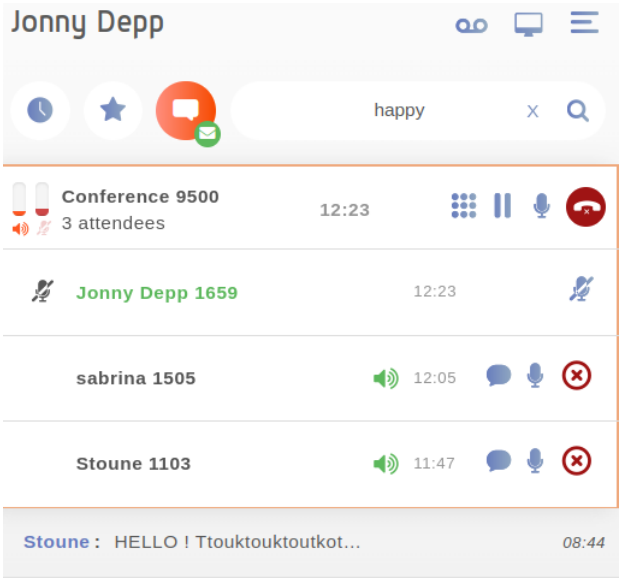
Attendee action:

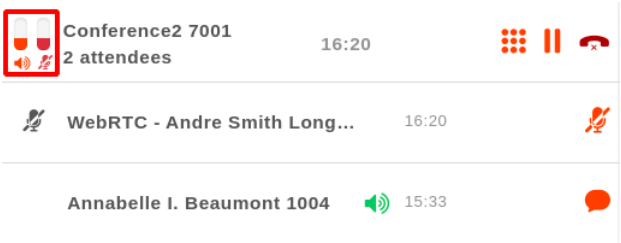
Participant can :

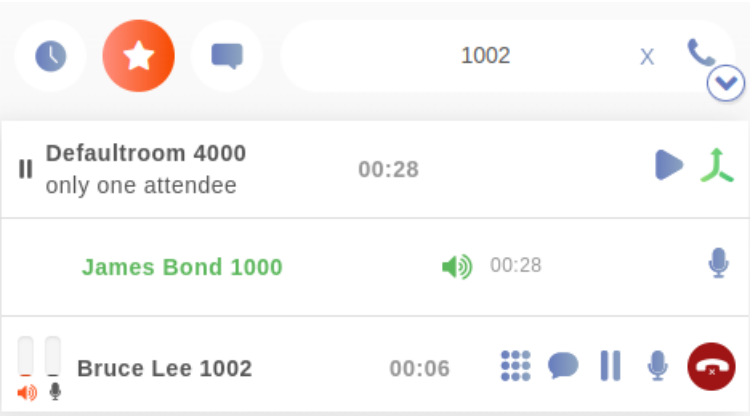
- Mute/Unmute itself. If the participant is muted and starts speaking, the microphone icon in the volume meter will blink in red as a reminder. This red warning only happens in audio conferences, not in normal phone calls.
- Include an ongoing secondary call to a conference via the include button.

Organizer can also:

- Mute/Unmute any attendee
- Kick out an attendee. A message will be played to the kicked out attendee before leaving the conference.


Jonny Depp
happy
Conference 9500
3 attendees
12:23
Jonny Depp 1659
12:23
sabrina 1505
12:05
Stoune 1103
11:47
Stoune : HELLO ! Ttouktouktoutkot...
08:44

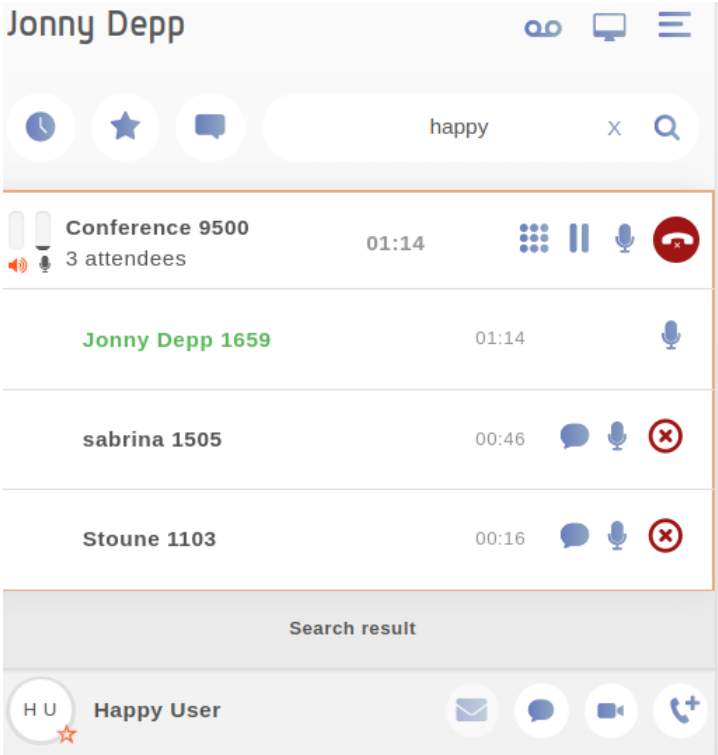

Conference2 7001
2 attendees
16:20
WebRTC - Andre Smith Long...
16:20
Annabelle I. Beaumont 1004
15:33


1002
Defaultroom 4000
only one attendee
00:28
James Bond 1000
00:28
Bruce Lee 1002
00:06

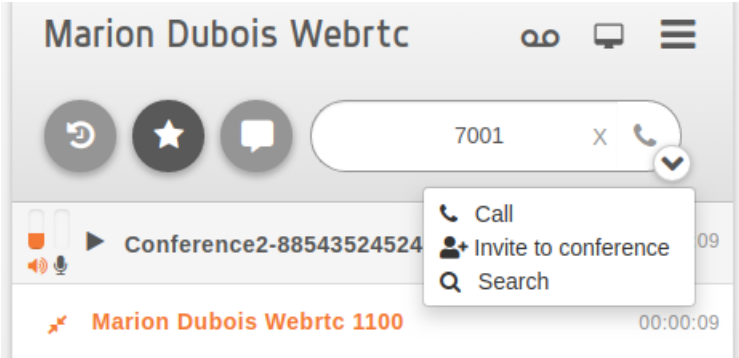
- Invite a user to the conference.

  This can be done by clicking the *invite to conference* button in the user dropdown menu :

  - from the *favorites* tab,
  - or from the *search results*, by searching for a username.



- The organizer can also invite a phone number to a conference via the additional call options. This option will be displayed after the organizer starts typing a phone number in the search bar.



5.1.12 Instant Messaging

It is possible from the assistant to send chat messages and have private text conversation. It is possible to deactivate it if this feature is not suitable for your needs or if you have already an external chat application, see *Disabling chat in UC Assistant and Switchboard*.

There are two possible behaviors for handling messages:

1. **No persistance** (default) : Each message is an *instant message* that can be sent to another user logged in. messages are not stored anywhere and will be discarded once you refresh the page or log out.
2. **With persistence** : Can be enabled with *Chat Backend*. This allows then

- To send messages to offline users
- To retrieve you conversation history with other parties
- To receive notifications when you logs in if you have unread messages that has been received while you were disconnected.

When you send a message, you are notified if the message was sent successfully or if the message was not received because the recipient is not logged in.

When you receive a message, you are notified with a orange badge on the message tabs. If you are using the desktop application, the electron tray icon also shows an orange badge. You will also have a notification from your web browser or a system notification if you are using the desktop application.

You can send links in message, they will be clickable. You can also write emojis from your keyboard (e.g., *:smile:*). You can find here some emojis exemple. The conversation window also allows you to call a user on their internal phone number directly from the chat. To do so, you can click on the phone icon next to the name of the user you're chatting with. If the video calls are available on your XiVO, you can launch one by clicking on the camera.

You can send a message from different places in the application :

- From the contact line :
- From an ongoing call :
- from a conference call :

You are also able to list, at any time, all your ongoing conversation from the message tab :

5.1.13 External directory

This feature will add an additional book icon on the left side of the search bar. When clicking on this *book icon*, it will open (or close) the defined **external directory** (the *external directory* being a directory accessible via an URL).

To enable it, see *configuration section*.

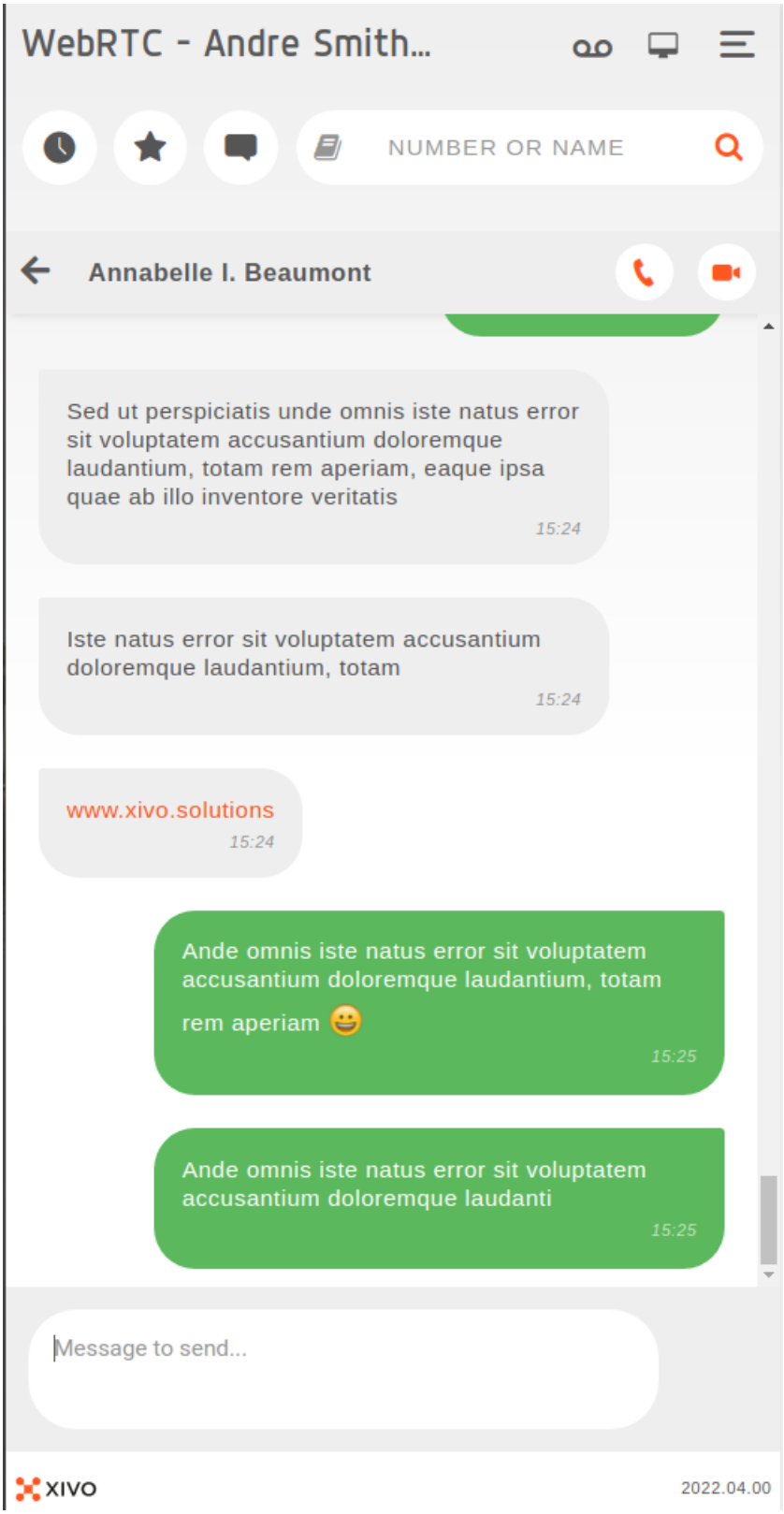
Note: The search in the search bar will not search in this directory. But a search could be implemented in this external directory.

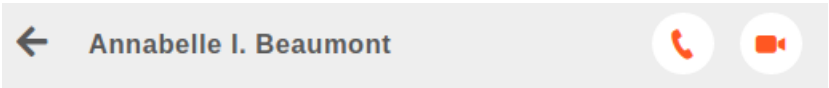
The screenshot below shows an example: when clicking on the *book icon* the *external directory* is shown. Here this *external directory* was developped to list users per site.

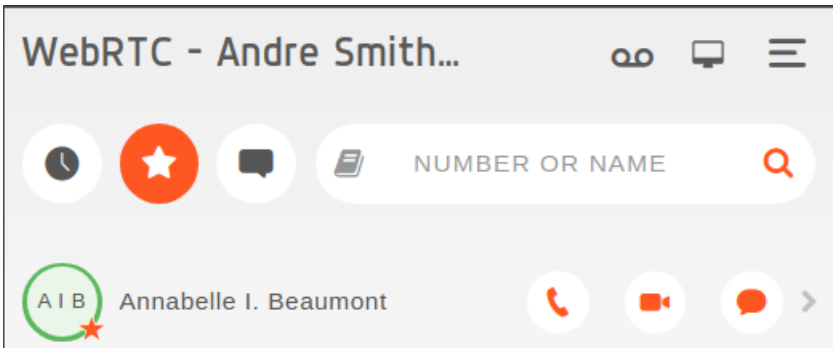
5.1.14 Logout

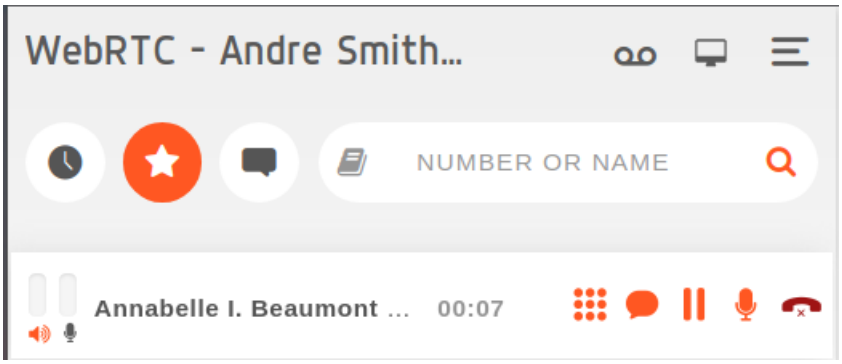
The logout button can be found after going in the top-right hamburger menu, which looks like the screenshot below :

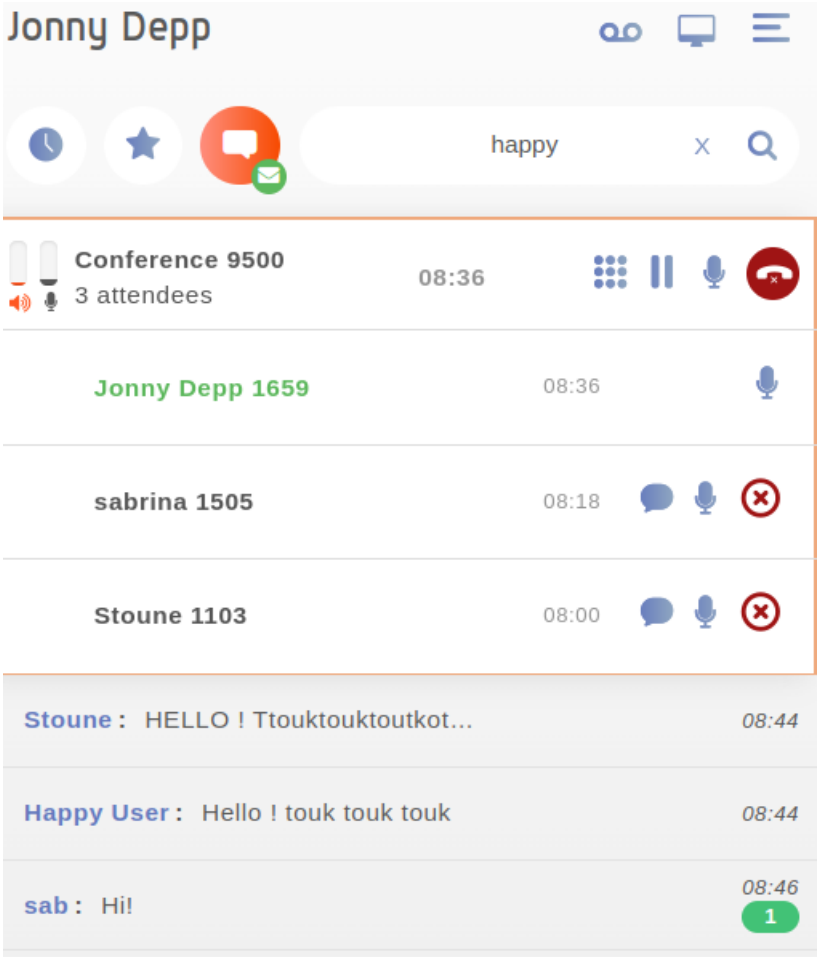
You will then be brought back to the login page.

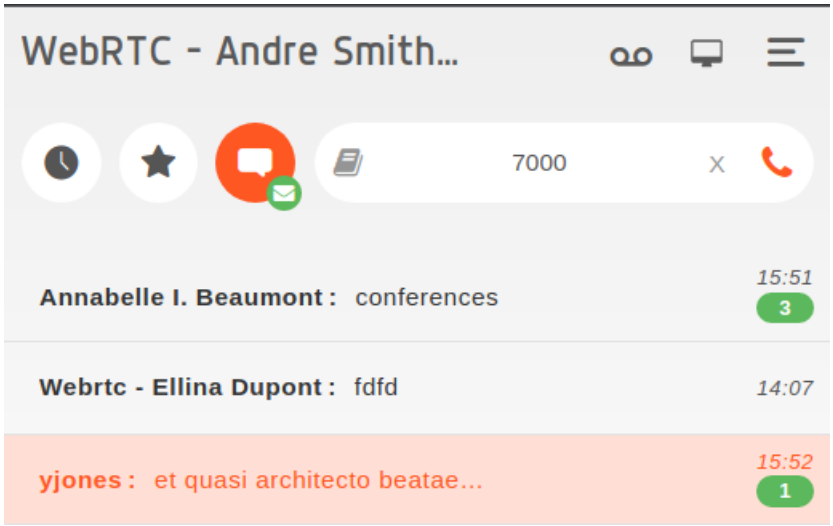

WebRTC - Andre Smith...
NUMBER OR NAME
Annabelle I. Beaumont
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis
15:24
Iste natus error sit voluptatem accusantium doloremque laudantium, totam
15:24
www.xivo.solutions
15:24
Ande omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam 😀
15:25
Ande omnis iste natus error sit voluptatem accusantium doloremque laudanti
15:25
Message to send...
XIVO
2022.04.00

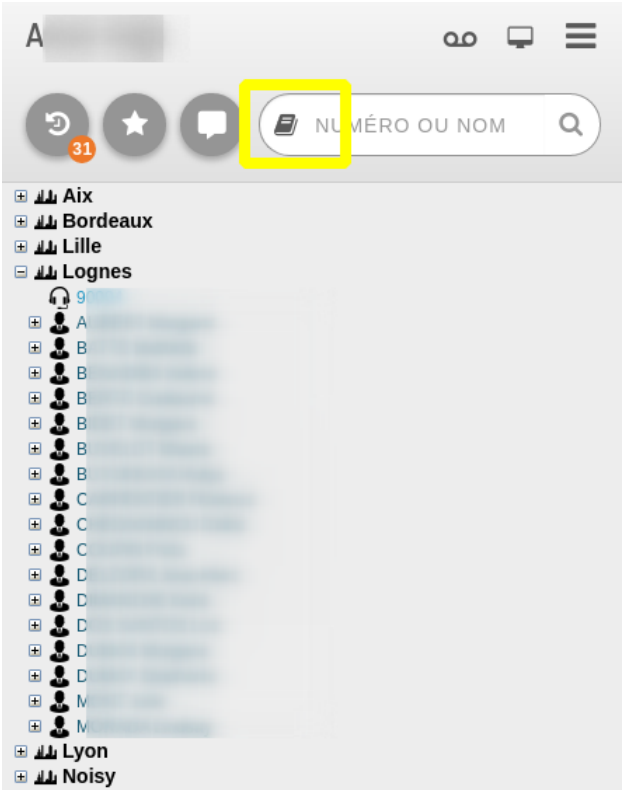

Annabelle I. Beaumont

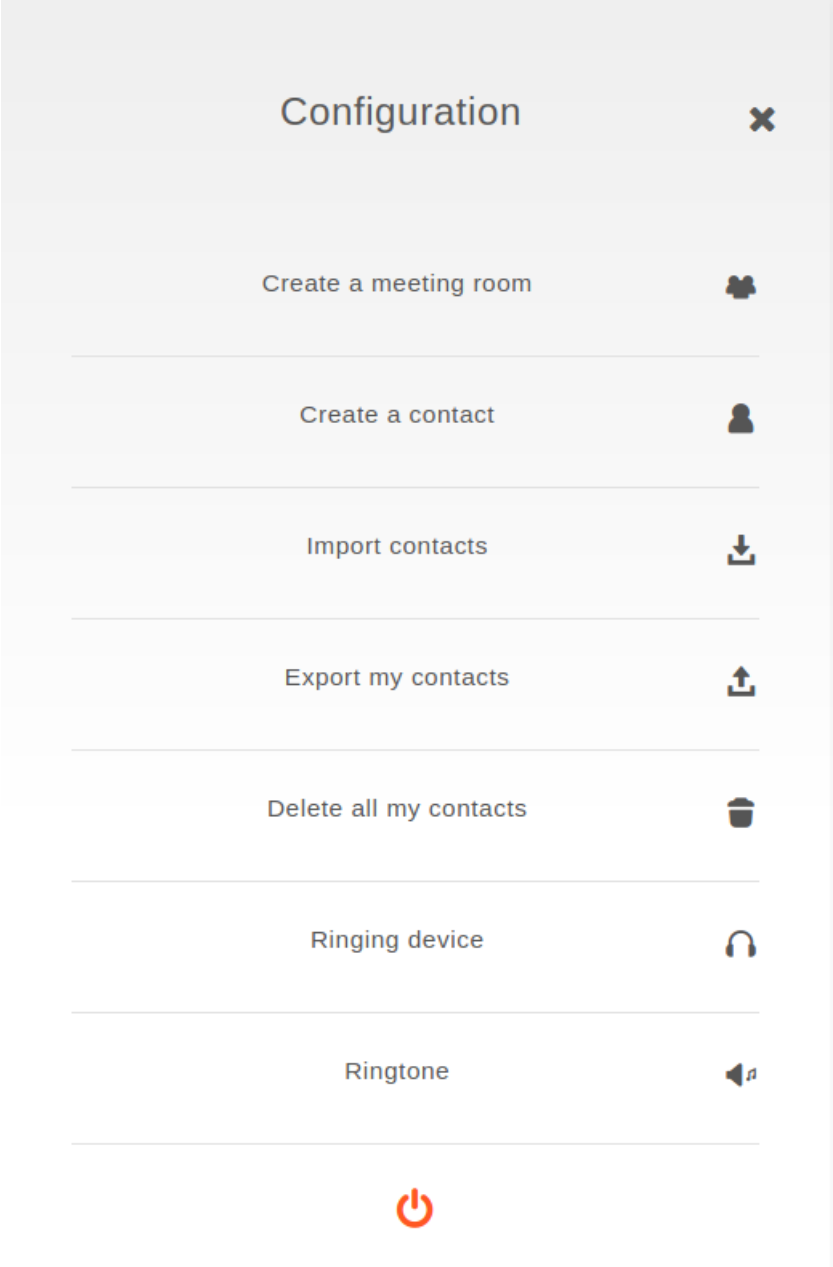

WebRTC - Andre Smith...
NUMBER OR NAME
A I B
Annabelle I. Beaumont


WebRTC - Andre Smith...
NUMBER OR NAME
Annabelle I. Beaumont ...
00:07


Jonny Depp
happy
Conference 9500
3 attendees
08:36
Jonny Depp 1659
08:36
sabrina 1505
08:18
Stoune 1103
08:00
Stoune : HELLO ! Ttouktouktoutkot...
08:44
Happy User : Hello ! touk touk touk
08:44
sab : Hi!
08:46
1


WebRTC - Andre Smith...
7000
Annabelle I. Beaumont : conferences
15:51
3
Webrtc - Ellina Dupont : fdfd
14:07
yjones : et quasi architecto beatae...
15:52
1


NUMÉRO OU NOM
31
Aix
Bordeaux
Lille
Lognes
Lyon
Noisy


Configuration
Create a meeting room
Create a contact
Import contacts
Export my contacts
Delete all my contacts
Ringing device
Ringtone

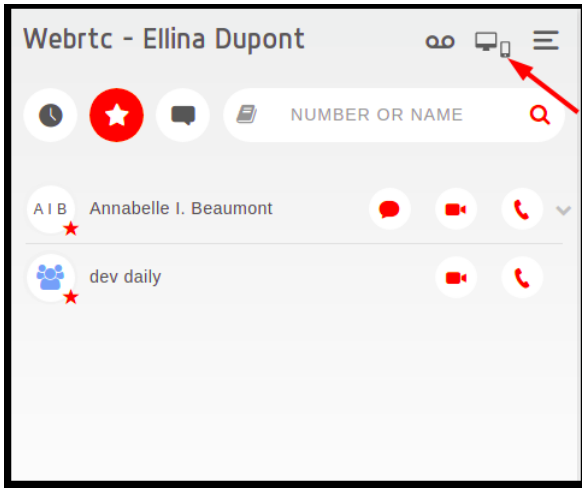
5.1.15 Meeting room

You can join an audio-video conference, called meeting room, which will open on the side of your application. See *Meeting Rooms* for more information.

5.1.16 Using web and mobile applications together

Note: This section concerns **only WebRTC users**, not Unique Account or SIP phone users.

After a user connects to the mobile application for the first time, a pop up notification will be sent on their UCAssistant. This is to inform the user that they will now receive calls on both mobile and web application. The user will be able to change this by clicking on the call management section :



They can then choose if they want to receive calls on the mobile application, web application, or on both.

After a user uninstalled the mobile application, they can unpair the mobile app from the assistant by clicking on the chain icon next to the mobile option.

5.2 Switchboard

Note: This section describes the Switchboard application features. It is available as a web application from your Web Browser. Currently it is not available as a desktop application.

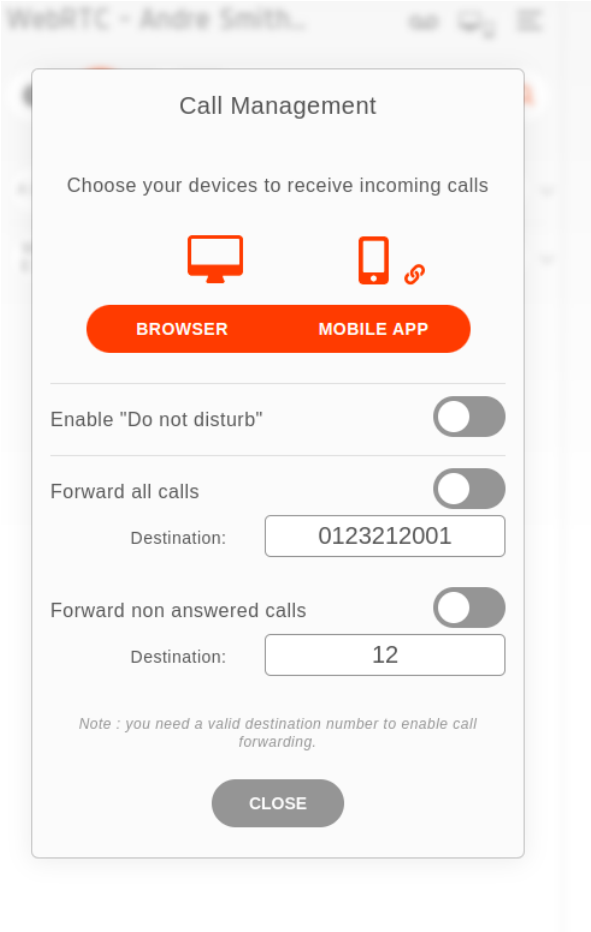
What is the Switchboard application ?

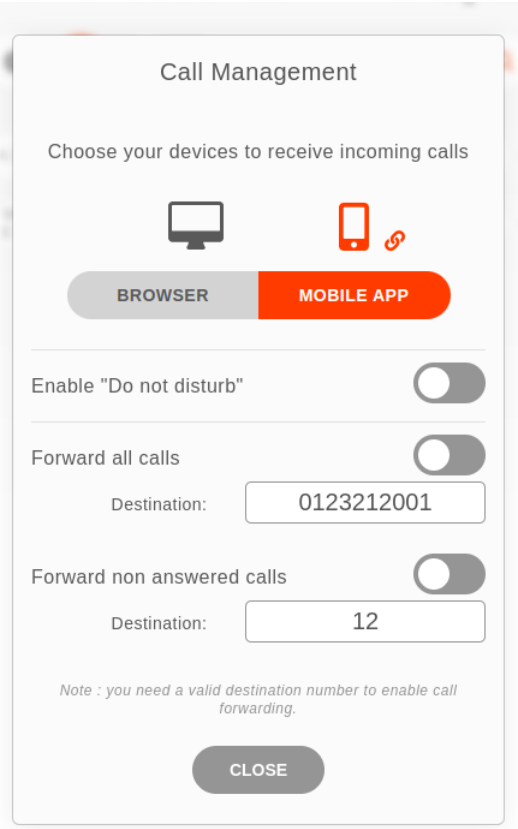
Switchboard is a Web application for switchboard operators. It is designed to handle the call flow of a company switchboard:

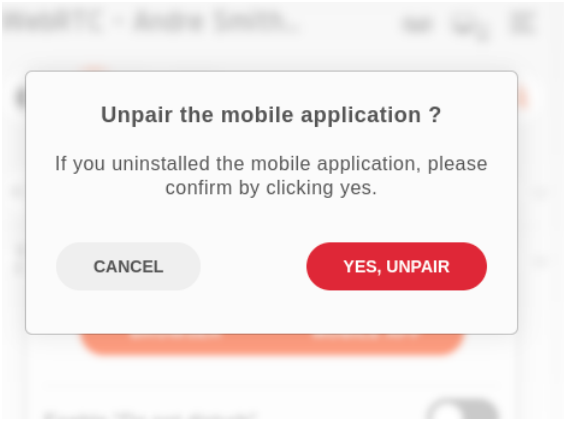
- see current incoming calls,
- answer incoming calls,
- easy search/transfer to user of the company
- being able to put calls in hold

All these actions can be used through keyboard shortcuts.

Important: You need to follow the *switchboard configuration section* to be able to use the switchboard application.


Call Management
Choose your devices to receive incoming calls
BROWSER
MOBILE APP
Enable "Do not disturb"
Forward all calls
Destination:
0123212001
Forward non answered calls
Destination:
12
Note : you need a valid destination number to enable call forwarding.
CLOSE


Call Management
Choose your devices to receive incoming calls
BROWSER
MOBILE APP
Enable "Do not disturb"
Forward all calls
Destination:
0123212001
Forward non answered calls
Destination:
12
Note : you need a valid destination number to enable call forwarding.
CLOSE



Contents

- *Switchboard*
  - *Login*
  - *Answer an incoming call*
  - *Hangup a Call*
  - *Handle Current Call*
    - *Transferring a call*
      - *Attended transfer to a user*
      - *Direct transfer to a user*
    - *Putting a call on hold*
  - *Retrieving a call on hold*
  - *Keyboard Navigation*
  - *Chat*
  - *Common features with CC Agent*
  - *Meeting room*

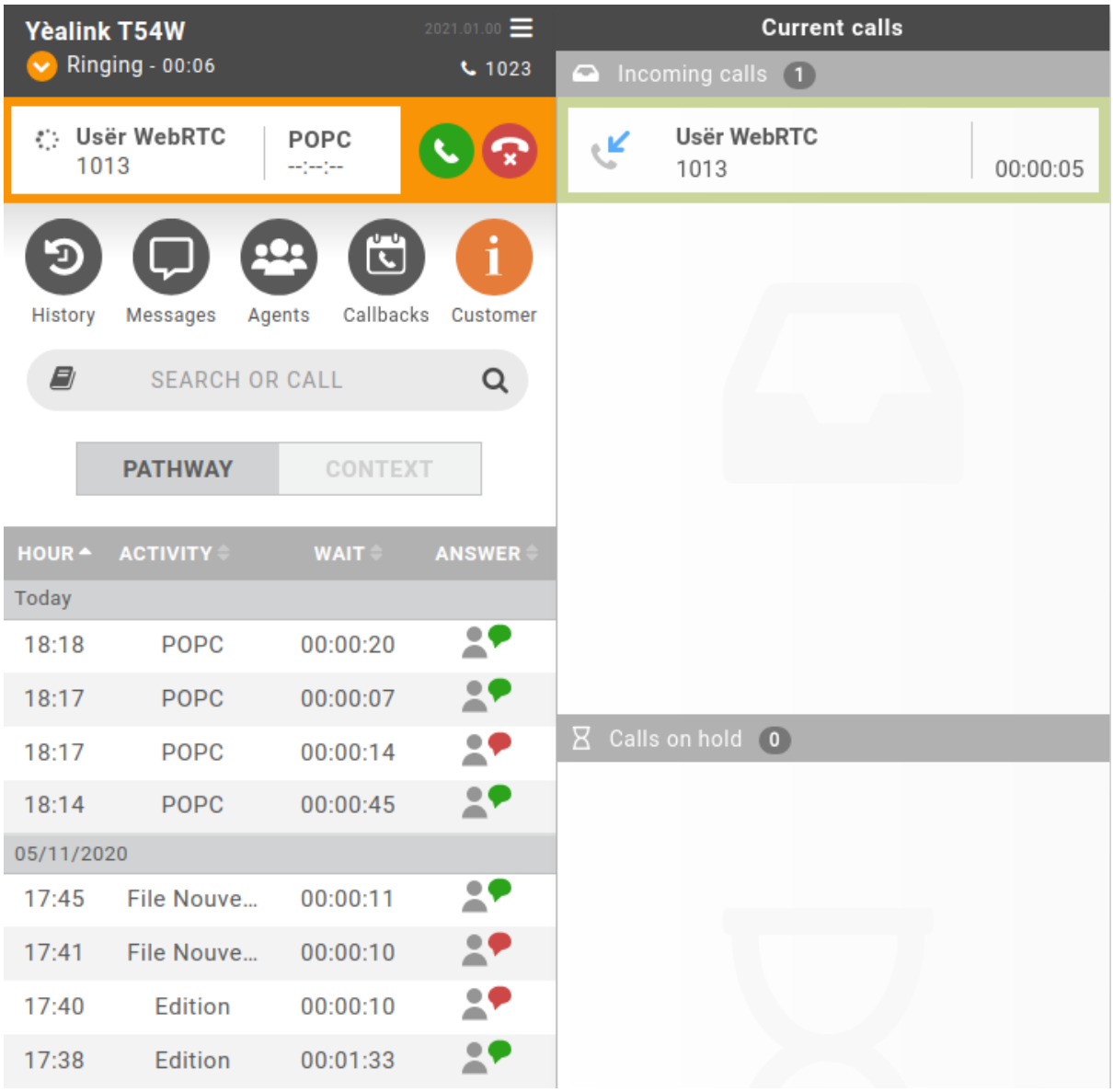
5.2.1 Login

The user can connect to the Switchboard application using https://YOUR_HOST/switchboard.

Note: Automatic login keeps you signed in until you log out.

5.2.2 Answer an incoming call

When the switchboard receives a call, the new call is added to the *Incoming Calls* list on the top right and the phone starts ringing. The ccagent panel displays the client history tab of the person who is calling.



The operator can answer this call by:

- clicking the *Answer* button on the left side, from the *CCagent* frame.
- Pressing the *F3* key.

Once the call has been answered, it is removed from the incoming calls list and only displayed in the *CCagent* frame on the left side.

5.2.3 Hangup a Call

The switchboard operator can hangup its current call by:

- clicking the *Hangup* button in the call control
- Pressing the *F4* key.

5.2.4 Handle Current 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
- put the call on hold by transferring it to the hold queue
- retrieve the call from the hold queue
- end the call using the *Hangup* button

Transferring a call

When the switchboard operator has answered the call, they can transfer it by making either an attended transfer or a direct transfer to a user.

Attended transfer to a user

To make an attended transfer the operator can :

- dial a number in the search bar and press Enter
- call a number from the search result

Then complete the transfer by either:

- clicking the *transfer button* from the ccagent call line
- pressing the *F7* key

Direct transfer to a user

To make a direct transfer the operator can :

- dial a number in the search bar and press the *F8* key
- search for a user and click the Direct Transfer button from the search results

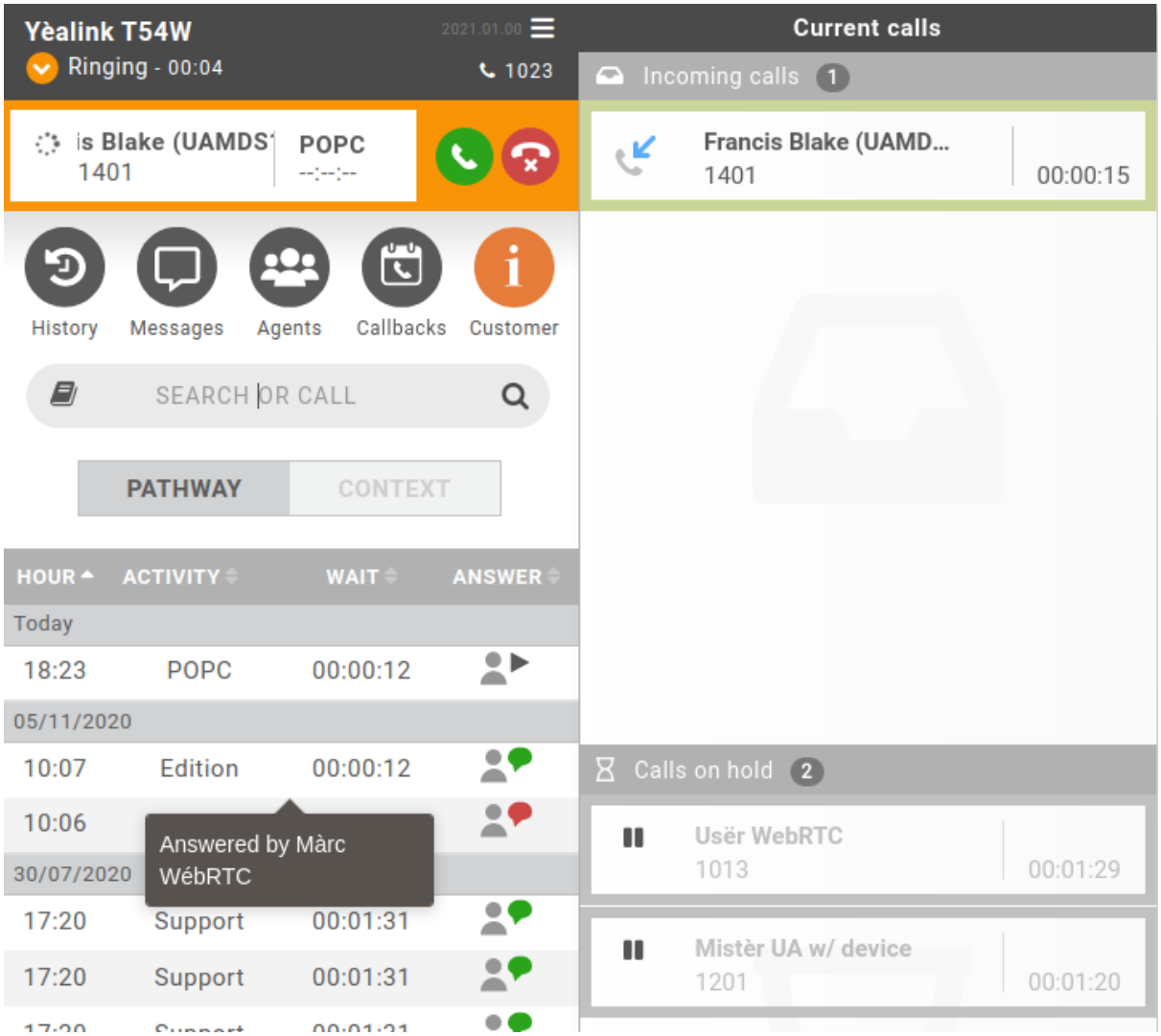
Putting a call on hold

The switchboard operator can put a call on hold by :

- clicking the *Hold* button (*hourglass* icon) in the call control
- pressing the *F9* key

When placing the call on hold, it will be removed from the *Call control* frame and displayed in the *Calls on hold* list on the bottom right. The time counter shows how long the call has been on hold. The calls are ordered from the oldest to the newest.

In the example below, the switchboard operator put two calls on hold and can now answer the incoming call that is currently ringing.

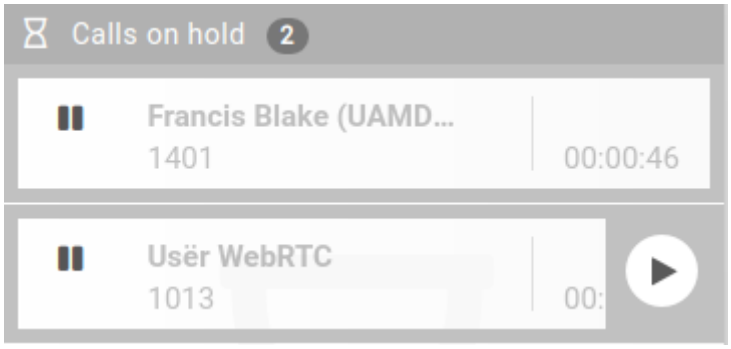


5.2.5 Retrieving a call on hold

After a call was put on hold, the switchboard operator can retrieve it.

To retrieve a call on hold you need to hover the call you want to retrieve and click on the "play" icon:

Note that you can chose which call you want to retrieve (i.e. you can retrieve the third one before the first one).



5.2.6 Keyboard Navigation

Note: The same shortcuts as the ones defined in CC Agent are available in the switchboard : see *CC Agent shortcuts*

In addition, the switchboard allow you to navigate in the hold queue calls list using **Pageup** and **Pagedown** keys, and to retrieve a call by pressing **Enter**.

It also has two exclusive keybindings :

- **F8** to **complete a direct transfer** the current ongoing call to the phone number in the search bar
- **F9** to **send to hold queue** the ongoing call

5.2.7 Chat

The switchboard operator can use the chat to start a conversation with an UC Assistant user or another switchboard operator. For more information about available chat features you can refer to *Instant Messaging*

5.2.8 Common features with CC Agent

The switchboard shares some of the CC Agent features:

- History - see *Agent Call history*

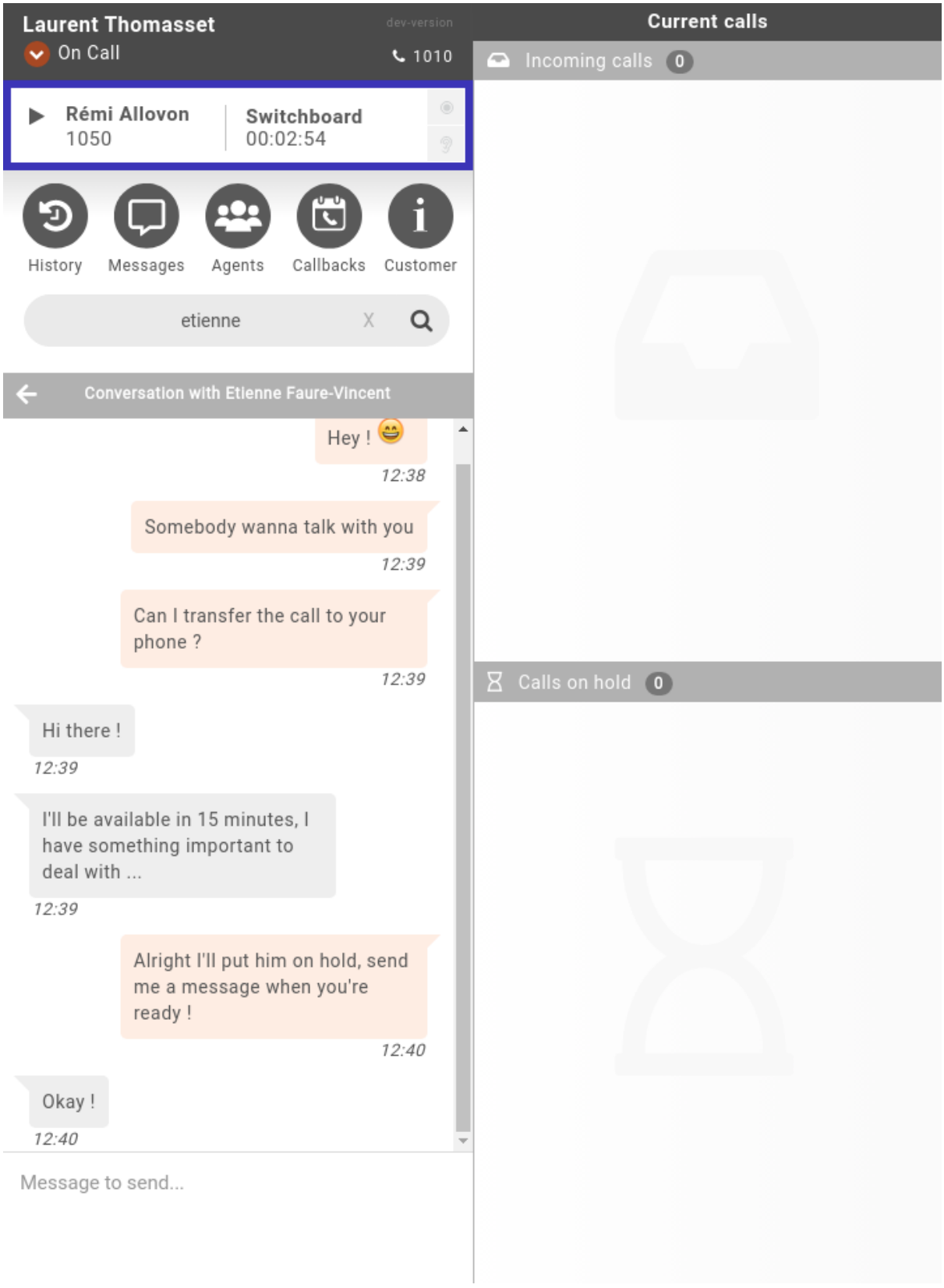
  Important: the Switchboard call history returns a maximum of 100 entries (instead of 20 entries in CC Agent).

- Search - see *CC Agent Search*

  Important: the Switchboard search result adds the possibility

  - to start a Chat conversation to another user of the system
  - to do a direct transfer when having an ongoing call

- Agents - see *Contacts*
- Callbacks - see *Callbacks*
- Customer Info - see *Customer Call History* and *Customer Call Context*
- External Directory - see *External directory*


Laurent Thomasset
On Call
dev-version
1010
Current calls
Incoming calls 0
Rémi Allovon
1050
Switchboard
00:02:54
History
Messages
Agents
Callbacks
Customer
etienne
Conversation with Etienne Faure-Vincent
Hey !
12:38
Somebody wanna talk with you
12:39
Can I transfer the call to your phone ?
12:39
Hi there !
12:39
I'll be available in 15 minutes, I have something important to deal with ...
12:39
Alright I'll put him on hold, send me a message when you're ready !
12:40
Okay !
12:40
Calls on hold 0
Message to send...

5.2.9 Meeting room

You can join an audio-video conference, called meeting room, which will open on the side of your application. See *Meeting Rooms* for more information.

5.3 Desktop Applications

The *XiVO Desktop Application* is a standalone executable for either *UC Assistant*, *Switchboard* and *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.

5.3.1 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 retrieve it on the login page.

You can retrieve it by clicking the Windows or Linux button on the login page from your browser, on the CCAgent or UCAssistant applications.

Note: You can disable download buttons, see *Disabling download desktop applications* section.

Those applications are retrieved from the official XiVO Solutions mirror https://mirror.xivo.solutions/xivo-desktop-assistant/

5.3.2 Update

The update process varies depending on your operating system.

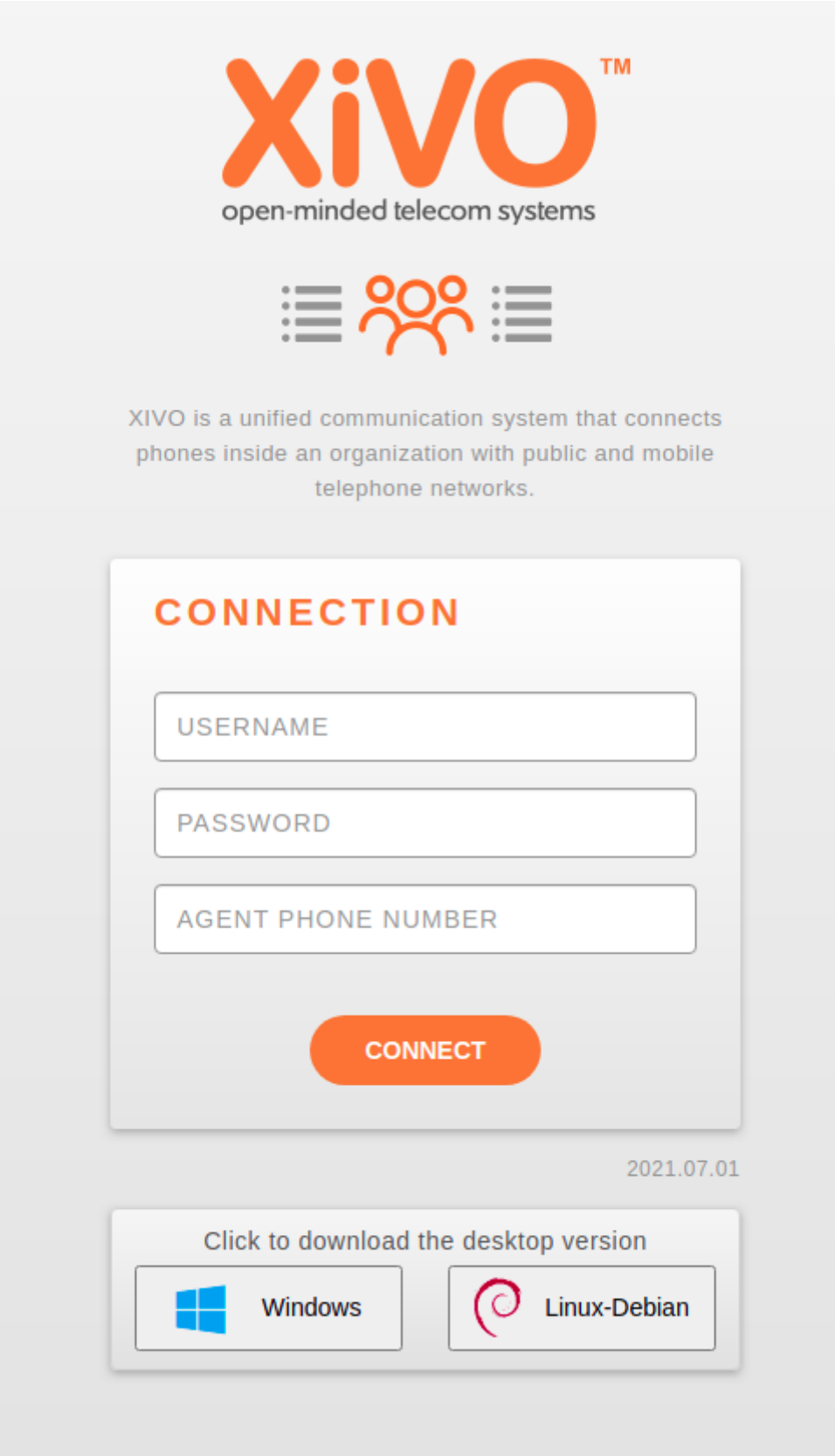
Windows

The desktop application uses the Squirrel framework to perform automatic updates.

1. Upon startup, the application checks *mirror.xivo.solutions* against your current LTS branch to detect if a new version is available.
2. If an update is found, it is downloaded automatically in the background.
3. Once the download is complete, you will receive a notification prompting you to restart the application or wait for next restart to apply the update.

Linux

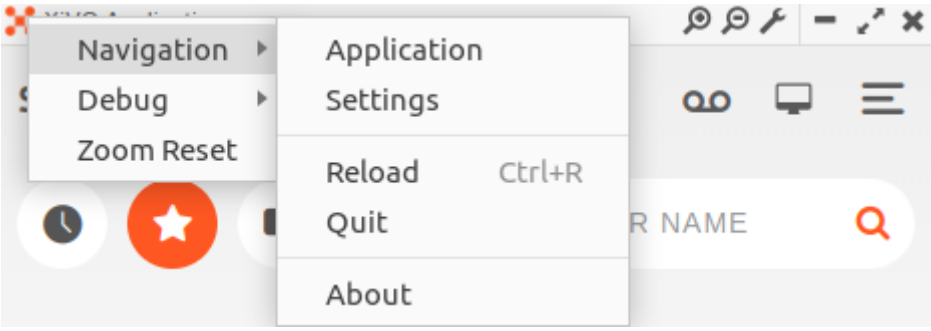
On Linux, updates are directly performed via your package manager.


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
CONNECT
2021.07.01
Click to download the desktop version
Windows
Linux-Debian

5.3.3 Configuration

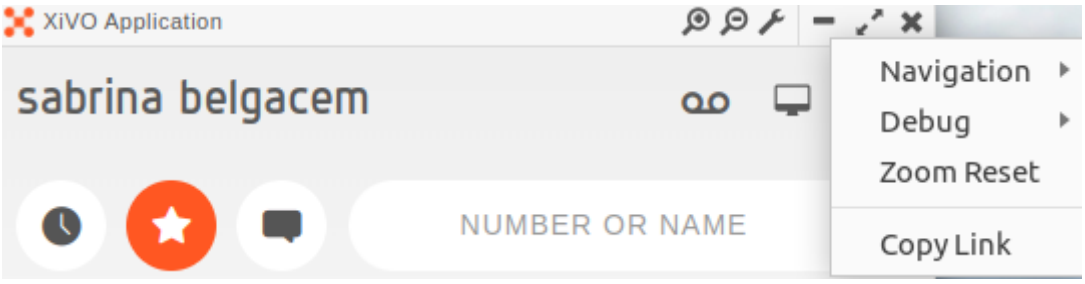
On first launch the application will display the settings page and ask for the application server. Basically it should be the IP address of your server If you don't know it, you need to ask your system administrator or refer to *Protocol and Application Server* paragraph.

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.

Zoom



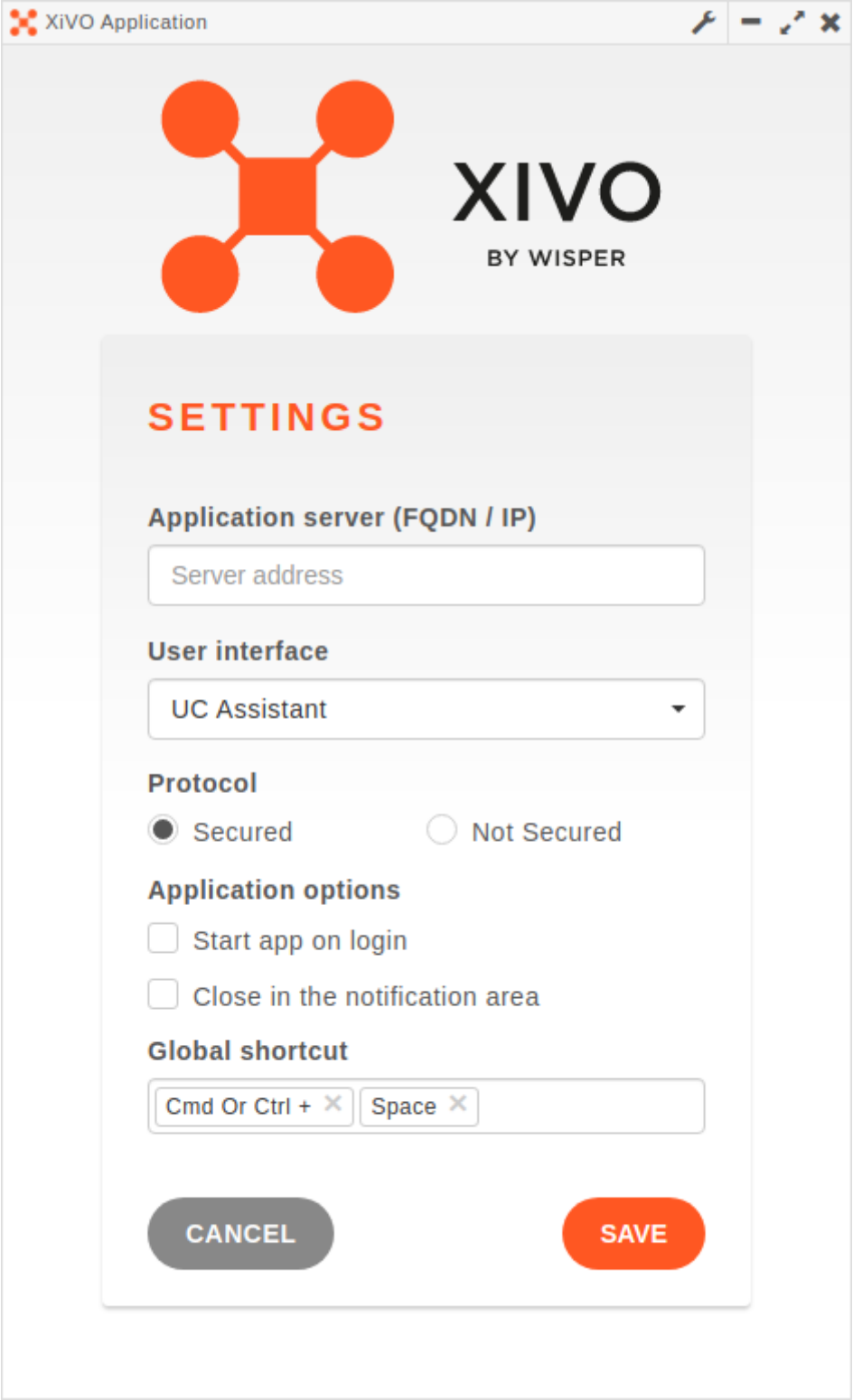
A zoom feature is available on the desktop assistant. To zoom in and zoom out click on zoom icons at the top right of the application. To reset the zoom, open the top menu (accessible through tray icon or right click on the logo) and click on "Zoom reset".

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.


XiVO Application
XIVO
BY WISPER
SETTINGS
Application server (FQDN / IP)
Server address
User interface
UC Assistant
Protocol
Secured
Not Secured
Application options
Start app on login
Close in the notification area
Global shortcut
Cmd Or Ctrl +
Space
CANCEL
SAVE

User Interface

Choose if you want to use the Desktop Assistant to access the UC Assistant application or the CC Agent application.

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 Application Server

In these two fields you need to specify the protocol and address to reach the application. The table below list the possible value:

	Connection URL	
Protocol	**Secure (recommended)**	Non Secure (should not be used)
Application server	`XiVOCC_IP`	`XiVOCC_IP:8070`

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).

Config file

You can specify your Desktop Assistant parameters by providing a `xivoconfig.ini` file or by editing the one that gets created automatically on first launch.

Note:

For Windows, this file must be located in user's **%APPDATA%/xivo-desktop-assistant/application/config**.

For Linux, this file must be located in user's **$HOME/.config/xivo-desktop-assistant/application/config**.

Below is an example of *xivoconfig.ini* file keys in use:

```
APP_PROTOCOL=HTTPS

APP_DOMAIN=xivo.mycorp.com

APP_INTERFACE=UCASSISTANT

APP_STARTUP=false

APP_CLOSE=true

[ALT_SERVERS]
My favorite server = server-favorite.com
Another server = some-other.server.com
My localhost server = 127.0.0.1:8070
```

APP_PROTOCOL

Define the use of HTTPS or HTTP connection.
Values: `HTTP` or `HTTPS` (default to `HTTPS`)

APP_DOMAIN

The server used by the Desktop Assistant.
Either an URL (e.g. xivo.mycorp.com) or an IP:PORT (e.g. 192.168.0.1:8070)

APP_INTERFACE

Which application to open between UC Assistant or CC Agent.
Values: `UCASSISTANT` or `CCAGENT`

APP_STARTUP (optional)

Enable open at startup of OS.
Values: `true` or `false`

APP_CLOSE (optional)

Enable minimization in the taskbar when closing the application.
Values: `true` or `false`

[ALT_SERVERS] (optional)

Override APP_DOMAIN to enable the use of a preset server list dropdown instead of a text input in the settings menu.

One pair of name-address per line.

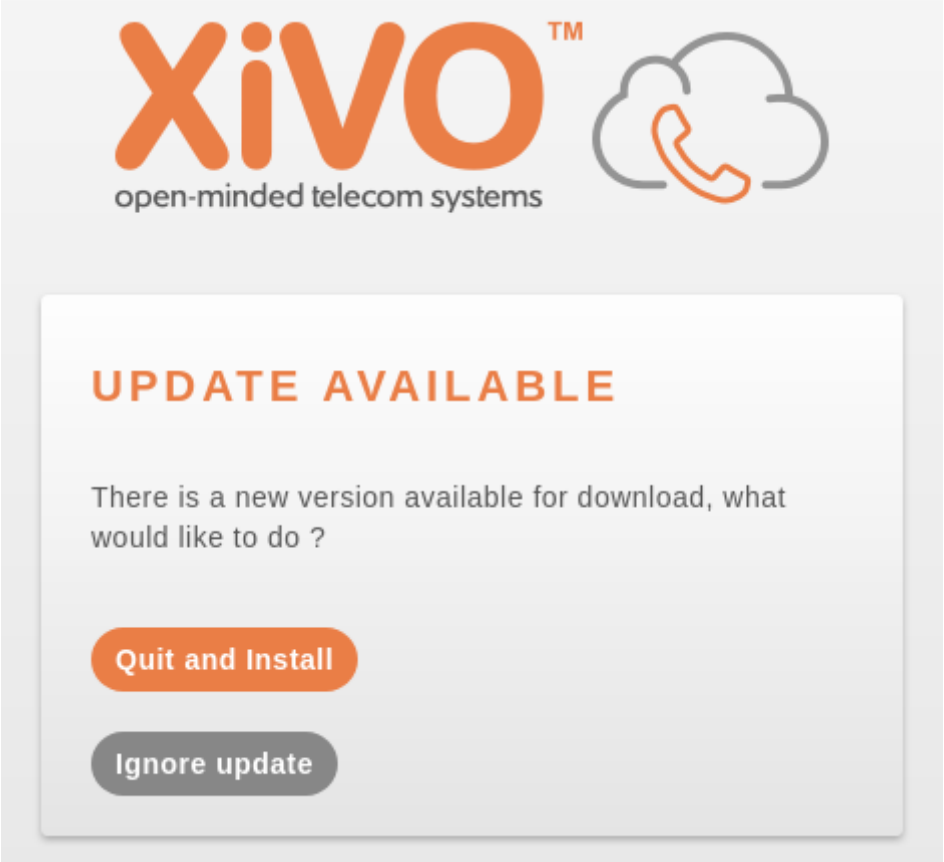
Values: `SERVER NAME = SERVER ADDRESS`

Update

On Windows, the application will check at startup for a new version of the application and offer to upgrade if one is available.

Note: The update will always use the protocol selected in the application settings.

Important: To have automatic update working on secured protocol, you need a valid SSL certificate (see *Install trusted certificate for nginx*).



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
- `-t` with an authentication token as parameter (see *Custom user data directory*)

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"
```

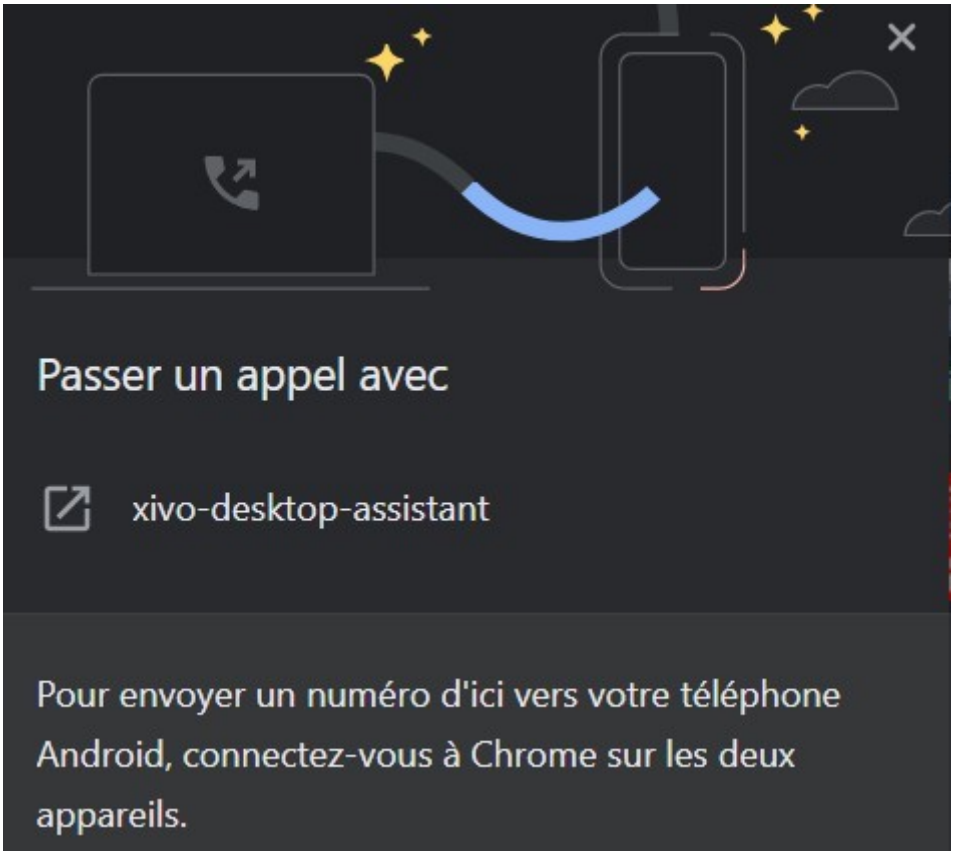
Note: You can set *Run* mode to *Minimized* in shortcut *General* tab to avoid **cmd.exe** blinking at startup.

Chrome integration

If you don't want to have a Chrome popup like this one when you click each time on a *tel:* link on **Windows** to make a call through your desktop application

You can edit your registry base to modify this Chrome key to avoid this display each time you want to call someone from a webpage

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome]
"ClickToCallEnabled"=dword:00000000
```



Known limitations

- Click on links using protocol *tel:* on **Windows** may not work if any version of *Skype* / *Lync* is installed on the PC.
- Autologin by token does not work with CCAgent because a phone number is required

Troubleshoot Application

Logging

Logs of application are available in order to debug a crash at startup or during usage of the application. They do not contain Javascript console logs, but only the interactions with operating system, like global shortcut key or callto and tel protocol.

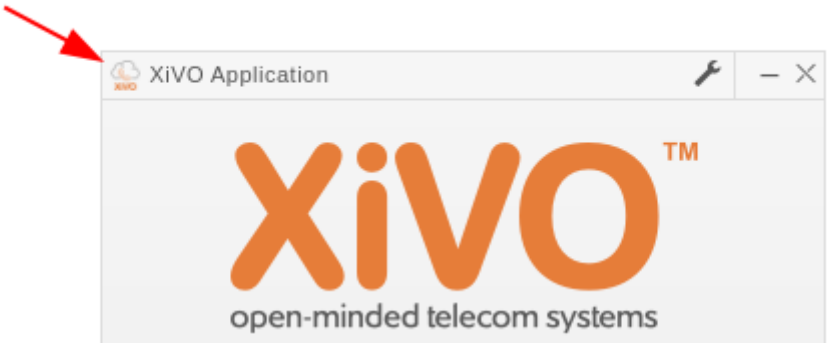
Note:

On Windows, logs are located in user's **%APPDATA%/xivo-desktop-assistant/application/logs**.

On Linux, logs are located in user's **$HOME/.config/xivo-desktop-assistant/application/logs**.

Desktop Application Developper Tools

If needed, you can use the developper console to diagnose a problem "live". To access the console, right click on the xivo cloud on the top left corner of your application

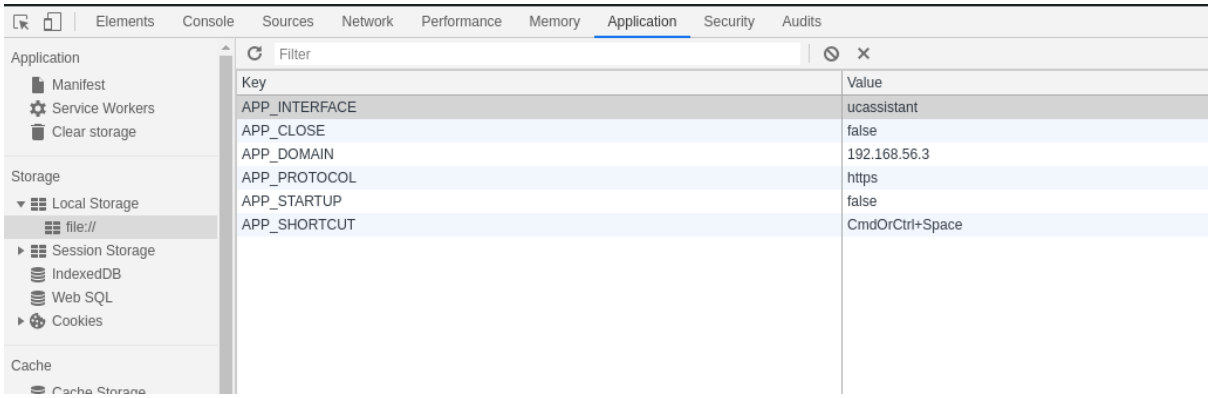


Then, in the 'debug' menu, you can either open the devtool, or the webview's devtool

The normal devtool is the one for the electron application. In this one, you will moslty find errors about shortcut keys, callto and tel protocol, network, and everything system-related.

The webview devtool is the one for what's opened inside the desktop application, so the ccagent or the ucassistant. this devtool is the same one that you get when you open the devtool from the web application in Chrome. You will mostly find network errors or the errors that are specific to the ucassistant and ccagent features.

The main reason to open devtools is to display the console, but you have another functionality used by the desktop application that you might need to debug : the localstorage. It's located in the application tab of the electron devtool. To check the variables set there, you need to click localstorage on the left, and click file://.



The variables that are set in there are the one coming from the xivoconfig.ini (except for the shortcut) and could be the cause of a wrong configuration.

Desktop Application Useful shortcuts

Ctrl + Shift + D toggle the docking of the devtool.

Ctrl + Shift + R refresh the page and clear the cache at the same time.

Ctrl + L clear the console.

Ctrl + Q kill the application.

Certificate

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.

5.3.4 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.

5.4 Mobile Application

Contents:

- *Environment and use cases*
- *Download*
  - *Google Play Store*
  - *App Store*
- *First launch*
- *First connection*
- *Home page*
- *Favorites*
- *History*
- *Mobile search*
- *Incoming call*
  - *IOS*
  - *Android*
- *Outgoing call*
- *Managing a call*
  - *Call transfer*
  - *Put on hold*

00:00:00
00:00:00
00:00:00

- *Notifications*
- *Call management*
  - *Device management*
  - *Do not disturb*
  - *Forward all calls*
  - *Forward calls on no answer*
- *Configuration*
- *Disconnecting / disabling application*

5.4.1 Environment and use cases

The XiVO mobile application comes in addition to the UC Assistant. It allows you to access the services deployed on your XiVO environment via your smartphone. You will need a Wifi or 4/5G connection.

Note: The mobile application can be used in WebRTC and Unique Account mode only.

Warning: Mobile VoIP is not suitable for critical or emergency calls.

Mobile operating systems, public network instability, and reliance on third-party notification services prevent the level of guaranteed reliability and Quality of Service (QoS) required for mission-critical communications. Dedicated telephony/softphony should be used for these purposes.

The use of the application can be :

- Exclusive, i.e. it is your unique Xivo environment
- Shared, i.e. a hybrid approach with your UC Assistant. Several combinations of call processing are then possible.

You can do the following actions:

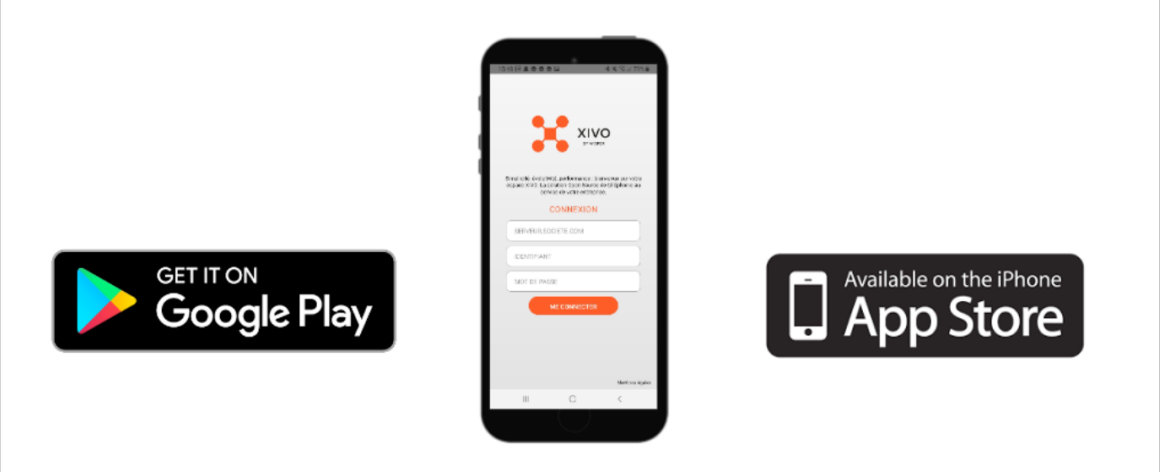
- Call
- Receive a call
- Transfer a call
- Call via your favorites
- Call by searching in the company directory
- Call by searching in your local cell phone directory
- Forwarding management
- Call management
- Visibility on the presence of your colleagues
- Access to voice messaging
- Access to the history
- Pop-up notification for missed calls
- Pop-up message notification on the voice mail.

The following services are not currently available :

- Video conferencing
- Chat

These features are in our roadmap.

5.4.2 Download



Google Play Store

The Xivo - Xivo mobile application is available on the Android Play Store. You need to search for the "XiVO" application. Once on the application you can tap *install* and then *open the application*.

App Store

The Xivo - Xivo mobile application will be available on the IOS App Store. You need to search for the application "XiVO".

5.4.3 First launch

When you open the application for the first time, it will ask for different authorizations : * Mandatory authorizations, for the application to function properly * An optional authorization, to allow access to your phone contacts.

- List of requested authorizations : • Stop battery optimization • View in the foreground of the application • Microphone • Phone account to make calls • Local phone book contacts (Optional)

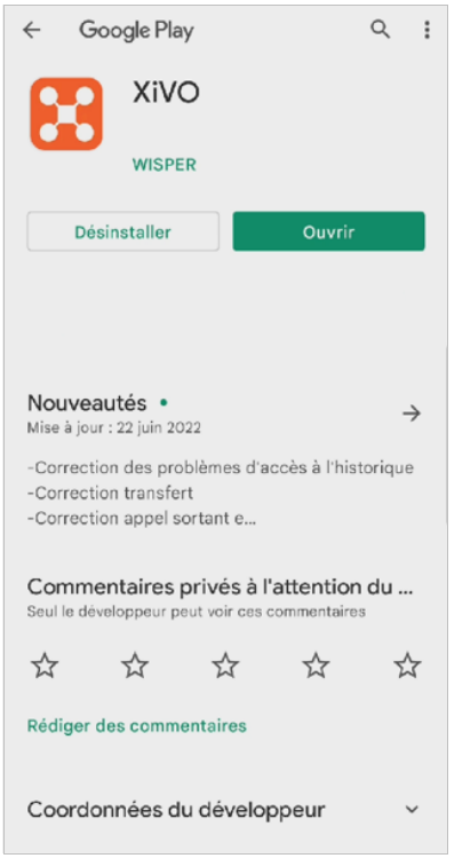
5.4.4 First connection

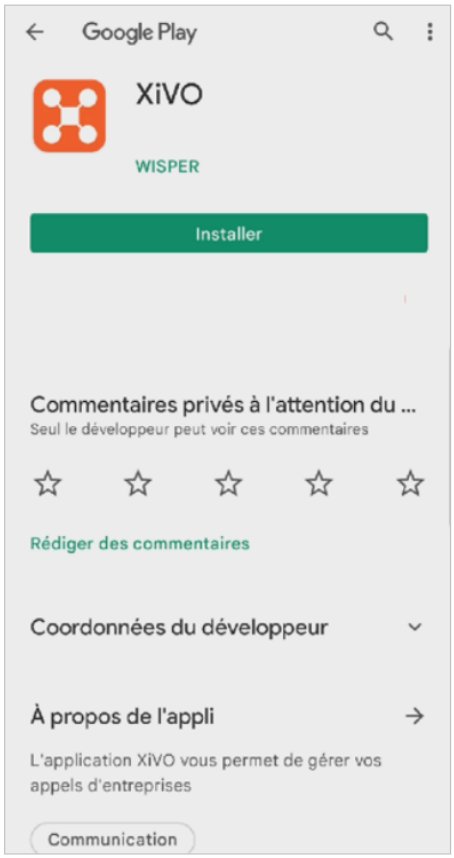
When you first connect to the application, a connection page will be displayed with the following fields :

- the address of the telephony server,
- your login,
- your password

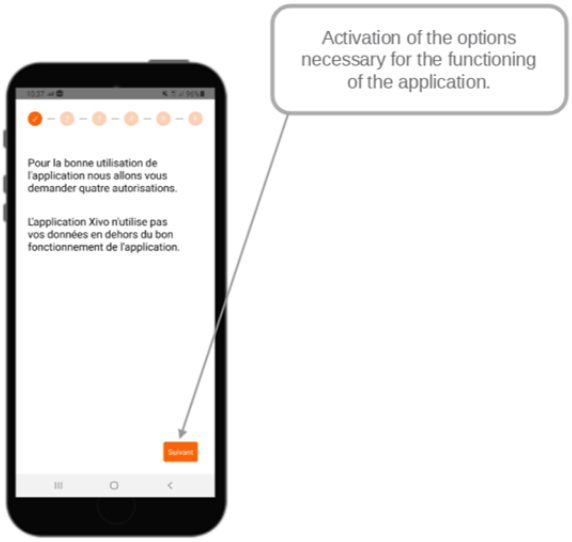
Where to find the server address:

- On your Web browser

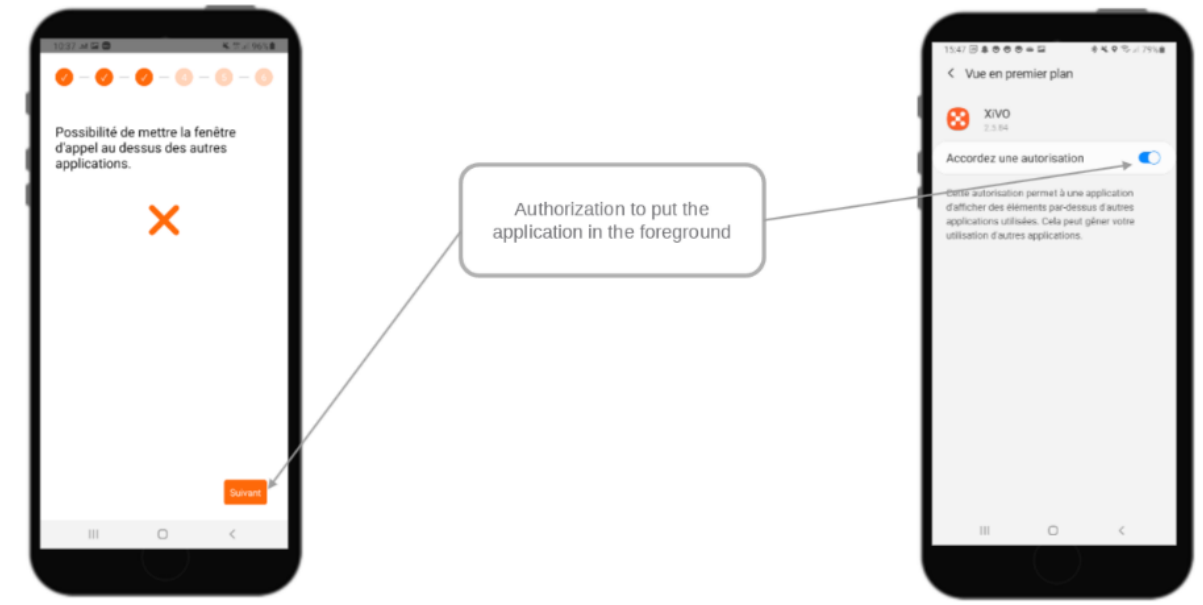


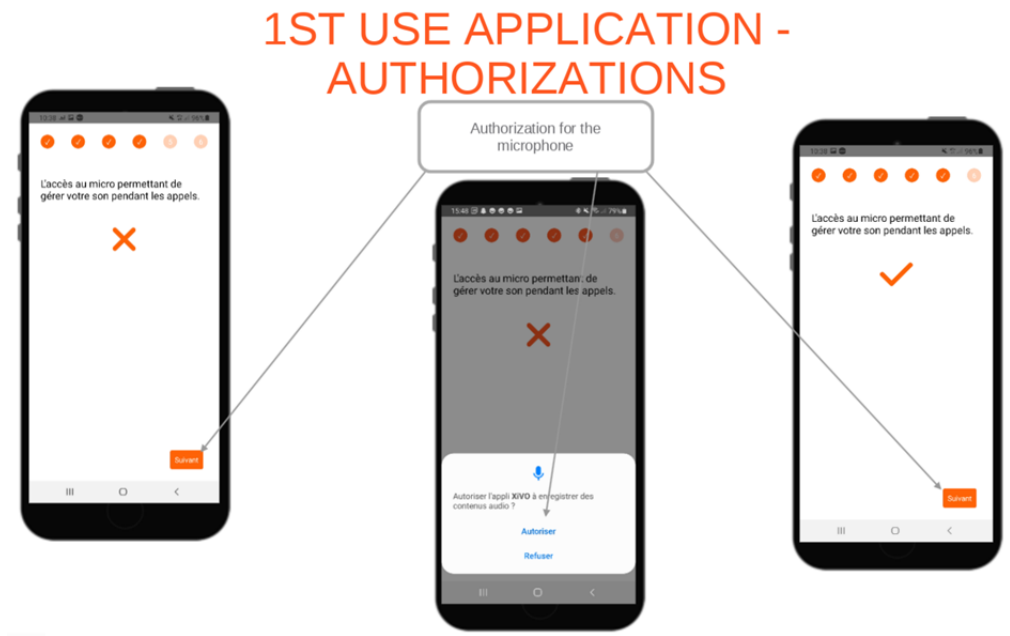


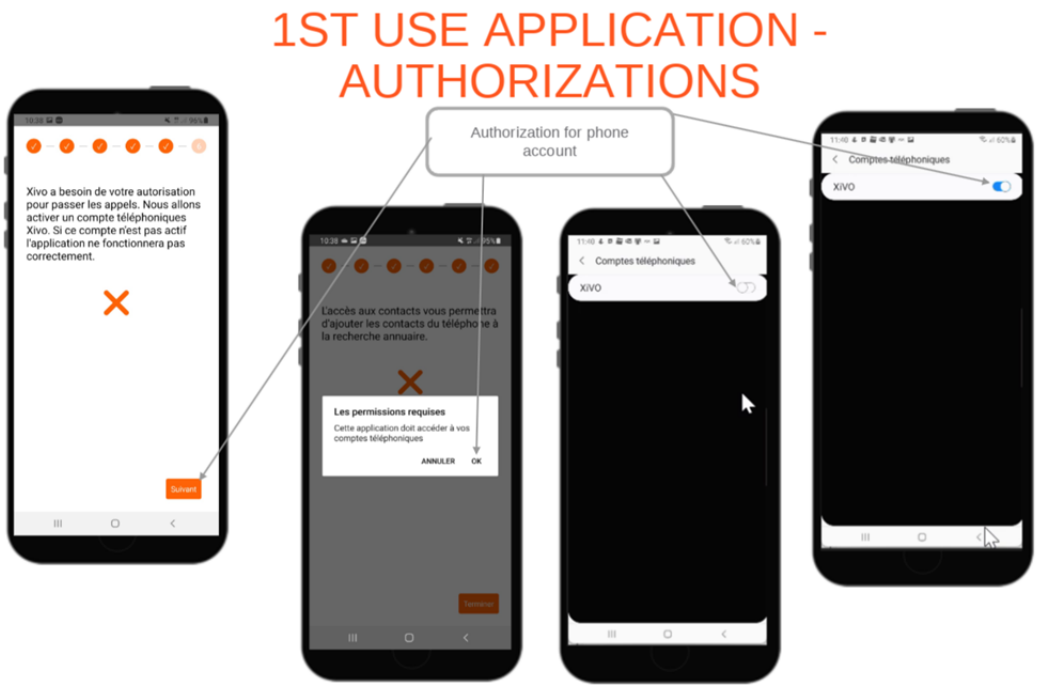
1ST USE APPLICATION

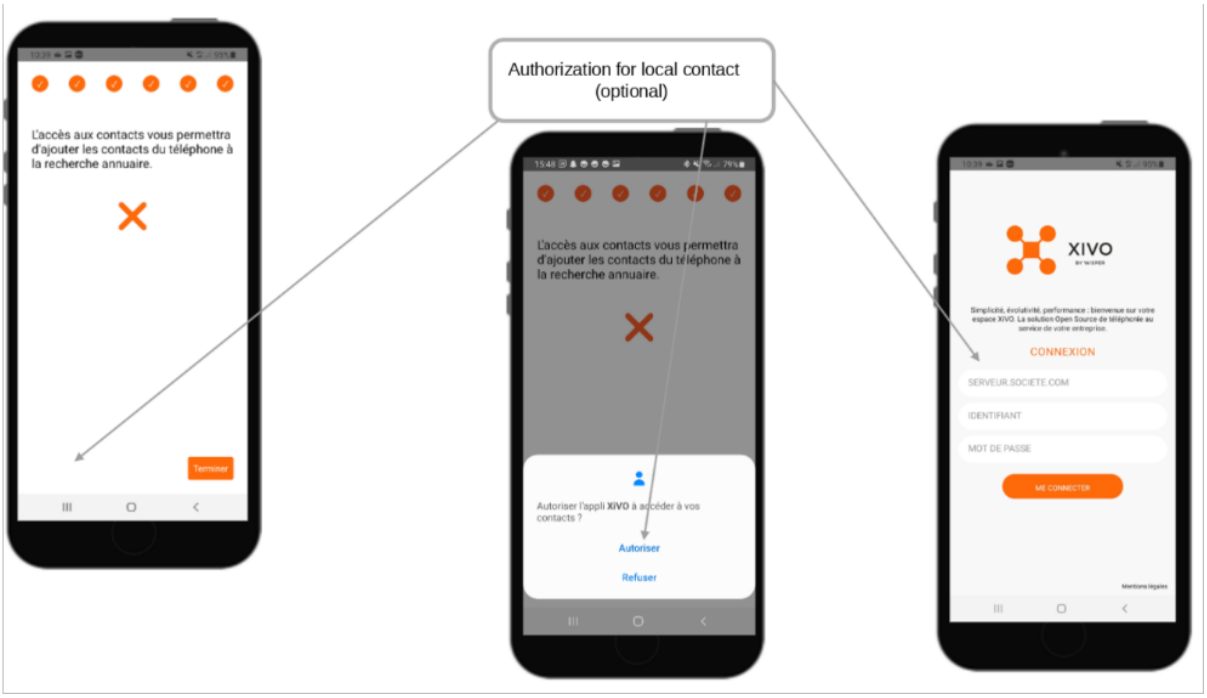


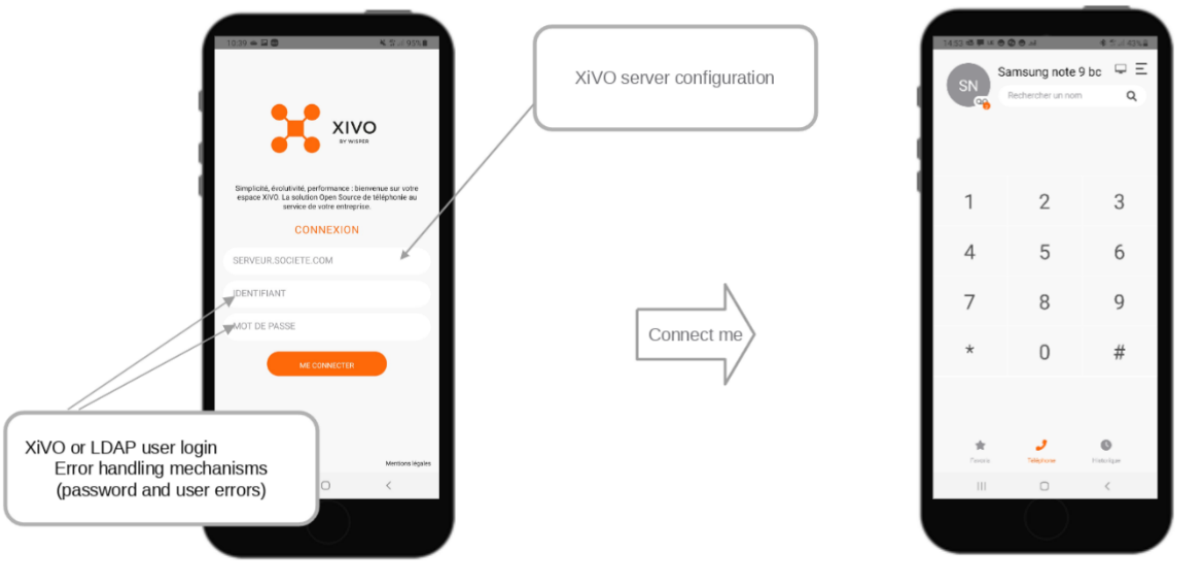
Authorization to stop battery optimization
L'arrêt de l'optimisation de la batterie pour que l'application soit priorisée par le téléphone même en économie de batterie.
Suivant
Arrêter optimisation utilisation batterie ?
Refuser
Autoriser

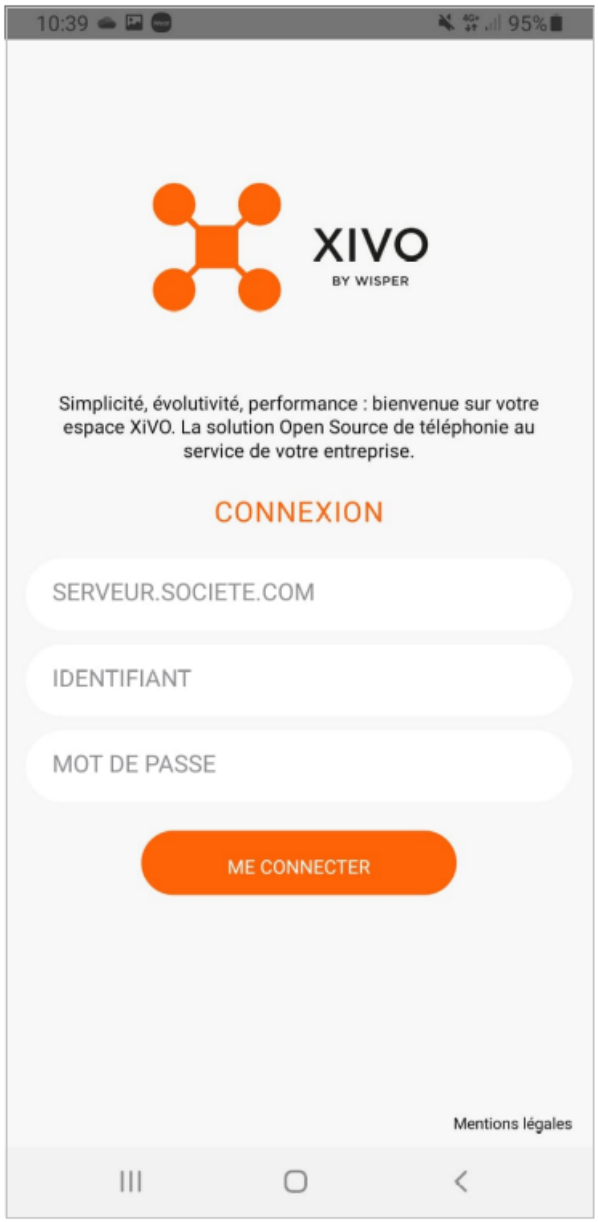

Possibilité de mettre la fenêtre d'appel au dessus des autres applications.
Suivant
Authorization to put the application in the foreground
Vue en premier plan
XiVO
Accordez une autorisation

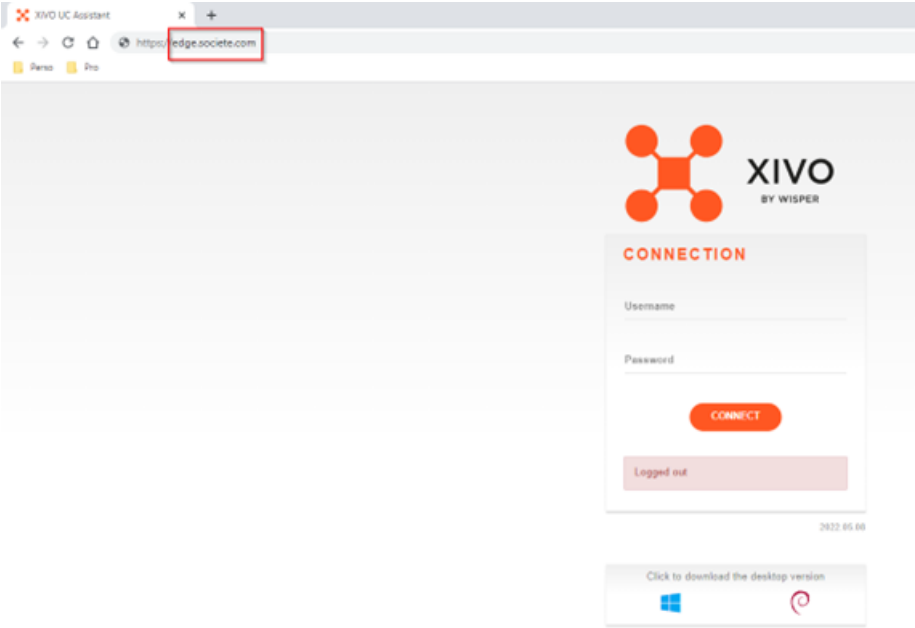

1ST USE APPLICATION - AUTHORIZATIONS
Authorization for the microphone
L'accès au micro permettant de gérer votre son pendant les appels.
Suivant
Autoriser
Refuser


1ST USE APPLICATION - AUTHORIZATIONS
Authorization for phone account
Xivo a besoin de votre autorisation pour passer les appels. Nous allons activer un compte téléphoniques Xivo. Si ce compte n'est pas actif l'application ne fonctionnera pas correctement.
Suivant
Les permissions requises
Cette application doit accéder à vos comptes téléphoniques
ANNULER
OK
Comptes téléphoniques
XIVO


Authorization for local contact (optional)
L'accès aux contacts vous permettra d'ajouter les contacts du téléphone à la recherche annuaire.
Terminer
Autoriser l'appli XiVO à accéder à vos contacts ?
Autoriser
Refuser
XIVO
CONNEXION
SERVEUR.SOCIETE.COM
IDENTIFIANT
MOT DE PASSE
ME CONNECTER


XiVO server configuration
XIVO
CONNEXION
SERVEUR.SOCIETE.COM
IDENTIFIANT
MOT DE PASSE
ME CONNECTER
XiVO or LDAP user login
Error handling mechanisms
(password and user errors)
Connect me
Samsung note 9 bc
SN
1 2 3
4 5 6
7 8 9
* 0 #

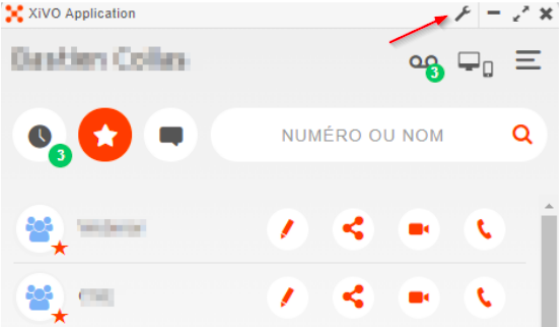

10:39
95%
XIVO
BY WISPER
Simplicité, évolutivité, performance : bienvenue sur votre espace XiVO. La solution Open Source de téléphonie au service de votre entreprise.
CONNEXION
SERVEUR.SOCIETE.COM
IDENTIFIANT
MOT DE PASSE
ME CONNECTER
Mentions légales



If you use your web browser to access Xivo, you will find the information directly in the address bar.

- On your Desktop Assistant

THe server address is in the parameters, on the upper right corner of your desktop assistant. Once in the settings, use the information in "application server".



If you use only the app for *Xivo connectivité*, please ask your IT manager for the url.

About the authentication fields: You must use the username and password that you usually use when logging in.

When you first login, a notification will be displayed in your UC Assistant. It shows you that you can choose, via the call management, on which devices you want to receive calls. See gif below :

5.4.5 Home page

Once you are logged in, you will be taken to the home page which is a dial pad. From this page you can access : * A dial pad to initiate a call via a number * Your Favorites * Your History * Your voice mail when you have one * A search bar * Access to the configuration

You can launch a call by typing a number on the keyboard. For example, if I type *55 to launch a call to the echo test, a call button appears at the bottom of the screen.

5.4.6 Favorites

You can access your favorites through the button located at the bottom left of the mobile application. Once in your favorites, you can launch a call by pressing directly on the handset on the right of the chosen contact. You will also have a view on the status of your colleagues.

5.4.7 History

You can access your call history through the button on the bottom right of the mobile application. In the history you will have the possibility to launch a call by pressing directly on the handset. You'll also be able to see the status of the other users.

5.4.8 Mobile search

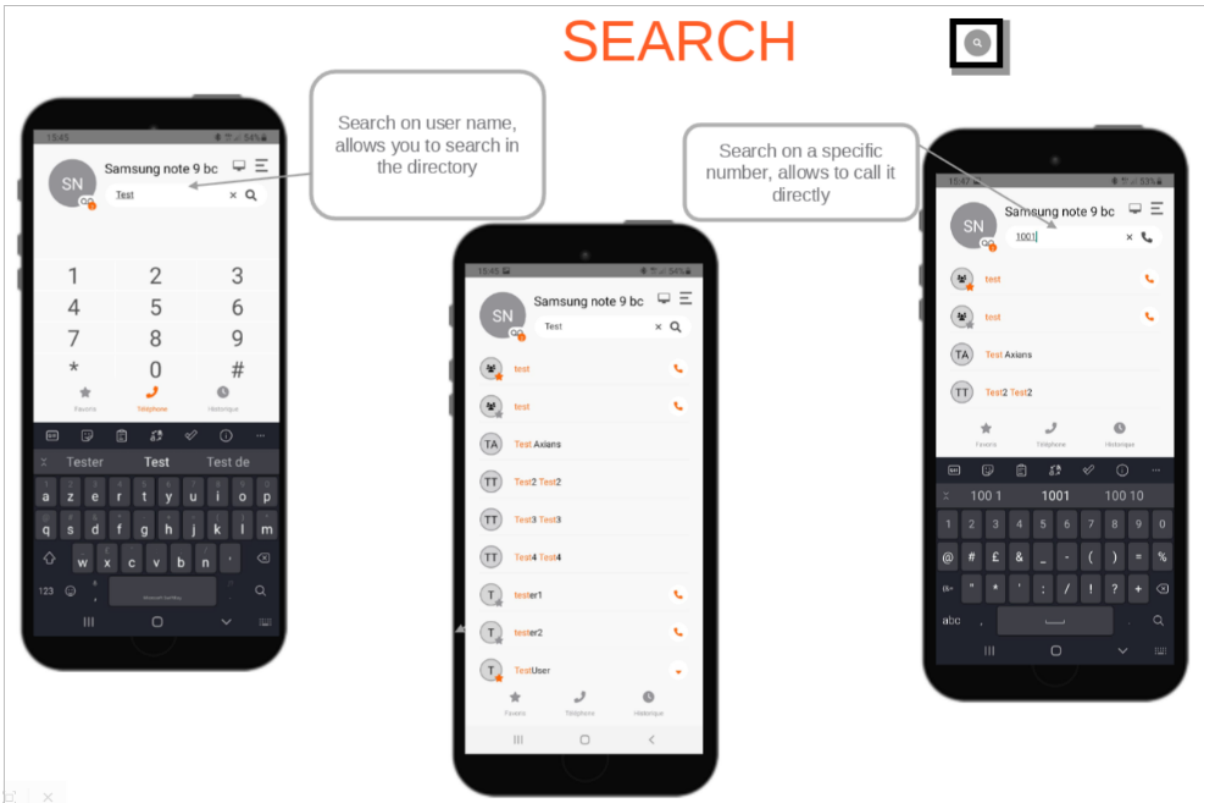
You can find the search bar on the top right of the mobile application on all of the following menus: Phone, History, Favorites.

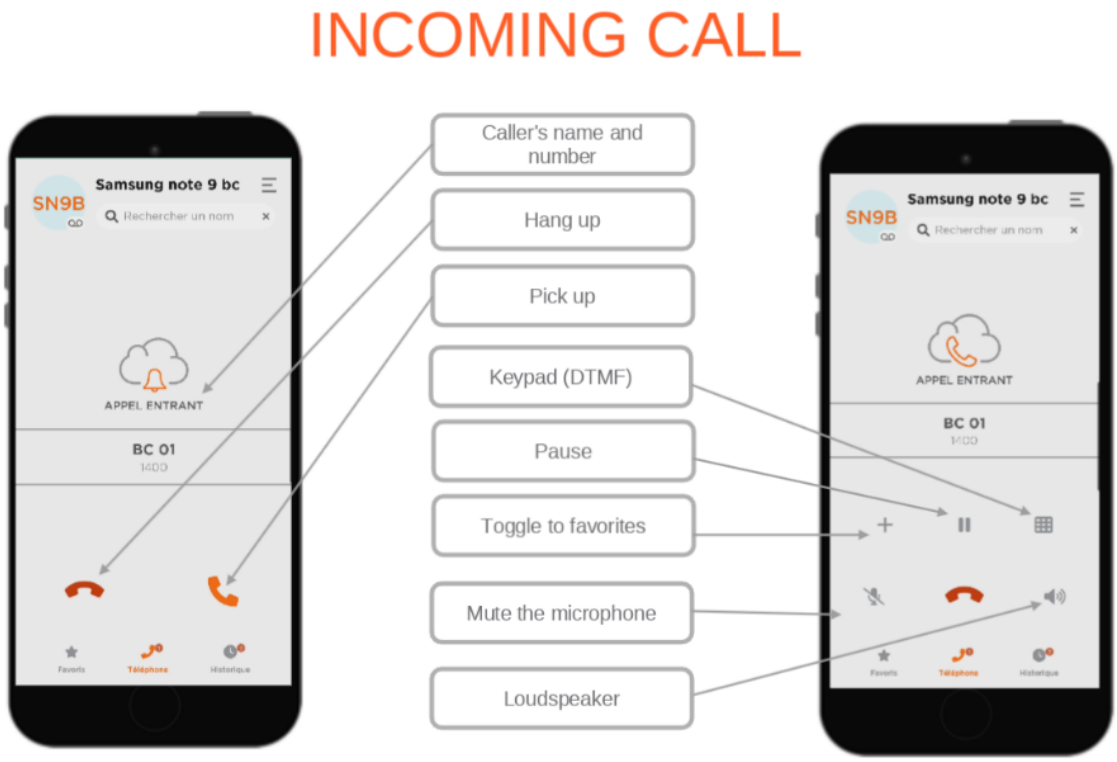
You will be able to search your personal XiVO directory, the company directory and also the local directory of your phone.

5.4.9 Incoming call

The reception of a call is done via push notification - this notification wapes up the app).

HISTORY
Name or number display
Timestamp
Incoming calls
Outgoing calls
Missed calls
Samsung note 9 bc
SN
Rechercher un nom
Aujourd'hui


SEARCH
Search on user name, allows you to search in the directory
Search on a specific number, allows to call it directly
Samsung note 9 bc
SN
Test
Tester
Test de
1001
100 1
100 10



IOS

When receiving a call on IOS, you can switch directly to the mobile application via the XiVO button by unlocking your mobile.

Android

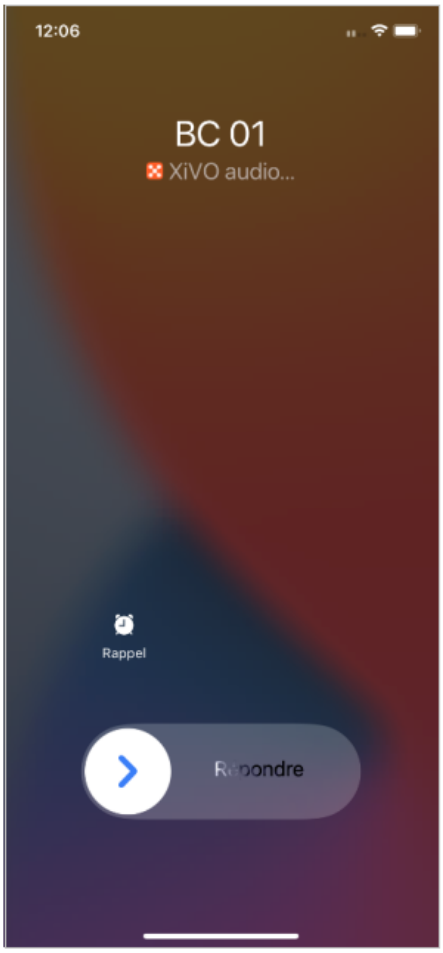
When receiving a call on Android, the call management will be accessible on the mobile application when you have unlocked your mobile.

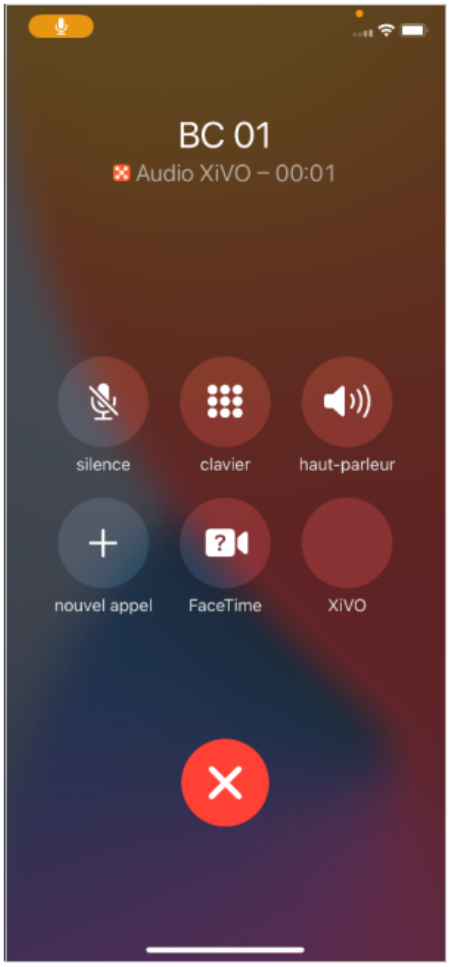
In call display :

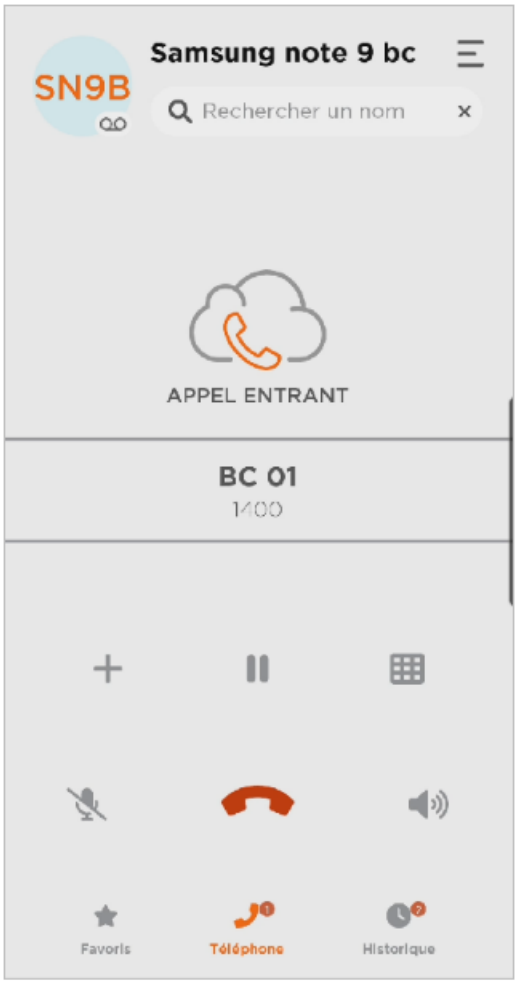
After picking up the phone you can pause a call, mute the microphone, activate speakers. The "+" button allows to call another number.

5.4.10 Outgoing call

In some cases, you might get "Your call is in transit, thank you for your patience" before getting the screen "your call is ringing". Once the person you are calling has picked up the phone you will be taken to the call management page.

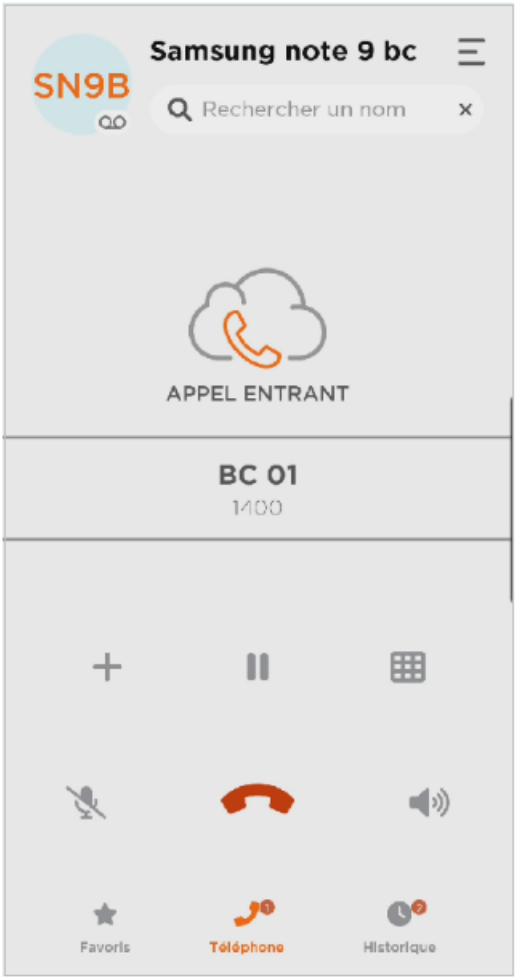

12:06
BC 01
XiVO audio...
Rappel
Répondre


BC 01
Audio XiVO – 00:01
silence
clavier
haut-parleur
nouvel appel
FaceTime
XiVO

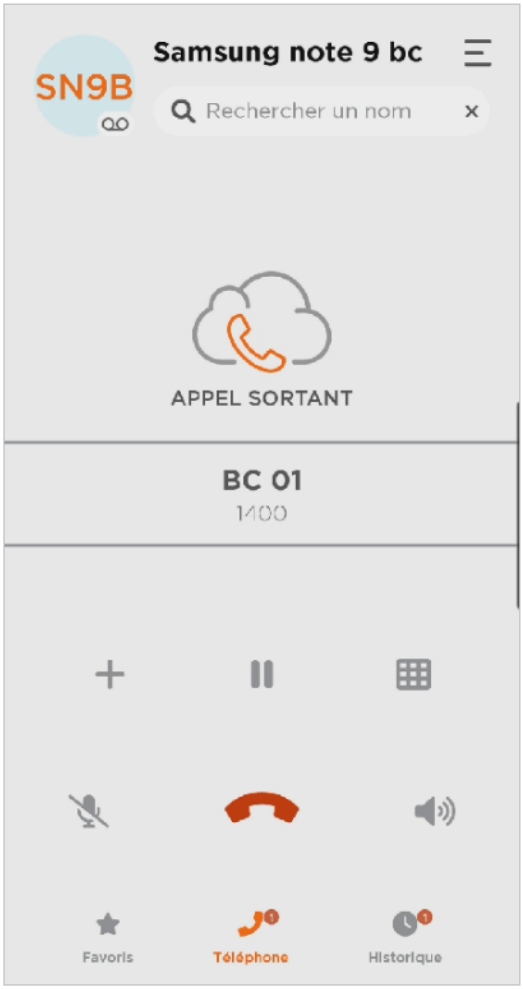

Samsung note 9 bc
SN9B
Rechercher un nom
APPEL ENTRANT
BC 01
1400
Favoris
Téléphone
Historique

12:10
70%
Appel entrant
BC 01
BC 01

Orange F
54%
14
45
mar. 31 mai
WISPER
CEBOX
BY WISPER
XIVO
BY WISPER
Charge : 54 %
3 h 44 min avant pleine charge


SN9B
Samsung note 9 bc
Rechercher un nom
APPEL ENTRANT
BC 01
1400
Favoris
Téléphone
Historique

SN9B
Samsung note 9 bc
Rechercher un nom
APPEL SORTANT
BC 01
1400
Votre appel sonne.
Favoris
Téléphone
Historique

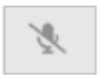

SN9B
Samsung note 9 bc
Rechercher un nom
APPEL SORTANT
BC 01
1400
Favoris
Téléphone
Historique

5.4.11 Managing a call

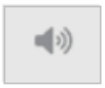
When you are on a call, whether it's an incoming or outgoing call, you get the call management options.

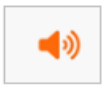
The following actions are available :

hang up


Microphone On

Microphone Off


Speaker not activated


Speakerphone On

Initiate a second call

Put the call on hold

Using the numeric keypad during a call

One call in progress

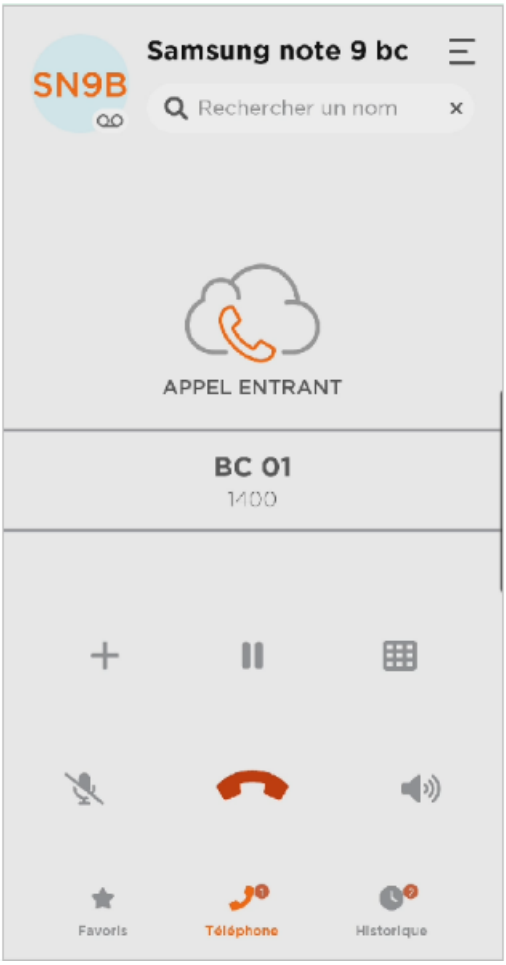
Two calls in progress

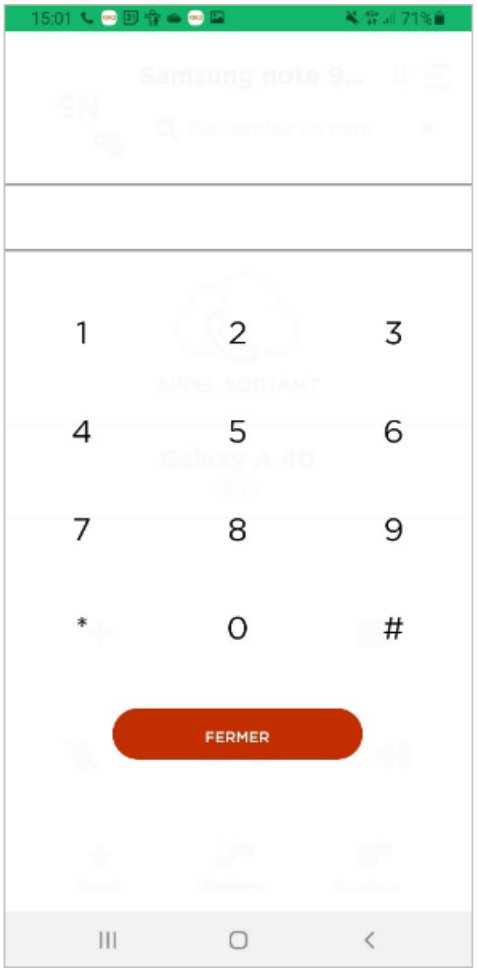
Call transfer

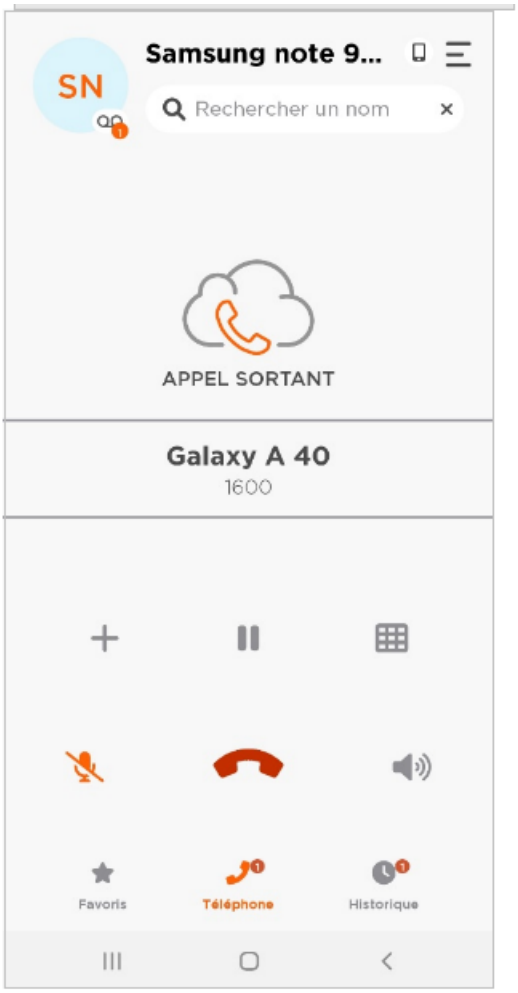
You can transfer a call while being on a call. There are two ways to do it: * Pressing the "+" button allows you to go directly to the Favorites and select the person you want to reach for your transfer. Once you have selected your favorite, the call will be launched and the person you were talking to will be put on hold.

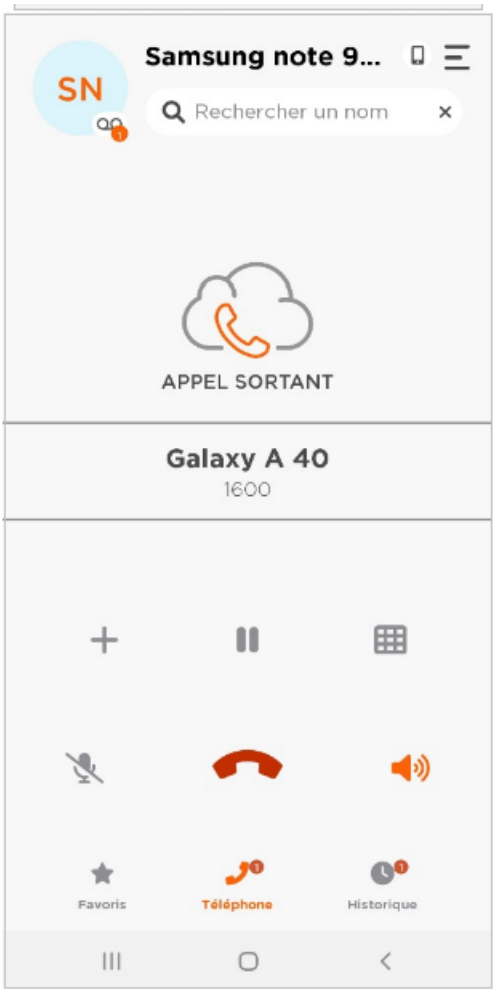
- You can access the pages :
  - Favorites
  - History
  - Search

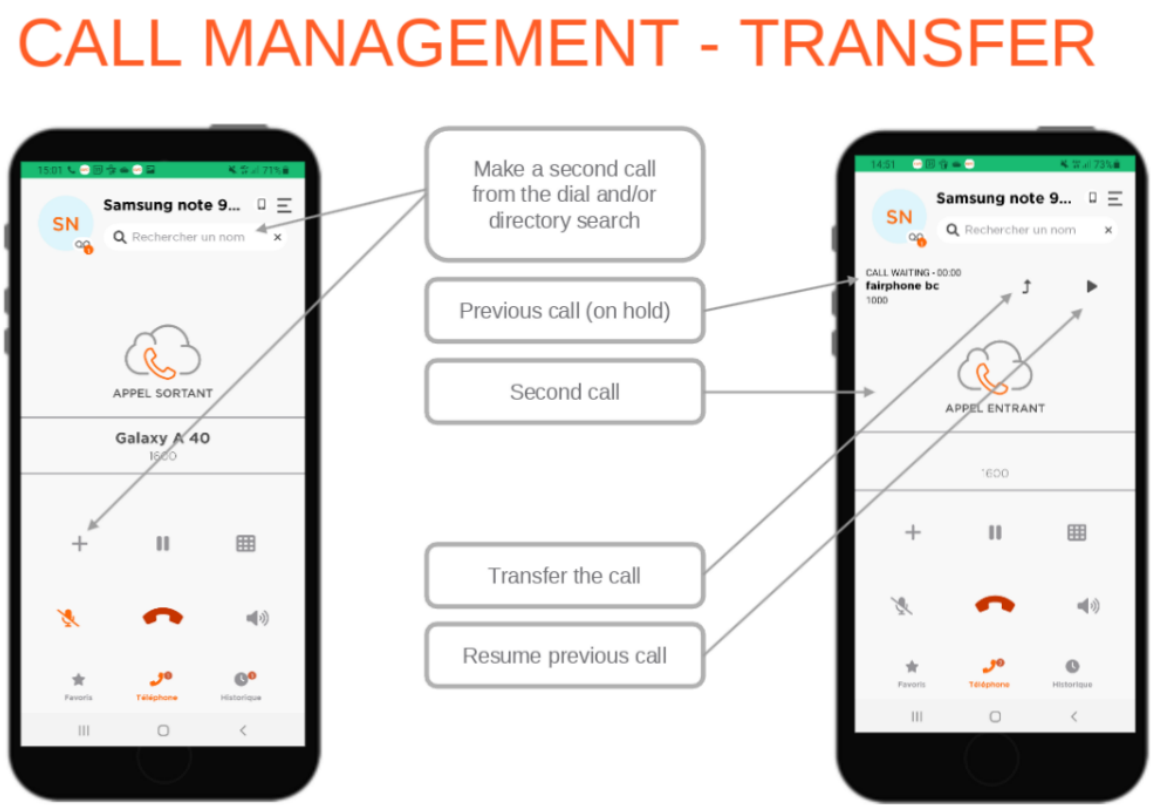
From one of these pages, you can launch a second call and put on hold the person with whom you were in a conversation.


SN9B
Samsung note 9 bc
Rechercher un nom
APPEL ENTRANT
BC 01
1400
Favoris
Téléphone
Historique


15:01
71%
1
2
3
4
5
6
7
8
9
*
0
#
FERMER


SN
Samsung note 9...
Rechercher un nom
APPEL SORTANT
Galaxy A 40
1600
Favoris
Téléphone
Historique


SN
Samsung note 9...
Rechercher un nom
APPEL SORTANT
Galaxy A 40
1600
Favoris
Téléphone
Historique



Put on hold

When you receive a call while being already on a call, you can put your current call on hold. In this context, the person you put on hold will hear a music on hold. You can then resume the call via the play button.

5.4.12 Notifications

Notifications are integrated in the mobile application. You will receive these notifications on your mobile application when you have a missed call.

5.4.13 Call management

Device management

During the installation and the first connection to the mobile application, you must have noticed a change in the call management of your UC assistant.

You can, on your UCAssistant select the ringing device:

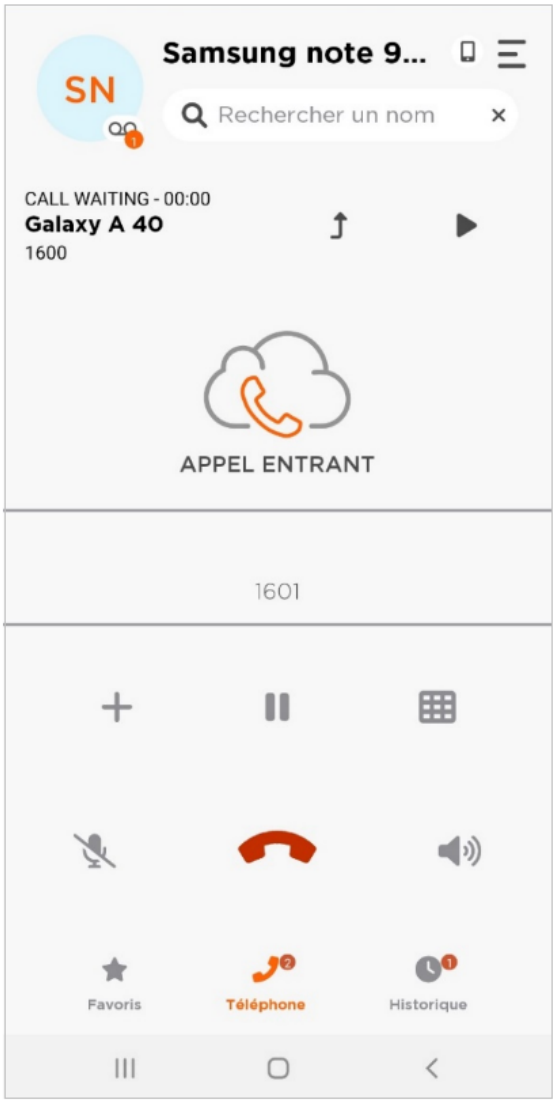
This allows you to select either your UC Assistant or the mobile application or both.

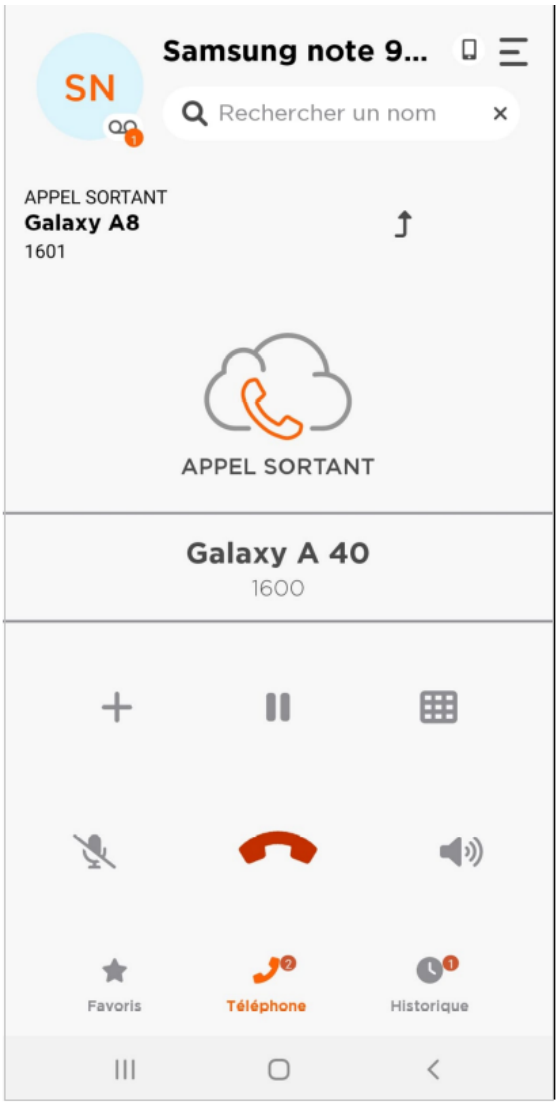
Documentation link below : https://documentation.xivo.solutions/en/2022.10/usersguide/uc_assistant/index.html#using-web-and-mobile-applications-together

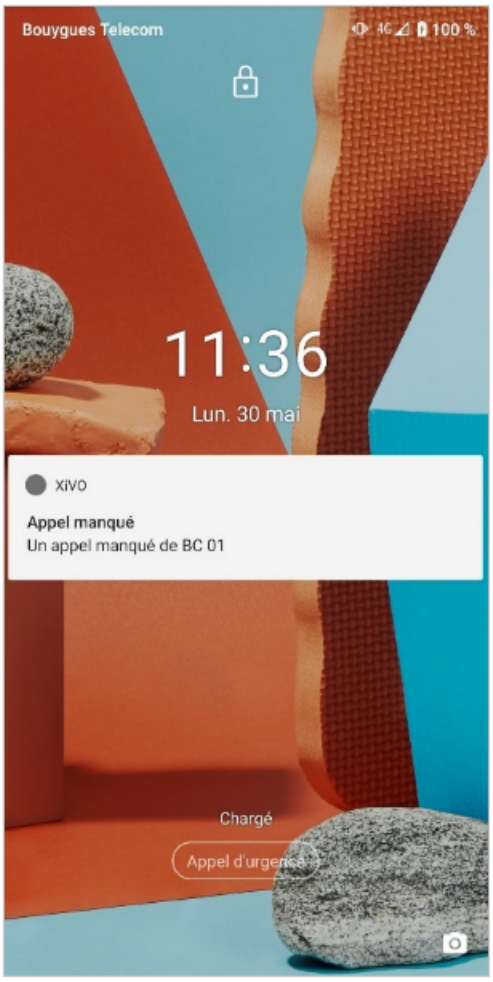
This management capabilities will also be available in the app.

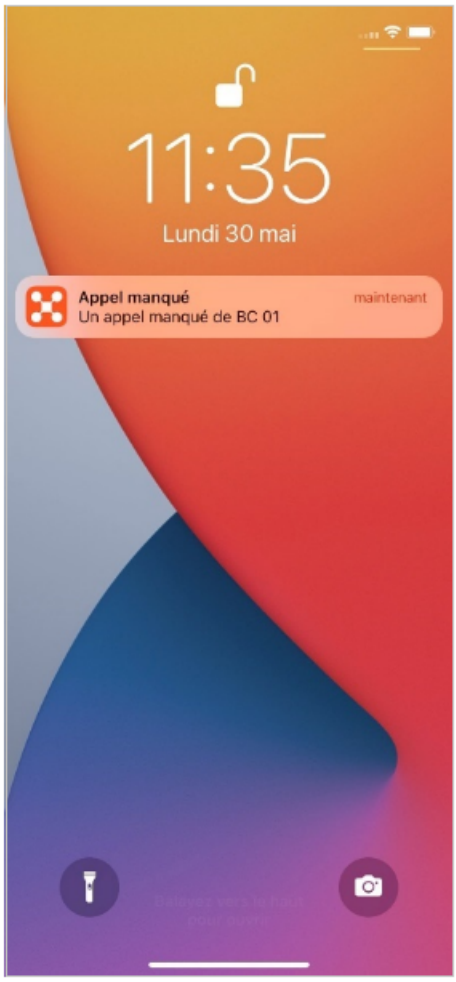
Please find some examples below:

When selecting the mobile application, you will see a mobile icon in the top right of the application.

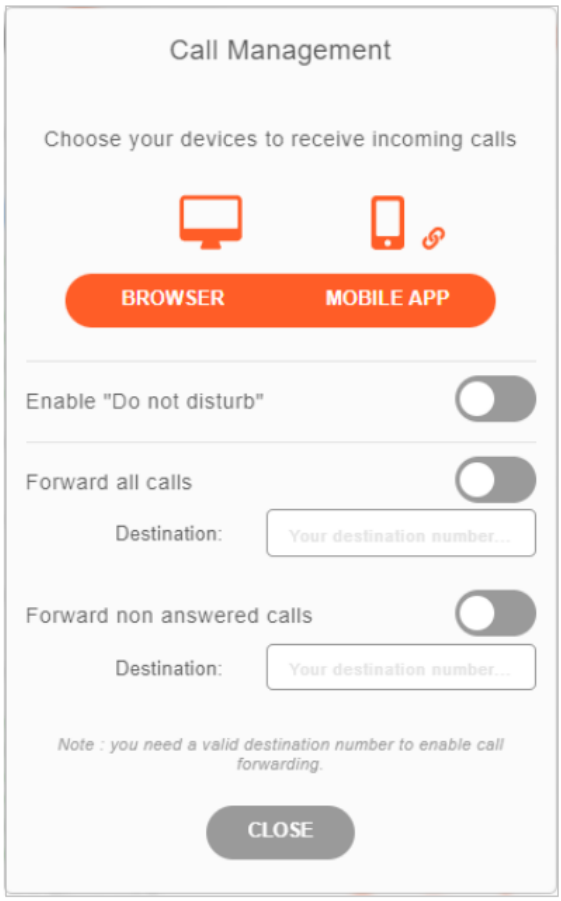

SN
Samsung note 9...
Rechercher un nom
CALL WAITING - 00:00
Galaxy A 40
1600
APPEL ENTRANT
1601
Favoris
Téléphone
Historique

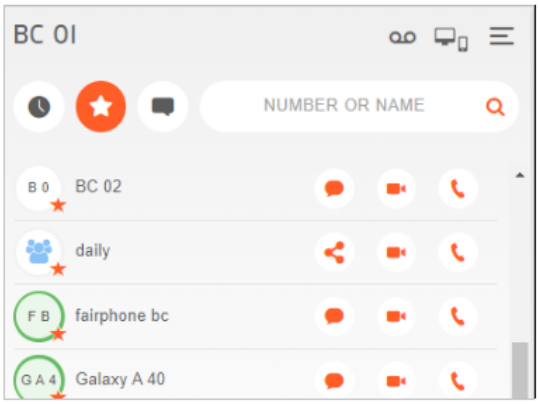

SN
Samsung note 9...
Rechercher un nom
APPEL SORTANT
Galaxy A8
1601
APPEL SORTANT
Galaxy A 40
1600
Favoris
Téléphone
Historique


Bouygues Telecom
100 %
11:36
Lun. 30 mai
XiVO
Appel manqué
Un appel manqué de BC 01
Chargé
Appel d'urgence


11:35
Lundi 30 mai
Appel manqué
maintenant
Un appel manqué de BC 01

Choice of call receiving devices
Configuration of an internal or external number (sda) for call forwarding to another line
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
1601
Renvoyer les appels sur non reponse
Destination
1601
FERMER
Activation of "do not disturb" mode
Activation of the "Forward all calls" mode
Activation of the mode " Forward calls on no answer ".





When selecting the mobile application and the web browser, you will see a mobile and a screen icon in the top right of the application.

When selecting the web browser, you will see a screen icon on the top right of the application.

When selecting the phone, you will see a phone icon on the top right of the application

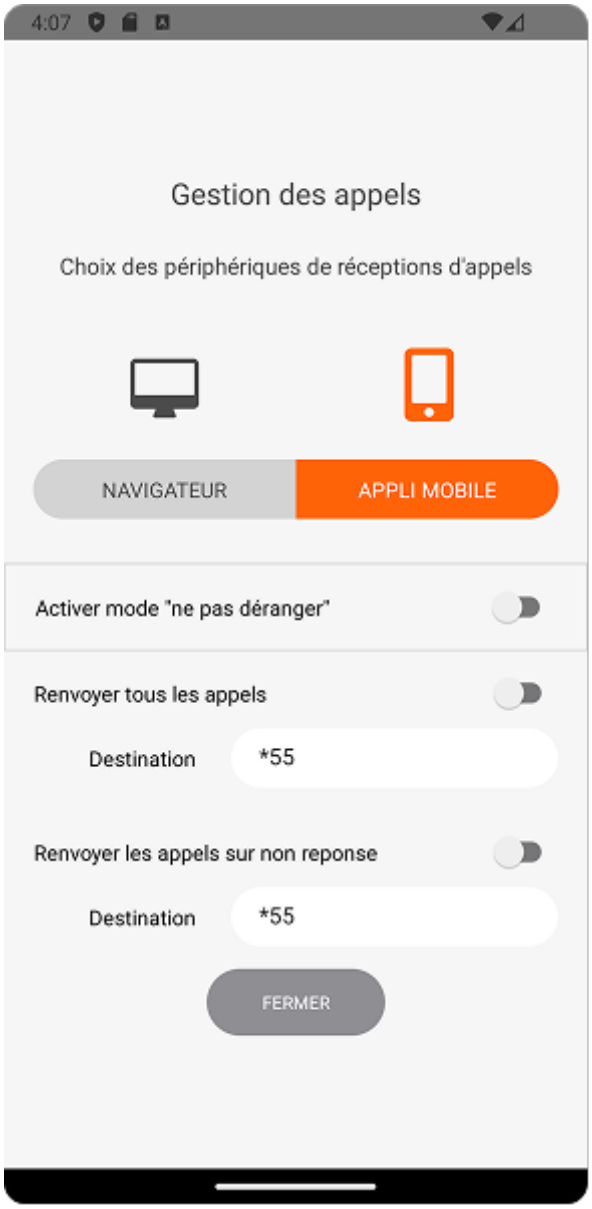
When selecting the phone and the web browser, you will see a phone and a screen icon on the top right of the application

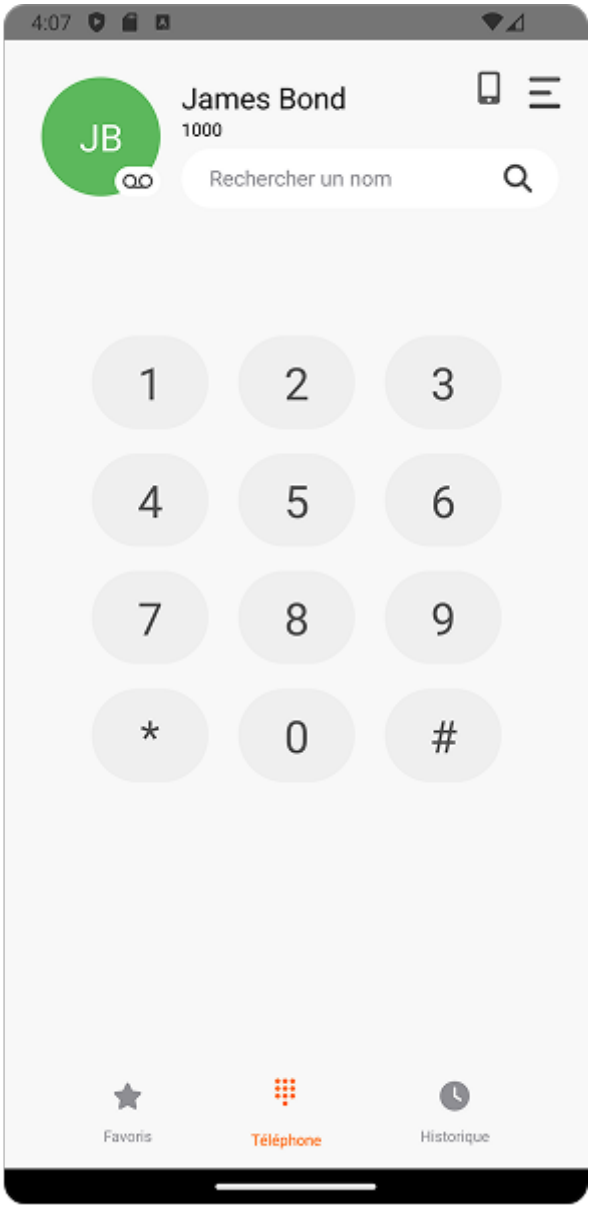
When selecting the phone and the mobile, you will see a phone and a mobile icon on the top right of the application

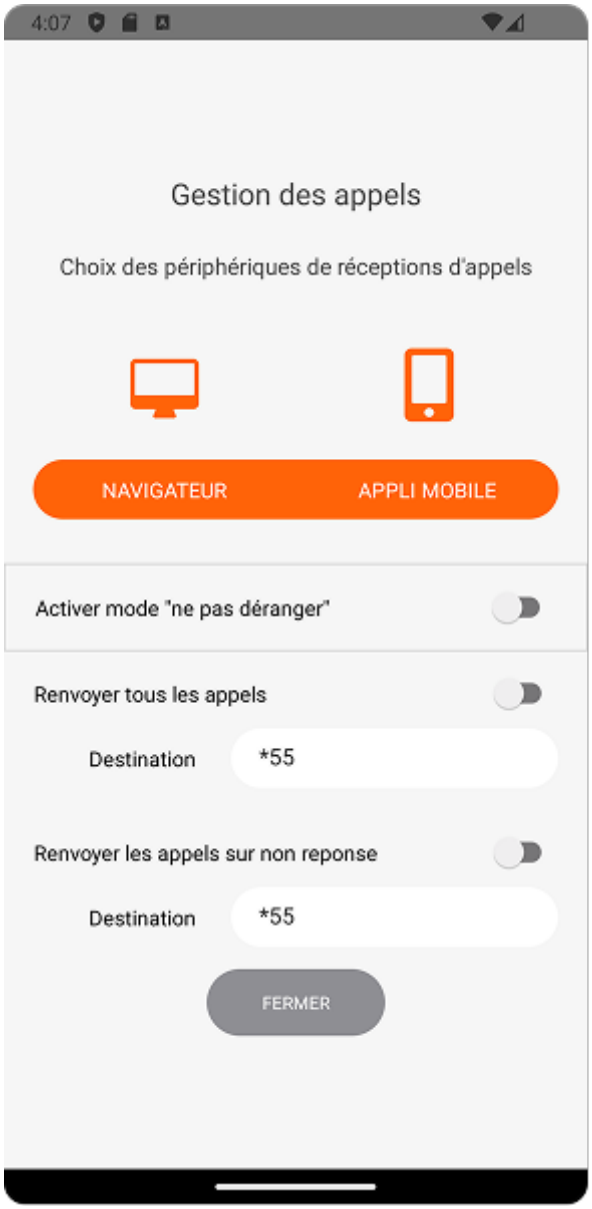
When selecting the phone and the mobile and the phone, you will see a phone, a mobile, and a screen icon on the top right of the application

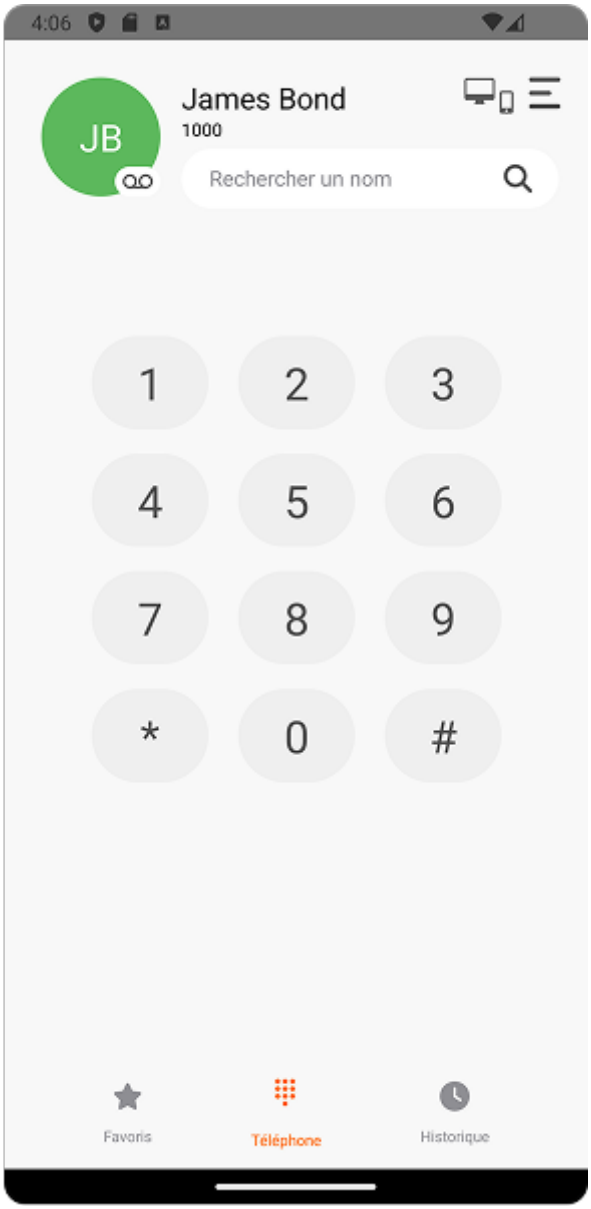
Do not disturb

When activating the “do not disturb” mode, you will see a “do not disturb” sign in the top right corner of the call management.

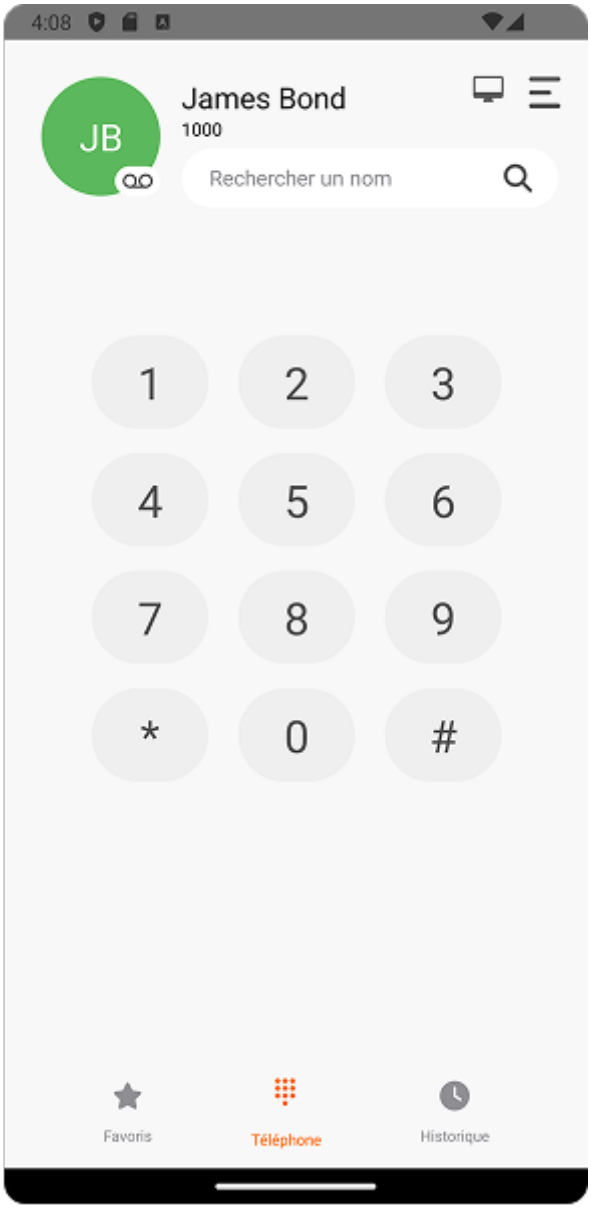

4:07
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*55
Renvoyer les appels sur non reponse
Destination
*55
FERMER

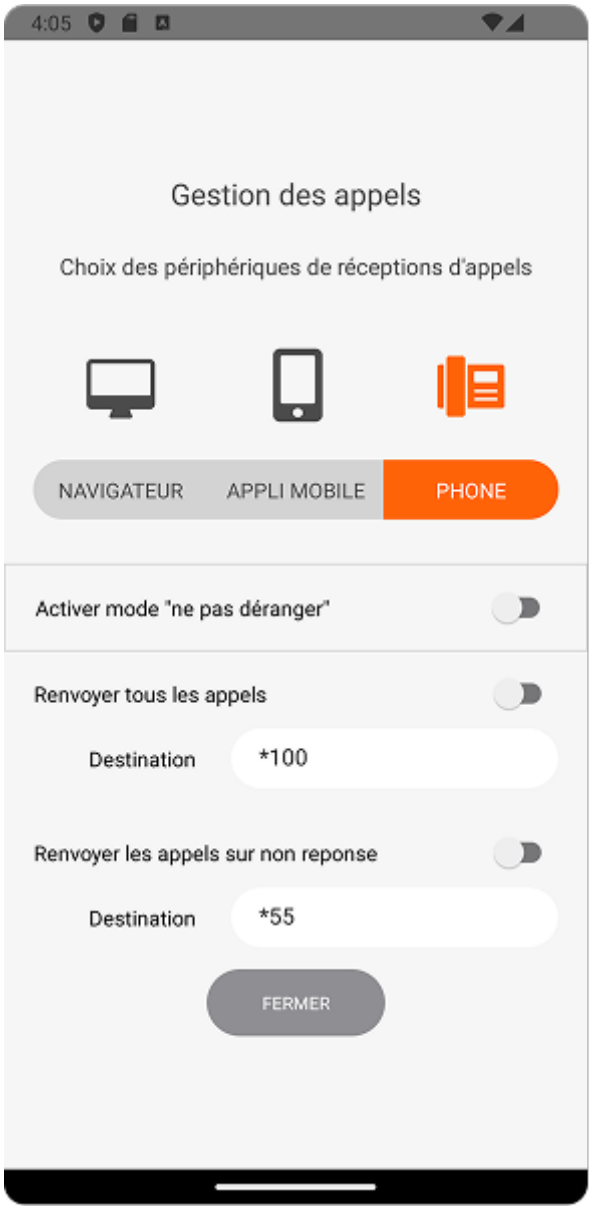

4:07
JB
James Bond
1000
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

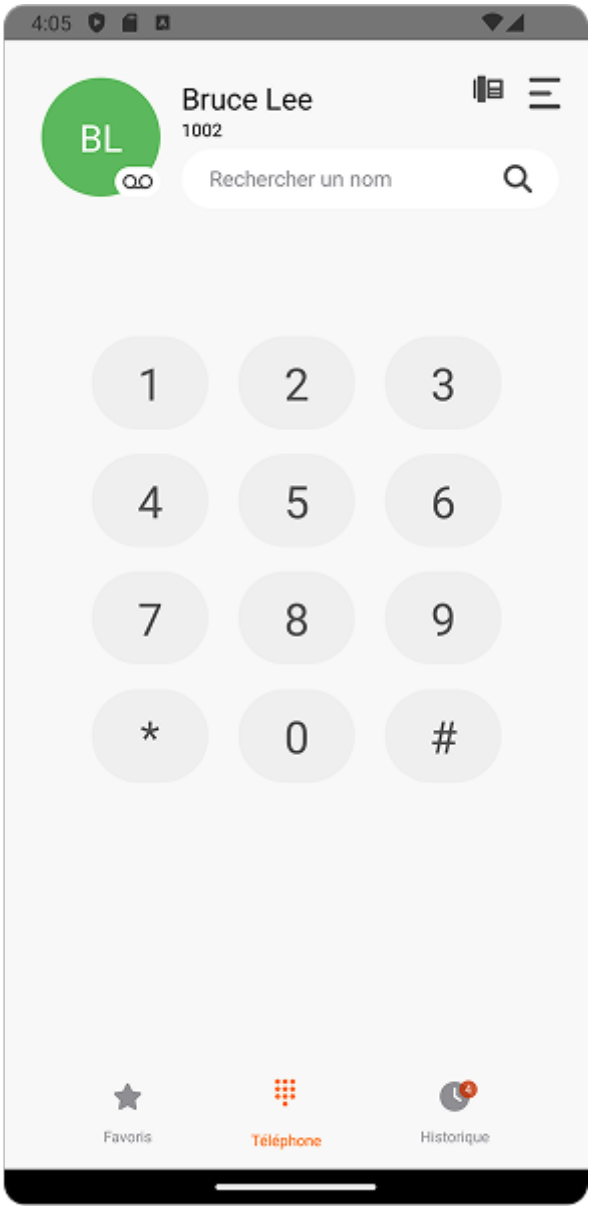

4:07
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*55
Renvoyer les appels sur non reponse
Destination
*55
FERMER

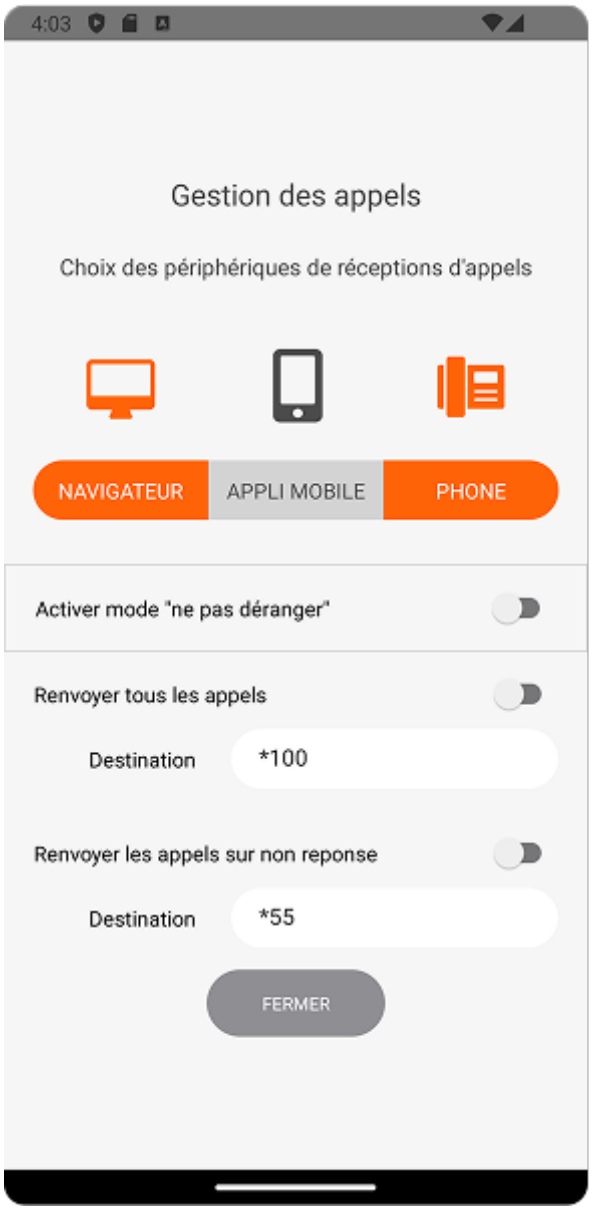

4:06
JB
James Bond
1000
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

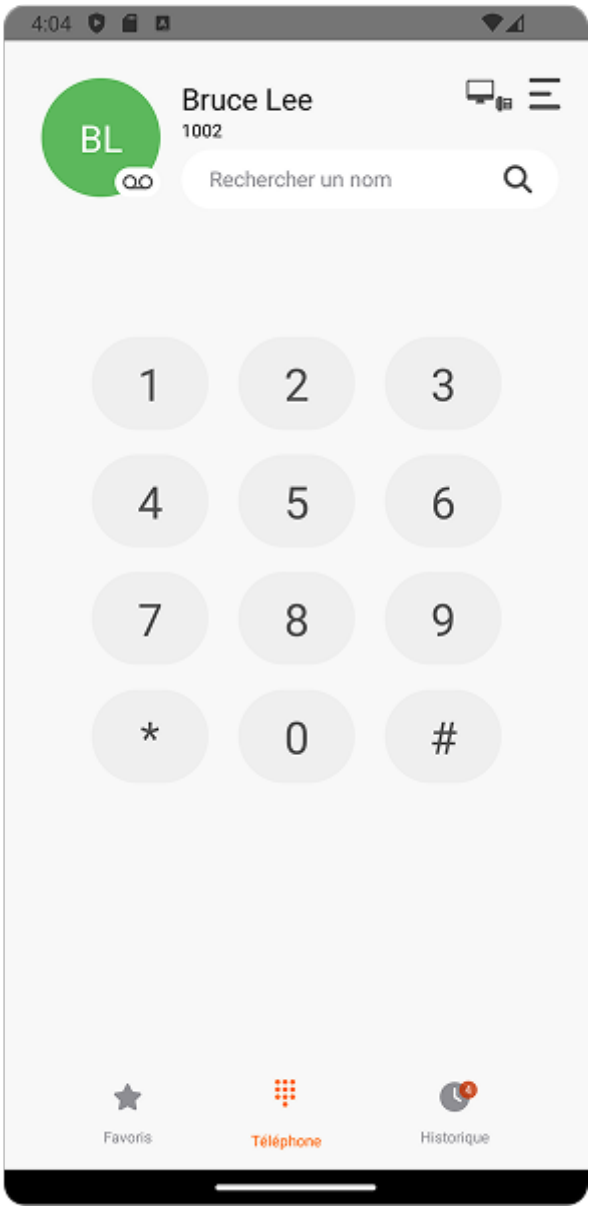
4:07
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*55
Renvoyer les appels sur non reponse
Destination
*55
FERMER

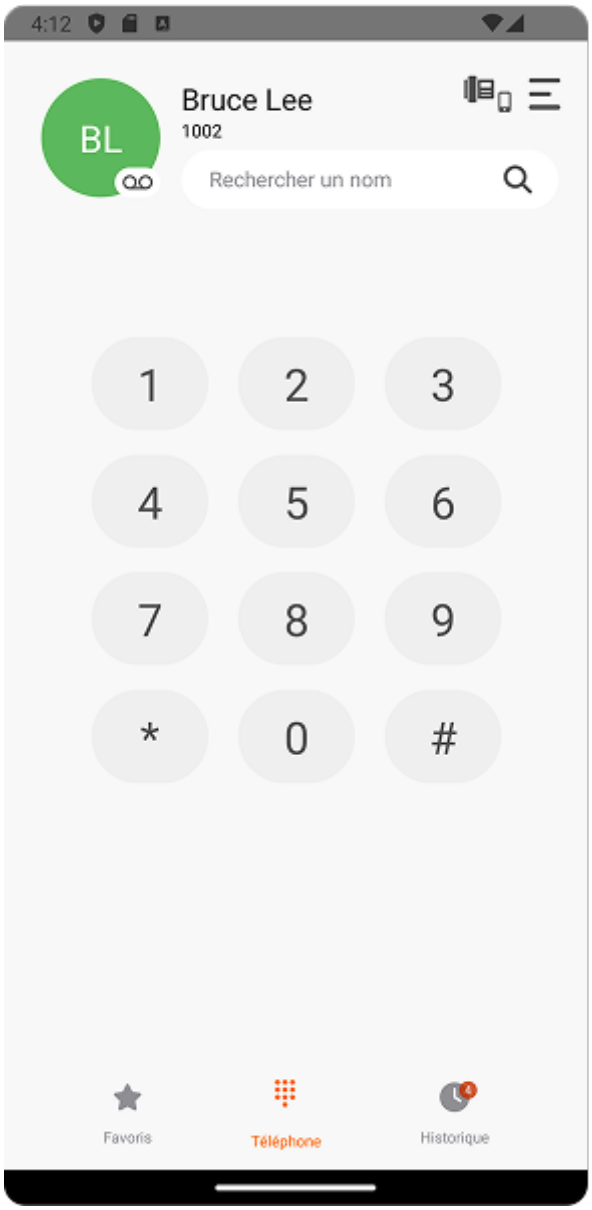

4:08
JB
James Bond
1000
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

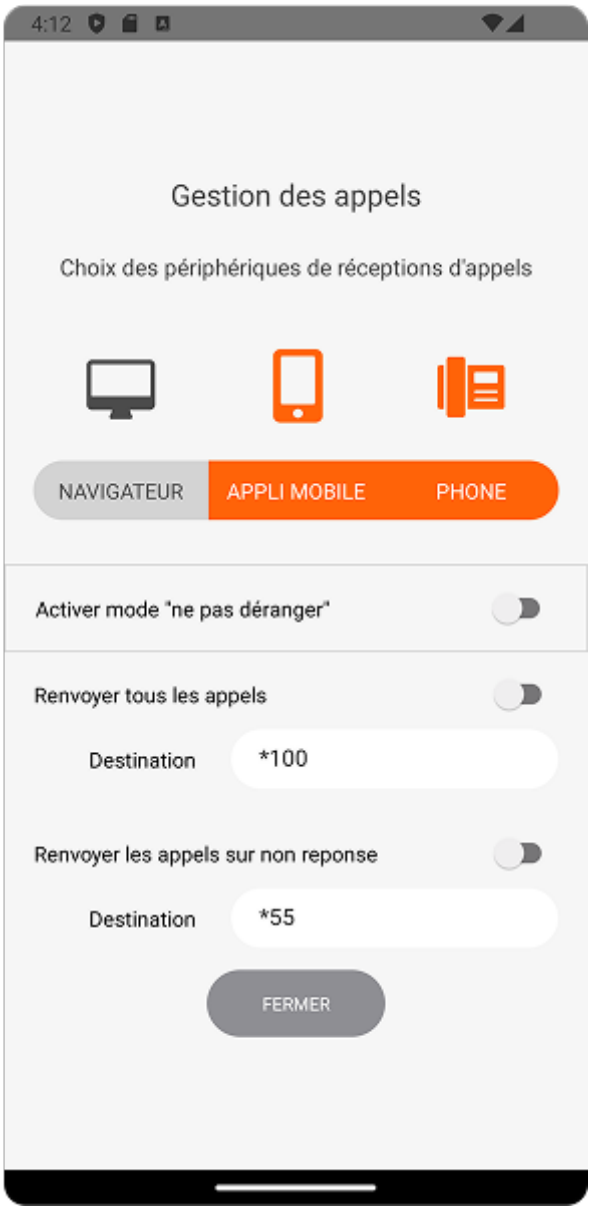

4:05
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
PHONE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*100
Renvoyer les appels sur non reponse
Destination
*55
FERMER

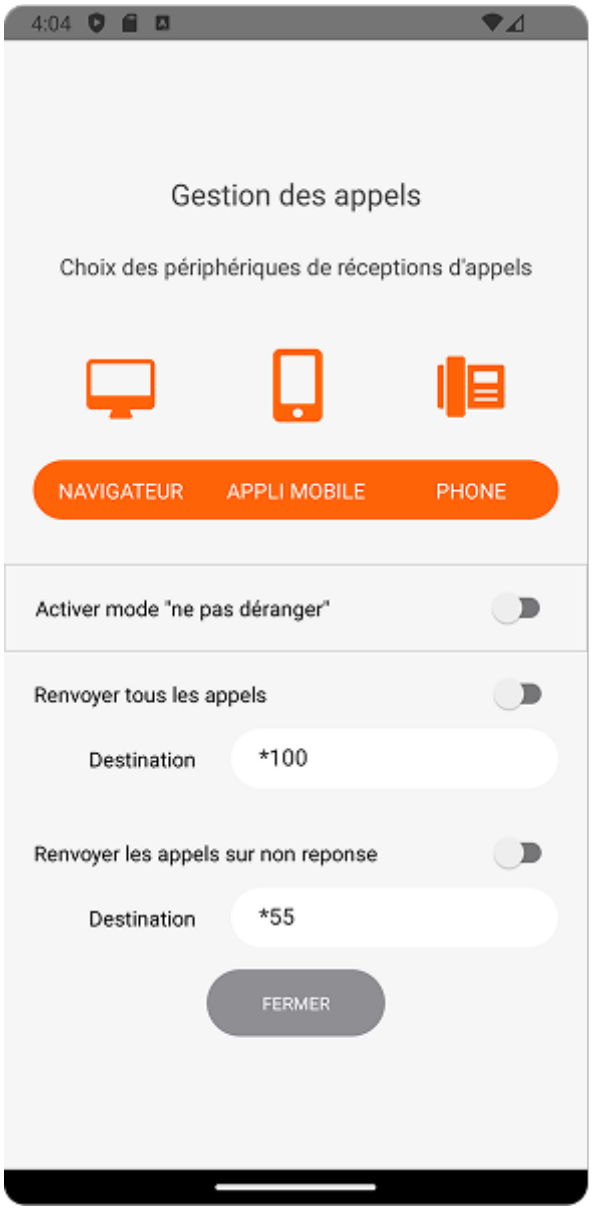

4:05
BL
Bruce Lee
1002
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

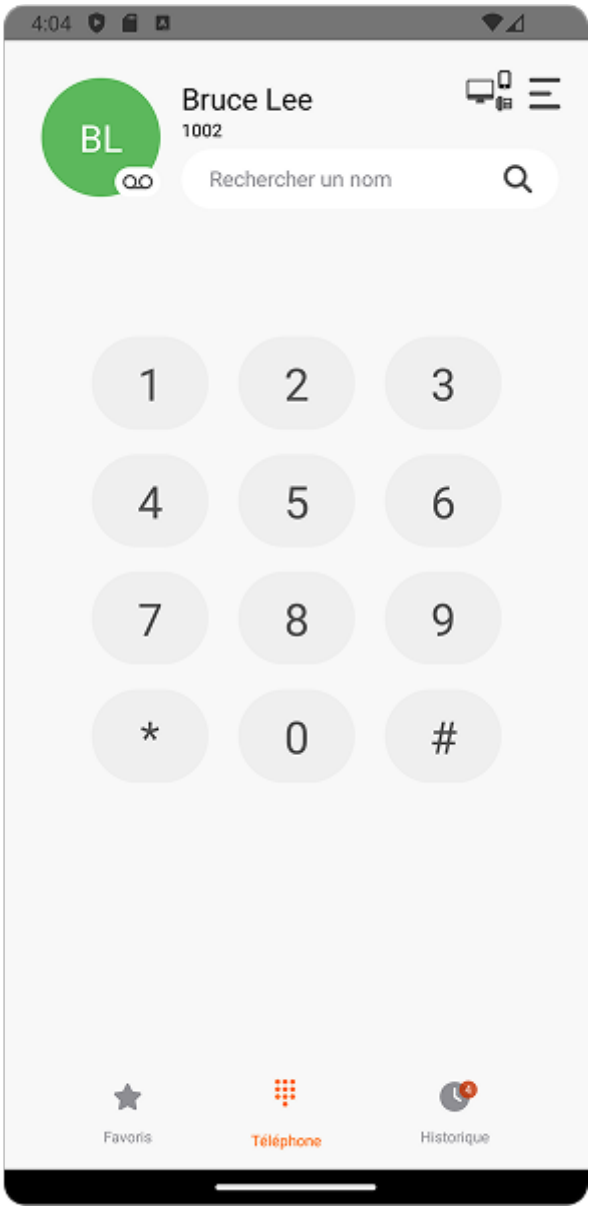

4:03
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
PHONE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*100
Renvoyer les appels sur non reponse
Destination
*55
FERMER


4:04
BL
Bruce Lee
1002
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

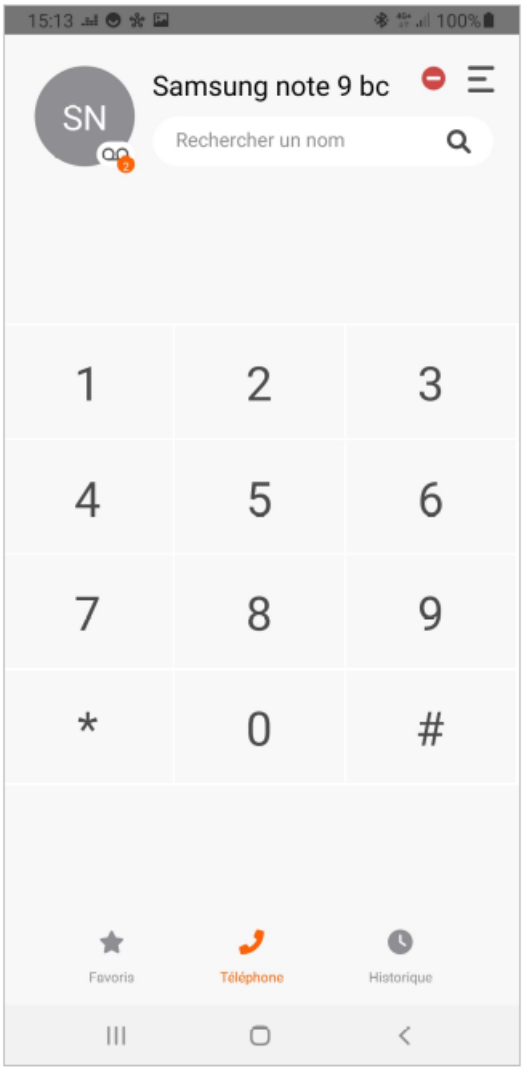

4:12
BL
Bruce Lee
1002
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique


4:12
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
PHONE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*100
Renvoyer les appels sur non reponse
Destination
*55
FERMER

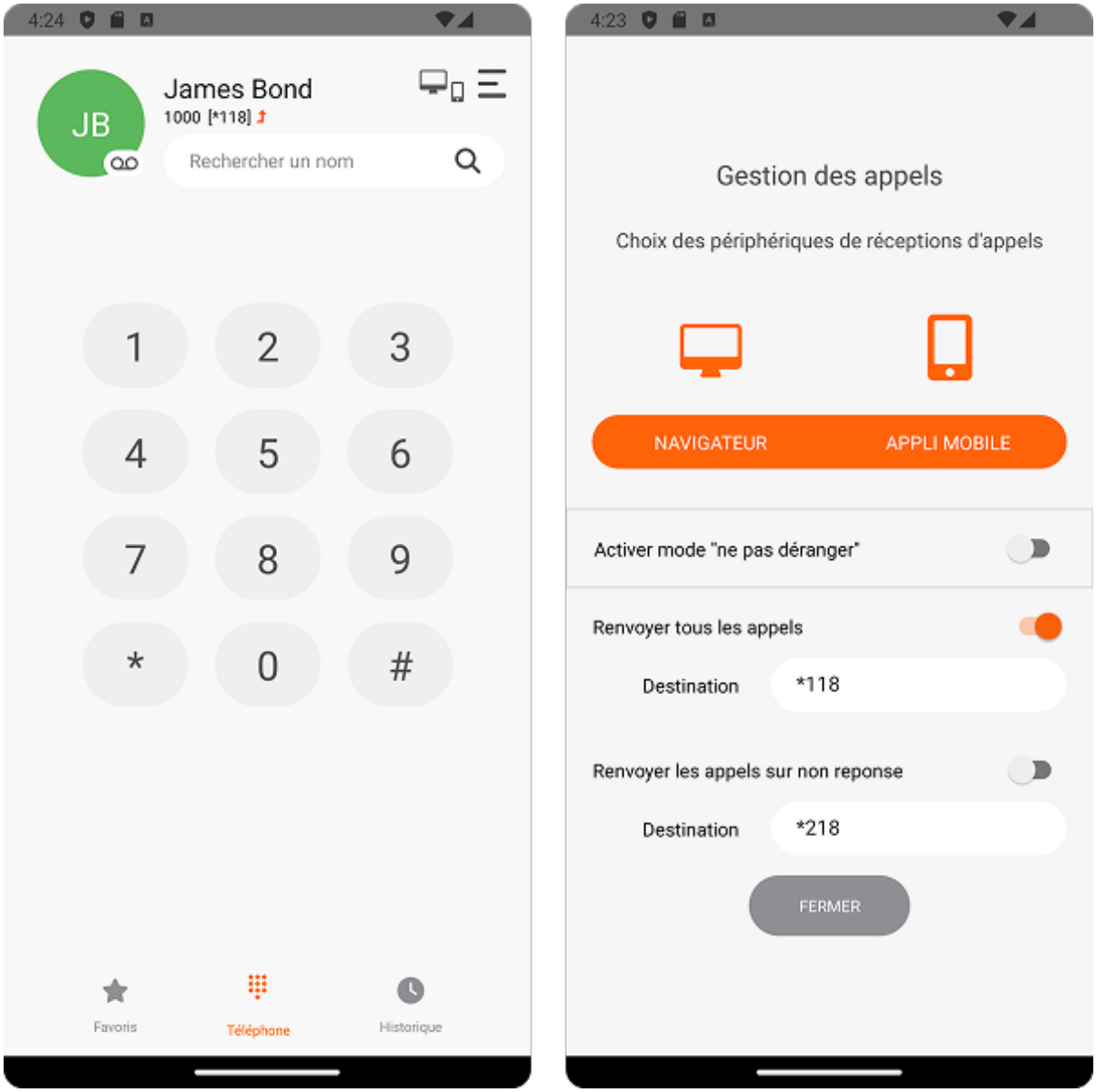

4:04
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
PHONE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*100
Renvoyer les appels sur non reponse
Destination
*55
FERMER


4:04
BL
Bruce Lee
1002
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

15:13
100%
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
1601
Renvoyer les appels sur non reponse
Destination
1601
FERMER


15:13
100%
SN
Samsung note 9 bc
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique

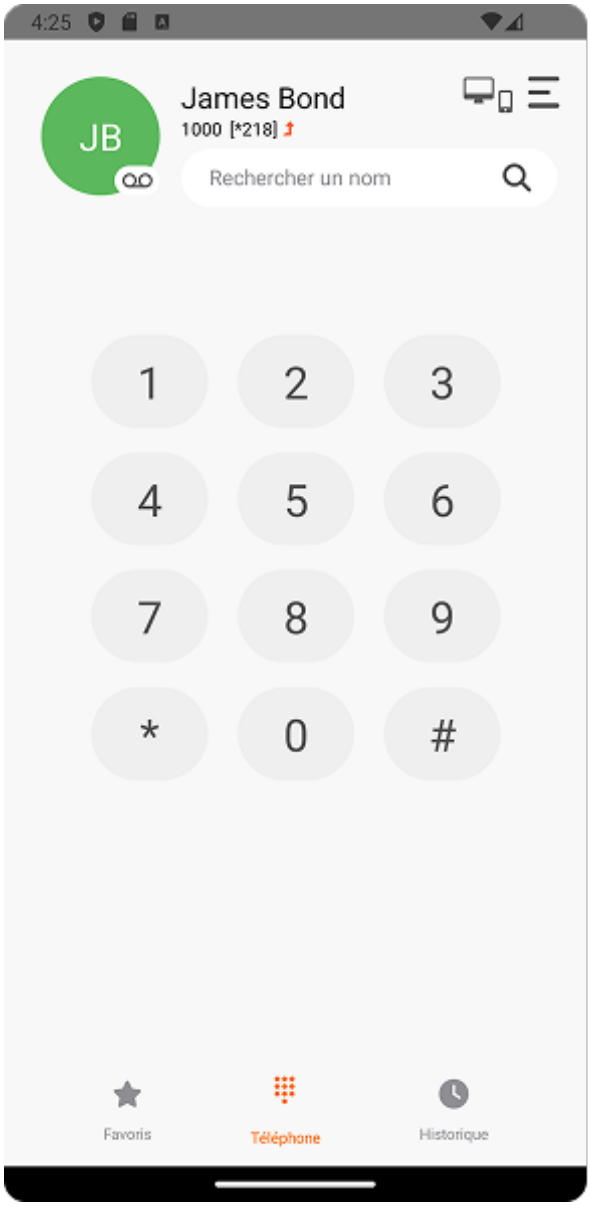
Forward all calls

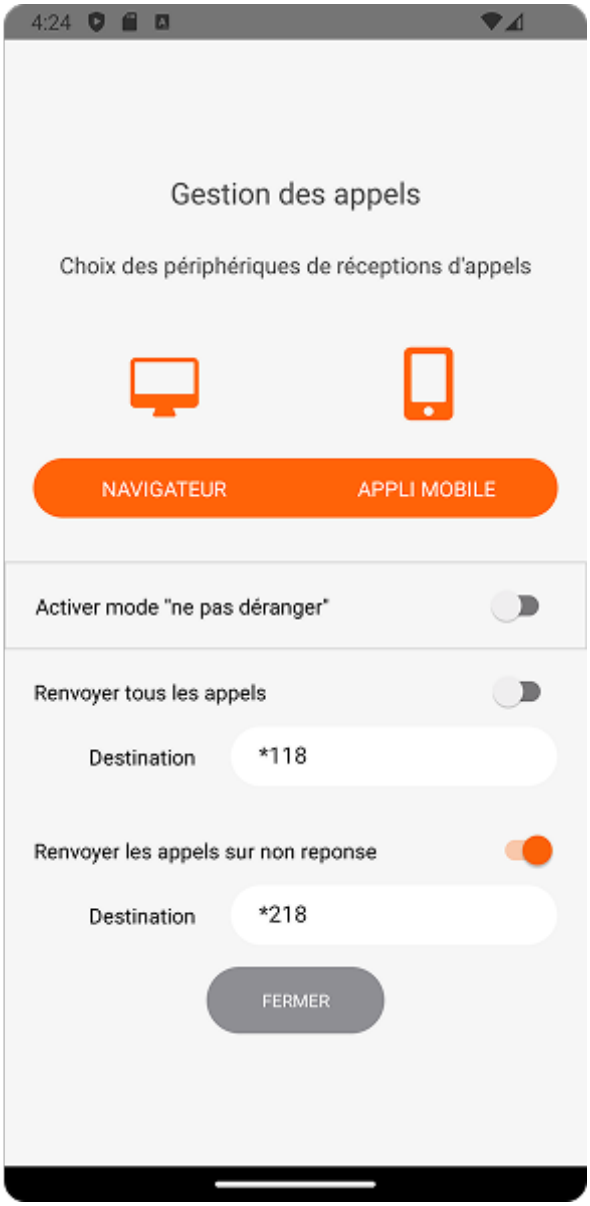


When activating the “forward all calls” you will see an arrow next to your internal number with the forward number where call will be forwarded.

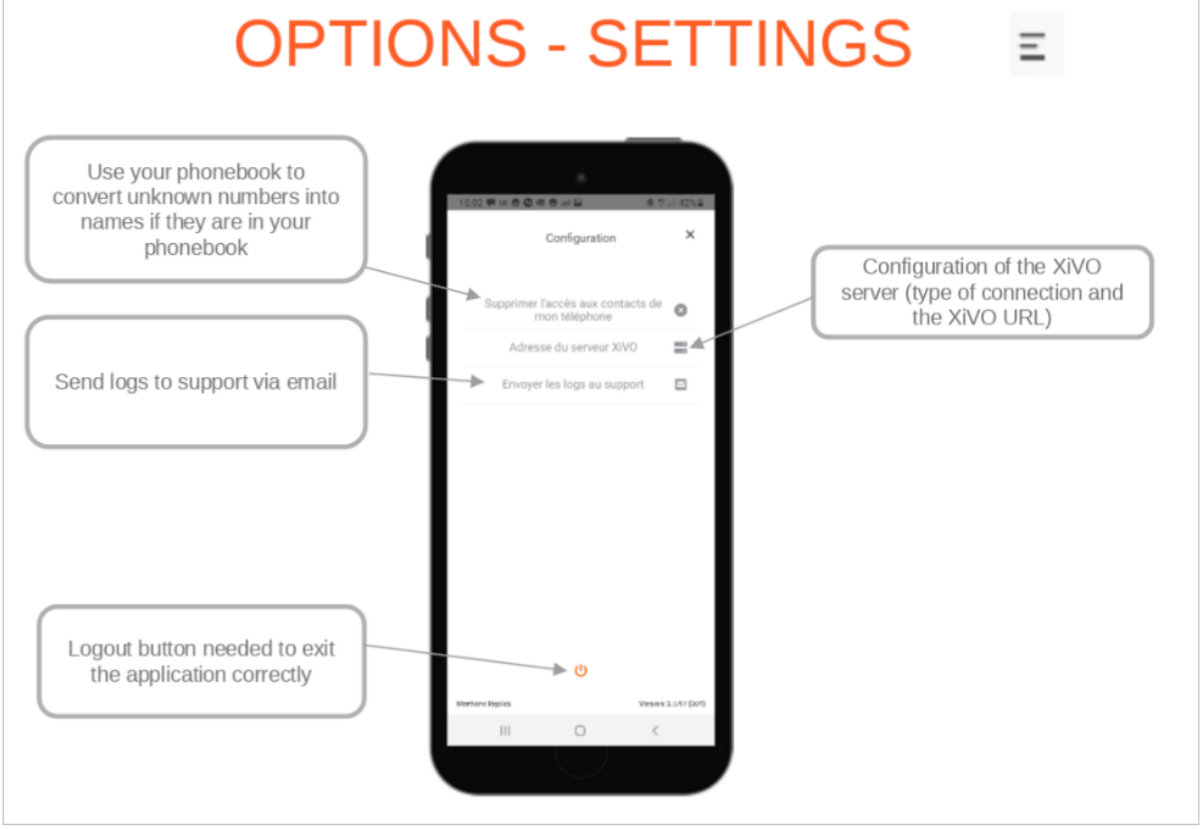
Forward calls on no answer

When activating the mode “forward calls on no answer”, you will see an arrow next to your internal number with the forward number where call will be forwarded.


4:25
JB
James Bond
1000 [*218]
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Favoris
Téléphone
Historique


4:24
Gestion des appels
Choix des périphériques de réceptions d'appels
NAVIGATEUR
APPLI MOBILE
Activer mode "ne pas déranger"
Renvoyer tous les appels
Destination
*118
Renvoyer les appels sur non reponse
Destination
*218
FERMER

5.4.14 Configuration



You can access the configuration by pressing the button on the top right corner of the call management. In this menu you have access to the following actions: -Remove access to my phone contacts -Integration of your forwarding number -Xivo server address -Activation of call forwarding

5.4.15 Disconnecting / disabling application

You can disconnect and disable the app by pressing the button at the bottom center of the page. When you log out, you wont be able to receive calls on your smartphone anymore. In this case, only the UC Assistant -or your browser - will ring if you are connected to it.

CHAPTER

SIX

ADDONS PRO

Important: This section provides information about available addons. Please note that these addons are exclusive features of the enterprise version and are not included in the standard XiVO package or available for free use. If you're interested in accessing these addons, please contact XiVO team for more information on how to enable these enterprise-level features.

Besides these considerations, the project stay **Open Source** and is available here: https://gitlab.com/xivo.solutions/

6.1 XiVO PBX

Important: This section provides information about available addons for XiVO PBX. Please note that these addons are exclusive features of the enterprise version and are not included in the standard XiVO package or available for free use. If you're interested in accessing these addons, please contact XiVO team for more information on how to enable these enterprise-level features.

6.1.1 IVR Editor

IVR is a new feature introduced in the Helios LTS.

Important: This is an enterprise version feature that is not included in XiVO and is not freely available. To enable it, please contact XiVO team.

- *Introduction*
- *IVR Installation & Upgrade*
- *Features*
- *Users' Guide*
  - *IVR management*
  - *Graphical IVR*
  - *Node types*
    - *Start Node*
    - *Update Node Name*
    - *Menu Node*

    * *User Node*
    * *Group Node*
    * *Queue Node*
    * *Voicemail Node*
    * *Playback Node*
    * *Hangup Node*
    * *Goto Node*
- *Use Case: Minimal Graphic IVR*
  - *Flowchart*
  - *Configure the Graphic IVR*
  - *Set Up Graphic IVR for External Dialing*
  - *Create a Schedule*
  - *Assign the Schedule to the Incoming Call Route*
- *Use Case: Graphic IVR with Submenu*
  - *Flowchart*
  - *Configure the Graphic IVR with a Submenu*

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 graphical editor allows users to design and configure parts of the dialplan through a visual interface. This approach makes it easier to build call flows by using drag-and-drop elements such as menus, call forwarding, and voicemail directly in the editor. By avoiding the need for manual configuration or scripting, the editor simplifies the creation and management of the dialplan. It also enables quick updates and testing of different scenarios, making it more accessible for administrators to implement changes and maintain the interactive voice response (IVR) system.

IVR Installation & Upgrade

Requirements

Important: Please contact support support@xivo.solutions to get access to the IVR Editor components

The IVR Editor component must be installed on a XiVO server.

Base installation

1. Docker override file Installation

These commands will install docker compose on the host.

```
# Download docker compose
TAG_OR_BRANCH=2026.05.00
cd /etc/docker/xivo
wget https://gitlab.com/xivo.solutions/ivr-editor/-/raw/${TAG_OR_BRANCH}/docker-xivo-
↪ivr.override.yml
```

2. IVR Editor configuration

1. Enable IVR in custom.env file

```
echo "IVR_INSTALLED=yes" >> /etc/docker/xivo/custom.env
```

Start the services

Start the services:

```
docker login -u xivoxc
(use the token provided by the XiVO team)

xivo-dcomp pull
xivo-dcomp up -d
```

Upgrade

Currently there is no *automatic upgrade* process. Here is the manual process that you need to follow on the XiVO host.

1. Make a backup of the IVR Editor compose override file:

```
cp -aR /etc/docker/xivo/docker-xivo-ivr.override.yml /var/tmp/docker-xivo-ivr.
↪override.yml
```

1. Re-download the ivr override compose file

```
# Download docker compose
TAG_OR_BRANCH=2026.05.00
cd /etc/docker/xivo
wget https://gitlab.com/xivo.solutions/ivr-editor/-/raw/${TAG_OR_BRANCH}/docker-xivo-
↪ivr.override.yml
```

1. Finally pull the new images and restart the containers:

```
xivo-dcomp pull
xivo-dcomp up -d
```

Features

- Create a edit part of dialplan (IVR) in graphical editor.
- Use created IVRs as an incall or a schedule destination
- Send call to queue or other defined dialplan object after IVR
- Upload and manage IVR specific audio files (voice prompts)

Users' Guide

IVR management

Navigate to *Services -> IPBX* in the Call Management section and select the *IVR*. The page will display a list of existing IVRs. Each IVR entry includes buttons that allow you to edit the dialplan metadata, modify the dialplan, or delete the dialplan.

Graphical IVR

The grafical IVR consists of nodes that represent dialplan actions and the connections between them. Each node type includes an entry point (a large circle at the top of the node rectangle) that can accept multiple connection lines, except for the Start node, which does not have an entry point. Depending on the type of node, there may be zero or more exit points (small circles at the bottom of the node rectangle). The specific exit point used is determined at runtime based on the outcome of the node's action.

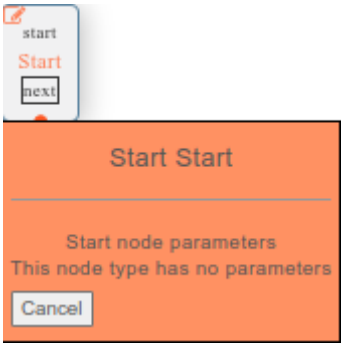
In the editor, you can:

- Add a node to the editor pane by selecting the desired node type from the left menu.
- Delete a node by clicking the red cross on the node.
- Configure node parameters by clicking the pencil icon in the top-left corner of the node and filling out the form.
- Connect nodes by dragging the exit point of one node to the entry point of another (or the same) node.
- Remove a connection line by clicking the rectangle with the minus sign on the line.

Node types

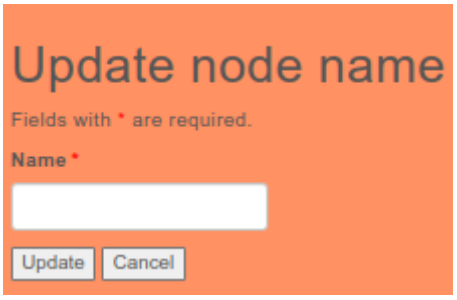
Start Node

- **Function**: Marks the beginning of the IVR flow. It is created automatically when an IVR is set up and cannot be manually added.
- **Parameters**: This node type does not require any configuration.
- **Screenshot**:



Update Node Name

- **Function**: Allows you to change the name of a node by double-clicking on the text (unset or orange text).
- **Parameters**:
  - *Name*: The name to be assigned to the node.
- **Screenshot**:

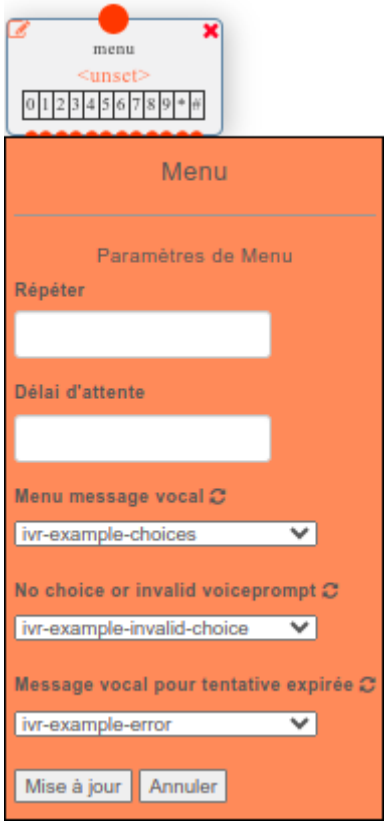


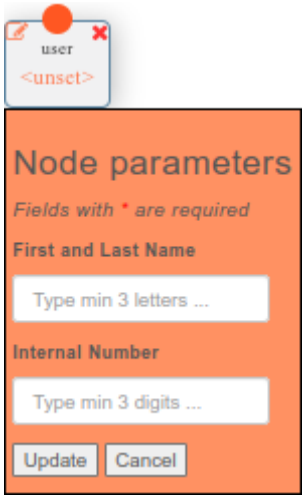
Menu Node

- **Function**: Presents a menu choice to the caller. Each key (0-9, *, #) can be configured to direct the call to different nodes or actions.
- **Parameters**:
  - *Repeat*: The number of times the message will be repeated if no response is given.
  - *Timeout*: The maximum duration to wait before considering that no response was given.
  - *Menu voiceprompt*: The audio file used for the menu announcement. (e.g., `ivr-example-choices`).
  - *No choice or invalid voiceprompt*: The message played if no choice is made or the choice is invalid. (e.g., `ivr-example-invalid-choice`).
  - *Outdated attempt voiceprompt*: The audio file played when all response attempts have failed. (e.g., `ivr-example-error`).
- **Screenshot**:

User Node

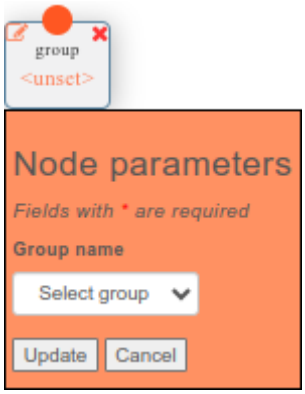
- **Function**: Allows you to direct the call to a specific user.
- **Parameters**:
  - *First and Last Name*: The full name of the user. You must enter at least three characters to trigger a dropdown list that helps with the selection. Once the name is selected, the **Internal Number** field is automatically filled.
  - *Internal Number*: The user's internal extension number. You must enter at least three digits to activate the dropdown suggestions. Once the number is chosen, the **First and Last Name** field is automatically completed.
- **Screenshot**:


menu
<unset>
0 1 2 3 4 5 6 7 8 9 * #
Menu
Paramètres de Menu
Répéter
Délai d'attente
Menu message vocal
ivr-example-choices
No choice or invalid voiceprompt
ivr-example-invalid-choice
Message vocal pour tentative expirée
ivr-example-error
Mise à jour
Annuler


user
<unset>
Node parameters
Fields with * are required
First and Last Name
Type min 3 letters ...
Internal Number
Type min 3 digits ...
Update
Cancel

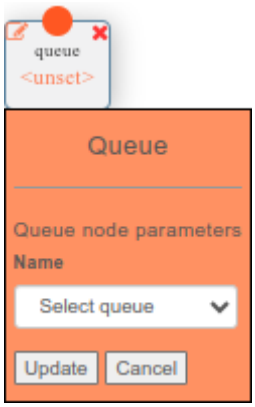
Group Node

- **Function**: Directs the call to a group of users.
- **Parameters**:
  - *Group name*: The name of the user group.
- **Screenshot**:



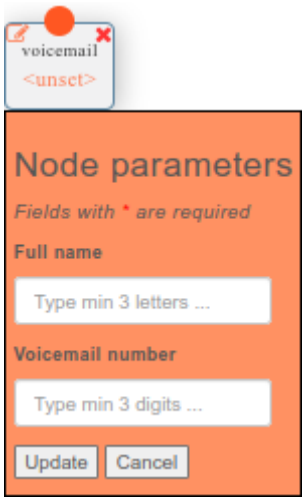
Queue Node

- **Function**: Sends the call to a specific queue, usually for support or a technical service.
- **Parameters**:
  - *Name*: Select the name of the queue.
- **Screenshot**:



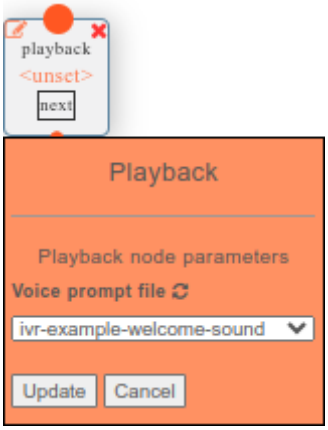
Voicemail Node

- **Function**: Allows you to transfer the call to a specific voicemail.
- **Parameters**:
  - *Full name*: The name of the voicemail recipient. Enter at least three characters to display a dropdown list of suggestions. Once the name is selected, the **Voicemail number** field is automatically filled.
  - *Voicemail number*: The number associated with the voicemail. You must enter at least three digits to activate the dropdown suggestions. Once the number is chosen, the **Full name** field is automatically completed.
- **Screenshot**:



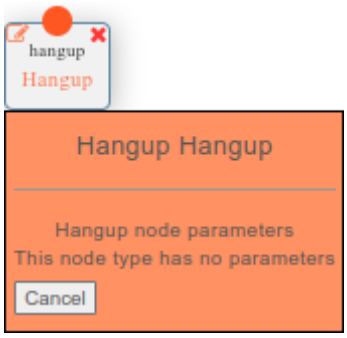
Playback Node

- **Function**: Plays an audio file for the caller. Typically used for welcome messages or instructions.
- **Parameters**:
  - *Voice prompt file*: Select the audio file to be played (e.g., `ivr-example-welcome-sound`).
- **Screenshot**:



Hangup Node

- **Function**: Ends the current call. Used to terminate an IVR flow.
- **Parameters**: No specific parameters need to be configured.
- **Screenshot**:



Goto Node

- **Function**: Allows you to redirect the call to another context or a specific extension.
- **Parameters**:
  - *Context*: The target context.
  - *Extension*: The extension to which the call should be directed.
- **Screenshot**:

Use Case: Minimal Graphic IVR

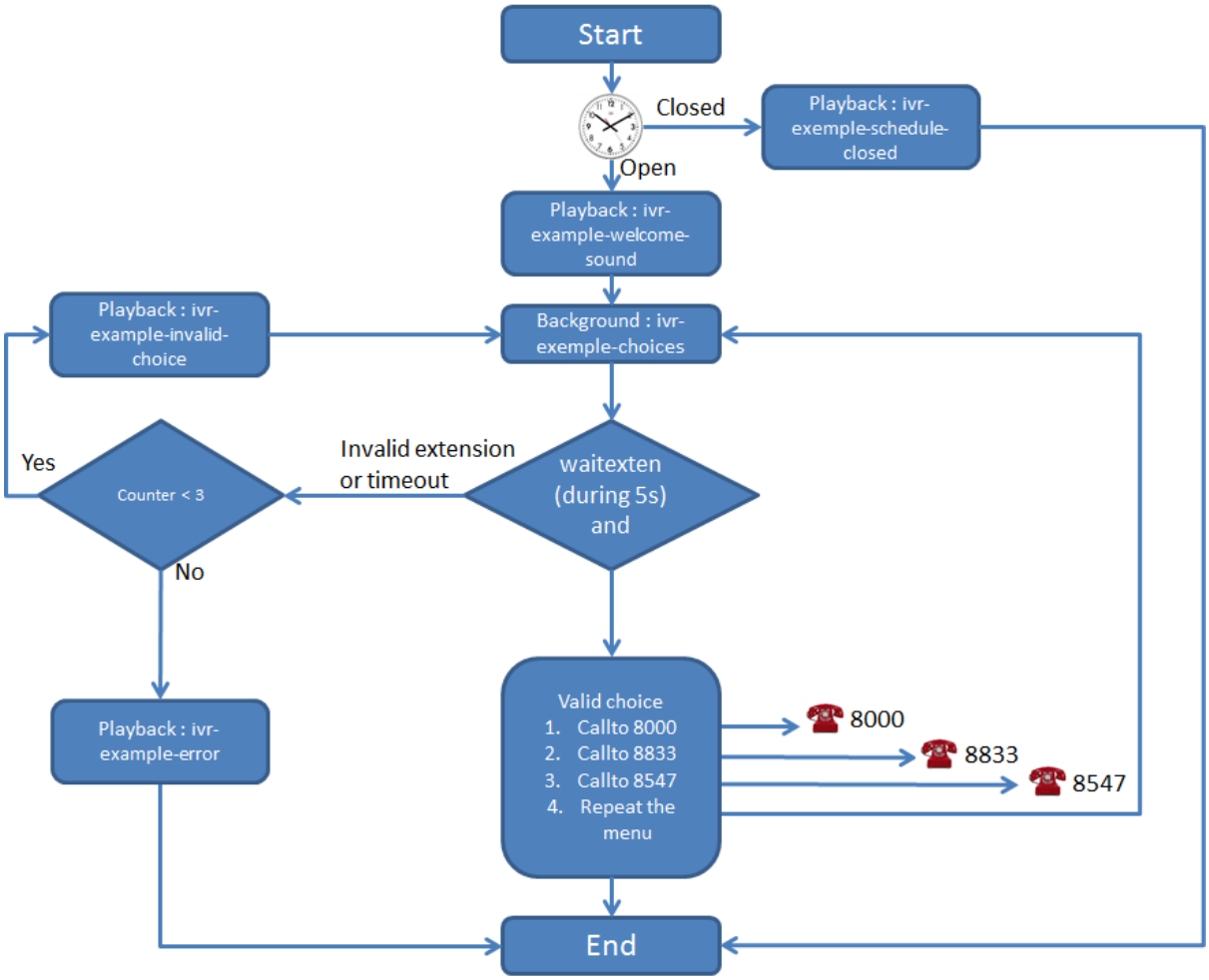
Configuration of a graphic IVR linked to a schedule and an incoming call.

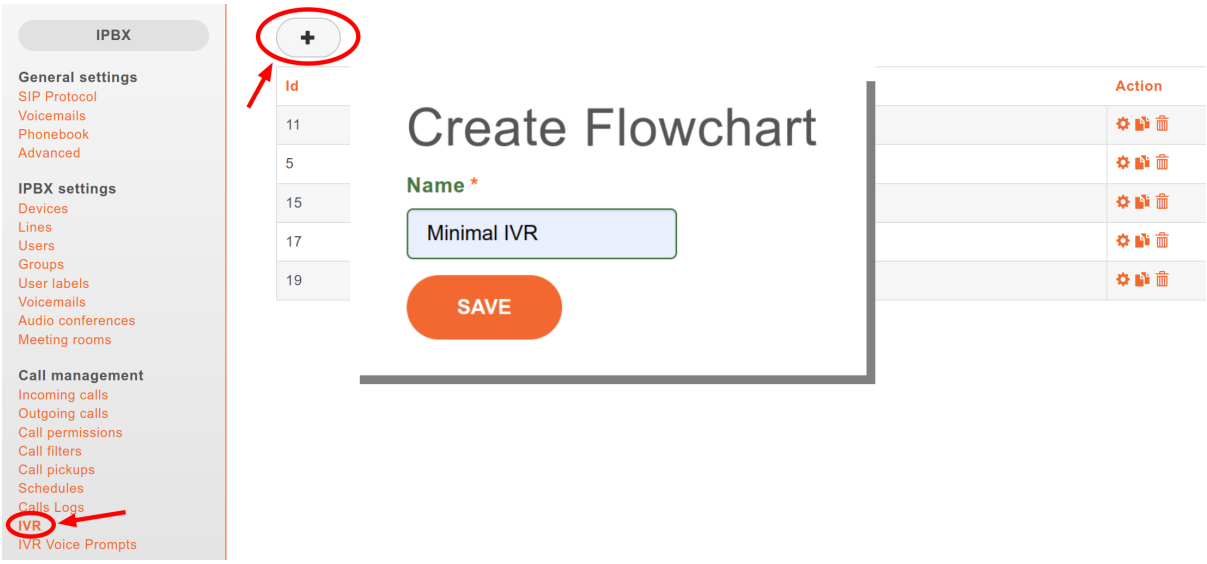
Flowchart

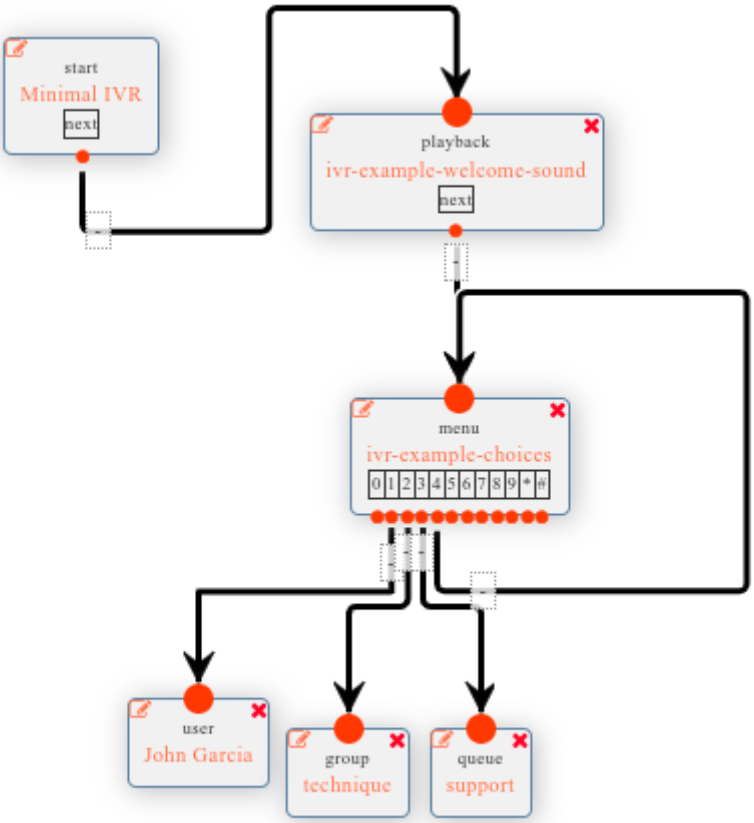
Configure the Graphic IVR

First, you need to create a graphic IVR. In our example, the graphic IVR is named `Minimal IVR`.

Once the IVR has been created, you can configure your graphic IVR.


Start
Closed
Open
Playback : ivr-exemple-schedule-closed
Playback : ivr-example-welcome-sound
Playback : ivr-example-invalid-choice
Background : ivr-exemple-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. Callto 8547
4. Repeat the menu
8000
8833
8547
End


IPBX
General settings
SIP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
User labels
Voicemails
Audio conferences
Meeting rooms
Call management
Incoming calls
Outgoing calls
Call permissions
Call filters
Call pickups
Schedules
Calls Logs
IVR
IVR Voice Prompts
Id
11
5
15
17
19
Create Flowchart
Name *
Minimal IVR
SAVE
Action



Set Up Graphic IVR for External Dialing

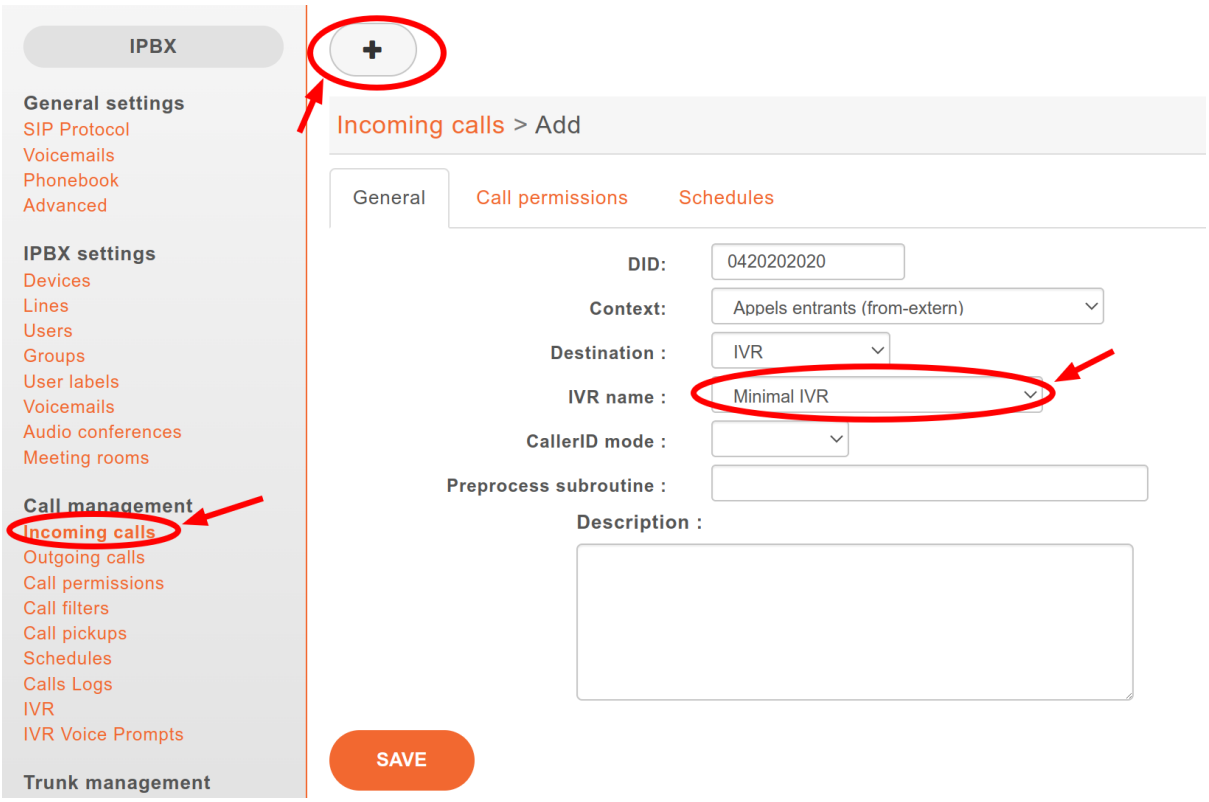
To allow calls to the graphic IVR from an external number, you need to create an incoming call route and redirect the call to the graphic IVR:

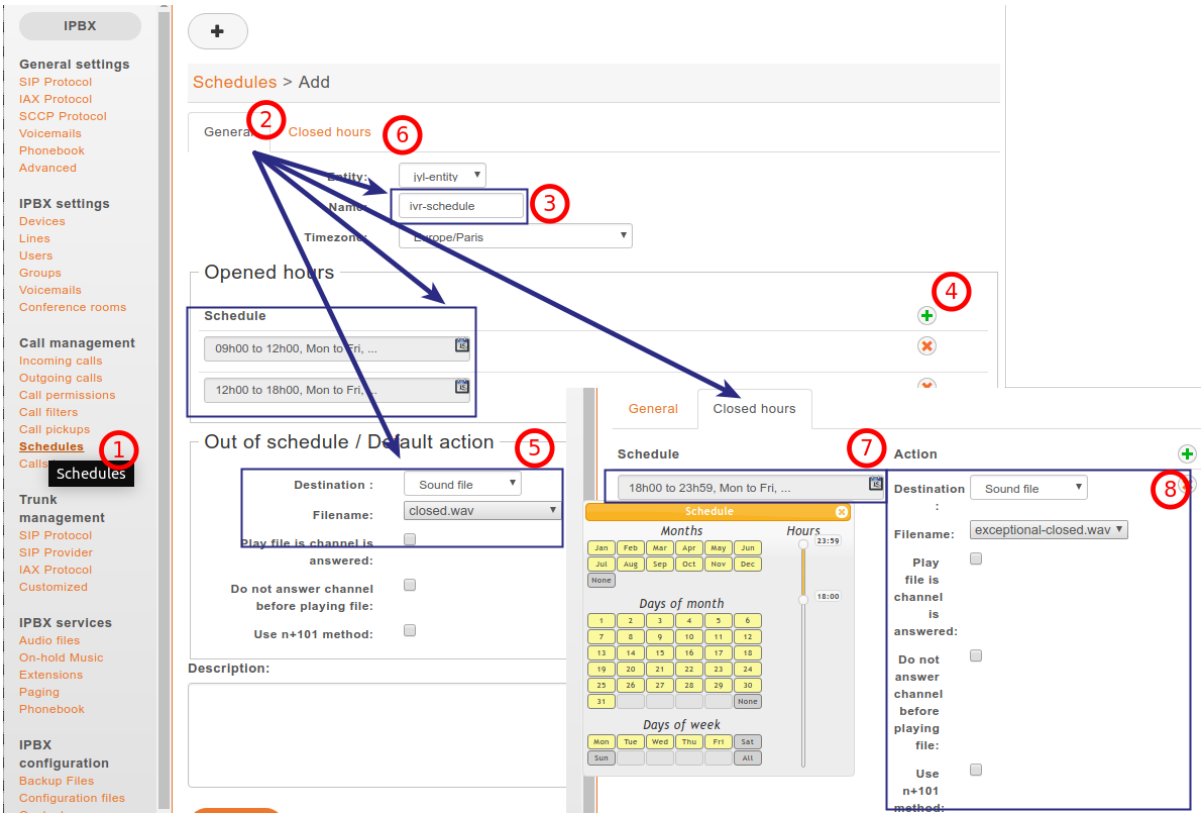
Create a Schedule

First, create your schedule (1) from the menu *Call Management* → *Schedules*. In the "General" tab, provide a name (3) for your schedule, configure the open hours (4), and select the audio file to be played when the company is closed.

In the "Closed Hours" tab (6), configure all special closed days (7) and select the audio file that will inform the caller that the company is exceptionally closed.

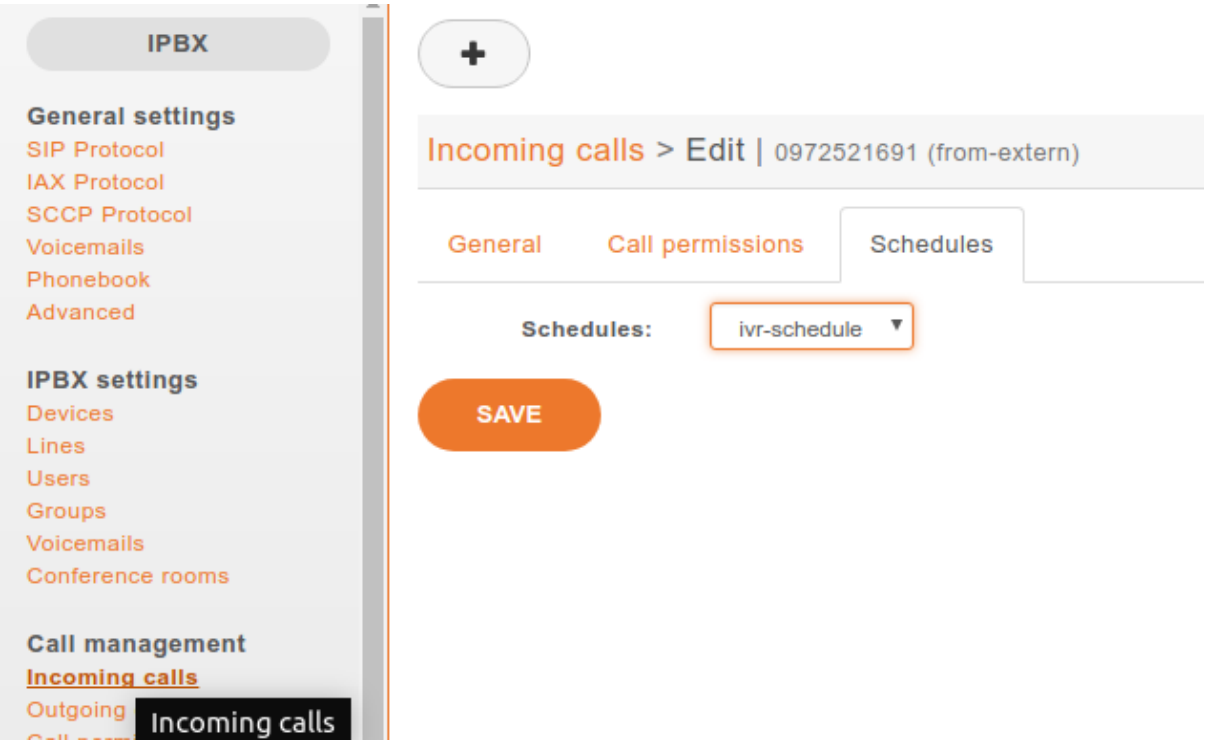
The IVR script will now only be available during working hours.


IPBX
General settings
SIP Protocol
Voicemails
Phonebook
Advanced
IPBX settings
Devices
Lines
Users
Groups
User labels
Voicemails
Audio conferences
Meeting rooms
Call management
Incoming calls
Outgoing calls
Call permissions
Call filters
Call pickups
Schedules
Calls Logs
IVR
IVR Voice Prompts
Trunk management
Incoming calls > Add
General
Call permissions
Schedules
DID:
0420202020
Context:
Appels entrants (from-extern)
Destination :
IVR
IVR name :
Minimal IVR
CallerID mode :
Preprocess subroutine :
Description :
SAVE


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
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 > Add
General
Closed hours
Entity:
iyl-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

Assign the Schedule to the Incoming Call Route

Return to editing your incoming call route (*Call Management* → *Incoming Calls*) and assign the newly created schedule in the "Schedules" tab.



Use Case: Graphic IVR with Submenu

Flowchart

Configure the Graphic IVR with a Submenu

Once the IVR has been created, you can configure your graphic IVR with a submenu.

6.1.2 Users Bidirectional Synchronization

Important: This is an enterprise version feature that is not included in XiVO and is not freely available. To enable it, please contact XiVO team.

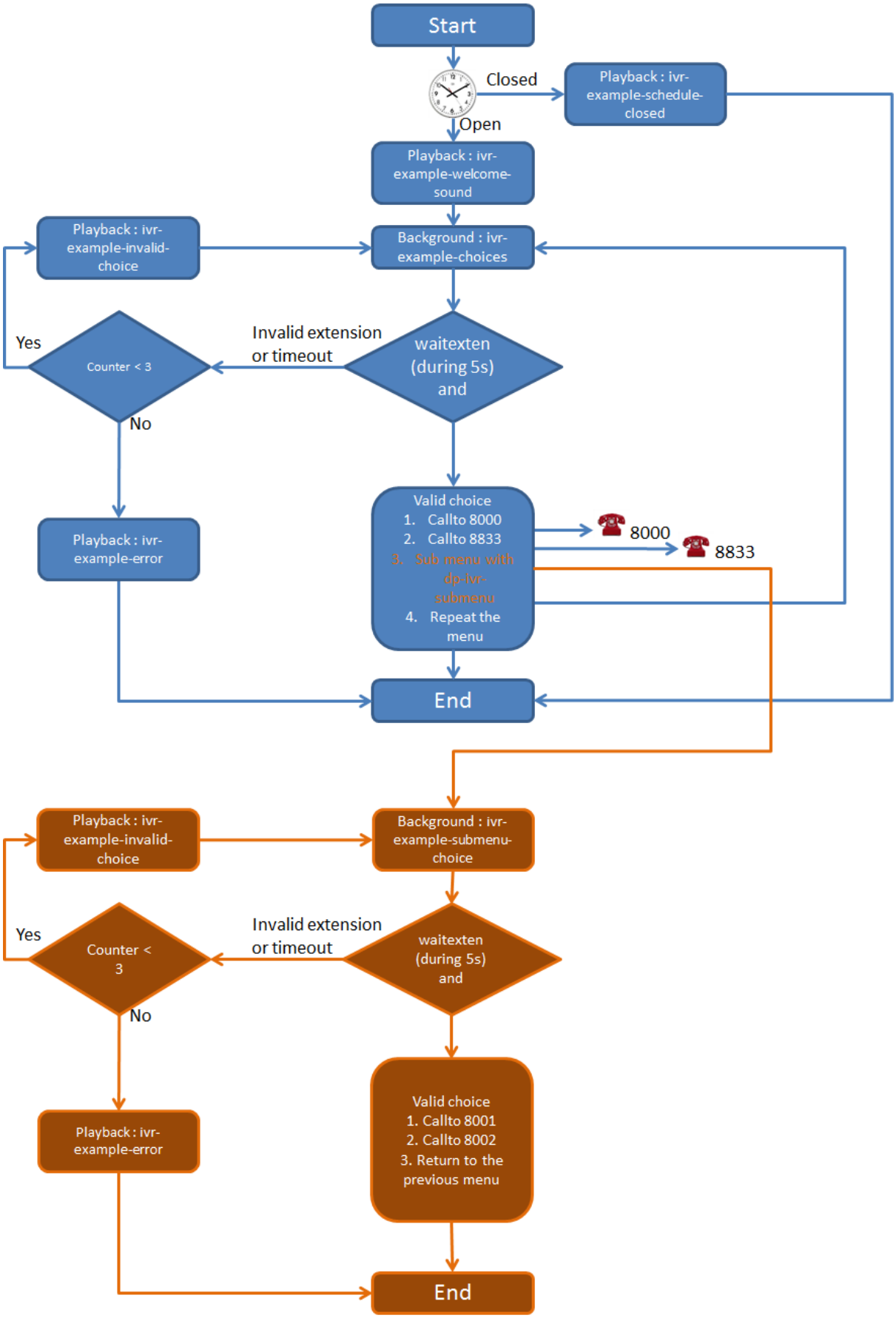
Overview

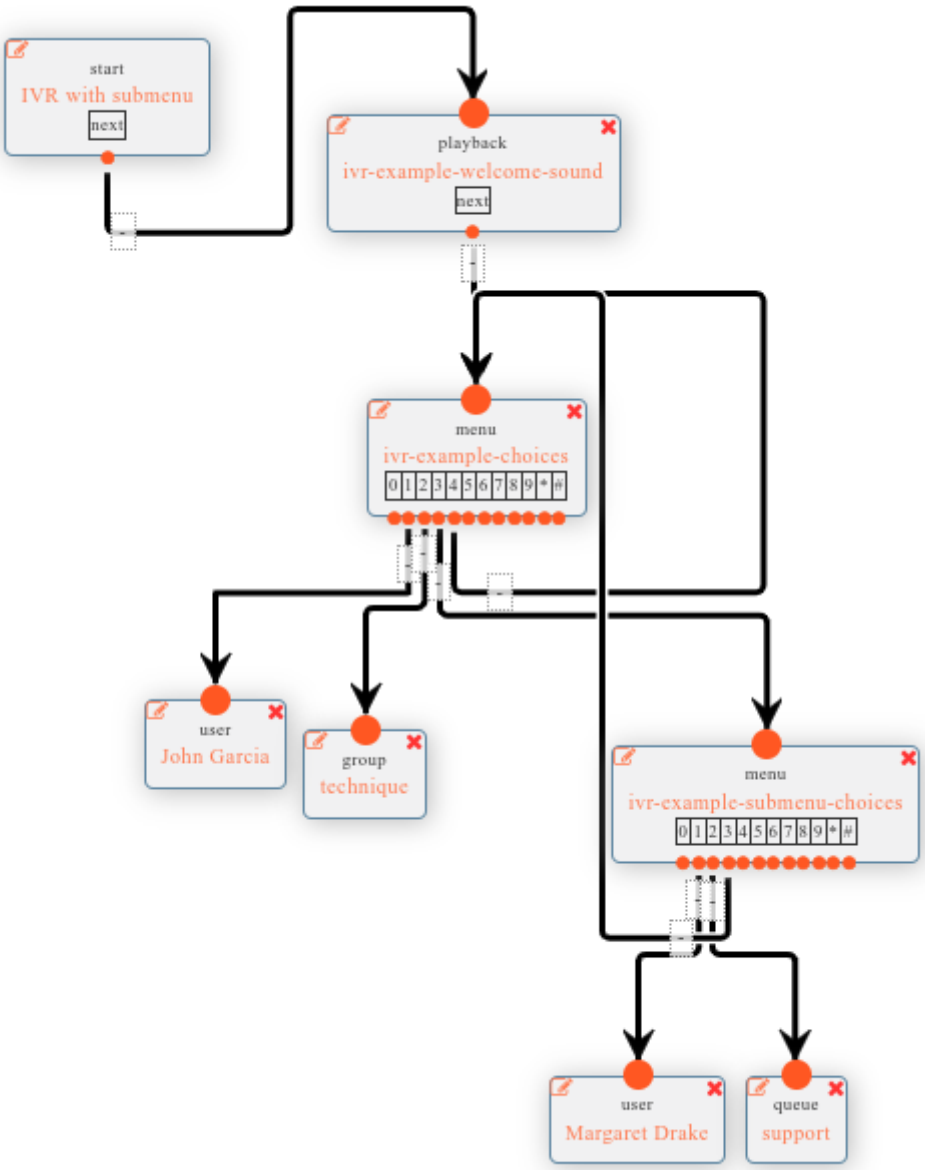
The LDAP Bidirectional Synchronization module enables automatic integration between an enterprise LDAP directory (such as Microsoft Active Directory or any compatible LDAP service) and XiVO.

Its purpose is to **simplify user administration**, **automatically provision telephony resources**, and **synchronize user information** based on data already maintained within the company's information system.

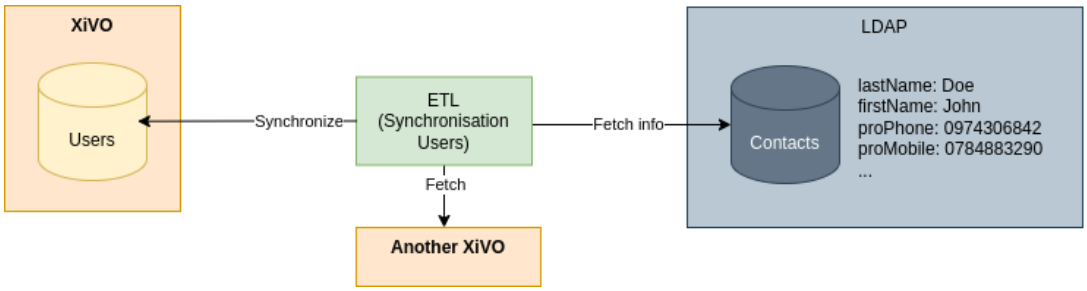
Through this mechanism, the LDAP directory becomes a trusted source used to create, update, and manage users in XiVO.

This module is particularly useful for organizations that want to:


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


start
IVR with submenu
next
playback
ivr-example-welcome-sound
next
menu
ivr-example-choices
0 1 2 3 4 5 6 7 8 9 * #
user
John Garcia
group
technique
menu
ivr-example-submenu-choices
0 1 2 3 4 5 6 7 8 9 * #
user
Margaret Drake
queue
support

- centralize user management
- avoid duplicate data entry between directory services and telephony systems
- automate telephony resource provisioning
- ensure consistency between identity management and telephony infrastructure
- streamline and industrialize user deployments



Key Features

The main objective of this module is to **automatically synchronize users and associated resources between an LDAP directory and XiVO**, ensuring consistent information while reducing manual administration.

This synchronization allows organizations to:

- automatically create XiVO users from LDAP accounts
- update user information when LDAP attributes change
- remove or deactivate users based on synchronization policies
- automate the assignment of telephony resources

Functional Description

The synchronization engine connects LDAP directory data with XiVO objects. The module supports:

- **automatic creation** of new XiVO users
- **automatic update** of existing users
- **removal or deactivation** of users according to defined synchronization rules

This ensures XiVO remains aligned with the enterprise directory without manual intervention. During synchronization, the module can automatically manage several XiVO resources, including:

- users
- internal numbers (line)
- Direct Inward Dialing numbers (DID)
- voicemail accounts
- call center agents

Depending on the LDAP attributes and configured mapping rules, XiVO can provision the necessary resources required for each user. The module supports multiple number formats, including:

- internal numbering
- national numbering formats
- international numbering
- **E.164 format**

This ensures numbering consistency regardless of how numbers are stored in LDAP or used within XiVO. Synchronization can allow telephony resources to be assigned directly from LDAP attributes, such as:

- internal extensions
- DID numbers
- information related to inbound routing

This enables deeper integration between telephony infrastructure and enterprise identity management. The module can also manage **Call Center agents**. Based on LDAP attributes, it becomes possible to:

- automatically create agents
- update their information
- integrate them into the call center configuration

This functionality is particularly useful for organizations operating support teams, customer service departments, or service desks. When a new user account is created, the module can automatically send a summary email containing useful information such as:

- login credentials
- assigned extension
- onboarding or usage information

This feature requires a properly configured **SMTP relay** on the client side.

Functional Benefits

The main benefit of the module is the elimination of repetitive manual tasks related to user creation and maintenance within XiVO. Administrators no longer need to re-enter information that already exists in the enterprise directory. By using LDAP as the reference directory, organizations ensure stronger consistency between:

- identity management systems
- telephony infrastructure
- user configuration

This reduces the risk of errors, omissions, or inconsistent information across systems. The onboarding process for new users becomes significantly faster thanks to automation:

- account creation
- telephony resource assignment
- synchronization of user information
- automatic delivery of login details

Users can become operational more quickly. The module is particularly valuable for organizations with:

- a large number of users
- multi-site environments
- industrialized deployment requirements
- integrated identity and telephony infrastructures

Use Cases

Enterprise user provisioning

In organizations where users are already managed in Active Directory, this module allows XiVO accounts to be created automatically without manual input.

Multi-site telephony deployments

LDAP attributes such as location, department, or role can be used to adapt XiVO provisioning according to the company's organizational structure.

Call Center workforce management

Call Center agents can be integrated directly from LDAP, simplifying their creation and lifecycle management.

Example synchronization flow

A typical synchronization scenario works as follows:

1. A user is created or modified in the LDAP directory.
2. The synchronization engine reads the configured LDAP attributes.
3. Data is analyzed and transformed according to mapping rules.
4. XiVO checks whether the user already exists.
5. If the user does not exist, XiVO creates the required resources:
   - user account
   - internal numbers
   - DID
   - voicemail
   - Call Center agent if applicable
6. If the user already exists, the allowed information is updated.
7. If notifications are enabled, an email is sent to the user.

```
[User created/modified in LDAP]
            |
            v
    [LDAP attribute retrieval]
            |
            v
      [Mapping / processing]
            |
            v
      [User lookup in XiVO]
         |             |
         | exists      | not found
         v             v
  [XiVO update]    [XiVO creation]
         |             |
         +------ +-----+
```

(continues on next page)

(continued from previous page)

```
                |
                v
        [Notification email]
                |
                v
         [User operational]
```

Pre-requisites

- Ensure proper configuration of LDAP server and XiVO for bidirectional sync
- Set up SMTP relay for user notification emails
- Have a webservice user. The permission of this user must be : confd.#
- Define which 14 information fields to include in the enriched contact sheet

Limitations

- The mobile phone number cannot be synchronized from Xivo to source.
- The callerid has to be mapped from a source field (it's not possible to concatenate the firstname and the lastname).
- The identifier field CANNOT be modified by a processing.
- A voicemail or a line already created can't be associated with a new or existing user. It must first be deleted by hand.
- If LDAP GUID attribute must be used as identifier_field, it must mapped to description XiVO field.
- Call permissions are not managed.
- When en incoming call is disassiociated from its user, the LDAP attribute is not updated: a 'None' value cannot be pushed towards LDAP.
- The outgoing_caller_id must be defined in field_mapping.yml even when it is not synchronised to avoid to overwrite with default as defined in User.py (because of LDAP update limitation against 'None' value).

6.2 XiVO UC

Important: This section provides information about available addons for UC app. Please note that these addons are exclusive features of the enterprise version and are not included in the standard XiVO package or available for free use. If you're interested in accessing these addons, please contact XiVO team for more information on how to enable these enterprise-level features.

6.2.1 SSO

Single Sign-On (SSO) automates the authentication process entirely, requiring no action from the user's end. For instance, technologies like Kerberos seamlessly authenticate the user by retrieving session information from Windows, eliminating the need for the user to enter their login credentials. SSO enhances user experience by providing seamless and secure access to various services and applications without repetitive logins.

- *Kerberos Authentication*
  - *Prerequisites*
  - *XiVOCC Configuration*
  - *Browser configuration*

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`).

6.2.2 Auth Delegation

Authentication delegation allows the user to be authenticated through an external source, but it does not eliminate the login step for the user. When using authentication delegation, the user's credentials are verified via an external system, ensuring a secure login process. However, the user is still required to initiate the login action.

- *Login Timeout Configuration*
- *LDAP Authentication*
  - *Note for LDAPs*
- *CAS Authentication*
  - *XiVOCC Configuration*
- *OpenID Connect (OIDC) Authentication*
  - *XiVOCC Configuration*
    - *Available env variables*
    - *Configuring XiVOCC*
    - *Optional env variables*

Login Timeout Configuration

When configuring external service as authentication provider, you may have trouble to login if the external system is slow. To overcome this issue, you can increase the default timeout (5000 milliseconds default) by setting the *LOGIN_TIMEOUT_MS* parameter in the `custom.env` of your xivocc installation. This value is the delay in milliseconds before aborting a login attempt.

```
LOGIN_TIMEOUT_MS=7000
```

LDAP Authentication

Configure LDAP authent for CCmanager, UC Assistant and CC Agent

1. Create a a configuration file named `xuc.conf` to add ldap configuration:

```
mkdir -p /etc/docker/xuc/
touch /etc/docker/xuc/xuc.conf
```

2. Edit file `/etc/docker/xuc/xuc.conf` with the following content:

```
include "application.conf"

authentication {
  ldap {
    managerDN = "uid=company,ou=people,dc=company,dc=com"
    managerPassword = "xxxxxxxxxx"
    url = "ldap://ldap.company.com:389"
    searchBase = "ou=people,dc=company,dc=com"
    userSearchFilter = "uid=%s"
```

(continues on next page)

(continued from previous page)

```
  }
}
```

Where

- `managerDN` is the LDAP user name of a user with read rights on the whole LDAP (at least the part which contains the users)
- `managerPassword` is the password for this user
- `url` is the URL to access the customer LDAP: [ldap|ldaps]://LDAP_HOST:LDAP_PORT (N.B. you MUST specify the port. Usually 389 for ldap and 636 for ldaps). Example:
  - to use LDAP `ldap://10.32.5.6:389`
  - to use LDAP s: `ldaps://ldap.corp.com:636` - see also *Note for LDAPs*
- `searchBase` is the LDAP OU where to look for users
- `userSearchFilter` is the LDAP filter to search for user by a given login (i.e. the filter to search by login)

3. Then customize the `docker-xivocc.yml` file to use this `xuc.conf` configuration by adding a 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
```

4. Finally recreate the container:

```
xivocc-dcomp stop xuc
xivocc-dcomp rm xuc
xivocc-dcomp up -d xuc
```

Note for LDAPs

As said above to connect to LDAP with a TLS/SSL connection you just need to put `ldaps` in the url of the ldap authentication configuration.

Then it should work out of the box **if the LDAP server uses a certificate signed by a known Certificate Authority of the xucserver JDK**.

If the certificate used by the customer LDAP server is not signed by a known Certificate Authority then you need to do the following:

Note: You're probably in this case if you get the following error in the xucserver log when someone is trying to log in:

```
play.api.UnexpectedException: Unexpected exception[
   AuthenticationDriverException: Unexpected authentication problem in LdapDriver␣
↪module,
   error: An error occurred while attempting to connect to server ldap.test.avencall.
↪com:636:IOException(
```

(continues on next page)

(continued from previous page)

```
      LDAPException(resultCode=91 (connect error), errorMessage='An error occurred␣
↪while attempting to establish a connection to server ldap.test.avencall.com/10.32.0.
↪1:636:
      SSLHandshakeException(PKIX path building failed: sun.security.provider.
↪certpath.SunCertPathBuilderException:
      unable to find valid certification path to requested target), ldapSDKVersion=6.
↪0.0, revision=524c20f3bbcc0d83fb56b9e136a2fd3a7f60437d'))]
```

1. The customer needs to give you the fullchain certificate file used by the LDAP server (in the following we will refer to it as `CUSTOMER_LDAP_CERT`),
2. Copy `CUSTOMER_LDAP_CERT` file to the xucserver host,
3. Then copy the `CUSTOMER_LDAP_CERT` file inside the xucserver container:

   ```
   docker cp PATH/TO/CUSTOMER_LDAP_CERT xivocc_xuc_1:/tmp/
   ```

4. Generate a Java Keystore file from this certificate:

   ```
   xivocc-dcomp exec xuc keytool -import -file /tmp/CUSTOMER_LDAP_CERT -alias␣
   ↪ldapcert -keystore /tmp/ldapCertTrustStore
   ```

   when prompt:

   - chose a password for the Java Keystore you're creating (you will need it afterwards and we will refer to it in the following as `LDAP_CERT_KSTORE_PWD`)
   - and answer *yes* to the question *Trust this certificate? [no]:*

5. Then on the xucserver host, create a folder to hold this truststore and copy the trusttore generated inside:

   ```
   mkdir /etc/docker/xuc-custom-truststore
   docker cp xivocc_xuc_1:/tmp/ldapCertTrustStore /etc/docker/xuc-custom-truststore/
   ```

6. Then customize the `docker-xivocc.yml` file to use this custom keystore by adding a volume and JDK option to specify the custom keystore location:

   > **Warning:** Take care to:
   >
   > - replace LDAP_CERT_KSTORE_PWD by the password you chose at step 4
   > - keep the correct Xms and Xmx values as configured on your system

   ```
   xuc:
     image: ...

     environment:
     - JAVA_OPTS=-Xms512m -Xmx2048m -Djavax.net.ssl.trustStore=/customSslTrustStore/
   ↪ldapCertTrustStore -Djavax.net.ssl.trustStorePassword=LDAP_CERT_KSTORE_PWD
     - CONFIG_FILE=/conf/xuc.conf

     volumes:
     - /etc/docker/xuc:/conf
     - /etc/docker/xuc-custom-truststore:/customSslTrustStore
   ```

7. Finally recreate the container:

```
xivocc-dcomp stop xuc
xivocc-dcomp rm xuc
xivocc-dcomp up -d xuc
```

CAS Authentication

To enable CAS authentication and sign delegation 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.
- The CAS authentication was **only** validated against CAS Server *``*Apereo Central Authentication Service*` in version `5.1.8`

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
```

OpenID Connect (OIDC) Authentication

To enable OIDC authentication and sign delegation feature, you need to have an existing OpenID Connect 1.0 infrastructure. You need to be able to create a realm and a client for the XiVOCC environment.

Warning:

- The OIDC authentication server must be accessible from the user and the server hosting the XiVOCC containers.
- OIDC Server users' username must match the XiVO username to allow login on the XiVOCC applications.
- We use Client Authentication with Keycloak (for more details see => https://openid.net/specs/openid-connect-core-1_0-15.html#ClientAuthentication)
- For Microsoft Entra Id configuration see => https://learn.microsoft.com/en-us/entra/fundamentals/whatis
- The OIDC authentication was **only** validated against OIDC Server `Keycloak` in version `10.0.2`.

XiVOCC Configuration

Available env variables

Here are the needed env variables to have the OIDC working :

ENABLE_OIDC

- It defines if the OIDC authentication is enabled
- If unset, the default value is false
- The syntax expect a simple boolean

OIDC_SERVER_URL

- It's the address where the UC Assistant redirect the user to be authenticated
- It challenges the `iss` field in the JWT token
- The syntax expect a single element

OIDC_CLIENT_ID

- It's the client id configured in the OIDC server
- It challenges the `azp` field in the JWT token
- It also challenges the `aud` field in the JWT token if `OIDC_AUDIENCE` is not configured
- The syntax accepts a single element

Configuring XiVOCC

- Edit the docker compose variables file `/etc/docker/compose/custom.env` and add these keys :

```
ENABLE_OIDC=true
OIDC_CLIENT_ID=myClientId
OIDC_SERVER_URL=https://oidc-server.example.org/auth/realms/myrealm
```

- Recreate and start the XiVOCC environment:

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

Optional env variables

Here are the additional variables for an advanced configuration :

OIDC_ADDITIONAL_SERVERS_URL optional

- It's an additional list of accepted token issuers that is allowed to provide token for authentication
- Useful if you want to login to the Xuc server with a token issued from a different OIDC server than OIDC_SERVER_URL
- It challenges the `iss` field in the JWT token
- If unset, only `OIDC_SERVER_URL` will challenge this token field, else, both will be used together
- The syntax accepts a single element, or a list of elements separated by comma

OIDC_ADDITIONAL_CLIENT_IDS optional

- It's an additional list of accepted client id
- Useful if you want to login to the Xuc server with a token issued from a different OIDC server than OIDC_SERVER_URL
- It challenges the `azp` field in the JWT token
- If unset, only `OIDC_CLIENT_ID` will challenge this token field, else, both will be used together
- The syntax accepts a single element, or a list of elements separated by comma

OIDC_AUDIENCE optional

- It's the list of accepted audiences
- It challenges the `aud` field in the JWT token
- If unset, it will be set to the `OIDC_CLIENT_ID` env variable value by default
- The syntax accepts a single element, or a list of elements separated by comma

OIDC_USERNAME_FIELD optional

- It's the field customizer for the XiVO username mapping
- It will decide the name of the field that will be considered as the XiVO username in the JWT token
- If unset, the XiVO username will be expected in the `preferred_username` token field
- The syntax expect a single element

OIDC_LOGOUT_ENABLE optional

- It defines if logging out of the XiVO application will also log the user out of the SSO
- If unset, the default value is `false`
- The syntax expect a simple boolean

6.2.3 Teams Presence Integration

Starting from release **Pollux**, users can now access **Microsoft Teams presence status** directly within the **UC Assistant** interface.

This new feature allows the UC Assistant to display Teams presence information such as "Available", "Busy", or "OutOfOffice" synchronized from Microsoft Teams. For now, presence is displayed **only during user searches or on the Favorites page**, and there is **no automatic refresh** of the presence status.

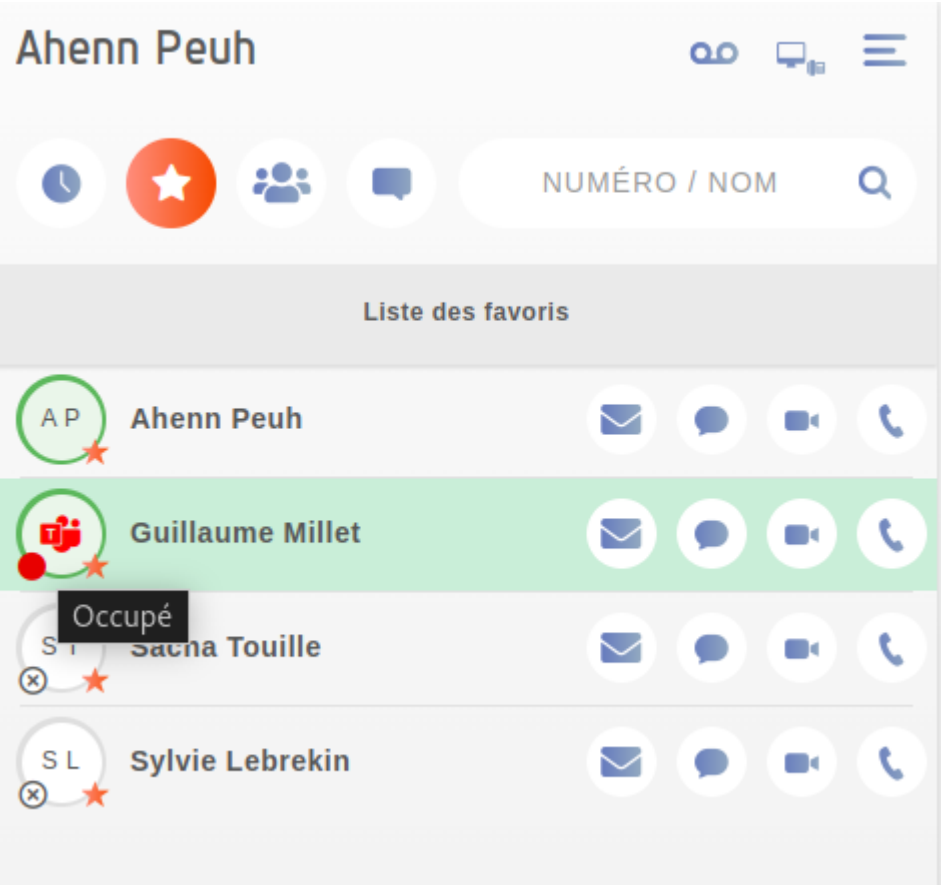
Configuration

To enable Teams presence in your environment, please contact our support team. A member of our technical staff will assist you with the configuration and ensure the integration is properly connected to your Teams tenant.

Examples

When you are in busy on teams:

This information is displayed on the UC Assistant



6.3 XiVO CC

Important: This section provides information about available addons for CC Agent. Please note that these addons are exclusive features of the enterprise version and are not included in the standard XiVO package or available for free use. If you're interested in accessing these addons, please contact XiVO team for more information on how to enable these enterprise-level features.

6.3.1 WebPanel Feature

The WebPanel feature provides dynamic, real-time visualization of call center activities, offering insights into agent performance and queue statistics. This powerful tool is designed to help managers and supervisors monitor and optimize call center operations efficiently.

Agent Activity Monitoring

The WebPanel offers a comprehensive view of agent activities, including:

- Answered calls per agent
- Calls picked up, hung up, and unanswered per agent
- Calls answered per queue
- Calls answered per agent group
- Total calls presented to agents

These metrics are updated in real-time, allowing for immediate assessment of agent performance and workload distribution.

Queue Performance Tracking

Monitor the dynamic evolution of queue activities with:

- Answered and abandoned calls across all queues
- General call volume trends (total, answered, and abandoned)
- Distribution of answered calls per queue

This information helps in identifying busy periods, optimizing staffing, and improving overall queue management.

Flexible Data Access

The WebPanel is designed for versatility, allowing access to critical data via:

- Desktop screens
- Tablet devices

This flexibility ensures that supervisors and managers can stay informed about call center performance, regardless of their location within the office.

Note: The WebPanel feature is part of our advanced suite of call center management tools. For more information on how to implement and utilize this feature, please contact XiVO team.

6.3.2 Call Qualifications

Important:

This is an enterprise version feature that is not included in XiVO and is not freely available.
To enable it, please contact XiVO team.

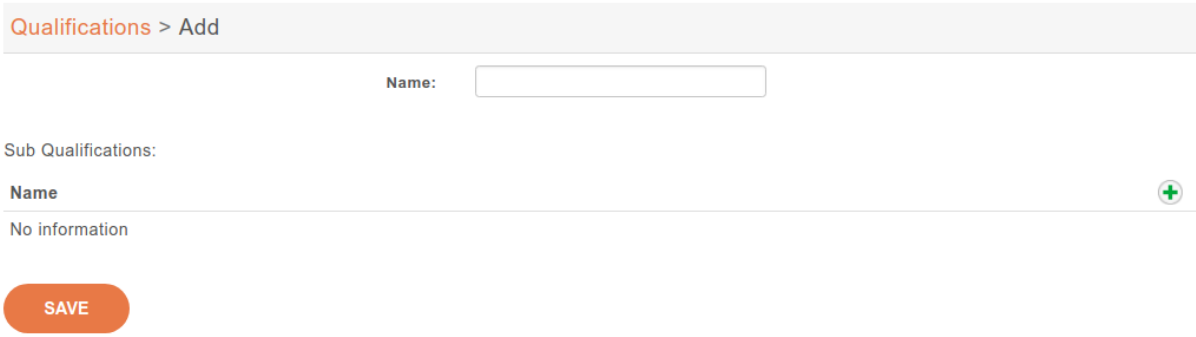
Besides these considerations, the project stay **Open Source** and is available here: https://gitlab.com/xivo.solutions/call-qualification

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**.

Note: Call qualifications will require some 3rd party application (see *Configuration*) complementary to CCAgent to display a sheet for agents to qualify the call and save results to database.

Qualifications are managed on the *Services* → *Call Center* → *Qualifications* page.

Services → *Call Center* → *Qualifications* → *Add*



Qualification

Qualification is a main description to which a sub qualification belongs.

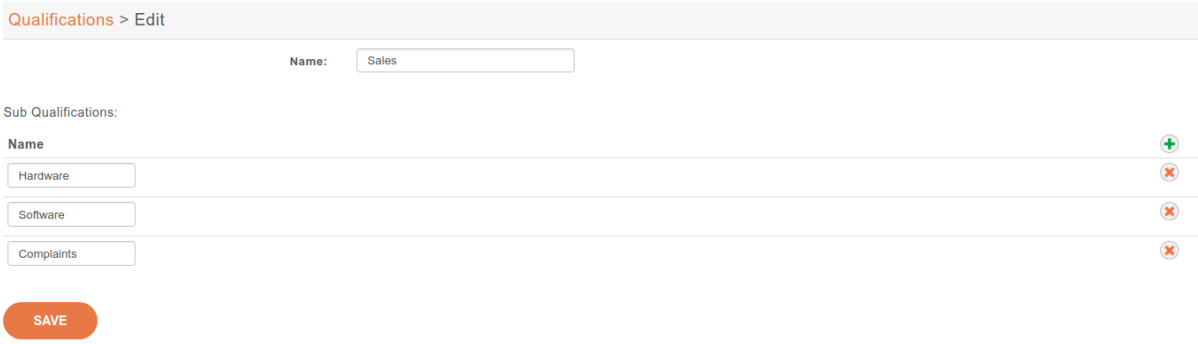
A qualification can be configured with the following options:

- Name: used as name for the qualification
- Sub Qualifications: used as name for the sub qualification

Sub Qualification

A sub qualification can be configured with the following options:

- Name: used as name for the qualification

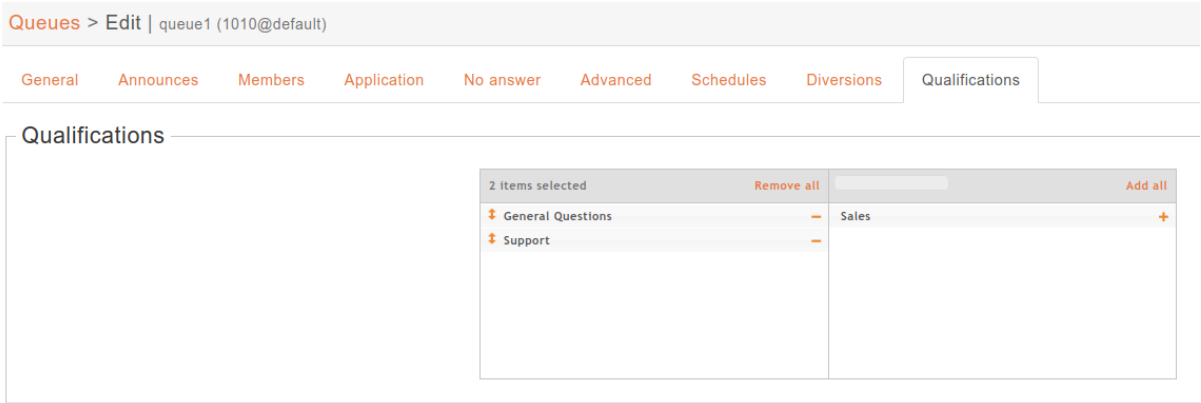


Assign qualification to queue

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.



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.

Qualification answers

A call in a queue can be qualified by the qualification assigned to this queue.

Exporting qualification answers

See *Exporting qualifications* feature in CCmanager.

6.4 XiVO Edge

Important:

This is an enterprise version feature introduced in the Gaia LTS that is not included in XiVO and is not freely available.
To enable it, please contact XiVO team.

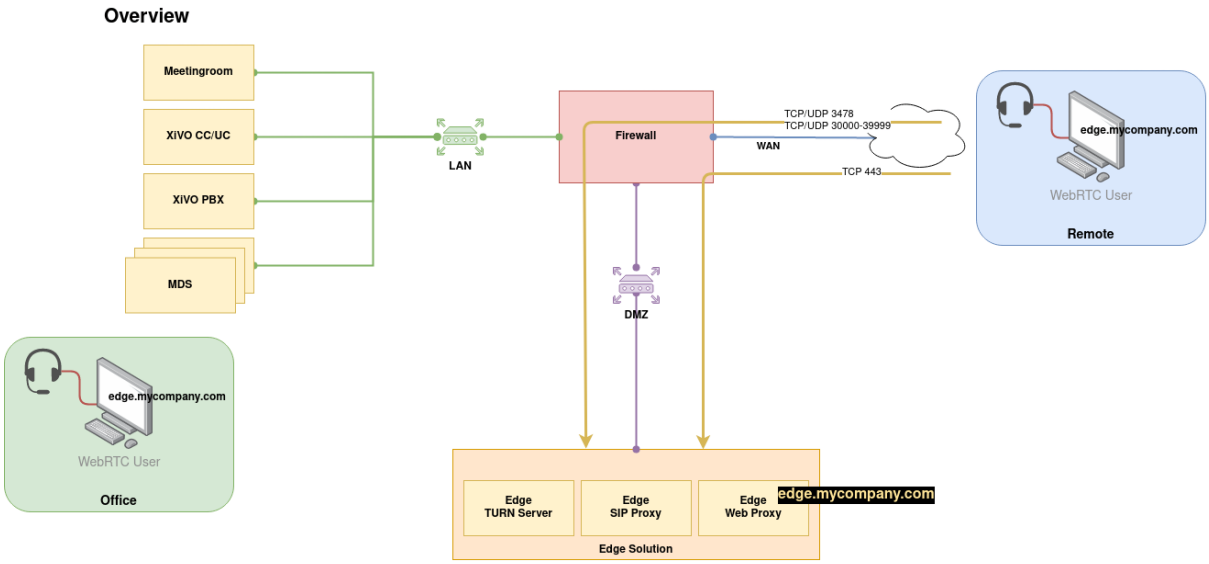
Besides these considerations, the project stay **Open Source** and is available here: https://gitlab.com/xivo.solutions/xivo-edge

His intended usage is to be able to use the WebRTC application without the need of a VPN. See architecture for more details.

6.4.1 Edge Architecture

- *Overall Architecture*
- *Network Flows*
  - *From the outside (WAN to DMZ)*
  - *Internally (DMZ to LAN)*
    - *Supplementary ports for Meetingroom (DMZ to LAN)*
  - *Internally (LAN to DMZ)*
    - *Supplementary ports for Meetingroom (LAN to DMZ)*
  - *Inside DMZ*

Overall Architecture



As you can see on the schema above the Edge Solution is to be put inside one's company DMZ.

From the external world only the Edge Solution will be seen. Only HTTPS and TURN flows should be authorized from the external - see *Network Flows* for more details.

The Edge Solution is composed of three components:

- a Web Proxy
- a SIP Proxy
- and a TURN Server

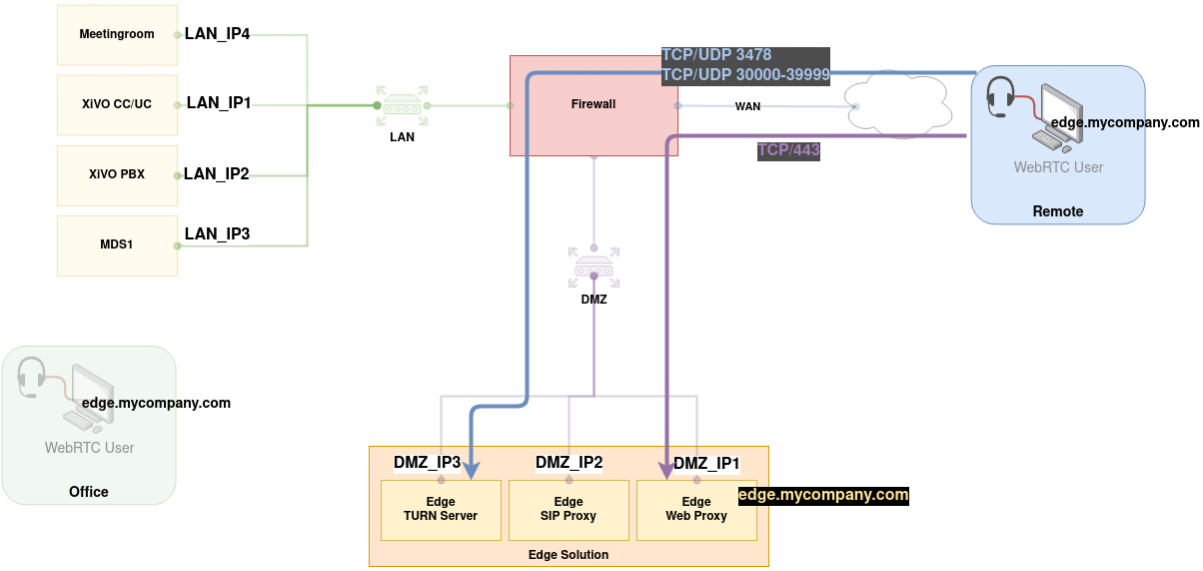
These components can be installed:

- on 3 separate servers - see *3 Servers Installation*
- or on 1 server - see *1 Server Installation*

Network Flows

From the outside (WAN to DMZ)

3 Servers Schema - External Flows (Outside - DMZ)

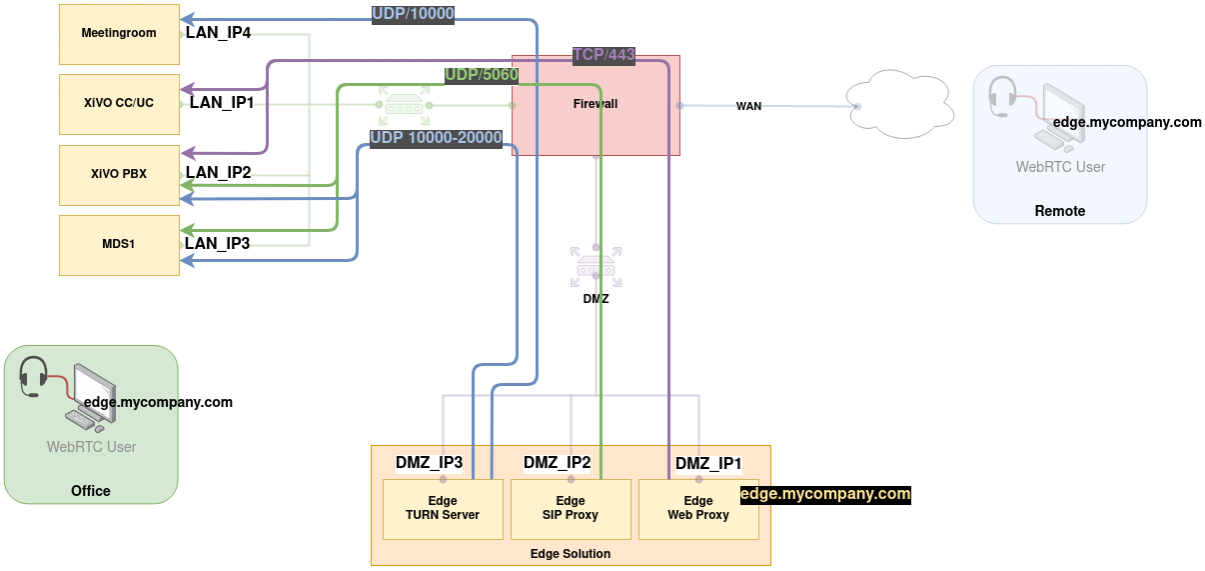


From	To	Ports to open	Usage
Outside `Any`	Web Proxy `DMZ_IP1`	• TCP/443	UC Application from External Users
Outside `Any`	TURN Server `DMZ_IP3`	• UDP/3478 • TCP/3478	STUN/TURN via UDP, TCP
Outside `Any`	TURN Server `DMZ_IP3`	• UDP/30000-39999 • TCP/30000-39999	RTP Flow between remote WebRTC client and TURN Server

Internally (DMZ to LAN)

From	To	Ports destination	Usage
Web Proxy `DMZ_IP1`	XiVO CC `LAN_IP1`	• TCP/443	Proxified UC application from Edge towards CC/UC Server
TURN Server `DMZ_IP3`	Mediaservers (inc. Main) `LAN_IP2`, `LAN_IP3`	• UDP/10000-20000	RTP flow between TURN Server and Asterisk(s)
SIP Proxy `DMZ_IP2`	Mediaservers (inc. main) `LAN_IP2`, `LAN_IP3`	• UDP/5060	SIP flow between Asterisk(s) and SIP Proxy

3 Servers Schema - Internal Flows (DMZ to LAN)



Supplementary ports for Meetingroom (DMZ to LAN)

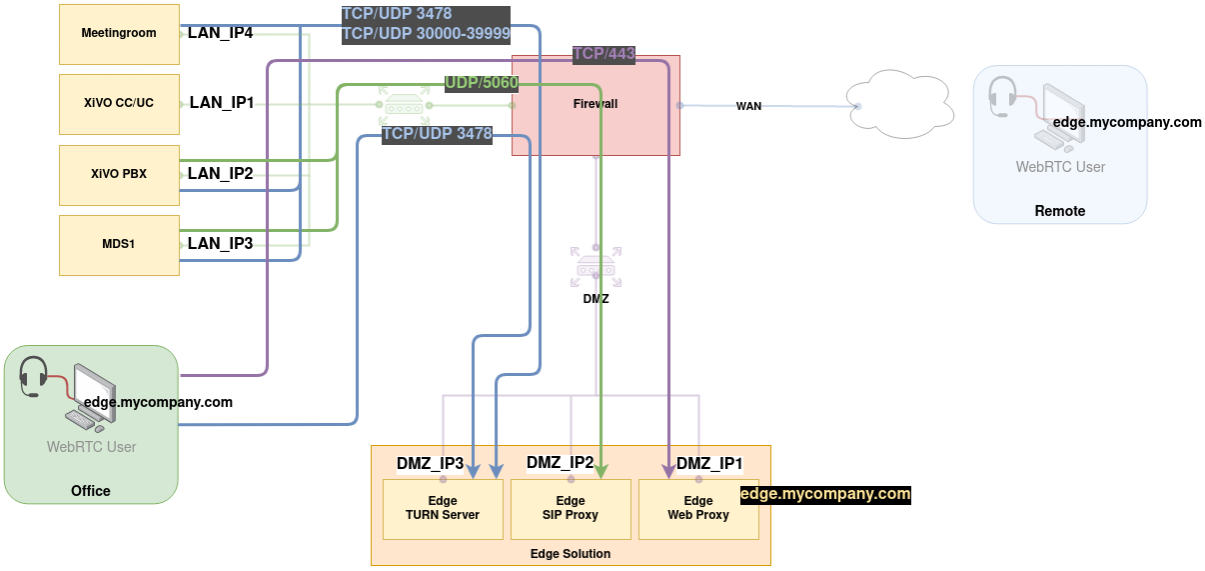
If you have a *Meetingroom server* in your installation you must also open these ports from the DMZ towards the LAN:

From	To	Ports to open	Usage
Web Proxy `DMZ_IP1`	XiVO `LAN_IP2`	• TCP/443	Proxified Meetingroom API from Edge towards XiVO Server
TURN Server `DMZ_IP3`	Meetingroom `LAN_IP4`	• UDP/10000	RTP flow between TURN Server and Meetingroom (when using TURN)

Internally (LAN to DMZ)

Here are the flow to open between the LAN towards the Edge servers (**LAN to DMZ**):

3 Servers Schema - Internal Flows (LAN to DMZ)



From	To	Ports to open	Usage
UC Clients `Any LAN`	Web Proxy `DMZ_IP1`	• TCP/443	UC Application for Internal Users
UC Clients `Any LAN`	TURN Server `DMZ_IP3`	• UDP/3478 • TCP/3478	STUN/TURN requests between UC Clients and TURN Server
Mediaservers (inc. Main) `LAN_IP2, LAN_IP3`	SIP Proxy `DMZ_IP2`	• UDP/5060	SIP flow between Asterisk(s) and SIP Proxy
Mediaservers (inc. Main) `LAN_IP2, LAN_IP3`	TURN Server `DMZ_IP3`	• UDP/3478 • TCP/3478	STUN/TURN requests between Mediaservers and TURN Server
Mediaservers (inc. Main) `LAN_IP2, LAN_IP3`	TURN Server `DMZ_IP3`	• UDP/30000-39999 • TCP/30000-39999	RTP flow between Asterisk(s) and TURN Server

Supplementary ports for Meetingroom (LAN to DMZ)

If you have a *Meetingroom server* in your installation you must also open these ports from the LAN towards the DMZ:

From	To	Ports to open	Usage
Meetingroom `LAN_IP4`	TURN Server `DMZ_IP3`	• UDP/3478 • TCP/3478	STUN/TURN requests between Meetingroom and TURN Server
Meetingroom `LAN_IP4`	TURN Server `DMZ_IP3`	• UDP/30000-39999 • TCP/30000-39999	RTP flow between Meetingroom and TURN Server

Inside DMZ

From/To	To/From	Ports to open	Usage
Web Proxy `DMZ_IP1`	SIP Proxy `DMZ_IP2`	• TCP/443	Websocket for SIP

This is probably unneeded (traffic between servers inside the DMZ should be in most of the cases unfiltered), but if it would be the case this flow must be authorized.

6.4.2 Edge Installation & Upgrade

- *Requirements*
  - *Server Requirements*
  - *Network Requirements*
- *Base installation*
  - *1. Docker & Docker compose Installation*
  - *2. XiVO Edge Launcher Setup*
- *3 Servers Installation*
  - *Web Proxy*
  - *SIP Proxy*
  - *TURN Server*
  - *Next step*
- *1 Server Installation*
  - *Web Proxy, SIP Proxy and TURN Server*
  - *Next step*
- *Upgrade Edge*

Requirements

Server Requirements

In case of *3 Servers Installation* each server should have:

- OS : **Debian 13** (Trixie), 64 bits.
- CPU: 2 CPU
- RAM: 2 Gb
- DD: 50 Gb

In case of *1 Server Installation* the server should have:

- OS : **Debian 13** (Trixie), 64 bits.
- CPU: 6 CPU
- RAM: 6 Gb

- DD: 50 Gb

Network Requirements

You will also need:

- 3 public IP addresses (or 1 for *1 Server Installation* deployment)
- 1 FQDN (i.e. edge.mycompany.com) that resolves:
  - to the Web Proxy server public IP address (in case of *3 Servers Installation*)
  - to the Edge server public IP address (in case of *1 Server Installation* deployment)
- a valid SSL certificate for this FQDN
- authorize the network flow as shown in the *Network Flows* section
- to make all your WebRTC users connect to the UC application via the Edge Solution

Important: otherwise it won't work !

Base installation

Whatever the type of installation (1 or 3 servers) you need to do the following on the host(s):

1. install docker and docker compose
2. download XiVO Edge service launcher

1. Docker & Docker compose Installation

These commands will install docker and docker compose on the host.

```
# Install docker prerequisites
apt update
apt install apt-transport-https ca-certificates curl gpg

# Add Docker's official GPG key:
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.
↪asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF

cat > /etc/apt/preferences.d/docker-ce <<EOF
Package: docker-ce*
Pin: version 5:28*
Pin-Priority: 1000
```

(continues on next page)

(continued from previous page)

```
EOF

cat > /etc/apt/preferences.d/containerd <<EOF
Package: containerd.io*
Pin: version 2.*
Pin-Priority: 1000
EOF

apt update
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
↪plugin

#Install docker compose v2
apt update
apt install docker-compose-plugin

mkdir -p /etc/systemd/system/containerd.service.d
cat > /etc/systemd/system/containerd.service.d/override.conf <<EOF
[Service]
LimitNOFILE=100000
EOF

systemctl daemon-reload
systemctl restart containerd.service
```

2. XiVO Edge Launcher Setup

The following commands describe how to install the Edge Solution launcher.

- Create `edge` directory:

```
mkdir -p /etc/docker/edge
```

- Download *XiVO Edge* configuration. In the following script replace `TAG_OR_BRANCH` by the name of a tag or a branch.

  Note: currently, to install the latest stable version of quantum, use `TAG_OR_BRANCH=2026.05.02`

```
TAG_OR_BRANCH=2026.05.02
cd /etc/docker/edge
wget "https://gitlab.com/xivo.solutions/xivo-edge/-/archive/${TAG_OR_BRANCH}/$
↪{TAG_OR_BRANCH}.tar.gz"
tar -zxvf ${TAG_OR_BRANCH}.tar.gz -C /etc/docker/edge --strip-components 1
rm ${TAG_OR_BRANCH}.tar.gz
```

- Create the `ssl` dir to put the ssl certificates:

```
mkdir -p /etc/docker/ssl
```

- Congrats: **You're done with the base installation**.
- **Next step** follow:
  - either the *3 Servers Installation*
  - or the *1 Server Installation*

3 Servers Installation

Note: If you're doing the 3 servers installation (1 host per service) you must follow the three subsections below.

You need one machine per service with *Base installation* done on all three.

Then you need to install the services on each host independently as explained below.

Web Proxy

Note: This step is to be done on the server which will host the **Web Proxy** (nginx) service.

- Create bash alias to launch services:

```
echo -e "Current alias for edge-dcomp is:\n$(alias edge-dcomp)"
sed -i '/alias edge-dcomp=/d' ~/.bashrc
echo "alias edge-dcomp='docker compose -p edge -f /etc/docker/edge/nginx-
↪edge.yml --env-file=/etc/docker/edge/.env'" >> ~/.bashrc
source ~/.bashrc
echo -e "New alias for edge-dcomp is now:\n$(alias edge-dcomp)"
```

SIP Proxy

Note: This step is to be done on the server which will host the **SIP Proxy** (kamailio) service.

- Create bash alias to launch services:

```
echo -e "Current alias for edge-dcomp is:\n$(alias edge-dcomp)"
sed -i '/alias edge-dcomp=/d' ~/.bashrc
echo "alias edge-dcomp='docker compose -p edge -f /etc/docker/edge/kamailio-
↪edge.yml -f /etc/docker/edge/kamailio-edge.override.yml --env-file=/etc/
↪docker/edge/.env'" >> ~/.bashrc
source ~/.bashrc
echo -e "New alias for edge-dcomp is now:\n$(alias edge-dcomp)"
```

TURN Server

Note: This step is to be done on the server which will host the **TURN Server** (coturn) service.

- Create bash alias to launch services:

```
echo -e "Current alias for edge-dcomp is:\n$(alias edge-dcomp)"
sed -i '/alias edge-dcomp=/d' ~/.bashrc
echo "alias edge-dcomp='docker compose -p edge -f /etc/docker/edge/coturn-
↪edge.yml --env-file=/etc/docker/edge/.env'" >> ~/.bashrc
source ~/.bashrc
echo -e "New alias for edge-dcomp is now:\n$(alias edge-dcomp)"
```

Next step

- Congrats: You're done with the Edge 3 Servers Installation.
- **Next step**: go to *Edge Configuration*

1 Server Installation

Note: If you're doing the mono server installation (all services on one host) you must follow the subsection below.

Web Proxy, SIP Proxy and TURN Server

- Create bash alias to launch services:

```
echo -e "Current alias for edge-dcomp is:\n$(alias edge-dcomp)"
sed -i '/alias edge-dcomp=/d' ~/.bashrc
echo "alias edge-dcomp='docker compose -p edge -f /etc/docker/edge/nginx-edge.
↪yml -f /etc/docker/edge/coturn-edge.yml -f /etc/docker/edge/kamailio-edge.yml -
↪-env-file=/etc/docker/edge/.env'" >> ~/.bashrc
source ~/.bashrc
echo -e "New alias for edge-dcomp is now:\n$(alias edge-dcomp)"
```

Next step

- Congrats: You're done with the Edge Mono Server Installation.
  - **Next step**: go to *Edge Configuration*

Upgrade Edge

Currently there is no *automatic upgrade* process. Here is the manual process that you need to follow on the 3 edge servers (or the edge server depending if it's a mono or three servers install).

- Make a backup of the Edge launcher:

```
EDGE_BCK_DIR=$(mktemp -d -p /var/tmp/ edge-backup.XXXXXXXX)
cp -arT /etc/docker/edge/ ${EDGE_BCK_DIR}
echo "Backup was created in ${EDGE_BCK_DIR}"
```

- Re-install the Edge Launcher (it will override the `.yml` files): follow the *Download and Extract of the Edge launcher files* by taking the new versions

  Note: This step overrides the current `.yml` files. If you had made some customization in them you will have to backport them by comparing the new one with the backup you did at previous step. You can use this to show the diff between the newly installed yml files and the backup:

  ```
  for ymlfile in $(ls -1 /etc/docker/edge/*.yml);
    do diff -Zu "${EDGE_BCK_DIR}/$(basename ${ymlfile})" "${ymlfile}";
  done
  ```

- Replace the `edge-dcomp` alias by what is described in the *3 Servers Installation* or *1 Server Installation* sections (replace the aliases or verify that they did not change).

- And then verify that the content of the `.env` file is correct:
  - Compare the old version and what is defined in the *Edge Configuration* section.

    ```
    diff -Zu "${EDGE_BCK_DIR}/.env" "/etc/docker/edge/.env"
    ```

  - Verify that the `XIVOCC_TAG` and `XIVOCC_DIST` correspond to what you want to install (it should be *2026.05* and *latest*).
- Finally pull the new images and restart the containers:

Warning: it will stop all WebRTC calls (and disconnect users from the application).

```
edge-dcomp pull
edge-dcomp up -d
```

6.4.3 Edge Configuration

- *Edge Server configuration*
  - *SSL Certificates*
  - *TURN Server Secret*
  - *TURN Server Relay Authorization*
  - *3 Servers Configuration*
    - *3 Servers Example Schema*
    - *Web Proxy*
    - *SIP Proxy*
    - *STUN/TURN server*
  - *1 Server Configuration*
    - *1 Server Example Schema*
    - *Web Proxy, SIP Proxy and TURN Server*
  - *Start the services*
- *XiVO CC configuration*
  - *XiVO UC Embdedded Configuration*
- *XiVO Configuration*
  - *STUN/TURN server configuration*
  - *Fail2ban*

Edge Server configuration

SSL Certificates

Note: This step is to be done on each host

The Edge Solution must be configured with valid certificate for your domain.

You have to put the:

- **fullchain** certificate: here `/etc/docker/ssl/xivo-edge.crt` and also here `/etc/docker/ssl/xivo-edge.chain` (this last location is mandatory for the SIP Proxy service)
- certificate key: here `/etc/docker/ssl/xivo-edge.key`

Make sure that the certificate key are given readonly permission to root and the folder is accessible for all users:

```
chmod +x /etc/docker/ssl
chmod 404 /etc/docker/ssl/xivo-edge.key
```

TURN Server Secret

You must generate a secret for the TURN server (in the following it will referred to as `<TURN_SECRET>`).

Generate it with the following command:

```
openssl rand -hex 16
```

Keep this `<TURN_SECRET>`. You will need it to configure

- the TURN Server
- the xucserver on the *XiVO CC/UC*
- and to generate the turn configuration on for asterisk (on XiVO Main and MDS)

TURN Server Relay Authorization

Important: You **MUST** follow this step on the TURN Server host. Otherwise the TURN relay **will not work**.

You **must add to the TURN configuration** the **IP addresses it is allowed to relay traffic to**. Therefore you must add the permission for the IP of:

- the XiVO
- the MDS (if in XDS architecture)
- and the Meetingroom (if installed)

In order to do this:

- lists the IP addresses towards which the **STUN/TURN Server** will be allowed to send traffic to. Which is at least:
  - the XiVO
  - (if applicable) the MDS
  - (if applicable) the Meetingroom

- then add to the `.env` file (of the server which will host **STUN/TURN Server** (coturn) service) a line with the following format:

```
TURN_ALLOWED_PEERS="--allowed-peer-ip=<XIVO IP> --allowed-peer-ip=<MDS1 IP> --
↪allowed-peer-ip=<MEETINGROOM IP> --allowed-peer-ip=<DMZ_IP3>"
```

For example:

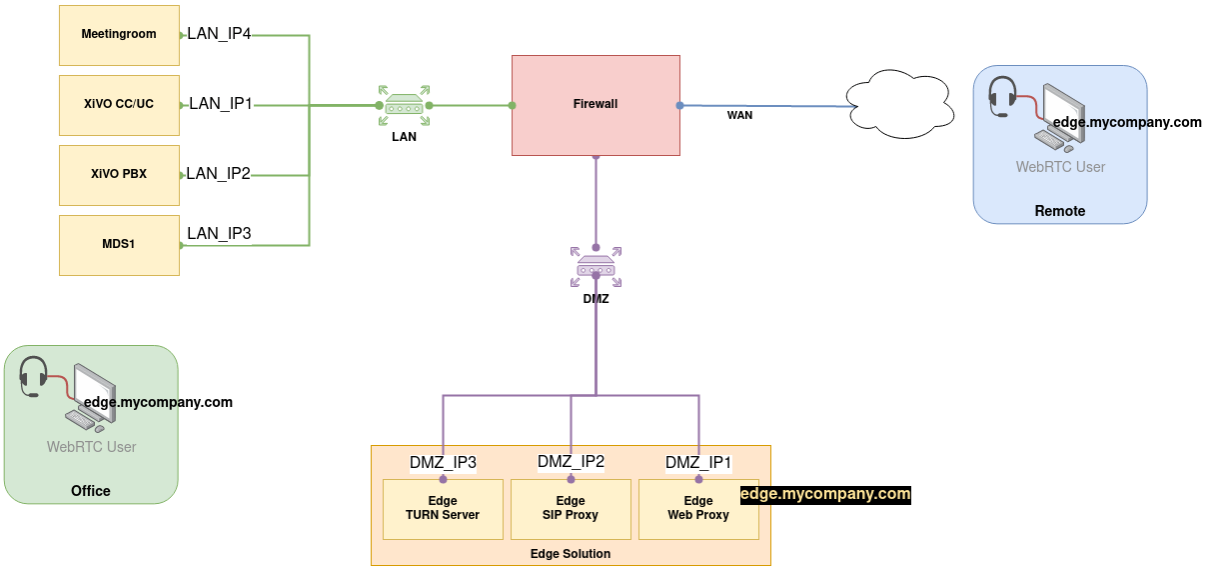
```
TURN_ALLOWED_PEERS="--allowed-peer-ip=192.168.240.2 --allowed-peer-ip=192.168.
↪240.4 --allowed-peer-ip=192.168.250.3"
```

3 Servers Configuration

Note: Follow this part if you configure the Edge Solution on 3 servers

3 Servers Example Schema

3 Servers Schema



Web Proxy

Note: This step is to be done on the server which will host the **Web Proxy** (nginx) service

Create the `.env` file with the following variables (replace values accordingly, and see example below):

```
cat > /etc/docker/edge/.env << EOF
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
XIVOCC_HOST=<IP ADDRESS OF THE XIVO CC/UC (xucserver/xucmgt/nginx server)>
XIVO_HOST=<IP ADDRESS OF THE XIVO>
EDGE_FQDN=<XIVO EDGE FQDN>
```

(continues on next page)

(continued from previous page)

```
EDGE_KAMAILIO_HOST=<IP ADDRESS OF THE KAMAILIO SERVER>
EOF
```

Example:

- given you install latest Luna version
- and given the *3 Servers Example Schema*

you should come up with:

```
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
XIVOCC_HOST=LAN_IP1
XIVO_HOST=LAN_IP2
EDGE_FQDN=edge.mycompany.com
EDGE_KAMAILIO_HOST=DMZ_IP2
```

SIP Proxy

Note: This step is to be done on the server which will host the **SIP Proxy** (kamailio) service

Create the `.env` file with the following variables (replace values accordingly, and see example below):

```
cat > /etc/docker/edge/.env << EOF
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
EDGE_FQDN=<XIVO EDGE FQDN>
XIVO_HOST=<IP ADDRESS OF THE XIVO>
EDGE_HOST_IP=<Edge server IP on which to listen>
EOF
```

Important: For a XDS installation you must:

- add to the `.env` file a line per MDS (excluding XiVO PBX) with the following format:

  ```
  XIVO_MDS_HOST_MDS1="<mds technical name>=<DATA IP ADDRESS TO MDS1>"
  ```

  For example: `XIVO_MDS_HOST_MDS1="mds1=192.168.240.2"`

  Mandatory step: You need to change any underscore in your mds naming to an hyphen For example, for an mds named media_server_1: `XIVO_MDS_HOST_MDS1="media-server-1=192.168.240.2"`

- and add to the `kamailio-edge.yml` file a `extra_hosts` section for kamailio with these variables:

  For example:

  ```yaml
  services:
    kamailio:
      image: "xivoxc/edge-kamailio:${XIVOCC_TAG}.${XIVOCC_DIST}"
      extra_hosts:
      - ${XIVO_MDS_HOST_MDS1}
  ```

Example:

- given you install latest Quantum version

- and given the *3 Servers Example Schema*

you should come up with:

```
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
EDGE_FQDN=edge.mycompany.com
XIVO_HOST=LAN_IP2
XIVO_MDS_HOST_MDS1="mds1=LAN_IP3"
EDGE_HOST_IP=DMZ_IP2
```

STUN/TURN server

Note: This step is to be done on the server which will host the **STUN/TURN Server** (coturn) service

Create the `.env` file with the following variables (replace values accordingly, and see example below):

```
cat > /etc/docker/edge/.env << EOF
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
EDGE_HOST_IP=<IP OF THE STUN/TURN SERVER>
XIVO_HOST=<IP ADDRESS OF THE XIVO>
TURN_EXTERNAL_IP=<External IP of the edge Server>
TURN_SERVER_SECRET=<TURN_SECRET>
TURN_REALM=<domain name of client>
TURN_ALLOWED_PEERS=<list of allowed peers IP>
EOF
```

Important: Don't forget to fill in the `TURN_ALLOWED_PEERS` var following *TURN Server Relay Authorization*

Example:

- given you install latest Luna version
- and given the *3 Servers Example Schema*

you should come up with:

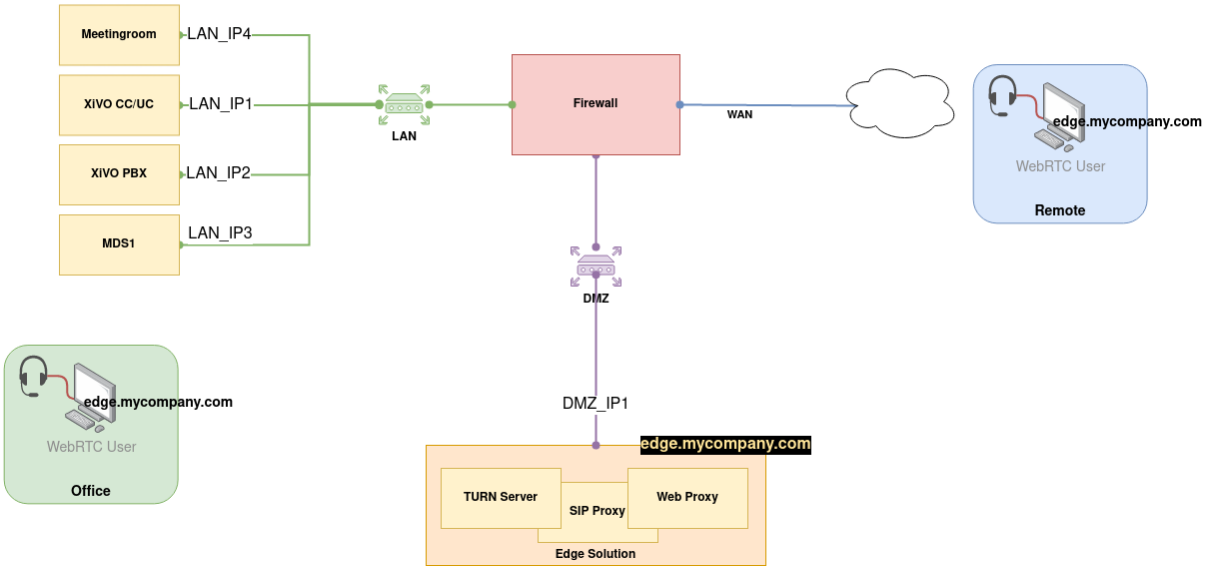
```
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
EDGE_HOST_IP=DMZ_IP3
XIVO_HOST=LAN_IP2
TURN_EXTERNAL_IP=DMZ_IP3
TURN_SERVER_SECRET=<TURN_SECRET>
TURN_REALM=mycompany.com
TURN_ALLOWED_PEERS="--allowed-peer-ip=LAN_IP2 --allowed-peer-ip=LAN_IP3 --allowed-
↪peer-ip=LAN_IP4 --allowed-peer-ip=DMZ_IP3"
```

1 Server Configuration

Note: Follow this part if you configure the Edge Solution 1 server

1 Server Example Schema

1 Server Schema



Web Proxy, SIP Proxy and TURN Server

Create the `.env` file with the following variables (replace values accordingly, and see example below):

```
cat > /etc/docker/edge/.env << EOF
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
XIVOCC_HOST=<IP ADDRESS OF THE XIVOCC>
XIVO_HOST=<IP ADDRESS OF THE XIVO>
EDGE_FQDN=<EDGE FQDN>
TURN_SERVER_SECRET=<TURN_SECRET>
TURN_REALM=<EDGE DOMAIN>
EDGE_HOST_IP=<EDGE HOST IP>
TURN_EXTERNAL_IP=<EDGE HOST IP>
TURN_ALLOWED_PEERS=<list of allowed peers IP>
EOF
```

Important: Don't forget to fill in the `TURN_ALLOWED_PEERS` var following *TURN Server Relay Authorization*

Important: For a XDS installation you must:

- add to the `.env` file a line per MDS (excluding XiVO PBX) with the following format:

```
XIVO_MDS_HOST_MDS1="<mds technical name>=<DATA IP ADDRESS TO MDS1>"
```

For example: `XIVO_MDS_HOST_MDS1="mds1=192.168.240.2"`

Mandatory step: You need to change any underscore in your mds naming to an hyphen For example, for an mds named media_server_1: `XIVO_MDS_HOST_MDS1="media-server-1=192.168.240.2"`

- and add to the `kamailio-edge.yml` file a `extra_hosts` section for kamailio with these variables:

  For example:

```
services:
  kamailio:
    image: "xivoxc/edge-kamailio:${XIVOCC_TAG}.${XIVOCC_DIST}"
    extra_hosts:
    - ${XIVO_MDS_HOST_MDS1}
```

Example:

- given you install latest Quantum version
- and given the *1 Server Example Schema*

you should come up with:

```
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
XIVOCC_HOST=LAN_IP1
XIVO_HOST=LAN_IP2
EDGE_FQDN=edge.mycompany.com
TURN_SERVER_SECRET=<TURN_SECRET>
TURN_REALM=mycompany.com
EDGE_HOST_IP=DMZ_IP1
TURN_EXTERNAL_IP=DMZ_IP1
XIVO_MDS_HOST_MDS1="mds1=LAN_IP3"
TURN_ALLOWED_PEERS="--allowed-peer-ip=LAN_IP2 --allowed-peer-ip=LAN_IP3 --allowed-
↪peer-ip=LAN_IP4 --allowed-peer-ip=DMZ_IP3"
```

Start the services

To start the services run the following command (you have to run it on all servers if you are in a 3 servers installation):

```
edge-dcomp up -d
```

XiVO CC configuration

Note: These steps are to be done on the XiVO CC.

For *XiVO UC Embedded* mode see *XiVO UC Embdedded Configuration* section.

On XiVO CC server (the server which hosts the xucmgt/xucserver and nginx) add the following variables in the `/etc/docker/compose/custom.env` file:

- add XUC_INTERNAL_HOST=<XIVOCC_IP>
- change XUC_HOST to the EDGE FQDN
- and add the TURN_SERVER_SECRET with the `<TURN_SECRET>` value (the `<TURN_SECRET>` is the secret generated for during *TURN Server Secret*)

- specify the `REPORTING_HOST` to the XiVO CC LAN IP address in order for SpagoBI report to work (note that SpagoBI work only if you access them via the internal LAN IP address).
- in order for the history to work add the RECORDING_SERVER_HOST=<IP of recording server>
- **Optionally** you can override ice gathering timeout (by default 500 milliseconds) to a shorter/greater value thanks to `ICE_GATHERING_TIMEOUT_MS`

```
XUC_INTERNAL_HOST=<XIVOCC_IP>
XUC_HOST=<XIVO EDGE FQDN>
TURN_SERVER_SECRET=<TURN_SECRET>
REPORTING_HOST=<LAN_IP1>
RECORDING_SERVER_HOST=<RECORDING_SERVER_IP>
RECORDING_SERVER_PORT=9400
```

- then you need to recreate components containers

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

XiVO UC Embdedded Configuration

Important: This section applies only if you are in the *XiVO UC Embedded* mode.

To configure the UC applications part of a XiVO, add the following variables in the `/etc/docker/compose/custom.env` file:

- change XUC_HOST to the EDGE FQDN
- and add the TURN_SERVER_SECRET with the `<TURN_SECRET>` value (the `<TURN_SECRET>` is the secret generated for during *TURN Server Secret*)

```
XUC_HOST=<XIVO EDGE FQDN>
TURN_SERVER_SECRET=<TURN_SECRET>
```

- then you need to recreate components containers

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

XiVO Configuration

Note: These steps are to be done on the XiVO

If you are using the *XiVO UC Embedded* mode you must also follow the section *XiVO UC Embdedded Configuration*

STUN/TURN server configuration

On the XiVO you need to configure the STUN/TURN server address.

Warning: Your Edge installation must be up and running when you do this step. As soon as you change this configuration all WebRTC users **won't work** without the Edge Solution.

- In the XiVO Admin webi *Services -> IPBX -> General Settings -> SIP Protocol* tab **Network**

- In parameter *XiVO Edge FQDN* enter the **TURN server** address with format `FQDN:3478` or `IP:3478` (you must put port **3478**). **Warning:** this FQDN must be resolvable and reachable from the XiVO & Public Internet.
  - 1 Server deployment: use the server public IP address or FQDN (i.e. `edge.mycompany.com` or `DMZ_IP1` in 1 Server Schema)
  - 3 Servers deployment: use the TURN server public IP address or FQDN (i.e. `DMZ_IP3` in 3 Servers Schema)
- Click save: it will reload the SIP and the RTP configuration of the XiVO (Main and MDS if any)
- Then **on each XiVO/MDS** run the following script to generate the credentials for the TURN server (the `<TURN_SECRET>` is the secret generated during *TURN Server Secret*):

```
xivo-edge-gen-turn-cred <TURN_SECRET>
```

- This script will generate new turn username and password in `/etc/asterisk/rtp.d/01-xivo-edge-turn-cred.conf`
- You then need to reload the rtp configuration

```
asterisk -rx 'module reload res_rtp_asterisk.so'
```

Note: The turn address `turnaddr` is generated by confgend (see conf in file `/etc/asterisk/rtp.conf`).

Fail2ban

Important: If you configure the Edge Solution for a XDS installation you must do it on each MDS.

You need to add Kamailio's ip address to protect it from being banned and resulting in complete block of webrtc users.

- Edit file `/etc/fail2ban/filter.d/asterisk-xivo.conf` and add to the `ignoreregex` section:
  - In *3 Servers Schema* replace `<SIP_PROXY_HOST_IP>` by `DMZ_IP2`
  - In *1 Server Schema* replace `<SIP_PROXY_HOST_IP>` by `DMZ_IP1`

```
ignoreregex = <SIP_PROXY_HOST_IP>
```

- Finally restart fail2ban service:

```
systemctl restart fail2ban.service
```

6.4.4 Edge Administration

- *Launch services*
- *See logs*
  - *Web Proxy Logs*
    - *Rate limiting*
  - *TURN Server Logs*
- *Troubleshooting*

- *XiVO*
  - *ICE negotiation debug in asterisk*
  - *Error sending STUN request log*
  - *Relay Permission Error*
- *TURN Server*
  - *Relay Permission Error*
- *SIP Server*

Launch services

```
edge-dcomp up -d
```

See logs

By default, logs are configured to keep 200M of log per service (5 files of 40M each).

You can see logs with the following command

```
edge-dcomp logs -tf
```

This one to see the last 1000 lines of logs for the nginx service

```
edge-dcomp logs -tf --tail=1000 nginx
```

Web Proxy Logs

Rate limiting

When a request is rate limited nginx answers with status **429**.

```
nginx_1   | 2021-03-03T09:52:43.227539911Z 81.185.165.93 - - [03/Mar/2021:09:52:43␣
↪+0000] "GET / HTTP/1.1" 429 169 "-" "Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/4.0.4"
↪"-"
```

You will also see an error log with the zone that was limited, the client address etc.:

```
nginx_1   | 2021-03-03T09:52:39.122434232Z 2021/03/03 09:52:39 [error] 41#41: *31␣
↪limiting requests, excess: 20.440 by zone "webapp", client: 81.185.165.93, server:␣
↪xivo-edge-access.dev.avencall.com, request: "GET / HTTP/1.1", host: "xivo-edge-
↪access.dev.avencall.com"
```

TURN Server Logs

Examples of logs:

TURN Allocation Request

- 91.194.178.211 is the client
- 91.194.178.210 is the TURN server

```
coturn_1  | 2021-03-03T12:24:25.079776718Z 361: : IPv4. tcp or tls connected to: 91.
↪194.178.211:45575
coturn_1  | 2021-03-03T12:24:25.079812744Z 361: : IPv4. tcp or tls connected to: 91.
↪194.178.211:45575
coturn_1  | 2021-03-03T12:24:25.079906231Z 361: : session 000000000000000005: realm
↪<dev.avencall.com> user <>: incoming packet message processed, error 401:␣
↪Unauthorized
coturn_1  | 2021-03-03T12:24:25.079948400Z 361: : session 000000000000000005: realm
↪<dev.avencall.com> user <>: incoming packet message processed, error 401:␣
↪Unauthorized
coturn_1  | 2021-03-03T12:24:25.080959417Z 361: : IPv4. Local relay addr: 91.194.178.
↪210:39824
coturn_1  | 2021-03-03T12:24:25.080979712Z 361: : IPv4. Local relay addr: 91.194.178.
↪210:39824
coturn_1  | 2021-03-03T12:24:25.080984747Z 361: : session 000000000000000005: new,␣
↪realm=<dev.avencall.com>, username=<1614779903>, lifetime=600
coturn_1  | 2021-03-03T12:24:25.080988937Z 361: : session 000000000000000005: new,␣
↪realm=<dev.avencall.com>, username=<1614779903>, lifetime=600
coturn_1  | 2021-03-03T12:24:25.081354852Z 361: : session 000000000000000005: realm
↪<dev.avencall.com> user <1614779903>: incoming packet ALLOCATE processed, success
coturn_1  | 2021-03-03T12:24:25.081375101Z 361: : session 000000000000000005: realm
↪<dev.avencall.com> user <1614779903>: incoming packet ALLOCATE processed, success
```

Troubleshooting

XiVO

ICE negotiation debug in asterisk

In asterisk CLI run command:

```
core set debug 2 res_rtp_asterisk.so
```

Then look in asterisk full logs. You'll see logs like:

```
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) RTP␣
↪allocated port 15094
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) ICE␣
↪creating session 0.0.0.0:15094 (15094)
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) ICE␣
↪create
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) ICE␣
↪add system candidates
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) ICE␣
↪add candidate: 10.32.4.2:15094, 2130706431
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) ICE␣
↪request TURN TCP RTP candidate
```

(continues on next page)

(continued from previous page)

```
[Mar 25 18:23:58] DEBUG[1806][C-0000001d] res_rtp_asterisk.c: (0x7ff5182e4a30) ICE␣
↪add candidate: 10.32.0.5:39715, 16777215
```

In the log above:

- 10.32.4.2 is the asterisk local IP address
- 10.32.0.5 is the TURN IP address

Error sending STUN request log

If you see this log:

```
[Mar 25 16:21:14] ERROR[1797]: pjproject: <?>:     icess0x7ff518338828 ..Error␣
↪sending STUN request: Network is unreachable
```

it means that XiVO can't reach (at the network level) one of the peer candidates. But it would that the xivo doesn't have network route toward this peer candidate. It should not happen if the XiVO is correctly configured with a default route.

Relay Permission Error

The TURN server is by default configured to not relay towards internal network. One must whitelist the at least the XIVO IP (and MDS and Meetingroom server if you have one) - see TURN section of *TURN Server Relay Authorization*.

If you see these logs **for an IP** of **either the XIVO or the MDS or the Meetingroom server** then it **may** mean that you didn't configure the relay permission correctly:

```
[Oct 18 18:32:03] ERROR[24183] pjproject:         tcprel0x7f7c343f59a0 .
↪CreatePermission failed for IP 10.32.0.1: 403/Forbidden IP
```

For example in the log above the coturn server does not accept to relay traffic towards **10.32.0.1** IP address. It's ok as long as **10.32.0.1** is not one of the XIVO or the MDS (or the Meetingroom) server IP address.

See *Relay Permission Error* for the corresponding logs in TURN Server.

TURN Server

Relay Permission Error

The TURN server is by default configured to not relay towards internal network. One must whitelist the at least the XIVO IP (and MDS and Meetingroom server if you have one) - see TURN section of *TURN Server Relay Authorization*.

If you see these logs **for an IP** of **either the XIVO or the MDS or the Meetingroom server** then it **may** mean that you didn't configure the relay permission correctly:

```
coturn_1    | 2021-10-18T16:35:01.310299945Z 807: : ERROR: A peer IP 10.32.0.5 denied␣
↪in the range: 10.0.0.0-10.255.255.255
coturn_1    | 2021-10-18T16:35:01.326137547Z 807: : session 001000000000000017: realm
↪<test.avencall.com> user <1634578239>: incoming packet CREATE_PERMISSION processed,␣
↪error 403: Forbidden IP
coturn_1    | 2021-10-18T16:35:01.326589074Z 807: : session 001000000000000017: realm
↪<test.avencall.com> user <1634578239>: incoming packet message processed, error␣
↪403: Forbidden IP
```

For example in the log above the coturn server does not accept to relay traffic towards **10.32.0.5** IP address. It's ok as long as **10.32.0.5** is not one of the XIVO or the MDS (or the Meetingroom) server IP address.

See *Relay Permission Error* for the corresponding logs in XiVO.

SIP Server

To debug Kamailio you can raise log level of module *xlog* (for dbg log in kamailio script) and *tm* for relay operation:

```
edge-dcomp exec kamailio bash
kamcmd
dbg.set_mod_level xlog 3
dbg.set_mod_level tm 3
```

6.4.5 Edge Features

- *Timezone*
- *Proxy Web Features*
  - *Expose TURNS server on port 443*
    - *How to activate this feature*
    - *How to deactivate this feature*
  - *Extend the Configuration*
  - *Number of Connections*
  - *Certificates and OCSP*
  - *Rate limiting*
  - *Token format validation*
- *TURN Server Features*
  - *Use TURN server on port 443*
  - *TURN Credentials*
  - *TURN Credentials lifetime*
  - *Optimize ICE (STUN/TURN) negotiation time*
    - *1. Before emitting the call*
    - *2. When call is answered*

Timezone

Timezone inside containers defaults to Europe/Paris. It can be changed by adding the env var `TIMEZONE` in the `.env` file.

Proxy Web Features

The Web proxy is configured to:

- proxify only the needed route towards the XiVO CC
- rate-limit the call to the application
- verify that calls to `/xuc/api/2.0/cti` endpoint (CTI WS) contains a token in the right format in the argument
- verify that calls to `/wssip` and `/wssip-MDSNAME` endpoint (SIP WS) contains a token in the argument

Expose TURNS server on port 443

(*Since 2025.02.00*)

Note: This applies only to the **1 Server Installation**

Important: Pre-requisite: have two different FQDN resolving to the `EDGE_HOST_IP`: one for the web the other for the turn.

In Edge deployment the TURN server is listening for TURN and TURNS request on port **3478**. In some environement this port may not be accessible: in this case the TURN server can not be used as a Relay.

To mitigate this issue it is possible to configure the Edge server to make TURNS available on port 443.

How to activate this feature

Important: Pre-requisite: have two different FQDN resolving to the `EDGE_HOST_IP`:

- a `EDGE_FQDN` (e.g. edge.mycorp.com) resolving to `EDGE_HOST_IP`
- **and** another `TURN_FQDN` (e.g. edge-turn.mycorp.com) resolving *also* to `EDGE_HOST_IP`

1. **Edit** the `/etc/docker/edge/.env` file adding the following variables:

```
EDGE_NGINX_TURN_PROXY=true
TURN_FQDN=<FQDN for TURN> # e.g. edge-turn.mycorp.com
```

2. Relaunch Edge services - see *Launch services*

How to deactivate this feature

1. **Remove** the `EDGE_NGINX_TURN_PROXY` and `TURN_FQDN` from the `/etc/docker/edge/.env` file.

```
sed -i '/^EDGE_NGINX_TURN_PROXY/d' /etc/docker/edge/.env
sed -i '/^TURN_FQDN/d' /etc/docker/edge/.env
```

2. Relaunch Edge services - see *Launch services*

Extend the Configuration

(*Since Naos.03*)

It is possible to extend the Nginx configuration. Our Edge Nginx is split under 2 server:

- one on port 80,
- one on port 443 with SSL

It is possible to extend each of these server section. It is mainly useful to add some location.

1. Create a directory on your Egde Web Proxy server:

```
mkdir -p /etc/docker/nginx/extra-conf.d/{http,https}/
```

2. **Edit** the Edge Nginx compose file `/etc/docker/edge/nginx-edge.yml` to add the following volume:

```
services:
  nginx:
  [...]
    volumes:
    - /etc/docker/ssl:/etc/nginx/ssl
    - /etc/docker/nginx/extra-conf.d/:/etc/nginx/templates/extra-conf.d/
    logging:
  [...]
```

3. Now add you can extend one of the server configuration section. For example:
   - create a file `/etc/docker/nginx/extra-conf.d/https/xivocc-deny-callback-api.conf.template`
   - with the folowing content:

```
location ~ ^/xuc/api/2.0/callback_lists {
  deny all;
}
```

   - this will deny the call to callback_lists API

Number of Connections

Since Maia.11

The number of connections Nginx can handle is given by the formula:

```
max connections = worker_processes * worker_connections
```

The Edge Nginx container:

- tunes the `worker_processes` variable at startup (this can be changed with the `NGINX_ENTRYPOINT_WORKER_PROCESSES_AUTOTUNE` to false)
- sets the `worker_connections` by default to 20000 - it means the Proxy Web should be able to handle ~5000 clients per worker process (given a client opens 2 WebSocket which are proxified, a client uses 4 connections)

  Note: It can be overriden wih `NGINX_WORKER_CONNECTIONS` env var

- displays in startup log the `worker_processes` and `worker_connections` configured and an estimation of the maximum number of connections given these figures

References:

- https://nginx.org/en/docs/ngx_core_module.html#worker_processes
- https://nginx.org/en/docs/ngx_core_module.html#worker_connections

Certificates and OCSP

By default OCSP is disabled. For OCSP to work the Intermediate Certificate must be correctly installed. Normally this should be ok if you did install the *fullchain* certificate as specified in the *SSL Certificates* config section.

Some certificate issuers dropped the compatibility with OCSP since 2025, nevertheless you can still reactivate OCSP check for the web proxy by adding the following in the `env` file:

```
OCSP_ENABLED=on
```

Rate limiting

Simplified explanation on how the nginx rate limit module works :

Rate limit does not apply to *Private IP Address* (RFC1918)

Rate limiting is configured in three zones and request limits are applied by client ip :

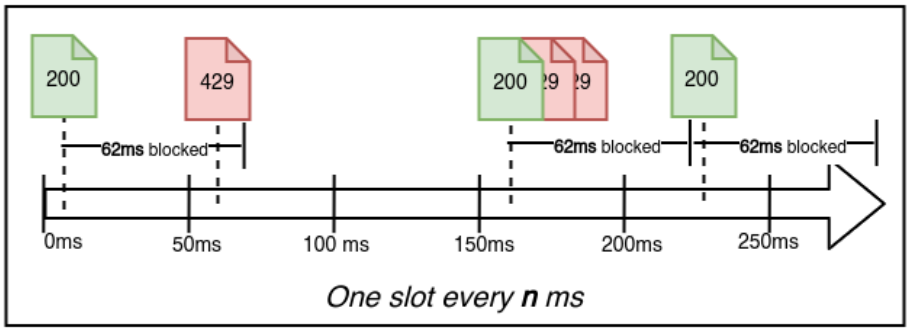
1. **webapp**
   - **30 req/s, burst of 20 req**
     - /ccagent
     - /ucassistant
     - /switchboard
   - **30 req/s, burst of 50 req (env variable), nodelay**
     - /assets
     - /config
     - /video/css
     - /video/images
     - /video/lang
     - /video/libs
     - /video/sounds
     - /pwa-worker.js
     - /static/offline.html
2. **apis**
   - **30 req/s, burst of 10 req**
     - /xuc
     - /version
   - **30 req/s, burst of 10 req, nodelay**
     - /video/external_api.js
3. **auth**
   - **20 req/m, burst of 12 req, nodelay**
     - /wssip

Rate limit

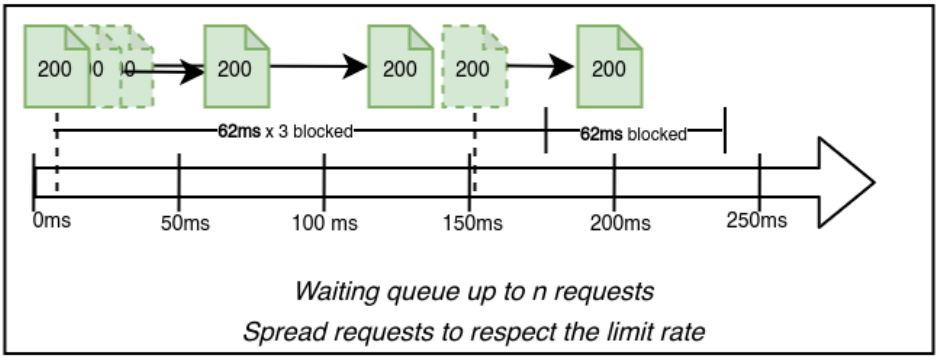
16 req/s = 1000ms / 16 = 1 req every **62ms** max

*It's just an **amount** of time*
*one slot for one request every **n** ms*

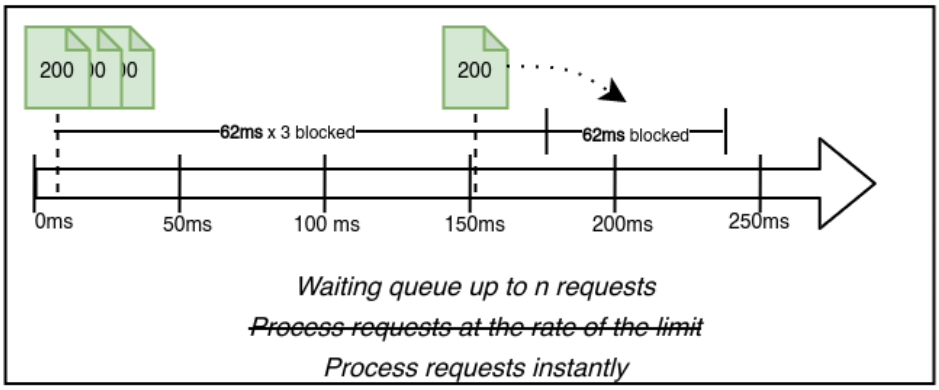
Rate limits applied to requests



Burst parameter



Burst parameter with nodelay (assets)



    - /xuc/api/2.0/auth/login
    - /xuc/api/2.0/cti
    - /invitation/video
    - /meetingrooms/token/validate/
4. **dapp**
    - **20 req/m, burst of 15 req, nodelay**
        - /install/win64
        - /updates/win64
        - /updates/debian
5. **video**
    - **20 req/s, burst of 12 req, nodelay**
        - /video/
        - /video/xmpp-websocket
        - /video/colibri-ws/

You can tweak the rate limit burst of the asset part by changing the value of the `ASSETS_BURST` variable.

Token format validation

The token format is checked with a regex. It checks the token contains 3 segments between 15 and 120 characters separated with dots.

TURN Server Features

Use TURN server on port 443

(*Since 2025.02*)

See *Expose TURNS server on port 443*

TURN Credentials

To be able to use the TURN server you must have valid credentials. Credentials are given by the xucserver to the WebRTC client.

These credentials are generated by the xucserver and validated/verified by the TURN server via a shared secret defined:

- in xuc via the `TURN_SERVER_SECRET` variable
- in TURN Server via the `static-auth-secret` parameter

By default only TURN is activated for both asterisk and WebRTC client. This can be changed via the `TURN_SERVER_ENABLE` and `STUN_SERVER_ENABLE` variables to be put in the `custome.env` of the XiVO CC

```
...
TURN_SERVER_ENABLE=false
STUN_SERVER_ENABLE=true
```

TURN Credentials lifetime

These credentials are valid for a default lifetime of **3600s**. The WebRTC client renew the credential every TTL/2.

The TTL can be changed via the `TURN_SERVER_CREDENTIAL_TTL` xuc environment variable. To change the credential TTL to 2hours add the following in the XiVOCC `custom.env` file:

```
...
TURN_SERVER_CREDENTIAL_TTL=7200
```

Optimize ICE (STUN/TURN) negotiation time

There are two places where the ICE mechanism can take some time:

1. Before emitting the call
2. When call is answered

1. Before emitting the call

When A wants to call B, A will start - before sending the SIP INVITE, by gathering what is called the "candidate". This is A's host IP addresses and also what the STUN/TURN server will give him.

The more host IP addresses A has, the longer the STUN/TURN gathering will take. Not to say that it also depends on the fact that these IP addresses can reach the STUN/TURN server.

- On the UC Application side and Mobile application the candidate gathering process is limited to **500 milliseconds**. See *XiVO CC configuration* for tweaking it.
- On asterisk side this can be enhanced by setting a stun and ice blacklist rule. This can be done by adding the following lines in a file in `/etc/asterisk/rtp.d/` directory (for example `/etc/asterisk/rtp.d/02-ice-optimization.conf`):

```
stun_deny = 0.0.0.0/0
stun_permit = 10.32.4.0/24

ice_deny = 0.0.0.0/0
ice_permit = 10.32.4.0/24
```

Where *10.32.4.0/24* is the network to which belongs the interface which can access the STUN/TURN server.

You need to reload rtp: *asterisk -rx 'module reload res_rtp_asterisk.so'*

Warning:

- Please read carefully : asterisk documentation https://github.com/asterisk/asterisk/blob/18/configs/samples/rtp.conf.sample
- Beware that a mis-configuration there can prevent your WebRTC user to make any call

2. When call is answered

If A calls B, when B answers, A and B will start to negotiate which RTP candidate they will use to communicate. The longer the candidate list for A and B the greater the number of possibility and connectivity check to do.

- On the UC Application side there is not much one can do except to shutdown some network interfaces. Besides, the candidate gathering process is limited to **500 milliseconds** and can be overriden to increase or reduce this timeframe. See *XiVO CC configuration*
- On asterisk side this can be enhanced by setting a stun and ice blacklist rule. This can be done by adding the following lines in a file in `/etc/asterisk/rtp.d/` directory (for example `/etc/asterisk/rtp.d/02-ice-optimization.conf`):

```
stun_deny = 0.0.0.0/0
stun_permit = 10.32.4.0/24

ice_deny = 0.0.0.0/0
ice_permit = 10.32.4.0/24
```

Where `10.32.4.0/24` is the network to which belongs the interface which can access the STUN/TURN server.

You need to reload rtp: `asterisk -rx 'module reload res_rtp_asterisk.so'`

Warning:

- Please read carefully : asterisk documentation https://github.com/asterisk/asterisk/blob/18/configs/samples/rtp.conf.sample
- Beware that a mis-configuration there can prevent your WebRTC user to make any call

6.5 Meeting Rooms

Meeting Rooms is a new feature introduced in the Helios LTS.

Important: This is an enterprise version feature that is not included in XiVO and is not freely available. To enable it, please contact XiVO team.

Besides these considerations, the project stay **Open Source** and is available here: https://gitlab.com/xivo.solutions/xivo-meetingrooms

It introduces Video Conference in XiVO.

6.5.1 Meeting Rooms Installation & Upgrade

- *Requirements*
  - *Server Requirements*
- *Base installation*
  - *1. Docker & Docker compose Installation*
  - *2. XiVO Meeting Rooms Launcher Setup*
  - *3. Next step*
- *Upgrade Meetingrooms*

Requirements

Important: The Meeting Rooms is an enterprise version feature. To enable it, please contact XiVO team.

Important: The Meeting Rooms components works only if you have the *XiVO Edge* component installed and configured.

Server Requirements

The Meeting Rooms components must be installed on a different server as the XiVO CC/UC server.

The minimal requirements would be:

- OS: **Debian 13** (Trixie), 64 bits
- CPU: 4 CPU
- RAM: 8 Gb

See: https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-scalable#machine-sizing

Base installation

1. Docker & Docker compose Installation

These commands will install docker and docker compose on the host.

```
# Install docker prerequisites
apt update
apt install apt-transport-https ca-certificates curl gpg

# Add Docker's official GPG key:
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.
↪asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF

cat > /etc/apt/preferences.d/docker-ce <<EOF
Package: docker-ce*
Pin: version 5:28*
Pin-Priority: 1000
EOF

cat > /etc/apt/preferences.d/containerd <<EOF
```

(continues on next page)

(continued from previous page)

```
Package: containerd.io*
Pin: version 2.*
Pin-Priority: 1000
EOF

apt update
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
↪plugin

#Install docker compose v2
apt update
apt install docker-compose-plugin

mkdir -p /etc/systemd/system/containerd.service.d
cat > /etc/systemd/system/containerd.service.d/override.conf <<EOF
[Service]
LimitNOFILE=100000
EOF

systemctl daemon-reload
systemctl restart containerd.service
```

2. XiVO Meeting Rooms Launcher Setup

1. Create `meetingrooms` directory:

```
mkdir -p /etc/docker/meetingrooms
```

2. Download *XiVO Meeting Rooms* configuration. In the following script replace `TAG_OR_BRANCH` by the name of a tag or a branch.

Note: currently to install the latest stable version of quantum, use `TAG_OR_BRANCH=2026.05.00`

```
TAG_OR_BRANCH=2026.05.00
cd /etc/docker/meetingrooms
wget "https://gitlab.com/xivo.solutions/xivo-meetingrooms/-/archive/${TAG_OR_
↪BRANCH}/${TAG_OR_BRANCH}.tar.gz"
tar -zxvf ${TAG_OR_BRANCH}.tar.gz -C /etc/docker/meetingrooms --strip-components␣
↪1
rm ${TAG_OR_BRANCH}.tar.gz
```

3. Create the jitsi configuration directory:

```
cd /etc/docker/meetingrooms
mkdir -p jitsi/{web/letsencrypt,transcripts,prosody/config,prosody/prosody-
↪plugins-custom,jicofo,jvb,jigasi,jibri}
```

4. Create bash alias to launch services:

```
echo -e "Current alias for meetingrooms-dcomp is:\n$(alias meetingrooms-dcomp)"
sed -i '/alias meetingrooms-dcomp=/d' ~/.bashrc
echo "alias meetingrooms-dcomp='docker compose -p meetingrooms -f /etc/docker/
↪meetingrooms/xivo-meetingrooms.yml --env-file=/etc/docker/meetingrooms/.env'" >
↪> ~/.bashrc
```

(continues on next page)

(continued from previous page)

```
source ~/.bashrc
echo -e "New alias for meetingrooms-dcomp is now:\n$(alias meetingrooms-dcomp)"
```

3. Next step

Congrats: You're done with the Meeting Rooms Installation.

- **Next step**: go to *Meeting Rooms Configuration*

Upgrade Meetingrooms

Currently there is no *automatic upgrade* process. Here is the manual process that you need to follow on the Meeting Room server.

- Make a backup of the Meeting Room launcher:

```
MR_BCK_DIR=$(mktemp -d -p /var/tmp/ meetingrooms-backup.XXXXXXXX)
cp -arT /etc/docker/meetingrooms/ ${MR_BCK_DIR}
echo "Backup was created in ${MR_BCK_DIR}"
```

- Re-install the Meeting Room Launcher (it will override the `.yml` files): follow the *2. XiVO Meeting Rooms Launcher Setup* steps to install new version

  Note: This step overrides the current `.yml` files. If you had made some customization in them you will have to backport them by comparing the new one with the backup you did at previous step. You can use this to show the diff between the newly installed yml files and the backup:

  ```
  for ymlfile in $(ls -1 /etc/docker/meetingrooms/*.yml);
    do diff -Zu "${MR_BCK_DIR}/$(basename ${ymlfile})" "${ymlfile}";
  done
  ```

- And then verify that the content of the `.env` file is correct:
  - Update the `.env` file using the script referenced in *Meeting Rooms Configuration* section.

    ```
    diff -Zu "${MR_BCK_DIR}/.env" "/etc/docker/meetingrooms/.env"
    ```

  - Verify that the `XIVOCC_TAG` and `XIVOCC_DIST` correspond to what you want to install (it should be *2026.05* and *latest*).
- Finally pull the new images and restart the containers:

  Warning: it will stop all Meeting Room calls (and disconnect users from the application).

```
docker login -u xivoxc
(use the token provided by the XiVO team)

meetingrooms-dcomp pull && docker logout
meetingrooms-dcomp up -d
```

6.5.2 Meeting Rooms Configuration

- *Meeting Rooms configuration*
  - *Create default env file*
  - *Start the services*
- *Edge Configuration*
- *XIVO CC Configuration*
  - *XiVO UC Embedded Configuration*
- *XiVO Configuration*
  - *Configure Jitsi properties in your custom env*
  - *Configure the Trunk to Jitsi*
  - *Customize proxy settings on XiVO*

Meeting Rooms configuration

Create default env file

1. **Launch the following commands** that will *generate* the `.env` file:

> **Warning:** Please **Copy** the **whole** code block below and **Paste** it in a terminal. It will (re)generate the `.env` file. Do not copy-paste only the `.env` content because it contains specific *here-document* syntax.

```
function generatePassword() {
    openssl rand -hex 16
}

function generateEnvFile() {

  cat > /etc/docker/meetingrooms/.env << EOF
# XiVO vars
XIVOCC_TAG=2026.05
XIVOCC_DIST=latest
XUC_HOST=${XUC_HOST}
MEETING_ROOMS_HOST_IP=${MEETING_ROOMS_HOST_IP}
XIVO_HOST=${XIVO_HOST}
#
# XiVO Edge vars
#
TURN_CREDENTIALS=${TURN_CREDENTIALS}
TURN_HOST=${TURN_HOST}
TURN_PORT=3478
TURNS_HOST=\${TURN_HOST}
TURNS_PORT=${TURNS_PORT}
#
# JWT Configuration
ENABLE_AUTH=1
ENABLE_GUESTS=0
AUTH_TYPE=jwt
```

(continues on next page)

(continued from previous page)

```
JWT_APP_ID=xivo
JWT_APP_SECRET=${JWT_APP_SECRET}
#
# Security
#
JICOFO_AUTH_PASSWORD=${JICOFO_AUTH_PASSWORD}
JVB_AUTH_PASSWORD=${JVB_AUTH_PASSWORD}
JIGASI_XMPP_PASSWORD=${JIGASI_XMPP_PASSWORD}
#
# Basic config
#
CONFIG=/etc/docker/meetingrooms/jitsi/
JVB_ADVERTISE_IPS=\${MEETING_ROOMS_HOST_IP}
HTTP_PORT=80
HTTPS_PORT=443
TZ=Europe/Paris
PUBLIC_URL=https://\${XUC_HOST}/video
XMPP_MODULES=muc_size
XMPP_MUC_MODULES=token_verification,token_affiliation
ENABLE_CLOSE_PAGE=true
ENABLE_END_CONFERENCE=false
#
# Basic Jigasi configuration options
#
JIGASI_SIP_URI=xivo-jitsi@\${XIVO_HOST}
JIGASI_SIP_PASSWORD=${JIGASI_SIP_PASSWORD}
JIGASI_SIP_SERVER=\${XIVO_HOST}
JIGASI_SIP_PORT=5060
JIGASI_SIP_TRANSPORT=UDP
#
# Advanced configuration options (you generally don't need to change these)
#
ENABLE_P2P=false
EOF
}

ENV_FILE="/etc/docker/meetingrooms/.env"

if [ -f "$ENV_FILE" ]; then
    source "$ENV_FILE"
else
    JICOFO_AUTH_PASSWORD=$(generatePassword)
    JVB_AUTH_PASSWORD=$(generatePassword)
    JIGASI_XMPP_PASSWORD=$(generatePassword)
    JIGASI_SIP_PASSWORD=$(generatePassword)
    XIVO_HOST="<FILL IN>"
    XUC_HOST="<FILL IN>"
    MEETING_ROOMS_HOST_IP="<FILL IN>"
    TURN_HOST="<FILL IN>"
    TURNS_PORT=3478
    TURN_CREDENTIALS="<FILL IN>"
    JWT_APP_SECRET="<FILL IN>"
fi

generateEnvFile
```

2. Then open the `/etc/docker/meetingrooms/.env` and fill in:

- `XUC_HOST` var with the same content as the XUC_HOST var in your **XiVO CC** configuration
- `MEETING_ROOMS_HOST_IP` var with the meeting rooms server IP address. It must be the meeting rooms server IP address accessible for your LAN client. It must be set otherwise nothing will work correctly.
- `XIVO_HOST` with the VoIP IP of the XiVO PBX It will be the IP towards which the Jitsi SIP gateway will register its SIP peer.
- `JWT_APP_SECRET` with the same content as the CONFIGMGT_AUTH_SECRET var defined in your **XiVO**'s /etc/docker/xivo/custom.env file
- Fill in also parameters linked to *XiVO Edge* (which is a prerequisite to use Meeting Room Server - see *Requirements*):
  - `TURN_HOST` var with the Edge STUN/TURN server FQDN (i.e. which must be a FQDN corresponding to the certificate configured on the Edge STUN/TURN server).
  - `TURN_CREDENTIALS` var with the value of the `TURN_SECRET` configure in the Edge TURN Server (the secret generated during *TURN Server Secret*).

Start the services

Start the services:

```
docker login -u xivoxc
(use the token provided by the XiVO team)

meetingrooms-dcomp pull && docker logout
meetingrooms-dcomp up -d
```

Edge Configuration

Note: These steps are to be done on the XiVO CC.

On the Edge server you must add the Meetingroom server IP address in the `TURN_ALLOWED_PEERS`.

Please refer to *TURN Server Relay Authorization* section.

XIVO CC Configuration

Note: These steps are to be done on the XiVO CC.

If you are using the *XiVO UC Embedded* mode you **must follow** the sub-section below: *XiVO UC Embedded Configuration*

On XiVO CC, to be able to use the video service you **must** configure the `NGINX_JITSI_HOST` variable.

1. Add the `NGINX_JITSI_HOST` value in the `/etc/docker/compose/custom.env` file:

   ```
   NGINX_JITSI_HOST=<IP Address of the Meeting Rooms server>
   ```

2. Relaunch the services

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

XiVO UC Embedded Configuration

Important: This section applies only if you are in the *XiVO UC Embedded* mode.

With XiVO UC applications, to be able to use the video service you **must** configure the `NGINX_JITSI_HOST` variable.

1. Add the `NGINX_JITSI_HOST` value in the `/etc/docker/xivo/custom.env` file:

```
NGINX_JITSI_HOST=<IP Address of the Meeting Rooms server>
```

2. Relaunch the services

```
xivo-dcomp up -d
```

XiVO Configuration

Note: These steps are to be done on the XiVO

If you are using the *XiVO UC Embedded* mode you must also follow the section *XiVO UC Embedded Configuration*

Configure Jitsi properties in your custom env

1. Edit `/etc/docker/xivo/custom.env` file :

```
MEETINGROOM_AUTH_DOMAIN=<The domain serving the service, i.e. "meet.jitsi" by␣
↪default, can be also set to "*">
MEETINGROOM_AUTH_APP_ID=<Value of the JWT_APP_ID defined in your jitsi .env file>
```

Configure the Trunk to Jitsi

On XiVO, go on page *Services* → *IPBX* → *Trunk management* → *SIP Protocol*, and create a SIP trunk with:

- Tab *General*:
  - *Name*: xivo-jitsi
  - *Username*: xivo-jitsi
  - *Password*: put the value of the `JIGASI_SIP_PASSWORD` variable in the `/etc/docker/meetingrooms/.env`
  - *Connection type*: Peer
  - *IP addressing type*: Dynamic
  - *Context*: default
  - *Media server*: MDS Main
  - *NAT*: Yes (force rport + comedia)
- Tab *Advanced*:
  - *Redirect media streams*: No

Then restart the jigasi docker on Meeting Room server:

```
meetingrooms-dcomp restart jigasi
```

Customize proxy settings on XiVO

Important: You need to do this only if XIVO is behind a proxy.

If XiVO PBX is behind a proxy (and you followed the *Proxy Configuration* guide) then you **must** tweak the proxy setting for agid container to add the Meetingroom server IP address in the NO_PROXY list.

See *Proxy Configuration for AGID*

6.5.3 Features

- *Overview*
- *Limitations*
- *Customization*
  - *Configure TURNS server to use port 443*
  - *Enable moderator for external participants*
  - *Enable lobby for external participants*
  - *Disable audio for meetingrooms*

Overview

- Open video from UC Assistant, CC Agent, Switchboard
- Call a user with video support
- Invite another internal user in the conference you're currently attending
- Link to access meetingroom for external participant
- Create a static meeting room (via *XiVO PBX* webi)
- Create a personal meeting room from the assistant (*UC Assistant* only)
- Join a meeting room via a phone
- Join a meeting room via incoming call
- Join a meeting room from an incoming call through an IVR to choose which MR to join
- Get instant system notifications when being invited to a video conference

Limitations

- Personal meeting room creation can only be done via the *UC Assistant* (it is not available for *CC Agent* or *Switchboard*)
- There is no history of the video calls/conference (no history of video conference you joined nor video conference invitation you received/missed)
- There must be at least one video participant for the meeting room to work. If only audio participants are in the conference they won't be able to speak to each other.
- *Invite to conference* / *Start a video call*:
  - these features currently works only if the chat is enabled
  - the invite to conference invitation has a timeout set to 15s which is not configurable
  - if PIN code is modified while you are already in one of your personal meeting room, you can't invite other people until you leave then rejoin it
  - CC Agent / Switchboard related notes:
    - these features do not work for a "roaming agent": i.e. an agent that is logged on another user's line (it applies to CC Agent or Switchboard application)
    - when an agent is in a meetingroom, he still receives queue calls. He has to put himself on pause manually to prevent it.

Customization

Configure TURNS server to use port 443

By default the Meetingrooms is configured to provide its clients a TURN server configuration to be requested on port 3478. In some environement this port may not be accessible: in this case it may prevent the Meetingroom (audio and/or video) to work properly.

If - **and only if** - Edge was configured to expose TURNS on port 443 - see *Expose TURNS server on port 443* - then Meetingrooms configuration can be changed to use it.

Important: Pre-requisite: configure *Expose TURNS server on port 443* on Edge server.

1. **Edit** the `/etc/docker/meetingrooms/.env` and change:
   - `TURN_HOST` by the TURN FQDN (same as `TURN_FQDN` env var in Edge configuration)
   - `TURNS_PORT` to **443**
   - for example:

```
#
# XiVO Edge vars
#
TURN_CREDENTIALS=4afe456ebdc0fe47ffb5f9a822b8404
TURN_HOST=edge-turn.mycorp.com # same as TURN_FQDN in Edge␣
↪configuration
TURN_PORT=3478
TURNS_HOST=${TURN_HOST}
TURNS_PORT=443
```

2. Relaunch the services:

```
meetingrooms-dcomp up -d
```

Enable moderator for external participants

By default users joining the Meetingroom *via a sharing link* join as **Guest** (this is a new behavior since Quantum.02).

This can be changed by editing the Meetingroom configuration as follows:

1. Edit `/etc/docker/meetingrooms/.env` and set in a new section at the bottom (if not already defined) which resets `XMPP_MUC_MODULES` var:

```
#
# Customized config
#

XMPP_MUC_MODULES=
```

2. Relaunch the services:

```
meetingrooms-dcomp up -d
```

Enable lobby for external participants

It is possible to configure Meetingrooms with "lobby rooms" for guests. With this setup:

- users joining the Meetingroom *via an sharing link* join as Guest **and wait in a *Lobby*** room till a Moderator accepts them
- internal users joining a Meetingroom *via their Assistant application* (UC Assistant, CC Agent) join the Meetingroom as *Moderator*

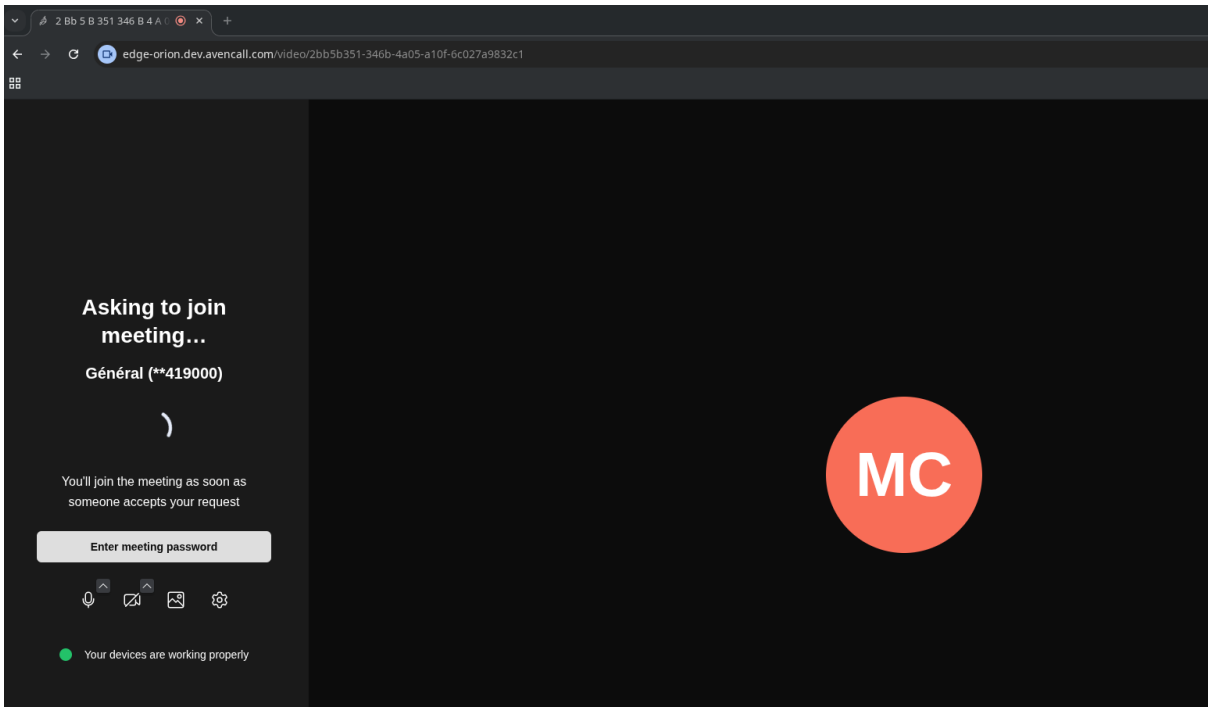


Fig. 1: User joining via sharing link wait in a Lobby room till they are accepted by a Moderator

To activate lobby rooms for guests, do the following:

1. Edit `/etc/docker/meetingrooms/.env` and set in a new section at the bottom (if not already defined):

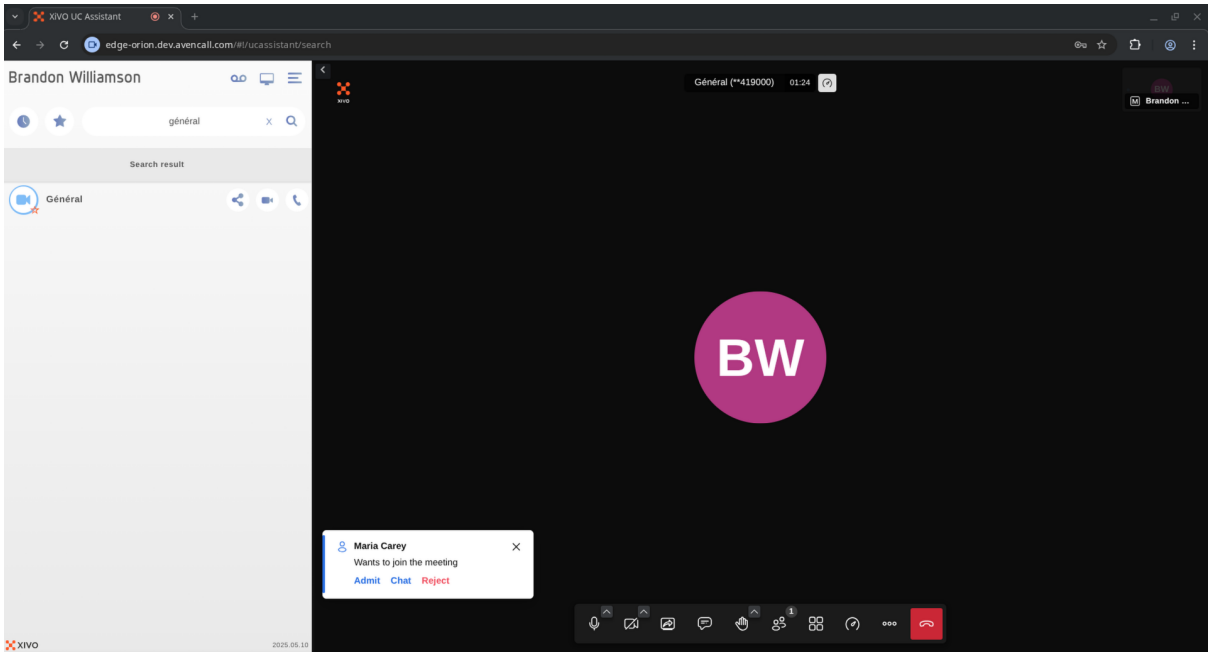


Fig. 2: With lobby rooms activated, external users must be allowed to the meetingroom by a Moderator

```
#
# Customized config
#

XMPP_MODULES=muc_size,persistent_lobby
XMPP_MUC_MODULES=token_verification,token_affiliation,token_lobby_bypass,lobby_
↪autostart
```

2. Relaunch the services:

```
meetingrooms-dcomp up -d
```

Disable audio for meetingrooms

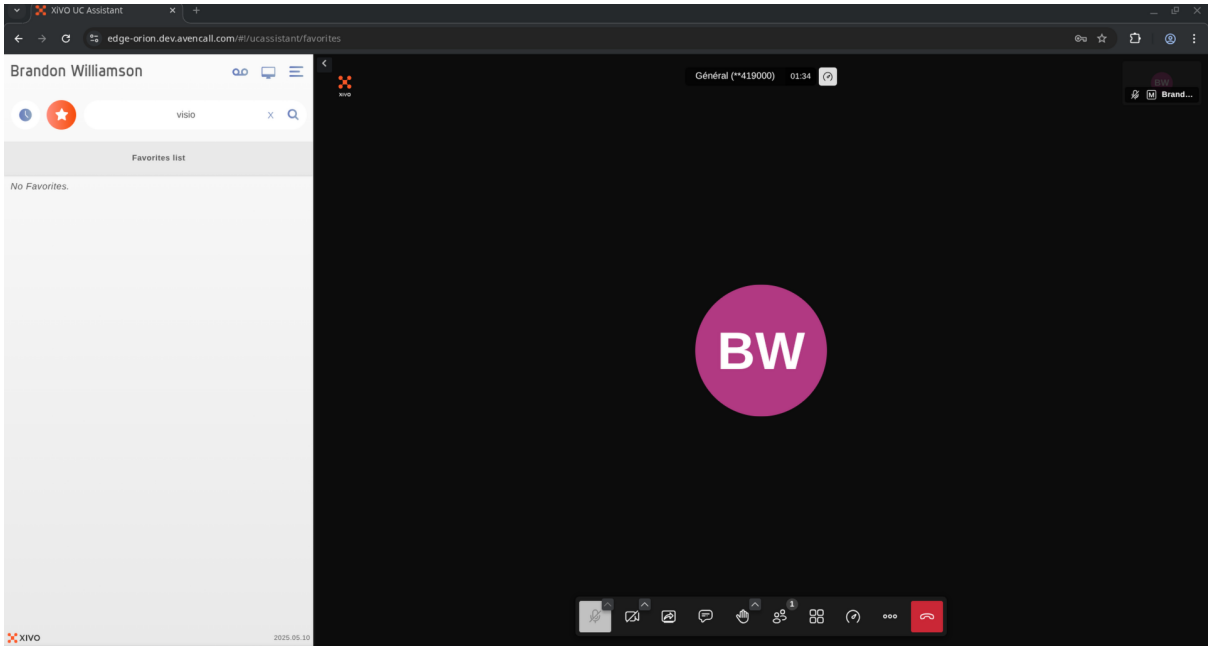
Meetingrooms can be used without audio. To do so globally:

1. Edit `/etc/docker/meetingrooms/.env` and set in a new section at the bottom (if not already defined):

```
#
# Customized config
#

START_SILENT=true
```

2. Relaunch the services:

```
meetingrooms-dcomp up -d
```



6.5.4 Users' Guide

Join Meeting Room from Assistant

To join a meeting room, you need to look for it from your Assistant applications (UC Assistant, CC Agent, Switchboard).

By typing part of its name it will display:

- your matching personal meeting rooms
- and the matching static meeting rooms.

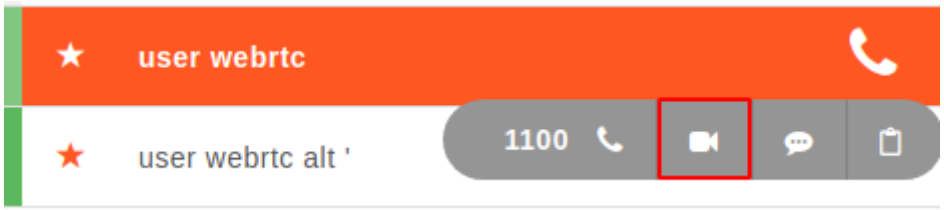
Then you can click on the call action and on the join action to join the meeting room.

Call Internal Users to join Meeting Room

You can call a internal user with video support. It will create a temporary meeting room for this call.

To do so:

- find user in assistant
- in the call action, click on the *video call* action
- you automatically enters temporary meeting room
- the other user receives an invitation:
  - when he accepts he will be joined into the call
  - if he rejects or miss the invitation you will be notified



Invite Internal Users to ongoing Meeting Room

When you are in a meeting room you can invite another internal user in it.

To do so:

- join a meetingroom (type the PIN if needed)
- then search in the assistant the user you want to invite
- in the call action, click on the *invite to meetingroom* action
- the other user receives an invitation:
  - when he accepts he will be joined into the conference
  - if he rejects or miss the invitation you will be notified

Note: If a user accepts an invitation to a meeting room while being currently on call, the call will be automatically put on hold.

Join Meeting Room for External Participants

It's possible to share a link to people that doesn't use the XiVO assistant. Thanks to *XiVO Edge* it is a public link.

To do so, search for a meeting room in your then click on share icon. The link will be copied in your clipboard.

Join Meeting Room from Phone

You can also access a Meeting Room with a phone by dialing `**MEETING_ROOM_NUMBER` where `MEETING_ROOM_NUMBER` is the Meeting Room number. You can only access Meeting Room which are configured the XiVO PBX with a number.

Given the XiVO PBX admin user created a Meeting Room:

- *Name*: ProjectManagerConf
- *Display Name*: Project Manager Conf
- *Number*: 5000

Then someone can join the conference via its phone set by dialing: `**5000`

Create a personal Meeting room

You can create a personal meeting room from the UCAssistant menu, with the name and pincode of your choice.

Other users can join a personal meeting room in a browser with an invitation link that can be found by the creator of the room, when clicking on his meetingroom in the search results or favorites (share button).

Search Meeting Room

When you search something in your Assistant (UC Assistant/CC Agent/Switchboard) the result will contain the matching meeting room.

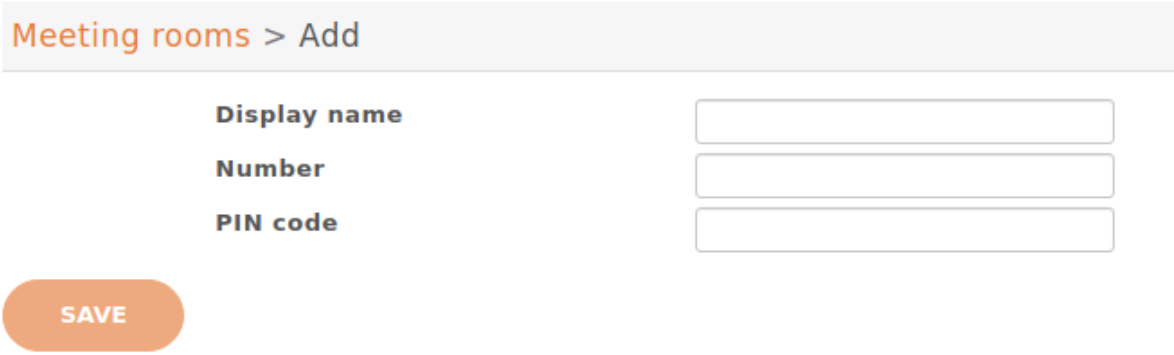
There is a configurable keyword that allows you to list all available rooms. By default it is `conference` and `visio`.

Note: For this to work, CTI Display Filter configuration must match the *xivo_meetingroom plugin configuration*.

6.5.5 Admin's Guide

Create a static Meeting Room

In order to create a static meeting room in XiVO (meaning available for all the users of the XIVO) go to *Services → IPBX → IPBX Settings → Meeting rooms*



Create an incoming calls for Meeting Rooms

In order to use built-in IVR for Meeting Rooms in XiVO PBX, create an incoming call that will be available for remote users to join MR externally go to *Services → IPBX → IPBX Settings → Call Management → Incoming Calls*

- *DID*: your incoming call extension
- *Context*: your incoming calls context (default is from-extern)
- *Destination*: Customized
- *Command*: Goto(meetingrooms-audio-ivr,s,1)

6.6 Mobile Application

Important: This is an enterprise version feature that is not included in XiVO community and therefore not freely available due to PUSH mecanism to be set in place to wake up mobile phones. To enable it, please contact XiVO team.

Besides these considerations, the project stay **Open Source** and is available here: https://gitlab.com/xivo.solutions/xivo-mobile

Warning: Mobile VoIP is not suitable for critical or emergency calls.

Mobile operating systems, public network instability, and reliance on third-party notification services prevent the level of guaranteed reliability and Quality of Service (QoS) required for mission-critical communications. Dedicated telephony/softphony should be used for these purposes.

Contents:

- *Release Notes*
  - *2.6.6 (08/05/2026)*
  - *2.6.5 (26/04/2026)*
  - *2.6.4 (5/10/2025)*
  - *2.5.136 (26/06/2025)*
  - *2.5.135 (22/02/2025)*
  - *2.5.134 (01/02/2025)*
- *Requirements*
  - *Limitations*
  - *Compatibility*
  - *Case of unlisted terminals*
- *Push Notifications*
  - *Some use cases*
- *Troubleshooting*
  - *Error Codes*
  - *No internet access*
  - *Sending logs*
  - *Debugging*
  - *Required permissions*
  - *Xiaomi stop battery optimization*
  - *Xiaomi authorizations*
  - *Xiaomi phone account*
- *Configuration*
  - *Type of Ringtone*
  - *Waiting While Mobile Application Is Waking Up*

6.6.1 Release Notes

2.6.6 (08/05/2026)

- Enhance display for high-end devices with a better use of the screen space
- Change log format to ISO 8601 format
- Retry mechanism for SIP Websocket when having network trouubles

2.6.5 (26/04/2026)

- SIP Register missing after error cases where line configuration is lost
- Contacts results are now ordered properly and not lexically anymore

Android

- Update to support Android 15 (SDK v35)

iOS

- Fix call hangup after 2 minutes when app is in background and screen locked

2.6.4 (5/10/2025)

- Introduces compatibility with Unique Account (with XiVO version >= Pollux)
- The arrow icon to indicate that forward is actived is now next to the internal number instead of the device management icon.
- Logging out from the Mobile Application now dissociates it from the user.

2.5.136 (26/06/2025)

- Fix the display of internal contacts (works only while giving full access permissions to the app)

2.5.135 (22/02/2025)

- Fix hanguped call after 90s when app were in background
- Avoid stopping SIP stack after a call is ended
- Remove deffered call mechanism that was sometimes launching unwanted calls
- Display server configuration in hamburger menu

iOS

- Reconnect properly to websocket after app was killed
- Fix lack of sound on some outgoing calls

2.5.134 (01/02/2025)

- New logging system to better identify issues
- Visualize that you are properly registered on the main screen
- Add a cancel button on login page
- Enhance resilience of call management when call is hanguped quickly

Android

- Update to support Android 14 (SDK v34)
- Fix automatic reconnection after app loosing focus and get it back

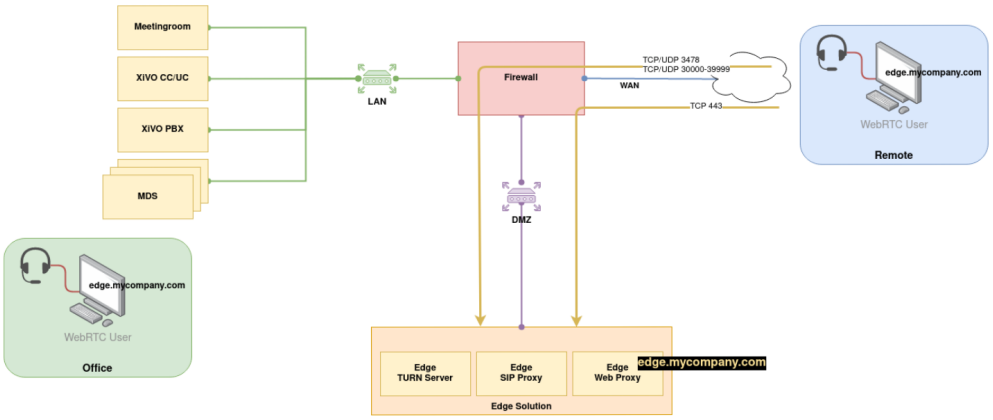
iOS

- Taking back a call after holding it no more loose the sound

6.6.2 Requirements

- Minimum XiVO version : **supported LTS version**

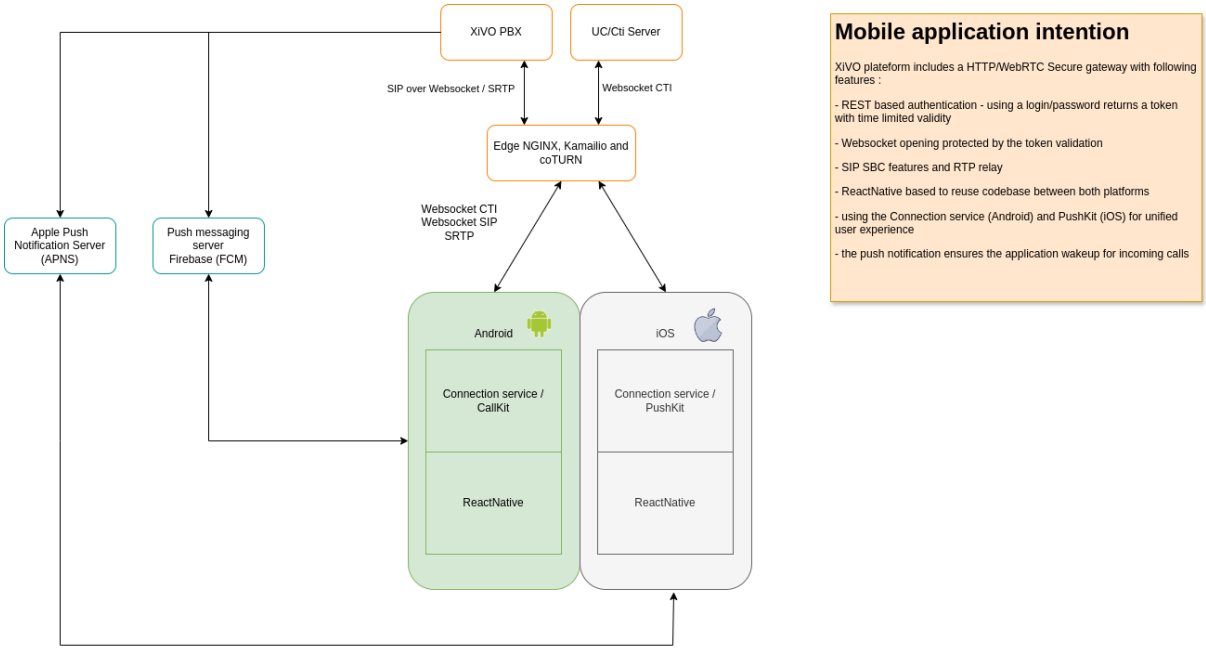
The XiVO mobile application should be used with at least the Jabbah version of XiVO. The XiVO must contain an EGDE infrastructure allowing teleworkers or travelling workers to be able to access XiVO from outside. Follow the official documentation on required ports - see *Network Flows*



When the user receives an incoming call, we use Firebase to wake up the mobile application by sending a push notification for Android devices and uses APNS (Apple Push Notification Server) for iOS devices.

The official firebase documentation describes the required ports and firewall configuration. See https://firebase.google.com/docs/cloud-messaging/concept-options#messaging-ports-and-your-firewall

Apple documentation describes the required ports and firewall configuration. https://support.apple.com/en-la/102266



Limitations

- No Video call
- No agent account
- No OIDC, CAS or Kerberos authentication, only basic or LDAP

Compatibility

Android

The tests are performed from Android 12. Even if compability is set Android from 11, previous versions are not supported. The application lifecycle is following Android 3 years LTS support Minimal phone requirements to find the application on store are:

- Shipped after 2020
- RAM >= 2Gb

iOS

The tests are performed from devices compatible with at least iOS 13. Previous versions are not supported.

Case of unlisted terminals

> **Warning:** Manufacturer's software layer may create unattended behavior on unsupported devices. Please open a ticket to describe device - with OS version and the bug.

It is imperative to make preliminary tests on the mobiles which would not be compatible. Be aware that some manufacturers are not keen with push notifications, and can cause issues for receiving VOIP calls. You can find more information about it here

If you encounter problems, you can send an email to the support directly via the mobile application. It is however important to have configured on your mobile an application to send emails. The logs of the mobile application are sent in an attached file in this email.

You can also send logs to your email directly for own debugging,

6.6.3 Push Notifications

Push notification is a technology used in the XiVO server and the mobile application.

Since the Izar version, you can ring multiple devices at the same time. When you choose to ring the mobile application and the UCAssistant, the system will know when the devices are both ready to receive the incoming call.

As the UC assistant is connected directly to the server via the web interface, this will be relatively quick.

For the mobile application, the operating systems (Android and IOS) are using a "DOZE" mode. This mode closes the application when it has not been used for some time, to reduce the consumption of the battery. To wake up the application, we use push notification that goes through a public push server either from Google or Apple to wake up the XiVO mobile application.

You might need to wait a certain time after you launched the call for the mobile application to wake up. This time can vary from one to ten seconds, depending on the push servers and the network quality. Once the mobile application responds, we start ringing the available devices.

Some use cases

Ringing device selected:

- *UC Assistant*: Fast ringing on the UC direct connection to the web browser or desktop
- *Mobile application*: Launching the ringtone on the mobile application when it will be awakened. The mode of operation used is the push notification. There can be a waiting time.
- *UC + Mobile Application*: There is a delay before launching the ringtone on both devices, the time for the mobile application to wake up.

6.6.4 Troubleshooting

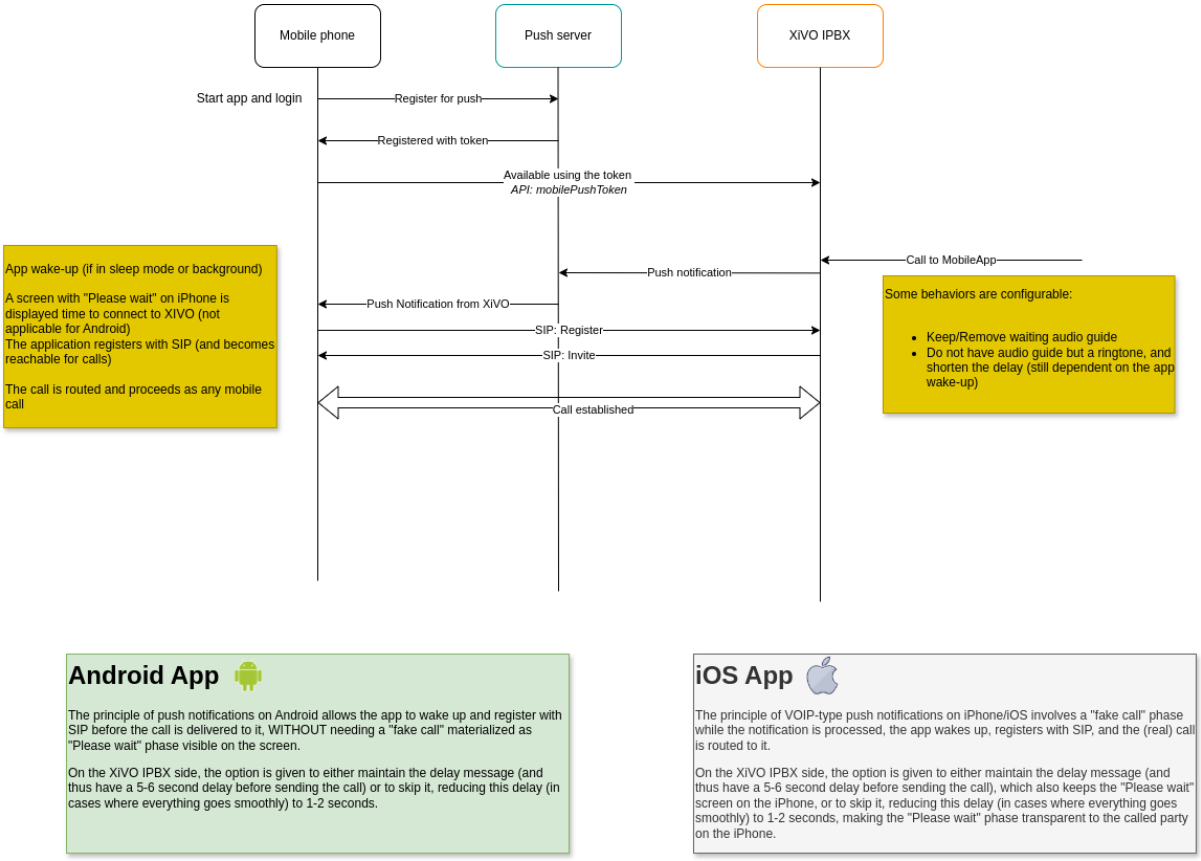
Error Codes

Error codes will be displayed in case of a problem on the application while trying to make an outgoing call :

- Error code 1:

This error code means the application is not receiving or sending the SIP signal, and that the user is not REGISTER in the asterisk.

- Error code 2:



You will get this error code if the application is able to receive or send the SIP signal, but the user is not REGISTER in the asterisk.

No internet access

If the user looses access to the Internet, the application switches to a non-usable mode until access is regained (see screenshot below). The connection status display is materialized by a greeen circle when there is an internet access, grey otherwise. If a call is done meanwhile, it will be dialed once internet connection is back.

Sending logs

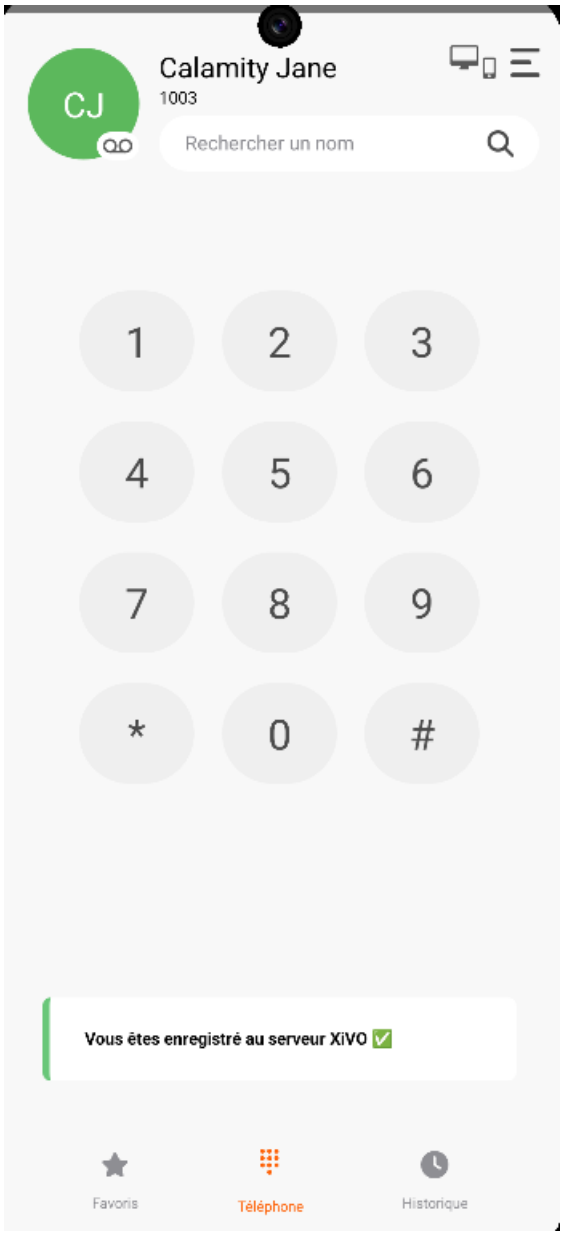
If you encounter an unsual behaviour and you manage to reproduce it on your application, you can send the logs directly via the configuration and click on *Send logs to support*.

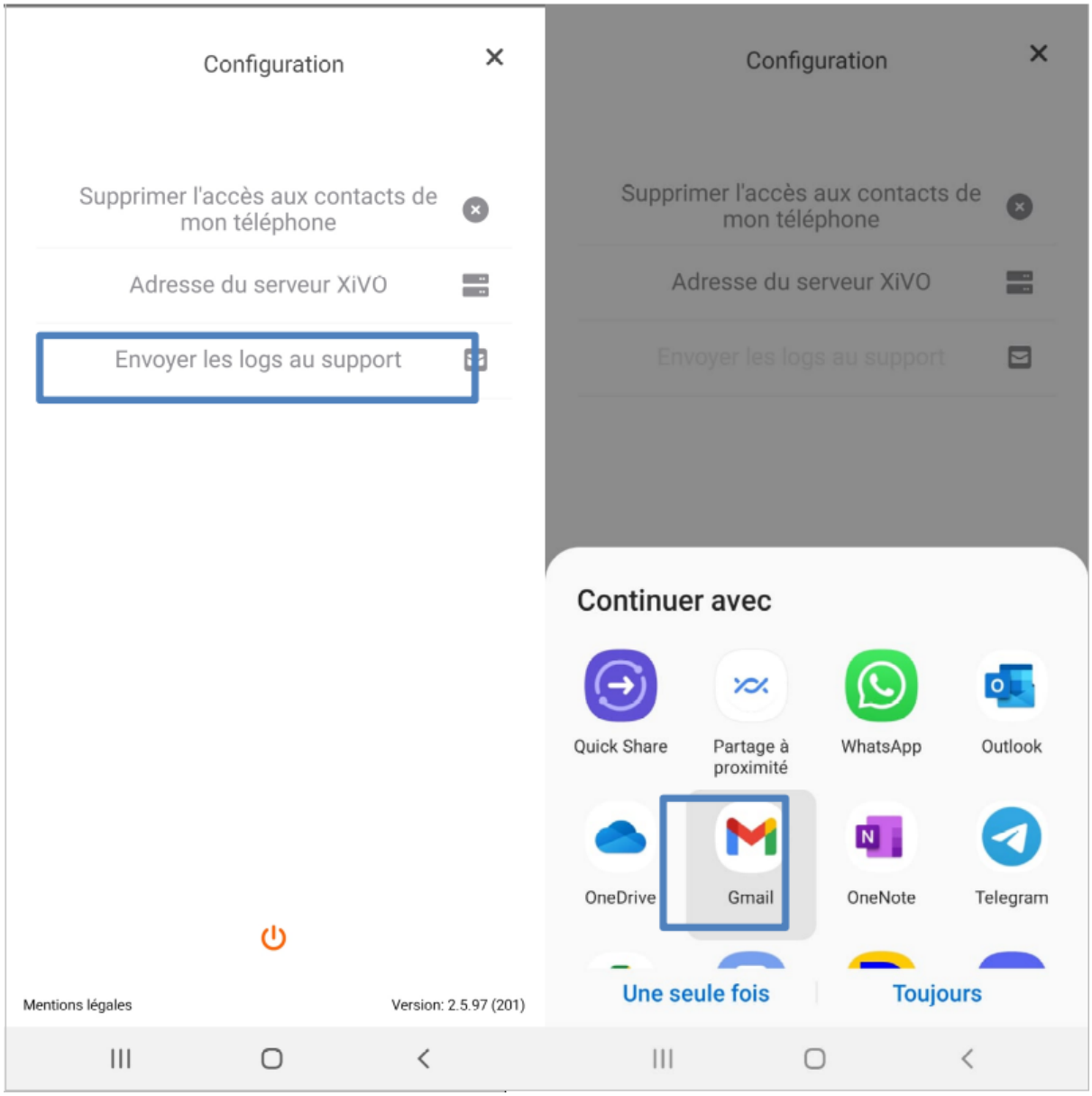
Warning: The logs are sent by email, you must therefore have a messaging system configured on your phone. You can ovveride the email to send it to yourself

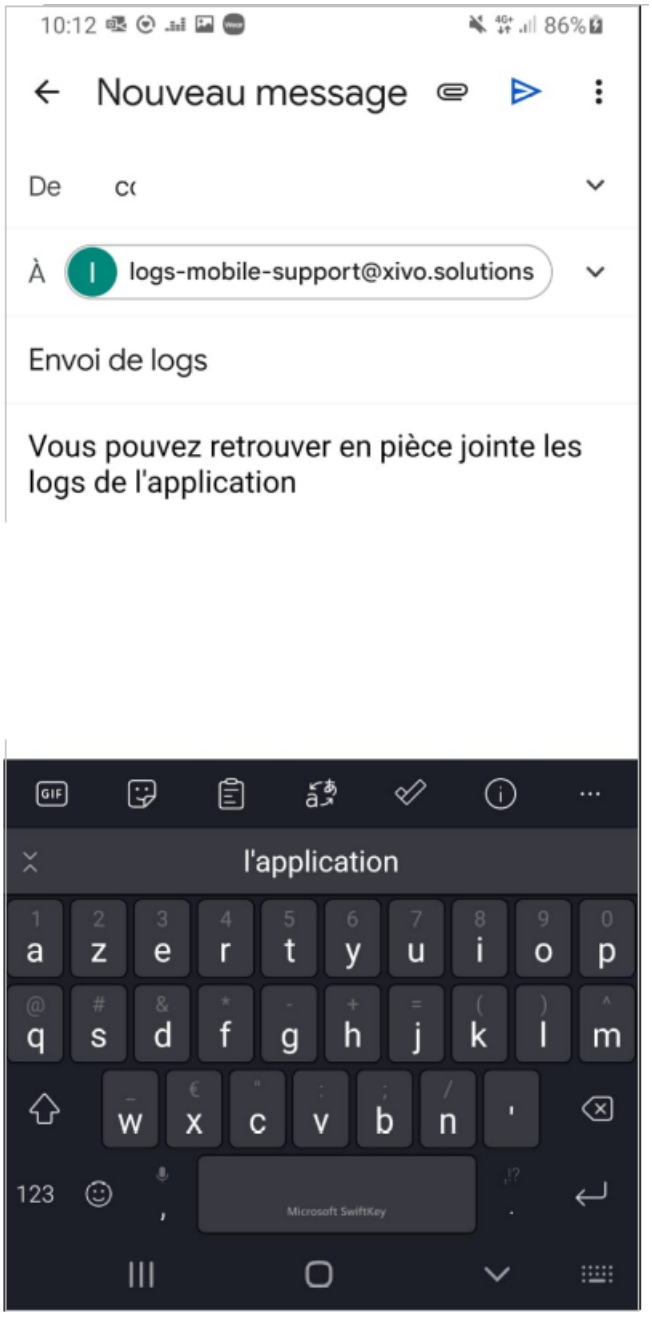
Understanding logs:

All business code is part of module, some modules have been pinpointed to focus on some app behaviour such as:

- *[Permission]* Related to permission granted by the user to the app
- *[Login]* Login process and token renewal
- *[Notification]* Local Notification for missed call and also remote push management registration/wake up
- *[System]* Low level foreground process to keep the app awake when receiving VOIP calls
- *[WebSockets]* Websockets lifecyle for both Xuc (CTI) and Asterisk (SIP)
- *[Webrtc]* actions / events related to the protocol


CJ
Calamity Jane
1003
Rechercher un nom
1
2
3
4
5
6
7
8
9
*
0
#
Vous êtes enregistré au serveur XiVO ✅
Favoris
Téléphone
Historique


Configuration
Supprimer l'accès aux contacts de mon téléphone
Adresse du serveur XiVO
Envoyer les logs au support
Mentions légales
Version: 2.5.97 (201)
Configuration
Supprimer l'accès aux contacts de mon téléphone
Adresse du serveur XiVO
Envoyer les logs au support
Continuer avec
Quick Share
Partage à proximité
WhatsApp
Outlook
OneDrive
Gmail
OneNote
Telegram
Une seule fois
Toujours


10:12
86%
Nouveau message
De
À
logs-mobile-support@xivo.solutions
Envoi de logs
Vous pouvez retrouver en pièce jointe les logs de l'application
l'application

- *[Audio]* Ringing tones and audio routes

Debugging

First, you have to look for AGI response on the Asterisk cli to confirm that the mobile push token is successfully executed. To do so :

- Check asterisk full logs on XiVO: `less /var/asterisk/full`
- search for the destination number of the call: `/DTSNUM=<NUMBER_CALLED>`
- look at the logs at the time the call was made and note the channel_id exemple: [C-XXXXX]
- Search the channel_id in the logs `/C-XXXXX`, then you can follow the process described before more easily.
- Check that the token has been pushed : `less /var/asterisk/full | grep webpush | grep C-XXXXX`

In edge-Kamailio logs, you can see the `INVITE` processing and the `Location` (mobileApp and/or WebApp): `edge-dcomp logs -tf kamailio | grep LOCATION | grep <sip_call_id>`

In mobile app logs, you will find token, the date and time when it's received, expired and renewed. Note that in mobile app logs the time is UTC based

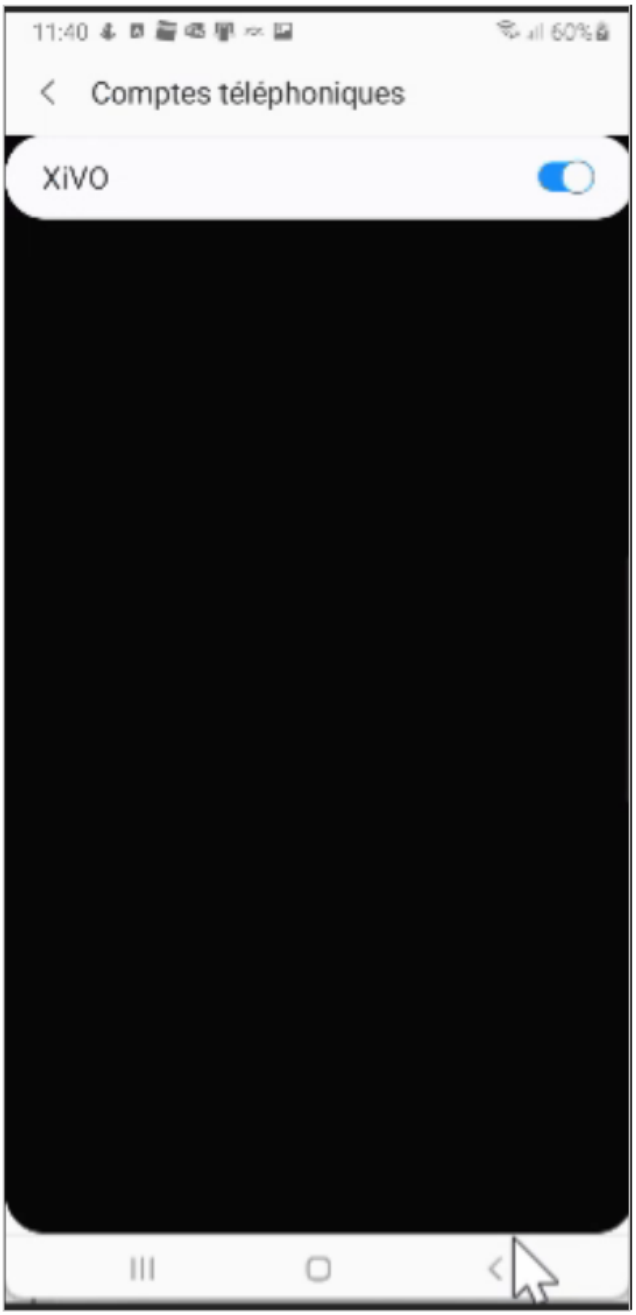
Required permissions

If you didn't give all the authorizations needed to use the mobile app, this error message will be displayed the next time you open the application. This means the XiVO phone account is not active.

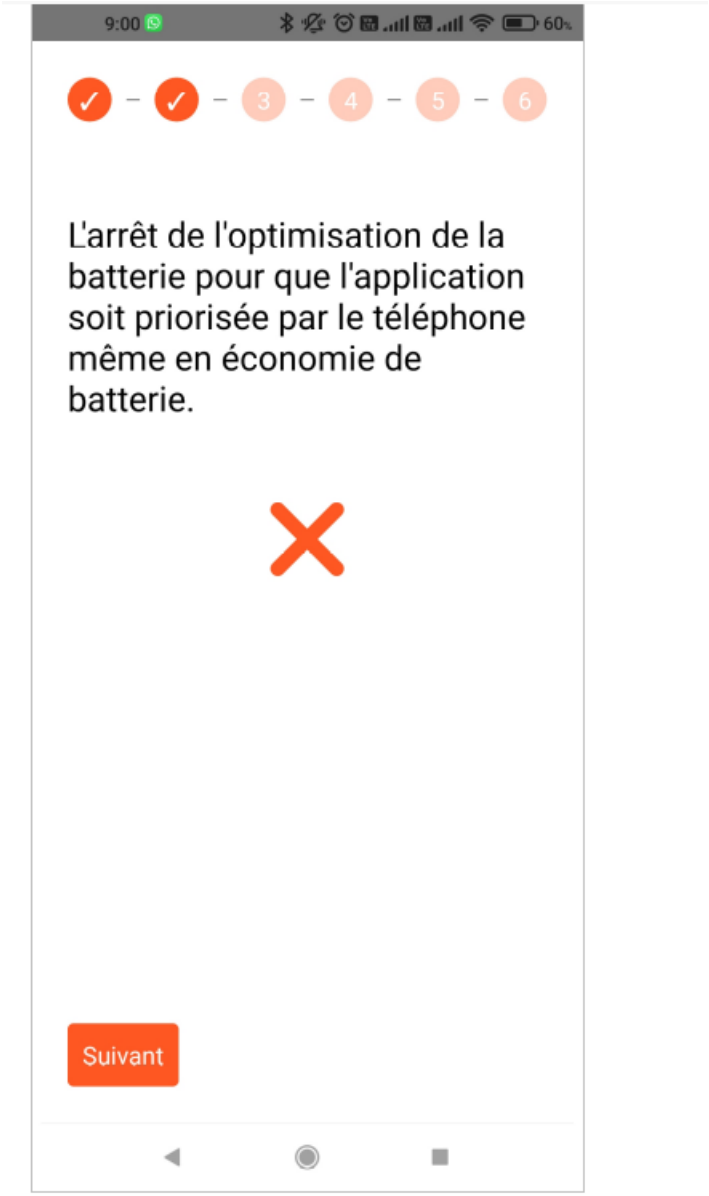
In this case, you should click the *OK* button and will be redirected to the settings.

If the phone is a "Samsung", the *phone accounts* settings will be opened. You just have to activate the check mark next to XiVO.

If the phone is *not* a Samsung, you will be redirected to the wrong settings. You will need to search in your phone settings where to activate the XiVO phone account.


11:40
60%
Comptes téléphoniques
XiVO

Xiaomi stop battery optimization

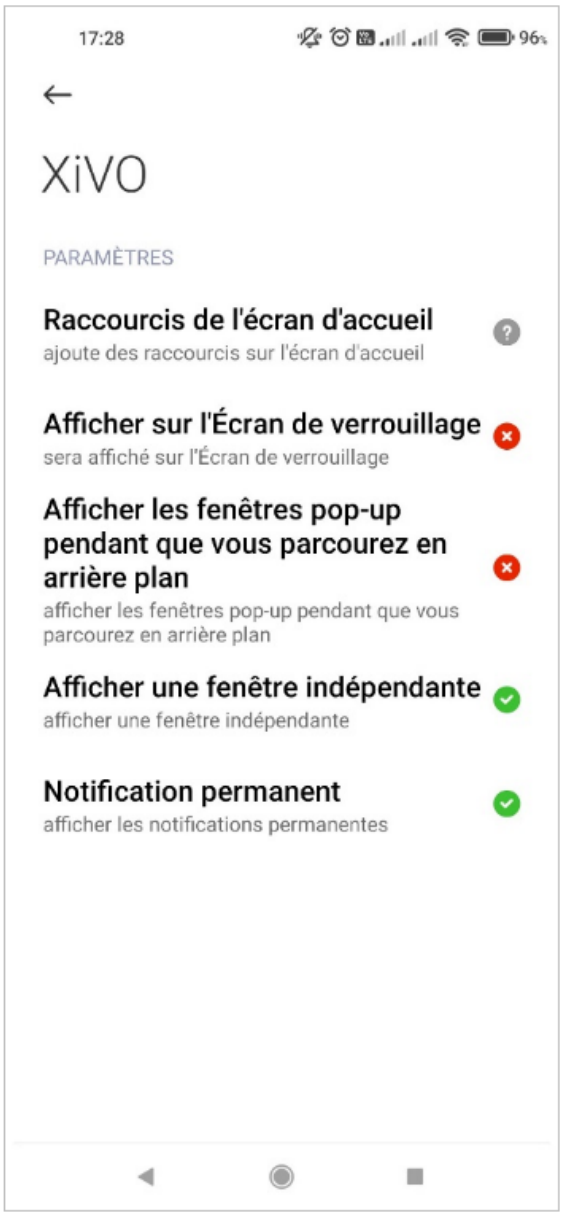


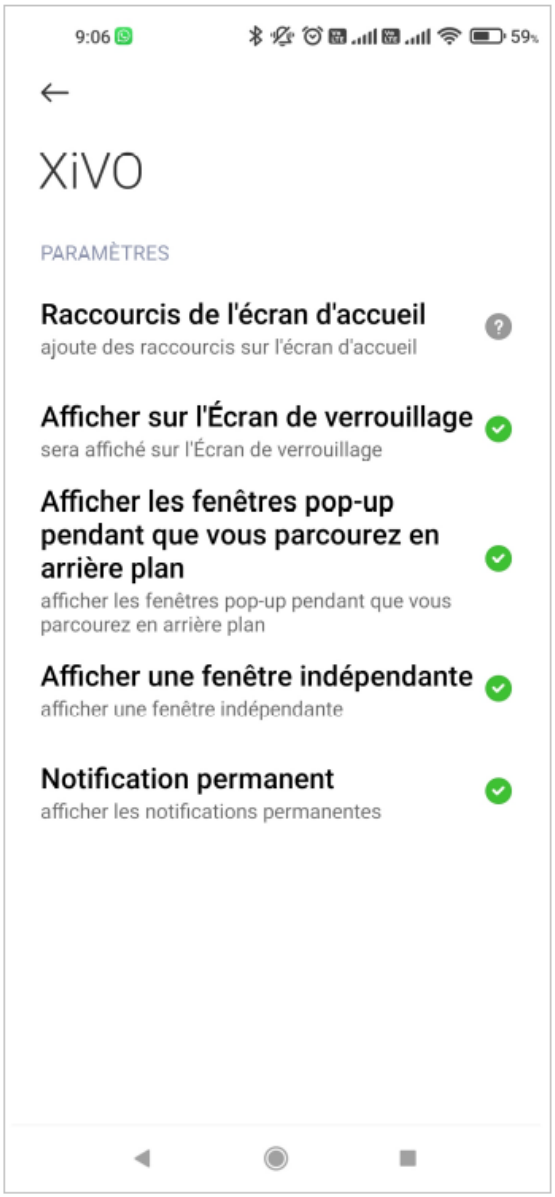
Xiaomi authorizations

In the settings of the Xivo application check if these settings are active:

Show on lock screen Show pop-up windows while browsing in the background

If you are in the above context, you may not receive calls when your phone is locked.

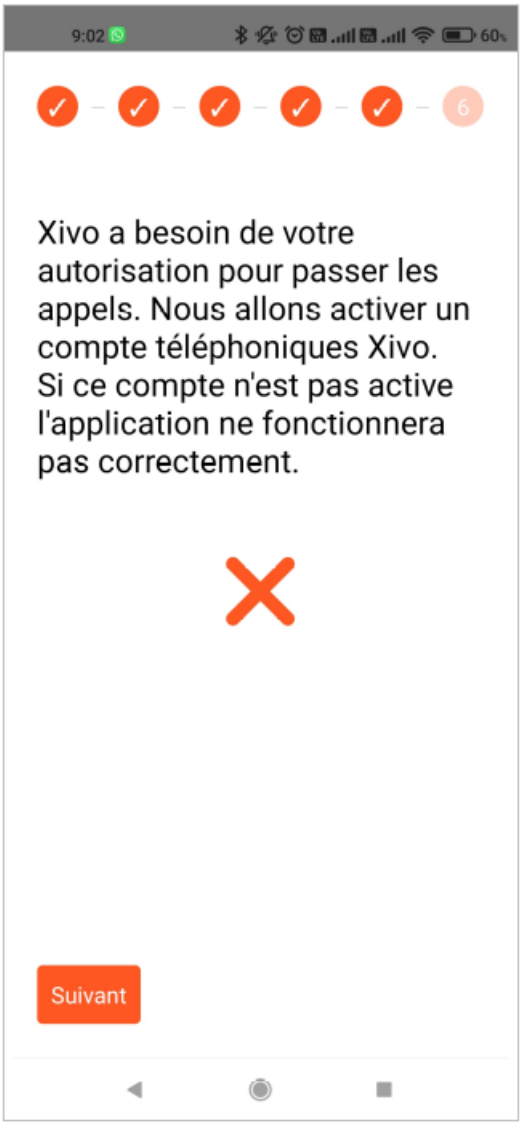

17:28
96%
XiVO
PARAMÈTRES
Raccourcis de l'écran d'accueil
ajoute des raccourcis sur l'écran d'accueil
Afficher sur l'Écran de verrouillage
sera affiché sur l'Écran de verrouillage
Afficher les fenêtres pop-up pendant que vous parcourez en arrière plan
afficher les fenêtres pop-up pendant que vous parcourez en arrière plan
Afficher une fenêtre indépendante
afficher une fenêtre indépendante
Notification permanent
afficher les notifications permanentes

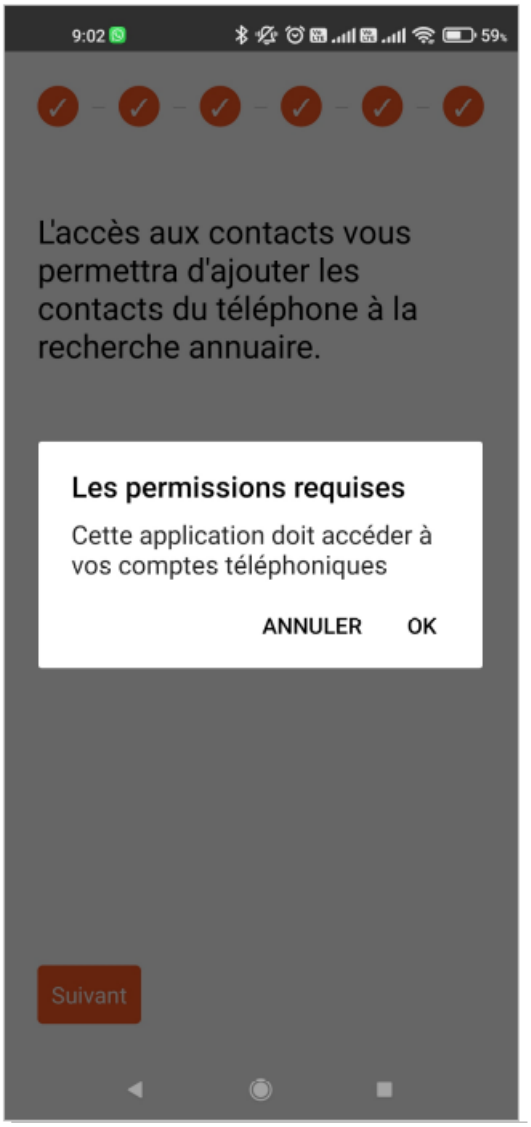

9:06
59%
XiVO
PARAMÈTRES
Raccourcis de l'écran d'accueil
ajoute des raccourcis sur l'écran d'accueil
Afficher sur l'Écran de verrouillage
sera affiché sur l'Écran de verrouillage
Afficher les fenêtres pop-up pendant que vous parcourez en arrière plan
afficher les fenêtres pop-up pendant que vous parcourez en arrière plan
Afficher une fenêtre indépendante
afficher une fenêtre indépendante
Notification permanent
afficher les notifications permanentes

Xiaomi phone account

When activating the XiVO phone account (step 5), you need to :

- Click *Ok* on the window *Les permissions requises*,
- go to *paramètres d'appel* => *Comptes des appels*,
- then in *comptes téléphoniques* you need to activate XiVO.



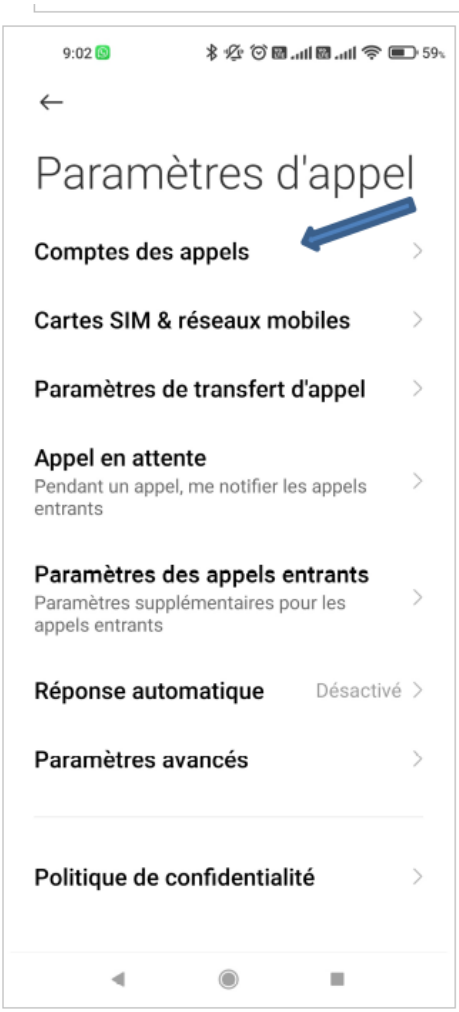


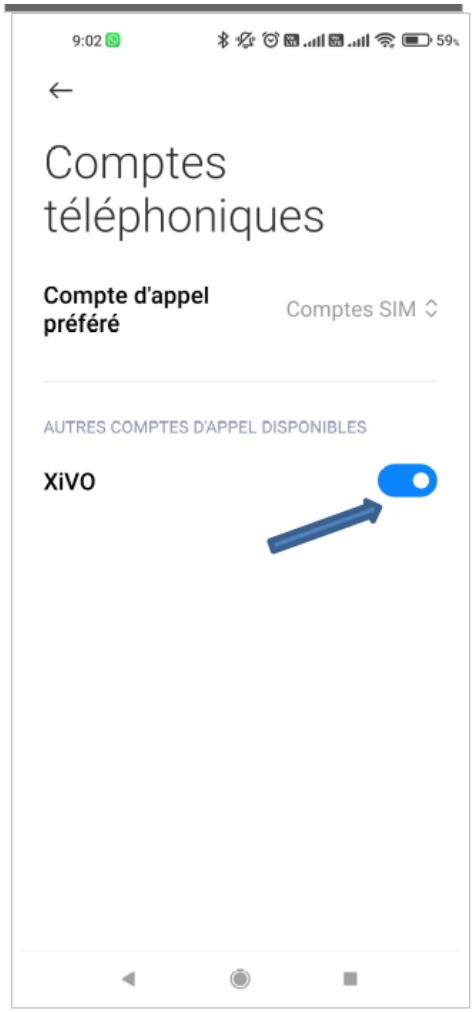
6.6.5 Configuration

Type of Ringtone

You can change the value of *XIVO_MAPP_WAIT_WITH_MUSIC* in the file `/etc/xivo/asterisk/xivo_globals.conf` to customize the ringtone behavior. If the value is set to *True*, you will hear the default music on hold while attempting to reach to a user with a mobile application. Otherwise, you will hear a regular ringtone.

```
XIVO_MAPP_WAIT_WITH_MUSIC = "True"
```


9:02
59%
Paramètres d'appel
Comptes des appels
Cartes SIM & réseaux mobiles
Paramètres de transfert d'appel
Appel en attente
Pendant un appel, me notifier les appels entrants
Paramètres des appels entrants
Paramètres supplémentaires pour les appels entrants
Réponse automatique
Désactivé
Paramètres avancés
Politique de confidentialité


9:02
59%
Comptes téléphoniques
Compte d'appel préféré
Comptes SIM
AUTRES COMPTES D'APPEL DISPONIBLES
XiVO

In case you do want to completely disable this, then you can set the value of *XIVO_PLAY_MSG* to *False* which will make caller hear a regular ringtone.

```
XIVO_MAPP_PLAY_MSG = "True"
```

Waiting While Mobile Application Is Waking Up

Before calling a user with a Mobile App, we wait for the mobile app to wake up (and (re)register (SIP level)). By default we wait 15s: 3 loops of 5s.

If the Mobile App never wakes up (or never (re)registers) the call will continue depending on the callee ringing device:

- if it was *Mobile application*: it will end up in user's no answer scenario
- if it was *UC Assistant + Mobile application*: it will call the *UC Assistant*

The time we wait for the Mobile App to wake up is configurable by configuring the number of loops.Interval between two loops is now configurable too. To do so, adjust the following two variables in the file `/etc/xivo/asterisk/xivo_globals.conf`:

```
XIVO_MAPP_LOOPS_MOBILEAPP = 3
XIVO_MAPP_LOOPS_WEBAPPANDMOBILEAPP = 3
XIVO_MAPP_LOOPS_INTERVAL = 5
```

Note that:

- `XIVO_MAPP_LOOPS_MOBILEAPP` configures the number of loops if the callee ringing device is *Mobile application*
- `XIVO_MAPP_LOOPS_WEBAPPANDMOBILEAPP` configures the number of loops if the callee ringing device is *UC Assistant + Mobile application*
- `XIVO_MAPP_LOOPS_INTERVAL` configures the time between two loops

Therefore you might, for example, want to:

- lower the number of loops when ringing device is *UC Assistant + Mobile application*: in order to fallback more quickly on the *UC Assistant* if the Mobile App does not wake up
- and increase the number of loops when ringing device is *Mobile application*: in order to give more time to the Mobile App to wake up before going to the callee no answer scenario
- another more dynamic but risky scenario is to disable the message via *XIVO_PLAY_MSG*, then reducing the time between loops via *XIVO_MAPP_LOOPS_INTERVAL*, in this scenario, we advice you not to forget to adapt the loops number to give sufficient time to the MobileApp to wake up.

6.7 XiVO Connect

Important: To get a demo or more information, please contact XiVO team.

Besides these considerations, the project is **Open Source** and code is available here: https://gitlab.com/avencall/xivo-connect

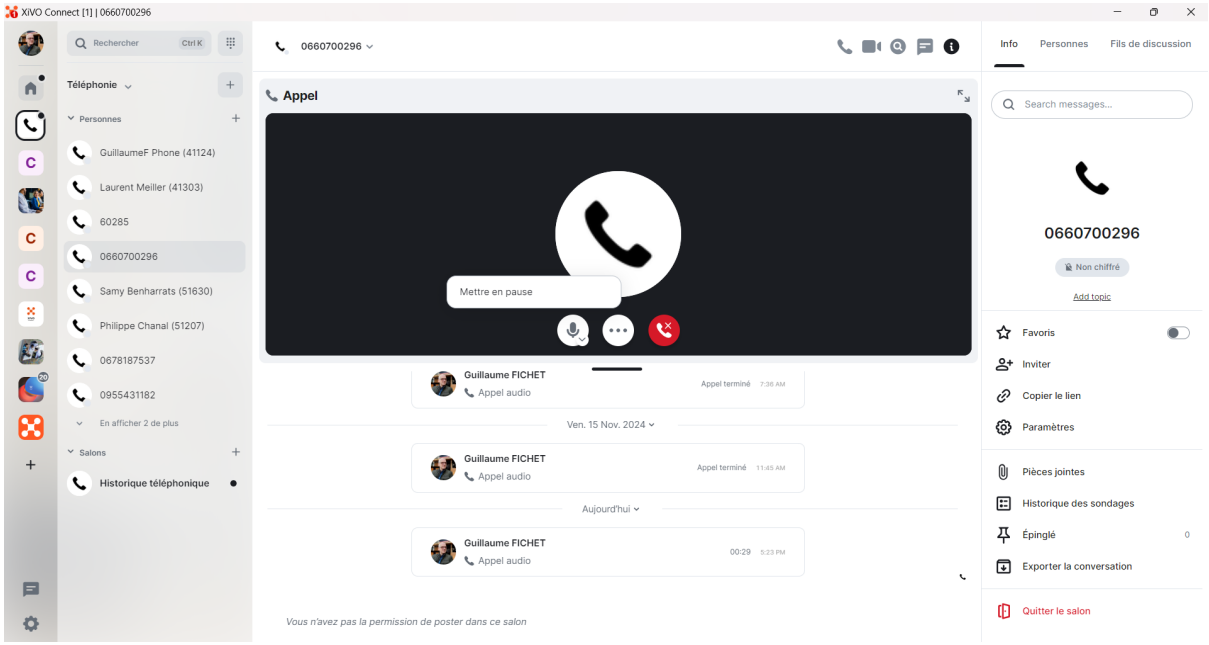
Contents:

- *Overview*
- *Functional Objectives*

- *Main Features*
- *Typical Use Cases*
- *Requirements*

6.7.1 Overview

XiVO Connect is a sovereign collaborative communication module developed on top of XiVO PBX. It combines instant messaging, video conferencing, file sharing and IP telephony in a unified interface. It is a customized soft-fork of **Element**, built on top of the **Matrix** protocol and deeply integrated with the XiVO ecosystem.



6.7.2 Functional Objectives

- Provide a **100% French**, open-source and sovereign alternative to proprietary collaborative platforms.
- Ensure **data protection** through end-to-end encryption.
- Offer full **mobility**: accessible from web browser, desktop application, and mobile app.
- Centralize all communications (chat, voice, video) and documents within a single tool, with **Nextcloud** integration for file storage and sharing.
- Integrate with XiVO telephony: calls are handled directly inside Connect, and call history is available from the interface.

6.7.3 Main Features

Instant Messaging

- Text chat with rooms, threads, mentions
- Private and public conversations
- "Spaces" for organizing teams or projects
- Voice messages

Polls

- Built-in polling tool for decision making inside rooms.

Video Conferencing

- High-quality audio and video
- Screen sharing
- Based on Jitsi technology

Document Sharing

- Native integration with Nextcloud
- Secure file storage and collaborative sharing within the chat interface

Directory

- User directory
- Quick contact search

Mobility

- Mobile application (smartphone/tablet) offering chat, calls and conferencing on the go

Integrated IP Telephony

- Direct interaction with XiVO PBX
- Incoming and outgoing SIP calls from the interface
- Call history accessible inside Connect

6.7.4 Typical Use Cases

- Organizations replacing Teams/Zoom with a sovereign alternative while keeping XiVO telephony.
- Remote or multi-site teams needing a single tool for chat, video, and file collaboration.
- Institutions focused on **confidentiality** and local data hosting requirements.
- Companies looking to unify telephony, messaging and collaboration into a modern platform.

6.7.5 Requirements

- A functional XiVO PBX instance from Naos version for telephony integration.

6.8 Metabase

Important: Metabase replaces SpagoBI for business intelligence and reporting in XiVOCC installations. SpagoBI technology is no longer actively maintained in its open-source version and has been deprecated starting with Maia (2024.05).

6.8.1 Introduction

Metabase is an open-source business intelligence and data visualization tool that provides interactive dashboards and reports for XiVOCC contact center statistics. It connects to the **xivo_stats** database to provide real-time and historical reporting capabilities.

The Metabase integration includes:

- Pre-configured dashboards and reports for queue and agent statistics
- Interactive data exploration tools
- Scheduled report delivery via email
- LDAP authentication support
- Custom report building capabilities

6.8.2 Installation

Note: Installation requires the **xivocc-metabase** package on your XIVOCC system. Itself requires specific docker access. If you don't have those, contact the XiVO team.

Install the Metabase package:

```
apt-get install xivocc-metabase
```

This installation will:

- Add the Docker override `10-docker-metabase.override.yml` in `/etc/docker/compose`, which defines the `metabase` service plus the one-shot `metabase-setup`, `metabase-reports` and `metabase-serializer` services (gated behind the `metabaseInit` Compose profile)
- Create the Metabase PostgreSQL database and user
- Generate the Metabase DB password and store it in `/etc/docker/compose/custom.env`
- On first install only, store the admin credentials and import log level in `/etc/docker/compose/custom.env` (`METABASE_ADMIN_USERNAME`, `METABASE_ADMIN_PASSWORD`, `METABASE_IMPORT_LOGLEVEL`)
- Start the Metabase container, automatically complete the first-run setup (admin user + `xivo_stats` database connection) and import the pre-configured reports

6.8.3 Accessing Metabase

- Metabase is available
  - From the fingerboard under Statistics `https://XIVOCC_IP/fingerboard`
  - Directly Access the Metabase instance at `https://XIVOCC_IP/reporting`
  - Under port *3000*` in the machine (XiVOCC) holding the metabase container.
- Log in
  - with the admin credentials. Retrieve them from `/etc/docker/compose/custom.env`:

```
grep METABASE_ADMIN /etc/docker/compose/custom.env
```

  - With LDAP credentials if the connection has been made. See *ldap authentication*

6.8.4 (Re-)Importing Pre-configured Reports

Pre-configured dashboards and reports are bundled in the `xivoxc/metabase-reports` image and imported automatically during installation into the **Rapports exemples** collection.

Re-running the Import

To re-run the setup and import (for example after updating the reports image), trigger the `metabaseInit` profile:

```
xivocc-dcomp --profile metabaseInit up metabase-setup metabase-reports metabase-
↪serializer
```

This runs three one-shot containers in order:

1. `metabase-reports` — copies the report definitions into the `metabase-reports-data` Docker volume
2. `metabase-setup` — waits for Metabase to be healthy, then completes the first-run wizard if it has not been done yet (idempotent: it exits cleanly if Metabase is already set up)
3. `metabase-serializer` — waits for the `xivo_stats` schema sync to complete, then imports the cards and dashboards

All three containers exit on their own once finished.

6.8.5 Pre-configured Reports

The default report collection includes:

Queue Statistics Reports:

- Queue performance metrics (answered, abandoned, timeout calls)
- Average wait time and call duration
- Service level metrics
- Call distribution by answer type

Agent Reports:

- Agent activity (login, pause, wrap-up time)
- Incoming and outgoing calls per agent
- Agent performance metrics
- Call handling statistics

For detailed information about the statistics model and database schema, see *Reporting and statistics*.

6.8.6 Advanced Configuration

Email Notifications

Metabase supports scheduled report delivery via email. To configure email:

1. Access Metabase as admin
2. Go to **Settings** > **Email**
3. Configure your SMTP server settings

For detailed email configuration instructions, see the official Metabase documentation: Metabase Email Configuration

LDAP Authentication

For enterprise deployments, you can configure LDAP authentication to use your existing directory service:

1. Access Metabase as admin
2. Go to **Settings** > **Authentication**
3. Configure LDAP settings

For detailed LDAP configuration instructions, see the official Metabase documentation: Metabase LDAP Authentication

6.8.7 External Resources

- Metabase Official Documentation
- Metabase Email Setup Guide
- Metabase LDAP Authentication Guide
- XiVO Statistics Database Schema: *Reporting and statistics*

6.8.8 Troubleshooting

First follow metabase logs

```
xivocc-dcomp logs -f metabase
```

Connection Issues

If Metabase cannot connect to the database:

- Verify the database credentials in the connection string
- Check that the `xivo_stats` database exists and is accessible
- Ensure the Docker compose stack is running: `xivocc-dcomp ps`

Import Issues

Try to reimport with more precise `METABASE_IMPORT_LOGLEVEL`, see *metabase import*

Uninstall

```
apt purge xivocc-metabase
```

Admin Password Reset

Consult XiVO support for assistance with password recovery.

6.9 XiVO Resilience

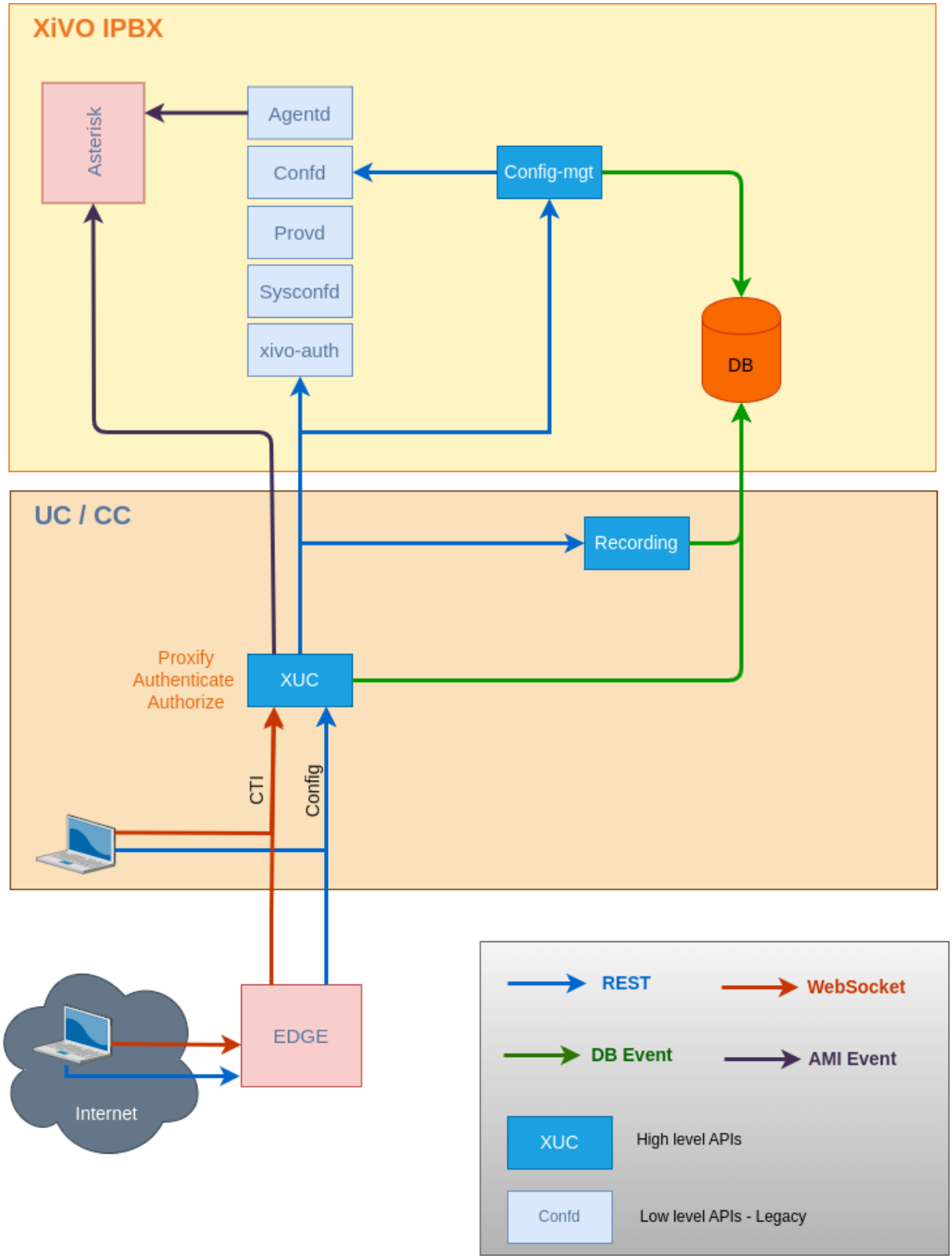
CHAPTER

SEVEN

API AND SDK

XiVO provides a rich set of interfaces to interact with the IPBX and Unified Communications components.

Depending on your needs, you will use either our high-level abstraction layer or direct low-level configuration services.


XiVO IPBX
Asterisk
Agentd
Confd
Provd
Sysconfd
xivo-auth
Config-mgt
DB
UC / CC
Recording
Proxify
Authenticate
Authorize
XUC
CTI
Config
EDGE
Internet
REST
WebSocket
DB Event
AMI Event
XUC
High level APIs
Confd
Low level APIs - Legacy

7.1 High-Level APIs

Note: For CTI Application Developers

The **XUC (XiVO Unified Communications)** server is the recommended entry point for most integrations. It acts as a proxy that authenticates and authorizes requests. It gives then access to:

- **CTI** Using **WebSockets** for real-time call control of telephony events.
- **Configuration** of unified communications features through high-level **REST** APIs.
- **Recording** to manage (listen, find, export…) call recordings.

7.1.1 CTI API

This framework is mainly provided by the xuc server to handle CTI integration. It provides

- Javascript API
- Rest Web services
- Sample application
- Real Time Statistics
- Third party application integration

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

Here is an example project with xuc authentication, both CTI and SIP websocket up and running, and an echo test plus a few examples of cti messages API. https://gitlab.com/xivo.solutions/cti-stack-example

Integration Principles

- Download the javascript API files from the project xucserver on our gitlab repository

```
wget https://gitlab.com/xivo.solutions/xucserver/raw/master/app/assets/javascripts/
↪cti.js
wget https://gitlab.com/xivo.solutions/xucserver/raw/master/app/assets/javascripts/
↪callback.js
wget https://gitlab.com/xivo.solutions/xucserver/raw/master/app/assets/javascripts/
↪membership.js
wget https://gitlab.com/xivo.solutions/xucserver/raw/master/app/assets/javascripts/xc_
↪webrtc.js
wget https://gitlab.com/xivo.solutions/xucserver/raw/master/app/assets/javascripts/
↪SIPml-api.js
```

- Include the files in your projects

```
<!-- jquery needed as a dependency CDN from https://code.jquery.com/ -->
<script src="https://code.jquery.com/jquery-2.2.4.min.js"
    integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44="
    crossorigin="anonymous"></script>

<script src="http://<xucserver>:<xucport>/assets/javascripts/shotgun.js" type="text/
↪javascript"></script>
<script src="cti.js" type="text/javascript"></script>
<script src="callback.js" type="text/javascript"></script>
<script src="membership.js" type="text/javascript"></script>

<!--  Optionnaly Include also the xc_webrtc and SIPml5 javascript APIs for the webRTC␣
↪support -->

<script src="xc_webrtc.js" type="text/javascript"></script>
<script src="SIPml-api.js" type="text/javascript"></script>
```

- Connect to the Xuc server using new Authentication token (see *User basic authentication*)

```
var wsurl = "ws://"+server+"/xuc/api/2.0/cti?token="+token;
Cti.WebSocket.init(wsurl,username,phoneNumber);
```

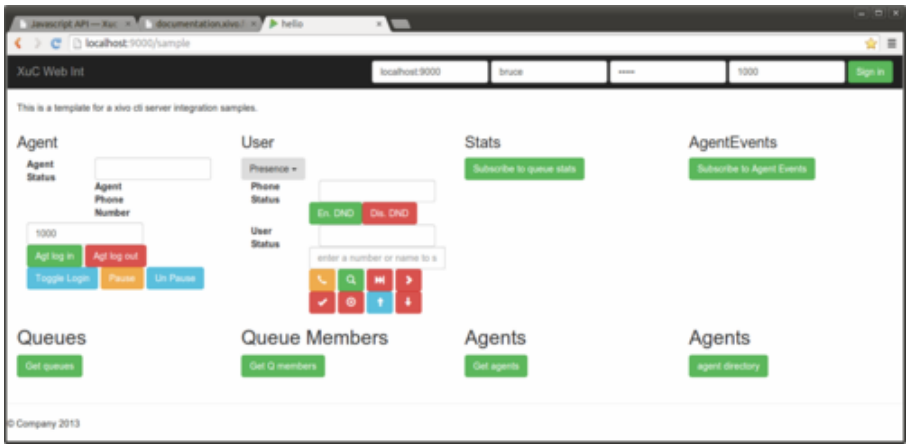
- **Setup event handlers to be notified on**
  - Phone state changes
  - Agent state changes
  - Statistics
  - ...
- **Eventually also webRTC handlers**
  - general
  - register
  - incoming
  - outgoing
- **Once web socket communication is established you are able to call XuC Cti javascript methods.**
  - 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) {
```

(continues on next page)

(continued from previous page)

```
    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_get_agent_call_history').click(function() {
    Cti.getAgentCallHistory(7);
 });
 $('#xuc_get_user_call_history').click(function() {
    Cti.getUserCallHistory(7);
 });
 $('#xuc_get_user_call_history_by_days').click(function() {
    Cti.getUserCallHistoryByDays(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);
```

(continues on next page)

(continued from previous page)

```
xc_webrtc.setHandler(xc_webrtc.MessageType.INCOMING, webRtcIncomingEventHandler);
xc_webrtc.setHandler(xc_webrtc.MessageType.OUTGOING, webRtcOutgoingEventHandler);
```

Throttling

The websocket server integrates a throttling mechanism to prevent flooding.

If you exceed more than 90 request messages in 30 seconds (with a burst of 180), your messages will be throttled and you will receive an error :

```
`{msgType: "Error", ctiMessage: {Error: "Maximum throttle throughput exceeded."}}`
```

By default, the server applies theses limits:

```
websocket.messagePerDuration = 90
websocket.maxBurst = 180
```

They can be adjusted if the default settings are not suitable due to a high volume of messages on a machine with heavy traffic. To adjust these values you have to set `WS_THROTTLING_NB_MESSAGES` and `WS_THROTTLING_NB_MESSAGES_BURST` environment variables in the `/etc/docker/compose/custom.env`

Example:

```
WS_THROTTLING_NB_MESSAGES=120
WS_THROTTLING_NB_MESSAGES_BURST=240
```

Then you need to recreate the xucserver container with `xivocc-dcomp up -d` to apply the changes. If these environment variables are not defined in the custom.env file, the default throttling limits will be applied.

This mechanism is analog to the rate-limiting from nginx, and concepts such as burst are *explained there*.

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"}}
```

Generic API

Generic CTI Methods

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

Generic CTI Events

Error

- Cti.MessageType.ERROR

Is triggered whenever service answered abnormally to the request.

LoggedOn

- Cti.MessageType.LOGGEDON

Is triggered once a user is properly authenticated and logged in.

LoggedOnError

- Cti.MessageType.LOGGEDONERROR

Is triggered when the CTI session establishment fails after a WebSocket connection. The WebSocket connection is closed immediately after this message is sent.

Two error codes are possible:

loggedOnFailed

Triggered when the CTI session establishment fails abnormally.

```
{
  "msgType": "LoggedOnError",
  "ctiMessage": {
    "Error": "loggedOnFailed"
  }
}
```

loggedOnTimeout

Triggered when the CTI session is not established within the configured delay after the WebSocket connection. The default timeout is 10 seconds, configurable via the `XUC_CTI_LOGGED_ON_TIMEOUT` environment variable (config key: `xivocti.loggedOnTimeout`).

```
{
  "msgType": "LoggedOnError",
  "ctiMessage": {
    "Error": "loggedOnTimeout"
  }
}
```

Link Status Update

- Cti.MessageType.LINKSTATUSUPDATE

Is triggered when connection to server is lost.

User API

Login and Authentication

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);
```

Contacts And Favorites

Cti.searchContacts: function(term)

Important: This command deprecates previously used *Cti.directoryLookUp(pattern)*.

Notes: For backward compatibility, when you call new APIs *Cti.searchContacts* you will both receive the new *Cti.MessageType.CONTACTSHEET* (see detail below) **and** the old *Cti.MessageType.DIRECTORYRESULT*.

Associated Handler

- Cti.MessageType.CONTACTSHEET

Triggered by command *Cti.searchContacts(pattern)*.

```
{
    "msgType": "ContactSheet",
    "ctiMessage": {
        "sheets": [
            {
                "name": "James Bond",
                "subtitle1": "",
                "subtitle2": "",
```

(continues on next page)

(continued from previous page)


```
                "picture": "",
                "isPersonal": false,
                "isFavorite": false,
                "isMeetingroom": false,
                "canBeFavorite": true,
                "status": {
                    "phone": 0,
                    "video": "Available"
                },
                "actions": [
                    {
                        "action": "Call",
                        "args": [
                            "1000"
                        ],
                        "default": true
                    },
                    {
                        "action": "Chat",
                        "args": [
                            "jbond"
                        ],
                        "default": false
                    },
                    {
                        "action": "Mail",
                        "args": [
                            ""
                        ],
                        "default": false
                    },
                    {
                        "action": "Video",
                        "args": [
                            "jbond"
                        ],
                        "default": false
                    }
                ],
                "sources": [
                    {
                        "name": "internal",
                        "id": "2"
                    }
                ],
                "details": [
                    {
                        "name": "Contacts",
                        "fields": [
                            {
                                "name": "Téléphone interne",
                                "data": "1000",
                                "dataType": "PhoneNumber"
                            },
                            {
                                "name": "Téléphone pro",
```


(continues on next page)

(continued from previous page)

```
                        "data": "",
                        "dataType": "PhoneNumber"
                    },
                    {
                        "name": "Téléphone mobile",
                        "data": "",
                        "dataType": "PhoneNumber"
                    },
                    {
                        "name": "Téléphone autre",
                        "data": "",
                        "dataType": "PhoneNumber"
                    },
                    {
                        "name": "E-mail",
                        "data": "",
                        "dataType": "Mail"
                    },
                    {
                        "name": "Télécopie",
                        "data": "",
                        "dataType": "PhoneNumber"
                    }
                ]
            },
            {
                "name": "Général",
                "fields": [
                    {
                        "name": "Fonction",
                        "data": "",
                        "dataType": "String"
                    },
                    {
                        "name": "Service",
                        "data": "",
                        "dataType": "String"
                    },
                    {
                        "name": "Manager",
                        "data": "",
                        "dataType": "String"
                    },
                    {
                        "name": "Société",
                        "data": "",
                        "dataType": "String"
                    },
                    {
                        "name": "Site web",
                        "data": "",
                        "dataType": "Url"
                    }
                ]
            },
            {
```

(continues on next page)

(continued from previous page)

```
                        "name": "Lieu d'affectation",
                        "fields": [
                            {
                                "name": "Bureau",
                                "data": "",
                                "dataType": "String"
                            },
                            {
                                "name": "Adresse",
                                "data": "",
                                "dataType": "String"
                            }
                        ]
                    }
                ]
            }
        ]
    }
}
```

Cti.getFavoriteContacts: function()

Important: This command depreactes previously used *Cti.getFavorites*

Notes: For backward compatibility, when you call new APIs *Cti.getFavoriteContacts* you will receive the new *Cti.MessageType.FAVORITECONTACTSHEET* **and** the old *Cti.MessageType.FAVORITES*.

Retrieve all the favorites defined for the user connected

Cti.addFavorite: function(contactId, source)

To set a contact (e.g. from search results) as favorite, source is the directory where favorite will be owned.

Cti.removeFavorite: function(contactId, source)

User Methods

Cti.changeUserStatus(reason)

Depreciated Update user status using a Cti server configured status name. (Use *Cti.pauseAgent(agentId, reason)* instead for agent only)

Cti.displayNameLookup(username)

Retrieves user's display name by providing it's username.

Cti.setUserPreference(key, value, value_type)

Update user's preference key. See *UserPreferences keys* for the list of existing keys.

User Events

User Statuses

- Cti.MessageType.USERSTATUSES : "UsersStatuses"

> **Warning:** The *UsersStatuses* message is **DEPRECATED** and **should not be used**. Use the *ctiStatus* message instead.

List all the statuses configured on XiVO to know which are possible pause reasons.

```
{
  "msgType":"UsersStatuses",
  "ctiMessage":[
      {"name":"disconnected","color":"#9E9E9E","longName":"Déconnecté","actions":[{
↪"name":"agentlogoff","parameters":""}]},
      {"name":"away","color":"#FFDD00","longName":"Sorti","actions":[{"name":
↪"enablednd","parameters":"false"}]},
      {"name":"berightback","color":"#F2833A","longName":"Bientôt de retour","actions
↪":[{"name":"enablednd","parameters":"false"}]},
      {"name":"available","color":"#9BC920","longName":"Disponible","actions":[{"name
↪":"enablednd","parameters":"false"}]},
      {"name":"ook","color":"#FF0F0F","longName":"Far Away","actions":[{"name":
↪"queuepause_all","parameters":"true"}]},
      {"name":"outtolunch","color":"#6CA6FF","longName":"Parti Manger","actions":[{
↪"name":"queuepause_all","parameters":"true"},
      {"name":"enablednd","parameters":"false"}]},
      {"name":"donotdisturb","color":"#D13224","longName":"Ne pas déranger","actions
↪":[{"name":"enablednd","parameters":"true"}]}
  ]
}
```

CTI Status

- Cti.MessageType.CTISTATUSES : "CtiStatuses"

List all the statuses configured on XiVO to know which are possible pause reasons.

```
{
  "msgType":"CtiStatuses",
  "ctiMessage":[
      {"name":"disconnected","displayName":"Déconnecté","status": 2},
      {"name":"ook","displayName":"Far Away","status": 1},
      {"name":"outtolunch","displayName":"Parti Manger","status": 1},
      {"name":"available","displayName":"Disponible","status": 0},
```

(continues on next page)

(continued from previous page)

```
  ]
}
```

User Status Update

- Cti.MessageType.USERSTATUSUPDATE : "UserStatusUpdate",

Depreciated Triggered when user changes status (while calling *Cti.changeUserStatus()*)

User Config Update

- Cti.MessageType.USERCONFIGUPDATE : "UserConfigUpdate",

Triggered when config of the user is updated. This happens if forward config is modified or voicemail for example. Any change of the following attribute might trigger this event.

```
{
    "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
    }
}
```

User Preference Update

- Cti.MessageType.USERPREFERENCE : "UserPreference",

Triggered on login to retrieve all user preferences. Then triggered when a user preference is updated.

```
{
    "msgType":"UserPreference",
    "ctiMessage":{
        "PREFERRED_DEVICE": {
            "value": "TypePhoneDevice",
            "value_type": "String"
        }, "MOBILE_APP_INFO": {
            "value": "true",
            "value_type": "Boolean"
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

UserPreferences keys

The existing keys are :

- *PREFERRED_DEVICE*: can be *TypePhoneDevice* or *TypeDefaultDevice*
- *MOBILE_APP_INFO*: can be *true* or *false*

Phone API

This API allow you to remotely control a device once a user associated to the device through a line is authenticated.

A sample of implementation is available in *app/assets/javascripts/pages/sample.js* and *app/views/sample/sample.scala.html*

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",
                   "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* If dial fails, a failure *Phone Events* will be sent back to application.

When placing a call using this api, *Phone Events* will be sent through the websocket when calls state change. If a call is made to a conference room hosted on the XiVO, *Conference Events* will be sent also.

> **Warning:** When entering a conference room **using this API**, if the conference room is configured with an administrator PIN, the user will **enter directly** the conference room **as an administrator** without having to enter a PIN code. If the conference room has only a user PIN, the user will also enter directly without having to enter a PIN code.

Cti.dialByUsername(username, variables)

Place a call to destination, defined by username, 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.originate(destination)

Originate a call

Cti.hangup(uniqueId)

Hangup established call. *uniqueId* is optional, but if you set it, you can explicitly hangup a call.

Cti.answer(uniqueId)

Answers a call. *uniqueId* is optional, but if you set it, you can explicitly answer a call among other ringing calls. (however, this parameter is for **WebRTC only**)

Cti.hold(uniqueId)

Put current call on hold. *uniqueId* is optional. If *uniqueId* is set it you can explicitly hold/unhold a call among other calls. (however, this parameter is for **WebRTC only**)

Note that you can't have two established. If you have one establish call and on call on hold, if you unhold this last call, the established will go automatically 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.getCurrentCallsPhoneEvents()

Request PhoneEvents for current device calls. See *Phone Events* for answer description.

Cti.naFwd(destination,state)

Forward on non answer

Cti.uncFwd(destination,state)

Unconditionnal forward

Cti.busyFwd(destination,state)

Forward on busy

Cti.dnd(state)

Set or unset do not disturb, state true or false

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",
        "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"
        },
        username: "jblack"
    }
}
```

fields	Description
Event types	• EventReleased • EventDialing • EventRinging • EventEstablished • EventOnHold • EventFailure
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_
username	Can be the username cti of called number, it's only defined when the called user is an internal user

An event Failure can be sent when Cti.dial cannot be completed. Currently it is very specific and is triggered only in case of Originate (webrtc) and peer initiator of the call is not registered

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",
          ...
        },
        ...
    }
}
```

Phone Status Update

When opening the websocket, the following message will be received automatically. It will indicate the current phone status (or the webrtc status) of the user and indicate if the user is ready to receive calls. A new event will be received if, for any reason, the phone (or the webrtc line) becomes unavailable.

Cti Message Type
`Cti.MessageType.PHONESTATUSUPDATE`

Example

```
{
    "msgType":"PhoneStatusUpdate",
    "ctiMessage":{
        status: "AVAILABLE"
    }
}
```

Possible `status` values are:

INDISPONIBLE
No contact is available, the user can't be reached.

AVAILABLE
At least one contact available, the user can be reached.

Important: This message is based on the contact status the user.

However if you want to monitor the status based on the extension state instead you can add the following line in `/etc/docker/compose/custom.env` file :

```
XUC_PHONE_STATUS_FROM_SIP_CONTACT=false
```

The PhoneStatus will then be based on the hint status instead of the contacts (SIP Registration).

In this case Possible `status` values are:

ONHOLD
The phone has at least one held call

RINGING
The phone is ringing

INDISPONIBLE
The phone is unavailable and is not able to receive calls

BUSY_AND_RINGING
The phone is busy and ringing

AVAILABLE
The phone is available and ready to receive calls

CALLING
The phone is on call

BUSY
The phone is busy

DEACTIVATED
The phone is deactivated and is not able to receive calls

UNEXISTING
The phone does not exist

ERROR
An error occured while monitoring the phone. The phone may not be able to receive calls

Phone Hint Status Methods

Cti.subscribeToPhoneHints(phoneNumbers)

- phoneNumbers (Array of string): list of phone numbers to subscribe

Subscribe to PhoneHintStatusEvents for a list of phone numbers. You will get an initial event with the current Phone Hint Status (see *Phone Hint Status Events* for details) for every subscribed phone number and then a new *Phone Hint Status Events* for every change to the phone number state. After the Xuc server restart it may happen that the current phone state is unknown, in which case you will not get the initial event, only the first event update. You can repeat the command to subscribe to more numbers. The subscription is valid for the current websocket and can be cancelled either by closing the websocket, or by the *Cti.unsubscribeFromAllPhoneHints()* command.

Cti.unsubscribeFromAllPhoneHints()

This command allows you to cancel all previous subscription to phone hints.

Phone Hint Status Events

You can subscribe to PhoneHintStatusEvents using *Cti.subscribeToPhoneHints*, allowing you to monitor state of the phone line with a given number. PhoneHintStatusEvents has the following format:

```
{
    "msgType":"PhoneHintStatusEvent",
    "ctiMessage":{
        "number":"1005","status":0
    }
}
```

The phone hint status code can be resolved to a name using the *Cti.PhoneStatus* object in the *cti.js*. E.g., the status 0 from the example above stands for "AVAILABLE".

Voice Mail Status Events

- VOICEMAILSTATUSUPDATE : "VoiceMailStatusUpdate",

```
{"msgType":"VoiceMailStatusUpdate","ctiMessage":{"voiceMailId":58,"newMessages":2,
↪"waitingMessages":1,"oldMessages":1}}
```

Agent API

This API is dedicated to handle Agents for CC activities.

A sample of implementation is available in *app/assets/javascripts/pages/sampleAgents.js* and *app/views/sample/sampleAgents.scala.html*

Login and Authentication

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

Security considerations

Defining a 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

Agent Methods

Cti.loginAgent(agentPhoneNumber, agentId)

Log an agent

Cti.logoutAgent(agentId)

Un log an agent

Cti.pauseAgent(agentId, reason)

Change agent state to pause.

- *agentId*: Id of the agent to set state for. Can be omitted to change current loggedin agent state.
- *reason*: Optional string to label the kind of pause to set.

Cti.unpauseAgent(agentId)

Change agent state to ready

- *agentId*: Id of the agent to set state for. Can be omitted to change current loggedin agent state.

Cti.listenAgent(agentId)

Listen to an agent

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.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.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.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 *Statistics* events.

Agent 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"
           },
           {
             "value": "http://www.google.fr/",
             "name": "popupUrl1",
             "order": 20,
```

(continues on next page)

(continued from previous page)

```
              "type": "url"
            }
          ]
        }
      }
    },
    "channel": "SIP/1k4yj2-00000013"
  }
}
```

Right Profile

- Cti.MessageType.RIGHTPROFILE: "RightProfile"

```
{"msgType":"RightProfile","ctiMessage":{"profile":"Supervisor", "rights":["dissuasion
↪","recording"]}}
```

This message is sent upon connection to the xuc websocket. The profile can be one of: "Supervisor", "Admin", "NoRight". The rights property contains an array with additional rights:

- "recording": User has access to recording management settings on queues
- "dissuasion": User has access to dissuasion management settings on queues

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]}
```

  - 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]}
```

  - AgentRinging

```
{"name":"AgentRinging","agentId":19,"phoneNb":"1000","since":0,"queues
↪":[8,14,170,4,1]}
```

  - AgentDialing

```
{"name":"AgentDialing","agentId":19,"phoneNb":"1000","since":0,"queues
↪":[8,14,170,4,1]}
```

– AgentOnCall

```
{"msgType":"AgentStateEvent","ctiMessage":
    {"name":"AgentOnCall","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1],
        "acd":false,"direction":"Incoming","callType":"External",
↪"monitorState":"ACTIVE"}}
```

– AgentLoggedOut

```
{"name":"AgentLoggedOut","agentId":19,"phoneNb":"1000","since":0,
↪"queues":[8,14,170,4,1]}
```

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}
]
```

Listen call

- 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}
```

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"}
]
```

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},
```

(continues on next page)

(continued from previous page)

```
        {"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"}
        ]
}
```

Callback Methods

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.

Callback.listenCallbackMessage(uuid)

Make your device rings to be able to listen recorded message if exists.

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"
   }
}}
```

Group API

This API is dedicated to manipulate groups fro UC Assistant.

Groups Events

UserGroups

- Handler on : *Cti.MessageType.USERGROUPS*

The handler is executed when the user logged in the UC Assistant. The message contains a list of groups the user belong to.

```
{
    "msgType":"UserGroups",
    "ctiMessage":[
        {"groupId":25,"groupName":"admin_team","groupNumber":"2300","membershipStatus
↪":"Available"},
        {"groupId":23,"groupName":"support_team","groupNumber":"2100",
↪"membershipStatus":"Available"}
    ]
}
```

UserGroupUpdate

- Handler on : *Cti.MessageType.USERGROUPUPDATE*

The handler is executed when a group the user belong to is either edited / deleted / created.

```
{
    "msgType":"UserGroupUpdate",
    "ctiMessage":{
        "groupId":25,"groupName":"admin_team","groupNumber":"2311","membershipStatus":
↪"Available"
    }
}
```

Groups Methods

The following methods allow you to manage the state of users within call distribution groups.

Cti.pauseUserGroup(groupId)

This command put user in the group in 'Paused' state, meaning he will not receive call from this group anymore.

Cti.unpauseUserGroup(groupId)

This command put user in the group in 'Available' state, meaning he will start receiving call from this group again.

MembershipStatus

User membership in group events indicate if the user is able to receive calls from the group or not.

The possible values are :

- `Available`: The user is connected to the group and can receive calls.
- `Exited`: The user is no longer member of the group.

Queue API

This API is dedicated to manipulate queues for CC activities.

A sample of implementation is available in *app/assets/javascripts/pages/sampleQueues.js* and *app/views/sample/sampleQueues.scala.html*

Queue Methods

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.subscribeToQueueCalls(queueId)

Cti.unSubscribeToQueueCalls(queueId)

Cti.retrieveQueueCall(queueCall)

Allow to pickup one call from queue and connect it to current agent connected.

a QueueCall is one element of *Queue Calls*, i.e

```
{"position":1,"name":"John Doe","number":"33356782212","queueTime":"2015-07-
↪16T10:40:16.626+02:00"}
```

> **Warning:** Retrieve a queue call may fail if your device is already ringing or connected.

Queue Events

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"},
        {"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}
        ]
}
```

Membership Methods

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]
```

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}
  ]
}
```

History API

This API to retrieve any kind of call history (user, agent, queue, customer...)

A sample of implementation is available in *app/assets/javascripts/pages/sampleHistory.js* and *app/views/sample/sampleHistory.scala.html*

History Methods

Cti.getUserCallHistory(size)

Get the call history of the logged in user, limited to the last *size* calls. If size is *null*, the last 10 days of history will be returned.

Cti.getUserCallHistoryByDays(days)

Get the call history of the logged in user, limited to the last *days* days.

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

History Events

Associated Handler CALLHISTORY

Received when calling *Cti.getUserCallHistory(size)* or *Cti.getUserCallHistoryByDays(days)*.

- CALLHISTORY : "CallHistory"

```
{
     "start":"2022-09-08 10:32:50",
     "duration":"00:00:25",
     "srcUsername":"usera",
     "dstUsername":"userb",
     "status":"answered",
     "srcPhoneStatus":4,
     "srcVideoStatus":"Available",
     "dstPhoneStatus":4,
     "dstVideoStatus":"Available",
     "srcNum":"1000",
     "dstNum":"1007",
     "srcFirstName":"User",
     "srcLastName":"A",
     "dstFirstName":"user",
     "dstLastName":"B"
 }
```

Received when calling *Cti.getAgentCallHistory(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

Conferences API

This api is to manipulate voice conference room

You can refer to *Conference Room* configuration for organizer feature

A sample of implementation is available in *app/assets/javascripts/pages/sampleConferences.js* and *app/views/sample/sampleConferences.scala.html*

Conference Methods

Cti.getConferenceRooms()

Warning: This API is **DEPRECATED** and will be replaced with breaking changes in the next version.

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",
        "username": "jbond"
      }
    ]
  },
  {
    "number": "4001",
    "name": "conference_support",
    "pinRequired": true,
    "startTime": 0,
    "members": []
  }
]
```

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.conferenceDeafen(conferenceNumber, index)

Make an attendee deaf by its index in the given conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin. After this command, the participant will not hear the conference but other participants can hear him.

Note: When a conference participant is deafened, a button will appear in the UC Assistant to undeafen him. This button is only available for conference organizers.

See also *conference_command_error*.

Cti.conferenceUndeafen(conferenceNumber, index)

Make an attendee undeaf by its index in the given conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin. After this command, the participant hear again the conference.

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*.

Cti.conferenceClose(conferenceNumber)

Close the given conference and hangup all participants. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

Cti.conferenceInvite(conferenceNumber, exten, role, earlyJoin, variables, marked, leaveWhenLastMarkedLeave, callerId)

Invite the given "exten" in the conference. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

Parameters :

- conferenceNumber: conference number
- exten: the guest extension to invite in the conference
- role: role of the guest once in the conference, either "User" or "Organizer"
- earlyJoin: boolean, set to "true" to ear the ringback in the conference or set to "false" if you want the guest to join once he answered
- variables: set of variables to be propagated
- marked: boolean, set to "true" to invite the extension as a marked user
- leaveWhenLastMarkedLeave: boolean, set to "true" for the invited extension to be removed from the conference when the last marked user in the conference leaves

- callerId: set a customized callerId on the invited extension. The callerId will be set as the invited extension's name inside the conference

Note: Variables need to be in object format {"key1": "value1", "key2": "value2"} and they will be passed to the channels variables, prepended by *USR_* for each key. (USR_key1, USR_key2 in this case)

See also *conference_command_error*.

Cti.includeToConference(role, marked, leaveWhenLastMarkedLeave, callerId)

Includes a call to a conference. The prerequisite is to be inside a conference and to have a second call.

Parameters :

- role: role of the invitee once in the conference, either "User" or "Organizer". Note that you cannot escalate your privileges.
- marked: boolean, set to "true" for the invitee to be a marked user
- leaveWhenLastMarkedLeave: boolean, set to "true" for the invitee to be removed from the conference when the last marked user in the conference leaves
- callerId: set a customized callerId on the invitee. The callerId will be set as the invitee's name inside the conference

See also *conference_command_error*.

Cti.conferenceReset(conferenceNumber)

Reset the participants state in the given conference to their default state. All muted or deaf participants are not muted or deaf anymore. This method is restricted to conference organizer so you must enter the conference with an organizer pin.

See also *conference_command_error*.

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",
```

(continues on next page)

(continued from previous page)

```
      "isMuted": false,
      "isMe": true,
      "username": "jbond",
      "isDeaf": false
    }
  ],
  "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,
  "username": "jbond",
  "isDeaf": 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
- username: username cti of the user, it's only defined for internal users
- isDeaf: indicate if the participant is deaf

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.

CHAT API

Allows user to send a text message to another user . If *xivo-chat-backend* package is installed messages can be persisted (even if user is disconnected).

A sample of implementation is available in *app/assets/javascripts/pages/sampleFlashText.js* and *app/views/sample/sampleFlashText.scala.html*

Chat Methods

Introduction

All chat methods contains *sequence* number that allows you setting any value to correlate your command with the answer. Basically if you send 3 chat messages to 3 different users, you will be able to know which acknowledgment is related to which destination just by looking at this number.

Send chat message

Cti.sendFlashTextMessage(username, sequence_number, message)

Example :

```
Cti.sendFlashTextMessage("user", 24, "This is my message");
```

Retrieve chat history

Cti.getFlashTextPartyHistory(username, sequence_number)

Example :

```
Cti.getFlashTextPartyHistory("user", 24);
```

FlashText Events

- Handler on : Cti.MessageType.FLASHTEXTEVENT

Message Events

- FlashTextUserMessage

Received by the user when a message is sent

```
{
  "msgType": "FlashTextEvent",
  "ctiMessage": {
    "event": "FlashTextUserMessage",
    "from": {
      "username": "bwillis",
      "phoneNumber": "1001",
      "displayName": "Bruce Willis",
      "guid": "goegjutxyt8c5dmsqy6xk4f85w"
    },
    "to": {
      "username": "jbond",
      "phoneNumber": "1000",
      "displayName": "James Bond",
      "guid": "eiifh5zietbi7k8ao6ndrzr7zo"
    },
    "sequence": 145,
    "message": "How are you James ?",
    "date": "2019-05-28T14:55:08.628+02:00",
  }
```

- FlashTextSendMessageAck

The message sent to the user can be delivered to the user, means the user is connected to the server, if the connected application is able to receive the message, message will be delivered.

```
{
  "msgType": "FlashTextEvent",
  "ctiMessage": {
    "event": "FlashTextSendMessageAck",
    "sequence": 1,
    "offline": false,
    "date": "2020-06-10T12:08:19.565Z"
  }
}
```

- FlashTextSendMessageAckOffline

If user is just disconnected, message is persisted and available to be retrieved later on further connection.

```
{
  "msgType": "FlashTextEvent",
  "ctiMessage": {
    "event": "FlashTextSendMessageAckOffline",
    "sequence": 1,
    "offline": true,
    "date": "2020-06-10T12:08:19.565Z"
  }
}
```

- FlashTextSendMessageNack

In case of error (network connection, service not running or not available, database error),

```
{
  "msgType": "FlashTextEvent",
  "ctiMessage": {
    "event":"FlashTextSendMessageNack"
    "sequence":28841,
  }
}
```

History Events

- FlashTextUserMessageHistory

Event sent each time when the chat history between two users is requested

```
{
      "msgType": "FlashTextEvent",
      "ctiMessage": {
              "event": "FlashTextUserMessageHistory",
              "users": [{
                      "username": "jbond",
      "phoneNumber": "1000",
                      "displayName": "James Bond",
                      "guid": "eiifh5zietbi7k8ao6ndrzr7zo"
              }, {
                      "username": "bwillis",
      "phoneNumber": "1001",
                      "displayName": "Bruce Willis",
                      "guid": "goegjutxyt8c5dmsqy6xk4f85w"
              }],
              "messages": [{
                      "from": {
                              "username": "bwillis",
        "phoneNumber": "1000",
                              "displayName": "Bruce Willis",
                              "guid": "goegjutxyt8c5dmsqy6xk4f85w"
                      },
                      "to": {
                              "username": "jbond",
        "phoneNumber": "1001",
                              "displayName": "James Bond",
                              "guid": "eiifh5zietbi7k8ao6ndrzr7zo"
                      },
                      "sequence": 0,
```

(continues on next page)

(continued from previous page)


```
                        "message": "hello1",
                        "date": "2020-03-06T16:41:18.789Z"
                }, {
                        "from": {
                                "username": "jbond",
        "phoneNumber": "1000",
                                "displayName": "James Bond",
                                "guid": "eiifh5zietbi7k8ao6ndrzr7zo"
                        },
                        "to": {
                                "username": "bwillis",
        "phoneNumber": "1001",
                                "displayName": "Bruce Willis",
                                "guid": "goegjutxyt8c5dmsqy6xk4f85w"
                        },
                        "sequence": 0,
                        "message": "Hello",
                        "date": "2020-03-09T10:19:12.763Z"
                }],
                "sequence": 1
        }
}
```


WebRTC API

webRTC Events

- 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.

If really you want to interact between SIPml5 session and XUC calls, you can use *SIP_CALL_ID* which is exposed in xc_webrtc and also in PhoneEvent than can be received in XUC websocket. See *Phone Events*

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, token, remoteAudio, ip)

Legacy Init - it asks for line configuration and ICE configuration before starting webRTC connection.

- 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,
- token - used when ssl is set to true, either reusing the CTI one, either create a new one, see *User basic authentication*
- 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.initByLineConfig(lineCfg, name, ssl, websocketPort, token, remoteAudio, ip)

Same as legacy init, but without asking for line configuration as you pass it as pamameter.

- lineCfg - user's line associated retrieved through LineConfig object in Cti websocket
- 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,
- token - used when ssl is set to true, either reusing the CTI one, either create a new one, see *User basic authentication*
- 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)

Start a webRTC call to the *destination*.

xc_webrtc.answer(sipCallId)

Answer an incoming webRTC call. *sipCallId* is optional, but if you set it, you can explicitly answer a specific call.

Note: The library accepts as an optional parameter the SIP_CALL_ID, which are currently reported in events. See *Phone Events*

xc_webrtc.conference()

Can be called when there's one call on hold and one active, then it resumes the call on hold and starts a 3-party conference hosted by the webrtc endpoint. It can be also used to resume a 3-party conference on hold.

xc_webrtc.hold(sipCallId)

Toggle hold on a webrtc call or a conference. A conference can be resumed either by new call of *xc_webrtc.hold()* or by a new call of *xc_webrtc.conference()*.

sipCallId is optional, but if you set it, you can explicitly hold a specific call. At least one call will always be ongoing even if you try to have two established calls at a time.

Note: The library accepts as an optional parameter the SIP_CALL_ID, which are currently reported in events. See *Phone Events*

---

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.toggleMicrophone(sipCallId)

Disable or enable microphone for a specific call id

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'}]
```

xc_webrtc.getIceUrls()

Get the list of currently configured STUN/TURN servers

Sip Configuration

Get sip configuration

Retrieve the ice configuration currently applied.

Example:

```
[
    {
        "urls": [
            "stun:xivo-edge:3478"
        ]
    },
    {
        "urls": [
            "turn:xivo-edge:3478",
            "turns:xivo-edge:3478"
        ],
        "username": "1617284035",
        "credential": "1o...BeLqsBh9g="
    }
]
```

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

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.

webRTC debugging

xc_webrtc.setDebug

The SIPml console debugging can be activated separately using this method.

Parameters are :

- debugLevel: String representing the lowest log level to display ('fatal', 'error', 'warn', 'info')
- event: Boolean to enable or disable event debugging
- handler: Boolean to enable or disable subscribe and unsubscribe debugging

Example :

```
xc_webrtc.setDebug('fatal', true, true)
```

SIPml.setCustomIceGatheringTimeout

You can set a custom ice gathering timeout to test some scenarios using this method.

Parameters are :

- ms: Integer representing the wanted timeout before shortcutting ice gathering (in milliseconds)

Example :

```
SIPml.setCustomIceGatheringTimeout(500)
```

xc_webrtc.setCustomLogger

You can set a custom logger object using this method.

Parameters are :

- logger: JS Object representing the custom logger. Must implement info, error, warn and log methods. Those methods will be called inside SIPml on the corresponding log level.

Example :

```
xc_webrtc.setCustomLogger({
    info: () => {...},
    error: () => {...},
    warn: () => {...},
    log: () => {...}
})
```

Rest API

The REST API allows you to control XiVO from a remote application. It can be accessed through HTTP requests using the following base URL: *http://localhost:${XUC_PORT}/xuc/api/2.0* where the default XUC port number is *8090*.

Type of users

Two types of users can request the API: CTI users and web service users.

CTI users

CTI users are XiVO users which are usually associated with a line (either plastic phone or WebRTC). They have have limited rights to the API, they can only request specific endpoints.

Web service users

Web service users are technical users. They are not associated with any line. They have specific and custom rights to use the API.

Permissions and ACLs

ACLs handle user permissions. CTI users all have the same permissions, they all are contained into the alias *alias.ctiuser*. They only can request some endpoints of the API.

However, web service user may have specific and custom permissions to request the API. A web service user can have access to specific endpoints, or to specific methods.

See *REST API Permissions* for more information.

ACLs of CTI users

All CTI users share the same permissions. The alias *alias.ctiuser* contains the following permissions:

```
["xuc.rest.contact.personal.read",
"xuc.rest.contact.personal.create",
"xuc.rest.contact.personal.delete",
"xuc.rest.contact.personal.*.read",
"xuc.rest.contact.personal.*.update",
"xuc.rest.contact.personal.*.delete",
"xuc.rest.contact.export.personal.read",
"xuc.rest.contact.import.personal.create",
"xuc.rest.contact.display.personal.read",
"xuc.rest.call_qualification.queue.*.read",
"xuc.rest.call_qualification.csv.#.read",
```

(continues on next page)

(continued from previous page)

```
"xuc.rest.call_qualification.create",
"xuc.rest.mobile.push.register.create",
"xuc.rest.config.meetingrooms.static.token.*.read",
"xuc.rest.config.meetingrooms.personal.token.*.read",
"xuc.rest.config.meetingrooms.temporary.token.*.read",
"xuc.rest.config.meetingrooms.personal.*.read",
"xuc.rest.config.meetingrooms.personal.create",
"xuc.rest.config.meetingrooms.personal.update",
"xuc.rest.config.meetingrooms.personal.*.delete",
"xuc.rest.config.#.read",
"xuc.rest.config.#.create",
"xuc.rest.config.#.update"]
```

User basic authentication

Only authenticated users can use the API. The authentication process provides a JWT token that must be used in all subsequent requests.

CTI user authentication

Request

Method	POST
URI	/auth/login
Header	Content-Type: application/json

Body parameters

Field	Type	Description
login	string	CTI user login
password	string	CTI user password

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/auth/login \
-d '{"login": "cti_user_login", "password": "cti_user_password"}' \
-H "Content-Type: application/json"
```

Response

```
{
   login: "$CTI_USER_LOGIN",
   token: "$CTI_USER_JWT_TOKEN",
   TTL: $CTI_USER_TTL
}
```

Model

Field	Type	Description
login	string	CTI user login
token	string	CTI user JWT token
TTL	integer	Time to live of the token in seconds

Error messages

Error code	HTTP header code	Possible cause
InvalidCredentials	401	Invalid login and/or invalid password
InvalidJson	400	The request body does not follow the body model
NotHandledError	500	Error not covered in current implementation

Web service user authentication

Request

Method	POST
URI	/auth/webservice
Header	Content-Type: application/json

Body parameters

Field	Type	Description
login	string	Web service user login
password	string	Web service user password

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/auth/webservice \
-d '{"login": "web_service_user_login", "password": "web_service_user_password"}' \
-H "Content-Type: application/json"
```

Response

```
{
   login: "$WEB_SERVICE_USER_LOGIN",
   token: "$WEB_SERVICE_USER_JWT_TOKEN",
   TTL: $WEB_SERVICE_USER_TTL
}
```

Model

Field	Type	Description
login	string	Web service user login
token	string	Web service user JWT token
TTL	integer	Time to live of the token in seconds

Error messages

Error code	HTTP header code	Possible cause
InvalidCredentials	401	Invalid login and/or invalid password
InvalidJson	400	The request body does not follow the body model
NotHandledError	500	Error not covered in current implementation

JWT token

The JWT token is necessary to use the API the other endpoints of the API. It must be provided in the *Authorization* header of all subsequent requests.

The contents of the token changes from Kuma. It now contains the following informations: - login: the login of the user - expiresAt: the date at which the token expires (in milliseconds since the epoch) - issuedAt: the date at which the token was issued (in milliseconds since the epoch) - userType: the type of the user (*cti* or *webservice*) - acls: the permissions of the user (see *REST API Permissions*)

As a consequence, any token generated before Kuma will not work anymore and users need to re-login.

This token can then be used with the *CTI Authentication* and *Check and renew JWT token*.

Single sign-in (SSO authentication)

Note: Only CTI users can be authenticated using SSO.

Authentication with Kerberos

Request

Method	GET
URI	/auth/sso

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/auth/sso
```

Response

```
{
   login: "$CTI_USER_LOGIN",
   token: "$CTI_USER_JWT_TOKEN",
}
```

Model

Field	Type	Description
login	string	CTI user login
token	string	CTI user JWT token

This token can then be used with the *CTI Authentication* and *Check and renew JWT token*.

Error messages

Error code	HTTP header code	Possible cause
UserNotFound	401	The user does not exist
SsoAuthenticationFailed	401	The user could not be authenticated using Kerberos
NotHandledError	500	Error not covered in current implementation

Error messages

Error code	HTTP header code	Possible cause
UserNotFound	401	The user does not exist
SsoAuthenticationFailed	401	The user could not be authenticated using Kerberos
NotHandledError	500	Error not covered in current implementation

Authentication with CAS

Request

Method	GET
URI	/auth/cas

URL parameters

Field	Type	Description
service	string	The URL of the CAS server

service | string | The URL of the CAS server |

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/auth/cas?service=https://cas.myplatform.
↪com&ticket=ST-11-Qsicgrh1mZ3dgoeOx7m6-af27d9025e0c
```

Response

```
{
   login: "$CTI_USER_LOGIN",
   token: "$CTI_USER_JWT_TOKEN",
}
```

Model

Field	Type	Description
login	string	CTI user login
token	string	CTI user JWT token

This token can then be used with the *CTI Authentication* and *Check and renew JWT token*.

Error messages

Error code	HTTP code	header	Possible cause
UserNotFound	401		The user does not exist
CasServerUrlNotSet	403		XiVOCC containers are not configured (see *CAS SSO Authentication*)
CasServerInvalidResponse	403		The CAS server returned an invalid response
CasServerInvalidParameter	403		The Parameters sent to the CAS Server are invalid
CasServerInvalidRequest	403		The Request to the CAS server is invalid
CasServerInvalidTicketSpec	403		The ticket specification is invalid
CasServerUnauthorizedServiceProxy	403		The CAS service proxy is not authorized
CasServerInvalidProxyCallback	403		The CAS service proxy callback is invalid
CasServerInvalidTicket	403		The ticket is invalid (probably expired or defined for another service)
CasServerInvalidService	403		The service is invalid
CasServerInternalError	403		CAS Server internal error
NotHandledError	500		Error not covered in current implementation

Authentication with OpenID Connect (OIDC)

Request

Method	GET
URI	/auth/oidc

Note: The token can also be provided in the *Authorization* header using the *Bearer* scheme.

URL parameters

Field	Type	Description
token	string	CTI user JWT token

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/auth/oidc?token=${OAUTH_TOKEN}
```

Response

```
{
   login: "$WEBSERVICE_USER_LOGIN",
   token: "$WEBSERVICE_USER_JWT_TOKEN",
   TTL: $WEBSERVICE_USER_TTL
}
```

Model

Field	Type	Description
login	string	CTI user login
token	string	CTI user JWT token

This token can then be used with the *CTI Authentication* and *Check and renew JWT token*.

Error messages

Error code	HTTP header code	Possible cause
UserNotFound	401	The user does not exist
OidcNotEnabled	403	OpenID connect authentication is not enabled (see *OpenID Connect SSO Authentication*)
OidcInvalidParameter	403	Missing access_token
OidcAuthenticationFailed	403	The access_token forwarded to XUC is invalid
NotHandledError	500	Error not covered in current implementation

Check and renew JWT token

Any user, either CTI or Web service, can check and renew their JWT token.

Request

Method	GET
URI	/auth/check
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/auth/check \
-H "Authorization: Bearer user_jwt_token"
```

Response

```
{
   login: "$USER_LOGIN",
   token: "$USER_JWT_TOKEN",
   TTL: $USER_TTL
}
```

Model

Field	Type	Description
login	string	CTI user login
token	string	CTI user JWT token
TTL	integer	Time to live of the token in seconds

Error messages

Error code	HTTP code	header	Possible cause
InvalidToken	403		Token does not match XUC signature tokens
InvalidJson	400		The request body is not a valid JSON or does not follow the body model
BearerNotFound	400		The token is not found in the *Authorization* header
AuthorizationHeaderNot-Found	400		The *Authorization* header is not found in the request
TokenExpired	403		The *Authorization* header is not found in the request
NotHandledError	500		Error not covered in current implementation

Personal contacts

CTI users can use the endpoints related to personal contacts.

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": ${ERROR_MESSAGE}
}
```

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 all personal contacts

Request

Method	GET
URI	/contact/personal
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/contact/personal \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Response

```
{
  "entries": [
      { "status": 0, "entry": [ "hawkeye", "pierce", "1002", "0761187406", "false"]},
      { "status": -2, "entry": [ "peter", "pan", "1004", "", "false"]}],
  "headers":
      ["Firstname", "Lastname", "Number", "Mobile", "Favorite"]
 }
```

Retrieve a specific personal contact

Request

Method	GET
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/personal/${CONTACT_UUID}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
CONTACT_UUID	string	UUID of the requested personal contact

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/contact/personal/28079dc0-2c6b-4332-
↪9075-61da9229389f \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Response

```
{
   "id":"28079dc0-2c6b-4332-9075-61da9229389f",
   "firstname":"doe",
   "lastname":"john",
   "number":"1111",
   "mobile":"2222",
   "fax":"3333",
   "email":"j.doe@my.corp",
   "company":"corp"
}
```

Create a personal contact

Request

Method	POST
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/personal
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
lastname	string	Last name of the personal contact
firstname	string	First name of the personal contact
company	string	Company of the personal contact
email	string	Email address of the personal contact
number	string	Phone number of the personal contact
mobile	string	Mobile number of the personal contact
fax	string	Fax number of the personal contact

Note: At least one of the following fields must be set: *number*, *mobile*, *fax*, *email* **and** *lastname* and/or *firstname*.

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/contact/personal \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token" \
-d '{"firstname":"doe","lastname":"john","number":"1111","mobile":"2222","fax":"3333",
↪"email":"j.doe@my.corp","number":"1111","mobile":"2222","fax":"3333"}'
```

Response

```
{
   "id":"28079dc0-2c6b-4332-9075-61da9229389f",
   "firstname":"doe",
   "lastname":"john",
   "number":"1111",
   "mobile":"2222",
   "fax":"3333",
   "email":"j.doe@my.corp",
   "company":"corp"
}
```

Update a personal contact

Request

Method	PUT
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/personal/${CONTACT_UUID}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
CONTACT_UUID	string	UUID of the requested personal contact

Body parameters

Field	Type	Description
lastname	string	Last name of the personal contact
firstname	string	First name of the personal contact
company	string	Company of the personal contact
email	string	Email address of the personal contact
number	string	Phone number of the personal contact
mobile	string	Mobile number of the personal contact
fax	string	Fax number of the personal contact

Note: All fields are optional

Curl example

```
curl -X PUT http://localhost:8090/xuc/api/2.0/contact/personal/28079dc0-2c6b-4332-
↪9075-61da9229389f \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token" \
-d '{"firstname":"doe","lastname":"john","number":"1111","mobile":"2222","fax":"3333",
↪"email":"j.doe@my.corp","number":"1111","mobile":"2222","fax":"3333"}'
```

Response

```
{
   "id":"28079dc0-2c6b-4332-9075-61da9229389f",
   "firstname":"doe",
   "lastname":"john",
   "number":"1111",
   "mobile":"2222",
   "fax":"3333",
   "email":"j.doe@my.corp",
   "company":"corp"
}
```

Delete a personal contact

Request

Method	DELETE
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/personal/${CONTACT_UUID}
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
CONTACT_UUID	string	UUID of the requested personal contact

Curl example

```
curl -X DELETE http://localhost:8090/xuc/api/2.0/contact/personal/28079dc0-2c6b-4332-
↪9075-61da9229389f \
-H "Authorization: Bearer user_jwt_token"
```

Response

It will return a 204 response.

Delete all personal contacts

Request

Method	DELETE
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/personal
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X DELETE http://localhost:8090/xuc/api/2.0/contact/personal \
-H "Authorization: Bearer user_jwt_token"
```

Response

It will return a 204 response.

Export personal contacts

Request

Method	GET
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/export/personal
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/contact/export/personal \
-H "Authorization: Bearer user_jwt_token"
```

Response

It will return a csv file with UTF-8 encoding containing all personal of a user

```
company,email,fax,firstname,lastname,mobile,number
corp,jdoe@company.corp,3333,John,Doe,2222,1111
```

Import personal contacts

Request

Method	POST
URI	http://localhost:${XUC_PORT}/xuc/api/2.0/contact/import/personal
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X POST http://localhost:${XUC_PORT}/xuc/api/2.0/contact/import/personal \
-H "Authorization: Bearer user_jwt_token" \
-F 'file=@/path/to/file.csv;type=text/csv'
```

Note: A CSV file must be sent as the body of the request. The first line of the CSV file must be the header. The file may contain the following fields: *company*, *email*, *fax*, *firstname*, *lastname*, *mobile*, *number*. The fields are optional. However, at least one of these fields is required: *number*, *mobile*, *fax*, *email* **and** *lastname* and/or *firstname* The fields must be separated by a comma. The CSV file must be encoded in UTF-8.

```
company,email,fax,firstname,lastname,mobile,number
corp,jdoe@company.corp,3333,John,Doe,2222,1111
```

Response

The request will create and return the response for each personal contact (either success or failure) (HTTP code 201).

Example

```
{
   "created": [{
      "id": "e62ee672-f74a-498c-b138-86ac1b0ae429",
      "lastname": "Wayne"
   }, {
      "id": "d3a67f8e-3d8a-465a-9633-bde65a41f1bb",
      "lastname": "Hawking"
   }],
   "failed": [{
      "line": 3,
      "errors": ["too many fields"]
   }]
}
```

Call 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": ${ERROR_MESSAGE}
}
```

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 call qualifications for a queue

Request

Method	GET
URI	/call_qualification/queue/${QUEUE_ID}
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/call_qualification/queue/42 \
-H "Authorization: Bearer user_jwt_token"
```

Response

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 call qualifications

Request

Method	POST
URI	/call_qualification
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/call_qualification \
-H "Authorization: Bearer user_jwt_token"
```

Response

```
{
   "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"
}
```

Export call qualifications

Request

Method	GET
URI	/call_qualification/csv/${QUEUE_ID}/${FROM}/${TO}
Header	Authorization: Bearer ${JWT_TOKEN}

Curl example

```
curl -X GET http://localhost:8090/xuc/api/2.0/call_qualification/42/home/office \
-H "Authorization: Bearer user_jwt_token"
```

Response

It will return a csv file with UTF-8 encoding containing all call qualification answers 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
```

Mobile application

To configure/handle XiVO mobile phone application

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	Token sent to API is not found
JsonErrorDecoding	400	Request sent to API is not JSON valid
NotHandledError	500	Error not covered in current implementation

Set push registration token

Warning: This Api is deprecated, it is replaced by Set Android push registration token & Set iOS push registration token APIs

Request

Method	POST
URI	/mobile/push/register
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
token	string	xuc token

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/mobile/push/register  \
-d '{"token": "1234-5678"}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Response

The response will register push notification token to wake up the mobile application answer (HTTP code 201).

Set Android push registration token

Request

Method	POST
URI	/mobile/push/register/android
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
token	string	xuc token

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/config/mobile/push/register/android?
↪apiVersion=2.0  \
-d '{"token": "1234-5678"}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Response

The response will register android push notification token to wake up the mobile application answer (HTTP code 201).

Set iOS push registration token

Request

Method	POST
URI	/mobile/push/register/ios
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
token	string	xuc token

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/config/mobile/push/register/ios?
↪apiVersion=2.0  \
-d '{"token": "1234-5678"}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Response

The response will register iOS push notification token to wake up the mobile application answer (HTTP code 201).

Dial

Multiple dial APIs are available to generate dial actions from 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": ${ERROR_MESSAGE}
}
```

Else, the API will return the following JSON format

```
{
   "res": "success"
}
```

Error codes

Error code	HTTP header code	Possible cause
BadRequest	400	The body is missing or is invalid
NotHandledError	500	Error not covered in current implementation

Dial a number

This request dials the destination using the username of a CTI user provided in the url, also passing optional variables along.

Note: The CTI user whose username is provided must be an authenticated.

Warning: This action is only available for web service users with *xuc.rest.dial.number.*.create* ACL.

Request

Method	POST
URI	/dial/number/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
destination	string	Number to be called someone is in queue
variables	object	Any other variables

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/dial/number/jbond \
-d '{"destination":"0123456789", "variables": { "varname1": "varval1",  "varname2":
↪"varval2"}}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Dial a user

This request dials the payload username using the username provided in the url, also passing optional variables along.

Note: The CTI user whose username is provided must be an authenticated.

Warning: This action is only available for web service users with *xuc.rest.dial.user.*.create* ACL.

Request

Method	POST
URI	/dial/user/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
username	string	Username of the user to call
variables	object	Any other variables

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/dial/user/jbond \
-d '{"username":"drno", "variables": { "varname1": "varval1",  "varname2": "varval2"}}
↪' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Dial from queue

This request dials the destination from a queue using the username provided in the url, also passing optional variables along.

Warning: This action is only available for web service users with *xuc.rest.dial.fromqueue.create* ACL.

Request

Method	POST
URI	/dial/fromqueue/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
destination	string	Number to be called someone is in queue
queueId	Int	Identifier of the queue
variables	object	Any other variables

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/dial/fromqueue/jbond  \
-d '{"destination":"0123456789", "queueId": 152, "variables": { "varname1": "varval1",
↪  "varname2": "varval2"}}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Dial to queue

This request dials the destination with a specific caller ID and then add it into a queue, also passing optional variables along.

Warning: This action is only available for web service users with *xuc.rest.dial.toqueue.create* ACL.

Request

Method	POST
URI	/dial/toqueue
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
destination	string	Number to be called someone is in queue
queueName	string	Name of the queue
callerIdNumber	string	Number displayed to people in queue
variables	object	Any other variables

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/dial/toqueue  \
-d '{"destination":"0123456789", "queueName":"support", "callerIdNumber":"0403010200",
↪ "variables": { "varname1": "varval1",  "varname2": "varval2"}}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Do not disturb status and forwarding

Multiple APIs are available to control user forwards and DND status

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>
}
```

Else, the API will return the following JSON format

```
{
   "res": "success"
}
```

Error codes

Error code	HTTP header code	Possible cause
JsonBodyNotFound	400	Token sent to API is not found
JsonErrorDecoding	400	Request sent to API is not JSON valid
UserNotFound	400	User not found
NotHandledError	500	Error not covered in current implementation

Set do not disturb status

This request will set the do not disturb status of the user provided in the url according to the state in the payload.

Warning: This action is only available for web service users with *xuc.rest.dnd.*.create* ACL.

Request

Method	POST
URI	/dnd/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
state	boolean	Whether the do not disturb status is enabled or not

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/dnd/jbond  \
-d '{"state": true}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Unconditional forwarding

This requests sets the unconditional forwarding of the user provided in the url according to the payload.

> **Warning:** This action is only available for web service users with *xuc.rest.forward.unconditional.*.create* ACL.

Request

Method	POST
URI	/forward/unconditional/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
destination	string	Destination of the forwarding
enabled	boolean	State of the unconditional forwarding

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/forward/unconditional/jbond  \
-d '{"destination": "0123456789", "enabled": true}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

No answer forwarding

This requests sets the no answer forwarding of the user provided in the url according to the payload

Warning: This action is only available for web service users with *xuc.rest.forward.noanswer.*.create* ACL.

Request

Method	POST
URI	/forward/noanswer/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
destination	string	Destination of the forwarding
enabled	boolean	State of the no answer forwarding

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/forward/noanswer/jbond  \
-d '{"destination": "0123456789", "enabled": true}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Busy forwarding

This requests sets the busy forwarding of the user provided in the url according to the payload.

Warning: This action is only available for web service users with *xuc.rest.forward.busy.*.create* ACL.

Request

Method	POST
URI	/forward/busy/${USERNAME}
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Body parameters

Field	Type	Description
destination	string	Destination of the forwarding
enabled	boolean	State of the busy forwarding

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/forward/busy/jbond  \
-d '{"destination": "0123456789", "enabled": true}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

User call history

Multiple APIs are available to retrieve user call history

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>
}
```

Else, the API will return the following JSON format:

```
{
   "res": "success",
   "data": {...}
}
```

Error codes

Error code	HTTP header code	Possible cause
JsonBodyNotFound	400	Token sent to API is not found
JsonErrorDecoding	400	Request sent to API is not JSON valid
UserNotFound	400	User not found
NotHandledError	500	Error not covered in current implementation

User call history

This request retrieves the call history of the user provided in the url.

Warning: This action is only available for web service users with *xuc.rest.history.*.read* ACL.

Request

Method	GET
URI	/history/${USERNAME}
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Path parameters

Field	Type	Description
size	integer	Number of calls to retrieve

Curl example

```
curl http://localhost:8090/xuc/api/2.0/history/jbond?size=10  \
-H "Authorization: Bearer user_jwt_token"
```

User call history by days

This request retrieves the call history of the user provided in the url.

Warning: This action is only available for web service users with *xuc.rest.history.days.*.read* ACL.

Request

Method	GET
URI	/history/days/${USERNAME}
Header	Authorization: Bearer ${JWT_TOKEN}

URL parameters

Field	Type	Description
USERNAME	string	Username of the CTI user to make the call

Path parameters

Field	Type	Description
days	integer	Number of days to retrieve

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/history/days/jbond?days=10  \
-H "Authorization: Bearer user_jwt_token"
```

Agent callbacks and tickets

Import and export callback requests and tickets

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>
}
```

Else, the API will return the following JSON format

```
{
   "res": "success",
   "data": {...}
}
```

Error codes

Error code	HTTP header code	Possible cause
JsonBodyNotFound	400	Token sent to API is not found
JsonErrorDecoding	400	Request sent to API is not JSON valid
UserNotFound	400	User not found
NotHandledError	500	Error not covered in current implementation

Import callback requests

This requests imports a csv of callback requests as text.

Warning: This action is only available for web service users with *xuc.rest.callback_lists.callback_requests.csv.create* ACL.

Request

Method	POST
URI	/callback_lists/callback_requests/csv
Header	Content-Type: text/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
listUuid	string	UUID of the list of callback requests
csv	string	CSV contents of the callback requests

Note: The *csv* parameter consists in one line: each line break should be replace with *n*.

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/callback_lists/callback_requests/csv \
-H "Content-Type: text/json" \
-H "Authorization: Bearer user_jwt_token" \
-d '{"listUuid": "1", "csv":
↪"phoneNumber|mobilePhoneNumber|firstName|lastName|company|description|dueDate|period\
↪n0230210092|0689746321|John|Doe|MyCompany|Call back quickly||\
↪n0587963214|0789654123|Alice|O'Neill|YourSociety||2016-08-01|Afternoon"}'
```

Export callback tickets

This requests exports the list of tickets associated with the callback list uuid.

> **Warning:** This action is only available for web service users with *xuc.rest.callback_lists.callback_tickets.csv.create* ACL.

Request

Method	POST
URI	/callback_lists/callback_tickets/csv
Header	Content-Type: application/json
Header	Authorization: Bearer ${JWT_TOKEN}

Body parameters

Field	Type	Description
listUuid	string	UUID of the list to export

Curl example

```
curl -X POST http://localhost:8090/xuc/api/2.0/callback_lists/callback_tickets/csv  \
-d '{"listUuid": "jsdfh-90298J-hdjkfd"}' \
-H "Content-Type: application/json" \
-H "Authorization: Bearer user_jwt_token"
```

Deprecated APIs

> **Warning:** The following APIs are now deprecated. You can re-enable them to specific client based on their IP address. To allow a client, add its IP address to the DEPRECATED_API_HOST key in the *XiVO CC* `custom.env` file. You can add multiple address separated by comma (,).

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>/
```

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].

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>/
```

[1] See Redmine ticket #150

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.

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>/
```

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 by size

Get the last X calls of the 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"}
]
```

User Call History by days

Get user call history of the last X days

```
curl -XGET -H "Content-Type: application/json"  http://localhost:8090/xuc/api/1.0/
↪historyByUsername/<domain>/<username>?days=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"}
]
```

Export 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"}}}}
```

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
PercentageAnsweredTotal	Percentage of calls answered compared to calls entered
PercentageAnsweredBefore15	Percentage of calls answered before 15 seconds over total number of calls entered
PercentageAbandonnedTotal	Percentage of calls abandonned compared to 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.

If configured, these additional stats threshold will be automatically available in queue view of CC Manager, see *Thresholds*.

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

name	Description
AvailableAgents	Number of agents available to take calls
TalkingAgents	Number of agents in communication
LongestWaitTime	Longest wait time achieved by a call currently waiting in the queue
WaitingCalls	Number of calls currently waiting in the queue
EWT	Estimated waiting time in the queue

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.

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.

Third Party Integration

Third party web application integration is possible inside the Xucmgt CCAgent application. Upon each call, you can display a custom panel next to the agent interface:

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"

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).

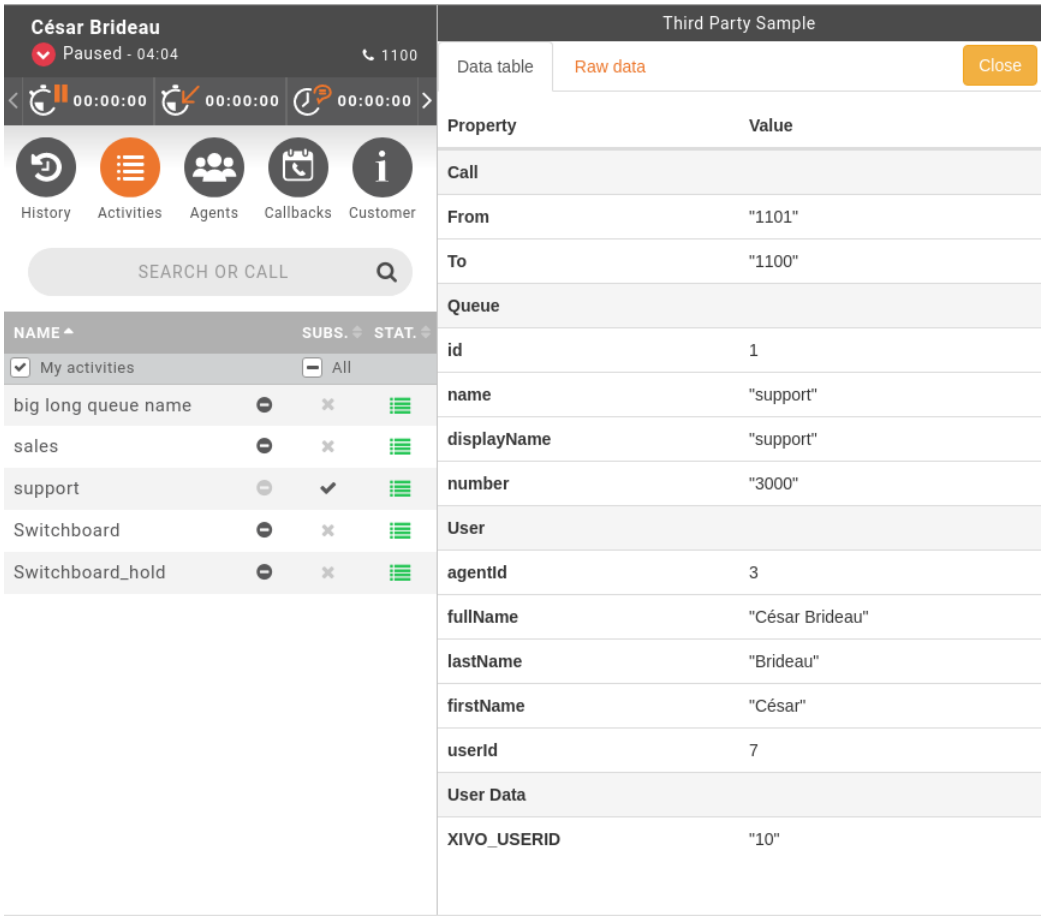
When the work is complete in the integrated application, the application must post a message to terminate the third party application pane inside the agent application (see *Completion*).

Configuration

You need to specify the third party application web service url to integrate this application inside the Xucmgt CCAgent 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.



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": 54,
    "agentId": 29,
    "firstName": "James",
    "lastName": "Bond",
    "fullName": "James Bond",
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJsb2dpbiI6Im13ZWJydGMiLCJleHBpcmVzQXQiOjE3MjQ3NjY5NzAsImlzc3VlZEF0IjoxNzI0NzYzMzcwLCJ1c2VyVHlwZS
↪EdKGjpNnV19BnQCiVXAiAfUEfirmZ6WvQhBDtEV6pQ4"
  },
  "callee": "1023",
  "caller": "1013",
  "userData": {
    "XIVO_USERID": "54",
    "SIPCALLID": "fd32a3a3-9647-457e-be04-16d7d408f271",
    "USR_XIVO_DSTNUM": "3001"
  },
  "callDataCallId": "1724763376.29",
  "queue": {
    "id": 2,
```

(continues on next page)

(continued from previous page)

```
    "name": "edition",
    "displayName": "Edition",
    "number": "3001"
  }
}
```

- `user` contains the connected user information
- `callee` contains the number called
- `queue` queue properties
- `callDataCallId` contains the linkedId of the call
- `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
  - `run`: Will run the given `url` as an executable, only works in desktop assistant.
  - `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.
- `executableArgs` can contain an array of argument for the `run` action.
- `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)

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);
})();
```

Third Party Login URL Integration

To know on which XiVO a user is logged, you can configure a URL with some information about the user when they log in the application.

Configuration

You need to specify the third party application web service url to integrate this application inside the XucMgt Assistant and Agent interfaces. This can be done by giving a value to the `THIRD_PARTY_LOGIN_URL` environment variable in the `/etc/docker/compose/custom.env` file

```
...
THIRD_PARTY_LOGIN_URL=https://some.url.com/logininfo?login=%{login}&token=%{token}&
↪xivoccHost=%{xivocchost}
```

The service will receive the url containing the username, token and XiVOCC host of the user who just logged in.

7.1.2 Recording API

This section describes the Recording server REST API.

In the following, all urls are relative to the recording server base url and port. For example a relative URL `/recording/records/search` is meant to be replaced by `http://192.168.29.101:9400/recording/records/search` assuming the recording server is available on port `9400` at `192.168.29.101`

Authentication

To use the recording server API you need to add an additional header in every HTTP Request. The header name is `X-Auth-Token` and its value must be the same as the `PLAY_AUTH_TOKEN` environment variable defined in the `custom.env` of the docker-compose environment hosting the recording server container (see *Shared token*).

Example:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json"
         -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}"  http://192.168.29.101:9400/recording/
↪records/search?pageSize=3  -d '{"agent": "1573"}'
```

Records

Search

This api allows to search for recorded **answered** calls in the database using criteria.

Description:

URL
`/recording/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 or part of the name to filter on

queue
The queue number or part of the name 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. % char can be used as wildcard.

caller
Return only calls with the give source number. % char can be used as wildcard.

direction
Filter calls based on the call direction, one of `incoming`, `outgoing` or `all`

queueCallStatus
Filter based on queue status `abandoned` `answered` `divert_ca_ratio` `divert_waittime` `closed` `full` `joinempty` `leaveempty` `timeout`

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: ${PLAY_AUTH_TOKEN}"  http://192.168.29.101:9400/recording/
↪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 **any** recorded call based on call id.

Description:

URL
`/recording/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: ${PLAY_AUTH_TOKEN}"  http://192.168.29.101:9400/recording/
↪records/callid_search?callid=1459435466.286075
```

Response:

```
{
    "hasNext": false,
    "records": [
        {
            "agent": "Dino Falconetti (1564)",
            "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
`/recording/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: ${PLAY_AUTH_TOKEN}"
          http://192.168.29.101:9400/recording/records/xivocc_gateway-1459435465.
↪15089/audio/download
```

Attach call data

This api allows to attach data to a given call

Description:

URL
`/recording/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: ${PLAY_AUTH_TOKEN}"
          http://192.168.29.101:9400/recording/call_data/761054/attached_data  -d '[{
↪"key":"color", "value": "green"}]'
```

History

Search

This API gives the call history of a given interface.

It returns 10 results by default - one can use the parameter `size` to specify how many results the query should return.

Description:

URL
`/recording/history`

Method
`POST`

Url parameters

size
The maximum number of results 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: ${PLAY_AUTH_TOKEN}" http://localhost:9400/recording/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 number of days

This API gives the call history of a given interface from today till a given number of days ago.

Use the parameter `days` to specify the number of days to return the history for.

With no parameters, the query will fall back on the previous API and display the 10 last calls.

Warning: do not use the parameter `days` from this API and the parameter `size` from the previous API at the same time, as it will cause an error.

Description:

URL
`/recording/history`

Method
`POST`

Url parameters

days
The number of days to display

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: ${PLAY_AUTH_TOKEN}" http://localhost:9400/recording/history?days=7  -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 agent number

This API gives the call history of a given agent number.

Description:

URL
`/recording/history/agent`

Method
`POST`

Url parameters

size
The maximum number of result to return (*default:* 10)

days
The number of days to retrieve (*default:* 7)

> **Warning:** Take care that the greater the number of days you ask to retrieve, the heavier the request will be to the databvase (depending on the number of calls in the dababase).

Request Body
A json object with a field named `agentNum` containing the agent number 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/recording/history/agent  -d '{"agentNum":"1004"}'
```

Response:

```
[
 {
  "start": "2021-01-19 16:19:54",
  "duration": "00:00:02",
  "src_num": "1008",
  "dst_num": "1023",
  "status": "emitted",
  "src_firstname": "Yealink",
  "src_lastname": "T54W",
  "dst_firstname": "Marc",
  "dst_lastname": "WebRTC"
 }
]
```

Search by customer

This api helps to find call history of a customer thanks to a list of predefined filters.

Description:

URL
`/recording/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: ${PLAY_AUTH_TOKEN}"
     http://localhost:9400/recording/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
`/recording/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: ${PLAY_AUTH_TOKEN}" 'http://localhost:9400/recording/last_agent?
↪callerNo=1002&since=100'
```

Response:

```
{"agentNumber":"2000"}
```

Channel Events Logging

Get CEL

This api retrieve the CEL from the whole XDS System (main and media servers).

Description:

URL
`/recording/cel`

Method
`GET`

Url parameters

from
The first cel id to retrieve

limit
The number of cel to retrieve

Example Query:

```
curl H "Content-Type: application/json" -H "Accept: application/json" -H 'X-Auth-
↪Token: u@pf#41[gYHJm<]9N[a0iWDQQ7`e9k' 'http://localhost:9400/recording/cel?from=10&
↪limit=1000'
```

Response:

```
[
{
  "id": "11",
  "eventtype": "LINKED_END",
  "eventtime": "2020-03-23 16:21:25.722316",
  "userdeftype": "",
  "cid_name": "James Bond",
  "cid_num": "1000",
  "cid_ani": "1000",
  "cid_rdnis": "",
  "cid_dnid": "",
  "exten": "s",
  "context": "echotest",
  "channame": "SIP/4uqyf9q-00000001",
  "appname": "",
  "appdata": "",
  "amaflags": "3",
  "accountcode": "",
  "peeraccount": "",
  "uniqueid": "1584976887.1",
  "linkedid": "1584976887.1",
  "userfield": "",
  "peer": "",
  "call_log_id": "",
  "extra": ""
},
{...}
]
```

7.1.3 Configuration API

This section describes XiVO Configuration server API. These APIs ease the usage of legacy apis and is supported by the configuration server dockerized component (config-mgt). These are mainly REST APIs.

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`

Important: All configmgt APIs are described in a Swagger UI available at `/api` for example : `http://192.`

168.29.101/configmgt/api.

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 `PLAY_AUTH_TOKEN` environment variable defined in the `custom.env` of the docker-compose environment hosting the configuration server container (see *Shared token*).

Example:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
     -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}"  http://localhost:9100/configmgt/api/1.0/
↪callback_lists
```

Dynamic filters

A **dynamic filter** is just a JSON representation to create lite look-a-like SQL assertions. It contains:

- **field** (string): **required** Field to make your query on
- **operator**: Can be one of = , *!=* , > , >= , < , <= , *like* , *ilike* , *is null* , *is not null*, *contains_all*
- **value** (string, number, boolean): value to filter on
- **list** (array): list of values to filter on. Used only with the operator *contains_all*
- **order**: can be *ASC* or *DESC*

Examples :

Return all the users ordered by the fullName ascending:

```
{"field":"fullName", "order": "ASC"}
```

Filter and return all the users with a fullName beginning with *Jack* and ordering them ascending:

```
{"field":"fullName", "operator":"like", "value":"Jack%", "order": "ASC"}
```

Return the users named *James Bond*:

```
{"field":"fullName", "operator":"=", "value":"James Bond"}
```

Return the users associated with (at least) labels *red* and *blue* (i.e. all users belonging to both labels *red* and *blue*):

```
{"field":"label", "operator":"contains_all", "list":["blue", "red"]}
```

Users

Following API allow to list and find XiVO users

Get all users

This API retrieves all XiVO users.

Description:

URL
`api/2.0/users`

Method
`GET`

Example Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/users
↪'
```

Response:

```
[
   {
      "fullName":"user A",
      "provisioning":583115,
      "lineType":"phone",
      "phoneNumber":"1000",
      "entity":"Test",
      "mdsName":"default",
      "mdsDisplayName":"MDS Main",
      "labels":[
         "blue",
         "green",
         "red"
      ]
   },
   {
      "fullName":"user B",
      "provisioning":273444,
      "lineType":"webrtc",
      "phoneNumber":"1001",
      "entity":"Test",
      "mdsName":"default",
      "mdsDisplayName":"MDS Main",
      "labels":[
         "green",
         "red"
      ]
   }
]
```

Find users

Finds some users using dynamic filters.

Description:

URL
`/api/2.0/users/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

limit
Max number of results to return

List of filterable columns

- phoneNumber
- fullName
- entity
- mdsDisplayName

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/find' \
  -d '
    {
      "filters":[
        {"field":"phoneNumber", "operator":"=", "value":"1000", "order": "ASC"}
      ],
      "offset":0,
      "limit":100}'
```

Response:

```
{
 "total":1,
 "list":[
    {
       "fullName":"user A",
       "provisioning":583115,
       "lineType":"ua",
       "phoneNumber":"1000",
       "entity":"Test",
```

(continues on next page)

(continued from previous page)

```
        "mdsName":"default",
        "mdsDisplayName": "MDS Main",
        "labels":[
           "blue",
           "green",
           "red"
        ]
     }
  ]
}
```

Get user's line

This API retrieves XiVO user line associated to it.

Description:

URL
`api/2.0/users/:id/line`

Method
`GET`

Line Types:

phone
Physical SIP device

webrtc
Web SIP softphone

ua
Unique account, which is hybrid line allowing webrtc if connected to assistant, otherwise use phone if not connected.

sccp
For Cisco non SIP phones (unsupported)

custom
Customized endpoint which is maybe not a device

Example Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/1/line'
```

Response:

```
{
      "id": 40,
      "lineType": "phone",
      "context": "default",
      "extension": "1000",
      "site": "default",
      "name": "nrzpjngu",
      "lineNum": 1,
      "provisioningId": 661997
}
```

Result Code

- 200 : The user line is found
- 404 : The user line is not found

Create / Update user's line

These API allow to create/update XiVO user line.

Description:

URL
`api/2.0/users/:id/line`

Method
`POST` for creation, `PUT` for update

Url parameters

id
user's id

Request body
Json object with field & value pair.

Allowed field names:

lineType
One of *phone*,*webrtc*,*ua*,*custom*

extension
user line phone number

context
Context of the user (e.g. default)

site
Xivo where will be located the user (e.g. default, mds1...)

device
Hash that reprensent device in XiVO (Optional)

lineNum
Line slot to use on the device itself (not relevant for webrtc line)

Example Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/1/line' \
  -d '
    {
      "lineType":"phone",
      "context":"default",
      "site":"default",
      "extension":"1000",
      "lineNum":1
    }'
```

Response:

```
{
  "id": 40,
  "lineType": "phone",
  "context": "default",
  "extension": "1000",
  "site": "default",
  "name": "nrzpjngu",
  "lineNum": 1,
  "provisioningId": 661997
}
```

Result Code

- 200 : The user line is created
- 404 : The user line cannot be found or created/updated

If the user line already exists:

```
{
  "error": "LineNotFound",
  "message": "Unable to create line for user Id 1"
}
```

Delete user's line

These API allow to delete a line of a XiVO user.

Description:

URL
`api/2.0/users/:id/line`

Method
`DELETE`

Url parameters

id
user's id

Example Query:

```
curl -XDELETE -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/1/line'
```

Result Code

- 204 : The user line is successfully deleted
- 404 : The user is not found

If the user line is not found:

```
{
  "error": "LineNotFound",
  "message": "Unable to delete line for user Id 1"
}
```

Validate a user

This API allow to validate a user used by authentication.

Description:

URL
`api/2.0/user/validate`

Method
`POST`

Example Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
-H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/user/
↪validate' \
-d '
  {
    "username":"john",
    "password":"doe"
    }'
```

Result Code

- 204 : The user is successfully validated
- 404 : The user is not found

If the user line is not found:

```
{
  "error": "NotFoundError",
  "message": "User not found"
}
```

User services

Following API allow to get and edit user services like forwards and do not disturb feature.

Get user services

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/1/services'
```

Response:

```
{
  "dndEnabled": false,
  "busy": {
    "enabled": false,
    "destination": "1002"
  },
  "noanswer": {
    "enabled": false,
```

(continues on next page)

(continued from previous page)

```
    "destination": "123"
  },
  "unconditional": {
    "enabled": false,
    "destination": "1001"
  }
}
```

Update user services

Query:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/1/services' \
  -d '
    {
      "dndEnabled": true,
      "busy": {
        "enabled": false,
        "destination": "1002"
      },
      "noanswer": {
        "enabled": false,
        "destination": "123"
      },
      "unconditional": {
        "enabled": false,
        "destination": "1001"
      }
  }'
```

Response:

```
{
  "dndEnabled": true,
  "busy": {
    "enabled": false,
    "destination": "1002"
  },
  "noanswer": {
    "enabled": false,
    "destination": "123"
  },
  "unconditional": {
    "enabled": false,
    "destination": "1001"
  }
}
```

It is also possible to only update partial information:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/1/services' \
  -d '{"dndEnabled": false}'
```

Response:

```
{
  "dndEnabled": false,
}
```

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 users' profiles

This API retrieves XiVO users having a login and a profile defined.

Description:

URL
`/users`

Method
`GET`

Example Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" '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 rights of a XiVO user having a 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: ${PLAY_AUTH_TOKEN}" '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,
  "dissuasionAccess":false}
}
```

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 (*admin*, *supervisor* or *teacher*)

data
The rights to update

queueIds
IDs of the queues the supervisor has access to (for supervisors and teachers)

groupIds
IDs of the groups the supervisor has access to (for supervisors and teachers)

incallIds
IDs of the incalls the supervisor has access to (for supervisors and teachers)

recordingAccess
whether or not the supervisor can access recordings (for supervisors only, true by default)

dissuasionAccess
whether or not the supervisor can update dissuasion destinations (for supervisors only, false by default)

Example Query:

```
curl -v -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/1.0/
↪rights/user/jbond' \
  -d '{"type":"supervisor",
        "data":{  "queueIds":[3,2,1,7],
                  "groupIds":[3,1,2],
                  "incallIds":[],
                  "recordingAccess":true,
                  "dissuasionAccess":false  }
      }'
```

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: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/1.0/
↪rights/user/jbond'
```

Users preferences

Following APIs allow the management of the users preferences.

List of preferences :

- `preferred_device` : For unique account users, this property is the value of the default device used by the user with the applications. It can be either `phone` or `webrtc`.

Get all the user's preferences

Get all the preferences for a user.

Description:

URL
`/users/:id/preferences`

Method
`GET`

Url parameters

id
user's id

Example

Query:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/3/preferences'
```

Result:

```
[{
  "key": "preferred_device",
  "value": "phone",
  "value_type" : "String"
},
{
  "key": "other_preference",
  "value": "42",
  "value_type" : "number"
}]
```

Get a user's preference

Get a specific preference for a user.

Description:

URL
`/users/:id/preferences/:preference_key`

Method
`GET`

Url parameters

id
user's id

preference_key
key of the preference

Example

Query:

```
curl -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/3/preferences/preferred_device'
```

Result:

```
{
  "value": "phone",
  "value_type" : "String"
}
```

Create user preference

Initialize the value of a preference for a user.

Description:

URL
`/users/:id/preferences/:preference_key`

Method
`POST`

Url parameters

id
user's id

preference_key
key of the preference

Request body
Json object with field & value pair.

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/3/preferences/preferred_device'
  -d {"value": "phone", "value_type": "String"}
```

Result Code

- 204 : The preference is successfully created
- 409 : The preference is already set for this user

Update user preference

Update the value of a preference for a user.

Description:

URL
`/users/:id/preferences/:preference_key`

Method
`PUT`

Url parameters

id
user's id

preference_key
key of the preference

Request body
Json object with field & value pair.

Result Code

- 204 : The preference is successfully updated
- 404 : The preference is not set for this user

Example

Query:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/3/preferences/preferred_device'
  -d {"value": "webrtc", "value_type": "string"}
```

Delete a specific user preference

Delete a specific preference for a user.

Description:

URL
`/users/:id/preferences/:preference_key`

Method
`DELETE`

Url parameters

id
user's id

preference_key
key of the preference

Example

Query:

```
curl -XDELETE -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/3/preferences/preferred_device'
```

Result Code

- 204 : The preference is successfully updated
- 404 : The preference is not set for this user

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: ${PLAY_AUTH_TOKEN}" '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: ${PLAY_AUTH_TOKEN}" \
  '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 journ\u00e9e",
          "periodStart": "09:00:00",
          "periodEnd": "17:00:00",
          "default": true
        }
      }
    ]
  },
  {
    "uuid": "75509ad3-3f81-40be-ad90-2c36d7a2c809",
    "name": "another",
```

(continues on next page)

(continued from previous page)

```
    "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: ${PLAY_AUTH_TOKEN}" '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: ${PLAY_AUTH_TOKEN}" \
  '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: ${PLAY_AUTH_TOKEN}" '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"
  }'
```

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: ${PLAY_AUTH_TOKEN}" '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": [
```

(continues on next page)

(continued from previous page)

```
    {
      "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 journ\u00e9e",
        "periodStart": "09:00:00",
        "periodEnd": "17:00:00",
        "default": true
      }
    }
  ]
}
```

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: ${PLAY_AUTH_TOKEN}" '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,
```

(continues on next page)

(continued from previous page)

```
      "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: ${PLAY_AUTH_TOKEN}" '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
```

(continues on next page)

(continued from previous page)

```
    }],
  "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 }]
```

Queue Dissuasion

Get the dissuasion for a queue

Returns the dissuasion of a given queue: if a queue has its *No Answer* → *Fail* case configured to

- *Destination*: *Sound file*
- *Filename*: `<some_file>`

then it returns the sound file name `<some_file>` configured.

Description:

URL
`/queue/:id/dissuasion`

Method
`GET`

Url parameters

id
queue's id

Example:

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/1.0/queue/3/
↪dissuasion
```

Response:

If the queue *sales* has its dissuasion configured towards the sound file `sales_audio_file.wav`:

```
{
  "id": 3,
  "name": "sales",
  "dissuasion": {
    "type": "soundFile",
    "value": {
      "soundFile": "sales_audio_file"
    }
  }
}
```

If the queue *sales* has its dissuasion configured towards an other queue with id *support*:

```
{
  "id": 3,
  "name": "sales",
  "dissuasion": {
    "type": "queue",
    "value": {
      "queueName": "support"
    }
  }
}
```

If the queue *sales* has its dissuasion configured neither towards a *sound file* nor a *queue*:

```
{
  "id": 3,
  "name": "sales",
  "dissuasion": {
    "type": "other"
  }
}
```

If the queue does not exist:

```
{
  "error": "QueueNotFound",
  "message": "Unable to perform getQueueDissusasion on queue 3 - QueueNotFound"
}
```

Get all the sound files dissuasion for a queue

Returns the list of dissuasions available for a given queue:

- the list of sound files is taken from directory `/var/lib/xivo/sounds/playback/`,
- a sound file is available for a given queue if it starts with the queuename (e.g. for queue *sales* the sound file must start with `sales_`).
- the default queue is defined in the `custom.env`

Description:

URL
`/queue/:id/dissuasions`

Method
`GET`

Url parameters

id
queue's id

Example:

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/1.0/queue/3/
↪dissuasions
```

Response:

For the queue named *sales* with sound files `sales_audio_file.wav` and `sales_audio_file_2` present in dir `/var/lib/xivo/sounds/playback/`, and the queue *switchboard* defined in `custom.env`:

```
{
  "id": 3,
  "name": "sales",
  "dissuasions": {
    "soundFiles": [
      {
        "soundFile": "sales_audio_file"
      },
      {
        "soundFile": "sales_audio_file_2"
      }
    ],
    "queues": [
      {
        "queueName": "switchboard"
      }
    ]
  }
}
```

If the queue does not exist

```
{
  "error": "QueueNotFound",
  "message": "Unable to perform getQueueDissusasionList on queue 3 - QueueNotFound"
}
```

Set the dissuasion sound file for a given queue

Sets the dissuasion sound file for a given queue.

Description:

URL
`/queue/<id>/dissuasion/sound_file`

Method
`PUT`

Url parameters

id
queue's id

Request body
Json object with soundFile

Example:

Query:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/1.0/queue/3/
↪dissuasion/sound_file -d '{"soundFile": "sales_newSoundFile"}'
```

Response:

If the change was successful (with the new file being, for instance, sales_newSoundFile.wav)

```
{
  id: 3,
  soundFile: "sales_newSoundFile"
}
```

If the queue does not exist

```
{
  "error": "QueueNotFound",
  "message": "Unable to perform updateQueueDissuasion on queue 3 - QueueNotFound"
}
```

If the file does not exist

```
{
  "error": "FileNotFound",
  "message": "Unable to perform updateQueueDissuasion on queue 3 with file sales_
↪newSoundFile - FileNotFound"
}
```

For a given queue, set the queue to redirect calls to in case of dissuasion

For a given queue, sets the queue to redirect calls to in case of dissuasion.

Description:

URL
`/queue/<id>/dissuasion/queue`

Method
`PUT`

Url parameters

id
queue's id

Request body
Json object with queueName

Example:

Query:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/1.0/queue/3/
↪dissuasion/queue -d '{"queueName": "sales_queue"}'
```

Response:

If the change was successful (with the new queue to redirect calls to being, for instance, sales_queue)

```
{
  id: 3,
  queueName: "sales_queue"
}
```

If the queue we want to change the dissuasion destination of does not exist

```
{
  "error": "QueueNotFound",
  "message": "Unable to perform updateQueueDissuasion on queue 3 - QueueNotFound"
}
```

If the queue to redirect calls to does not exist

```
{
  "error": "QueueNotFound",
  "message": "Unable to perform updateQueueDissuasion on queue 3: could not find the␣
↪queue 'sales_queue' - QueueNotFound"
}
```

Sip Configuration

Get sip configuration

Retrieve the configured stun address.

Example:

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/1.0/sip/ice_
↪servers
```

Response::

```
{
  "stun_address": "stun:stun.l.google.com:19302"
}
```

Meeting Rooms

The following API allows to retrieve, edit and delete meeting rooms.

Get a meeting room

Retrieve a meeting room by id

Description:

URL
/meetingrooms/<id>

Method
GET

Url parameters

id
meeting room's id

Example:

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/2.0/meetingrooms/
↪1
```

Response::

```
{
 "id":1,
 "name":"myConfRoom1",
 "displayName":"My Conf Room 1",
 "number":1050,
 "userPin":1234
}
```

Result Code

- 200 : The meeting room is retrieved
- 500 : The meeting room is not retrieved due to an exception

Get all meeting rooms

Description:

URL
/meetingrooms

Method
GET

Example:

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/2.0/meetingrooms
```

Response::

```
[
 {
    "id":1,
    "name":"myConfRoom1",
    "displayName":"My Conf Room 1",
    "number":1050,
    "userPin":1234
 },
 {
    "id":2
    "name":"myConfRoom2",
    "displayName":"My Conf Room 2",
    "number":1051
 }
]
```

Result Code

- 200 : The meeting rooms are retrieved
- 500 : The meeting rooms are not retrieved due to an exception

Create / Update a meeting room

Description:

URL
`/meetingrooms`

Method
`POST` for create, `PUT` for update

Request body
Json object with field & value pair.

Allowed field names:

id
Id of the meeting room, only required when updating the meeting room

name
Technical unique name of the meeting room (e.g. myMeetingRoom1)

displayName
Display name of the meeting room (e.g. My Meeting Room 1)

number
Number of the meeting room

userPin
Optional, the pin to access the meeting room (e.g. 1234)

Example:

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/2.0/meetingrooms
  -d {"name": "myConfRoom1", "displayName": "My Conf Room 1", "number": 1050, "userPin
↪": 1234}
```

Query:

```
curl -XPUT -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/2.0/meetingrooms
  -d {"id": 1, "name": "myConfRoom1", "displayName": "My Conf Room 1", "number": 1050,
↪ "userPin": 1234}
```

Response::

```
{
    "id":1,
    "name":"myConfRoom1",
    "displayName":"My Conf Room 1",
    "number":1050,
    "userPin":1234
}
```

Result Code

- 200 : The meeting room is created/updated
- 400 : The meeting room is not created/updated (bad JSON or duplicate)
- 500 : The meeting room is not created/updated

If the meeting room already exists:

```
{
  "error": "Duplicate",
  "message": "duplicate key value violates unique constraint <constraint_name>"
}
```

Delete a meeting room

Description:

URL
/meetingrooms/<id>

Method
DELETE

Example:

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" localhost:9000/configmgt/api/2.0/meetingrooms/
↪1
```

Response::

```
{
    "id":1,
    "name":"myConfRoom1",
    "displayName":"My Conf Room 1",
    "number":1050,
    "userPin":1234
}
```

Result Code

- 200 : The meeting room is deleted

- 500 : The meeting room is not deleted

Find meeting rooms

This API allows to find meeting rooms using dynamic filters.

Description:

URL
`/meetingrooms/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

limit
Max number of results to return

List of filterable columns

- name
- displayName
- number
- userPin

Example

Query:

```
curl -XPOST -H "Content-Type: application/json" -H "Accept: application/json" \
  -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪meetingrooms/find' \
  -d '
    {
      "filters":[
        {"field":"name", "operator":"=", "value":"myConfRoom1conf", "order": "ASC"}
      ],
      "offset":0,
      "limit":100
    }'
```

Response:

```
{
 "total":1,
 "list":[
    {
      "id":1,
```

(continues on next page)

(continued from previous page)

```
      "name":"myConfRoom1",
      "displayName":"My Conf Room 1",
      "number":1050,
      "userPin":1234
    }
 ]
}
```

Get Xivo Users With contact informations

This API allows to get Xivo Users with contact informations

Description:

URL
/api/2.0/users/ascontact

Method
GET

Example

Query:

```
curl -XGET -H "Content-Type: application/json" -H "Accept: application/json" \
     -H "X-Auth-Token: ${PLAY_AUTH_TOKEN}" 'http://localhost:9100/configmgt/api/2.0/
↪users/ascontact'
```

Response:

```
[
   {
      "firstname":"toto",
      "lastname":"user",
      "email":"t.user@xivo.solutions",
      "mobilephoneNumber":"1000",
      "internalphonenumber":"9999",
      "labels":[
        "red",
        "green",
        "blue"
      ],
      "externalphonenumber":"2000"
   },
   {
     "firstname":"Albert",
      "lastname":"Einstein",
      "email":"a.einstein@xivo.solutions",
      "mobilephoneNumber":"2012",
      "internalphonenumber":"2019",
      "labels":[
        "green",
        "yellow"
      ],
      "externalphonenumber":"0007"
   }
]
```

7.2 Low-Level & Admin APIs

Note: For System Administrators & Provisioning

Direct access to the core components of the XiVO IPBX for fine-grained configuration and legacy support:

- **Configuration** through *confd* to manage unitary devices, users, lines…
- **Authentication** with *xivo-auth* to manage tokens and permissions (acls)
- **Provisioning** via *provd* for device management and auto-configuration.
- **System** with *sysconfd* to access network and OS-level settings.

7.2.1 IPBX API

The XiVO REST APIs are the privileged way to programmatically interact with XiVO.

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

2022.06

- New key in wizard json: * `run_scripts`

2020.18

- One new parameter have been added to the users resource:
  - `line_type`

2020.14

- A new API for *User Labels* has been added:
  - PUT `1.1/labels/<label_id>`

2020.13

- A new API for *User Labels* has been added:
  - POST `1.1/labels`

2020.12

- A new API for *User Labels* has been added:
  - DELETE `1.1/labels/<label_id>`

2020.11

- A new API for *User Labels* has been added:
  - GET `1.1/labels`

Deneb.03

- A new API for agents has been added:
  - GET `/1.1/agents`
  - POST `/1.1/agents`
  - DELETE `/1.1/agents/<agent_id>`
  - GET `/1.1/agents/<agent_id>`
  - PUT `/1.1/agents/<agent_id>`
  - GET `/1.1/agents/<agent_id>/queues`

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`

REST API 1.1 examples

Create User for a line and a exten

Add user, line and exten with association

Add voicemail with association

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.

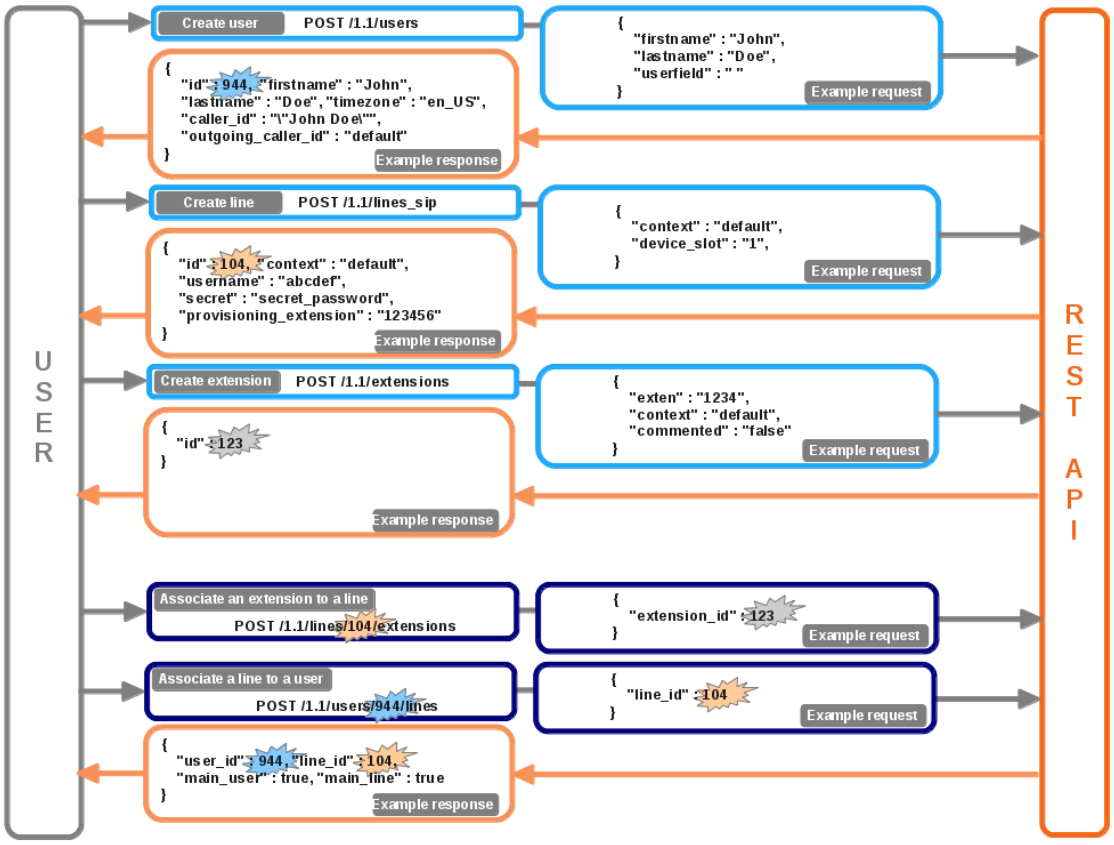


Fig. 1: Download source. (source)

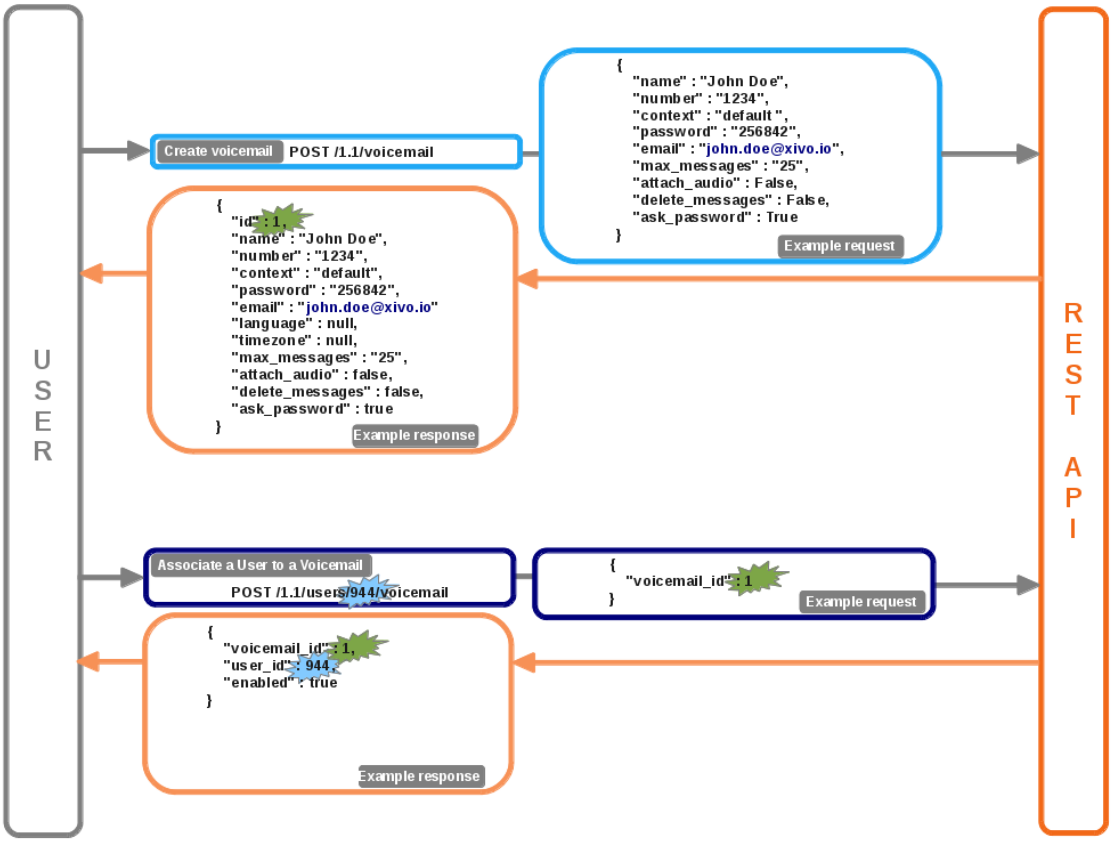


Fig. 2: Download source. (source)

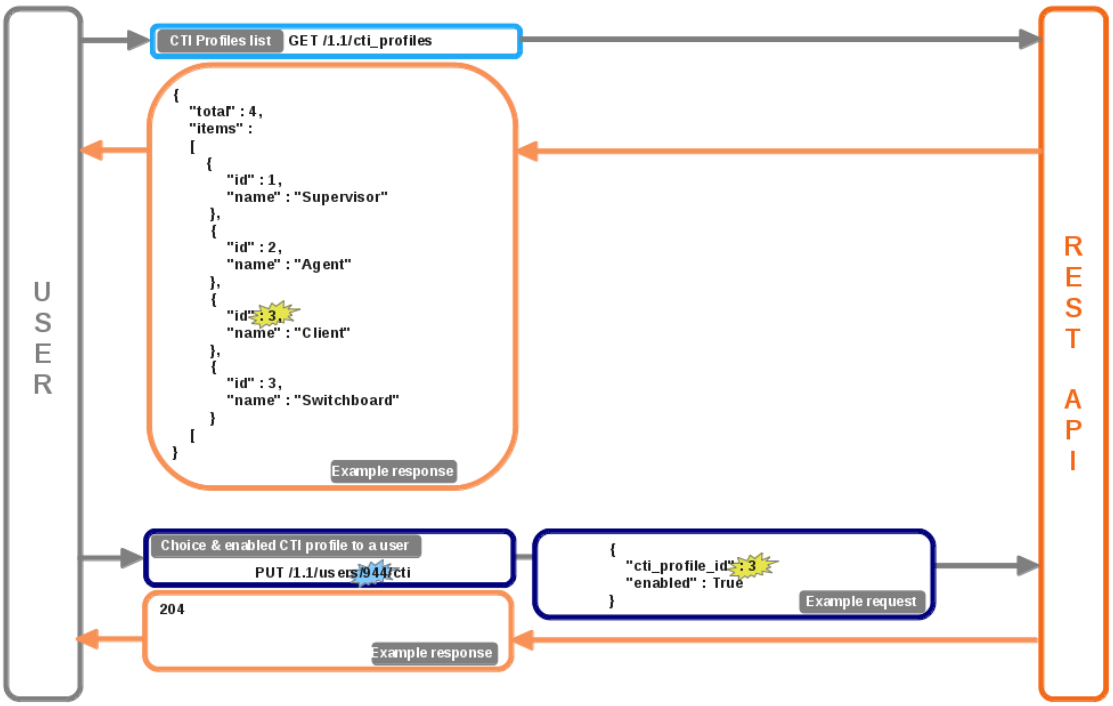


Fig. 3: Download source. (source)

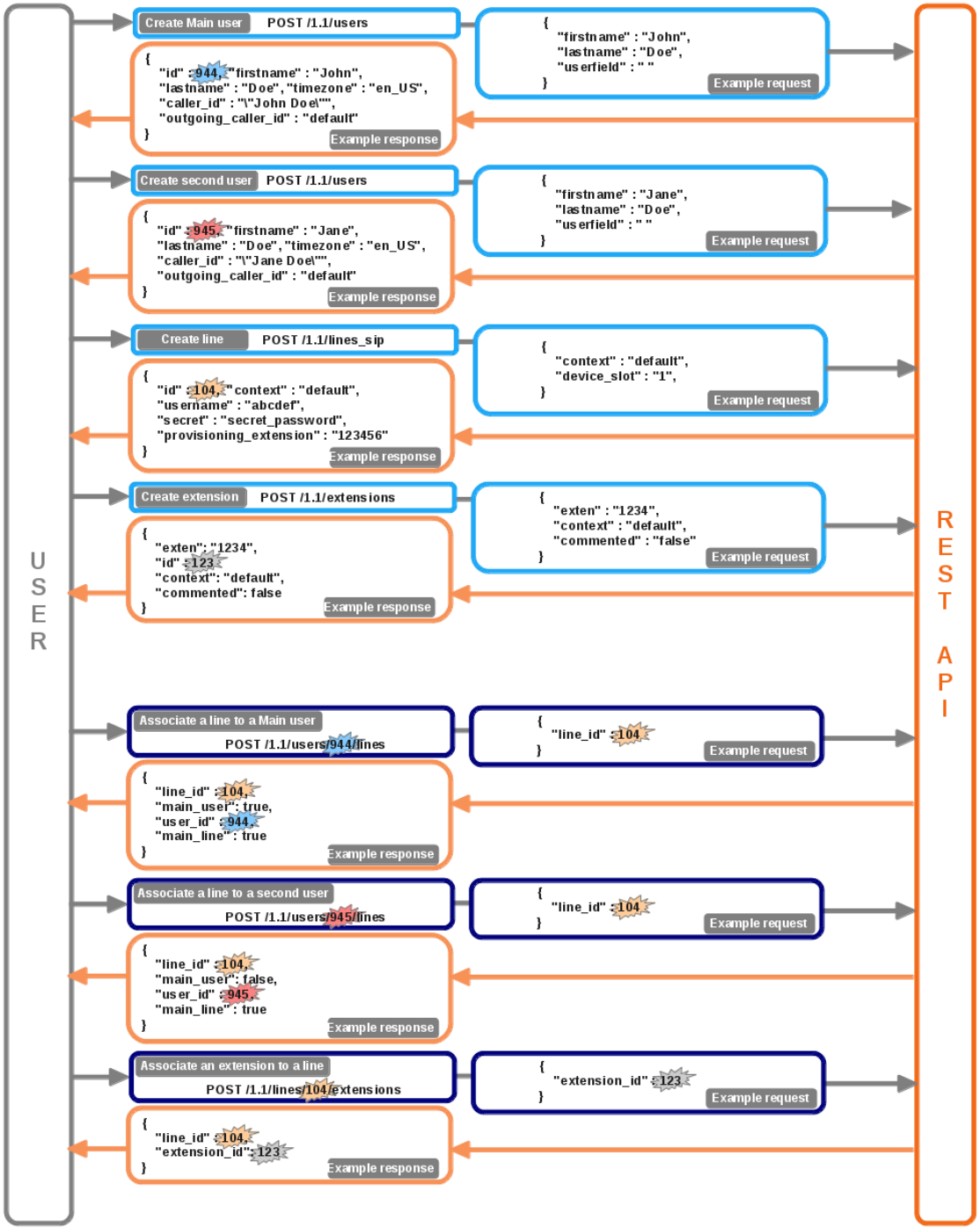


Fig. 4: Download source. (source)

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.

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
             }
         },
         "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",
        "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",
                "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",
```

(continues on next page)

(continued from previous page)

```
                "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",
                        "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"
                    }
                }
            }
        }
    }
}
```

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

{
    "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"
                }
            ]
        }
    }
}
```

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,
```

(continues on next page)

(continued from previous page)


```
                "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",
```

(continues on next page)

(continued from previous page)

```
                    "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"
                }
            ],
            "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",
            "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": {
                    "sip.lines": 6,
                    "switchboard": true
                },
                "Aastra, 6735i, 3.3.1.5089": {
                    "sip.lines": 9
```

(continues on next page)

(continued from previous page)

```
                },
                "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,
       "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,
```

(continues on next page)

(continued from previous page)

```
        "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
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
```

Update HA state

Query

```
POST /update_ha_state
```

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 using the authentication form in the header of the Swagger UI:

1. In the header of the page, fill in the authentication form with:
   - `Username`: the username from the Web Services Access you created earlier
   - `Password`: the password from the Web Services Access you created earlier
   - `Backend`: select `xivo_service` from the dropdown
2. Click the "Get Token" button
3. The token will be automatically retrieved and stored for use with all API requests

The token has a default expiration time. After the expiration time, you will need to re-authenticate to get a new token using the same form.

For more informations about the backends of xivo-auth, see *xivo-auth plugins*.

Use the authentication token

Once you have obtained a token using the authentication form in the header, it will be automatically used for all API requests.

You can now choose a REST API endpoint and "Try it out".

Access

Each REST API is available via HTTPS on *different ports*.

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
```

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
```

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.

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.

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,
```

(continues on next page)

(continued from previous page)

```
    "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, ... ]
```

7.2.2 API Security

Shared token

Accessing the *Configuration API* or the *Recording API* requires a token. This token is used by the Xuc process but can also be used to access API from custom server processes. This token must remain on the server and should not be leaked as it would allow API access without restrictions.

To define or change it, you can add or update the value of the `PLAY_AUTH_TOKEN` environment variable defined in the `custom.env` of **all** the following docker environment (where applicable):

XiVO
`/etc/docker/xivo/custom.env`

MDS
`/etc/docker/mds/custom.env`

XiVO CC
`/etc/docker/compose/custom.env`

XiVO UC
`/etc/docker/compose/custom.env`

Warning:

- Remind that the value, if defined, **must be the same** across all XiVO components.
- When the XiVO CC is splitted on separate server, the value must also be defined consistently.

For more information, see also the *Components Configuration* and *Multi-server installation* sections.

Unified Communication Credentials

Depending on the authentication mode you will require different kind of informations to access the *CTI API*. To simplify and unify authentication, API access is based on a custom token that can be used one of the following way, depending of the API you access:

X-Auth-Token Header
Adding a HTTP header with a valid token

Query string parameter
Adding a *token* parameter with a valid token in the query string part of the API url.

The following sections explain how to get a valid token or trade an external token to a *CTI API* token.

Login based access

You can use any valid user (see *Users*) with the following properties:

- CTI Login enabled
- Login
- Password
- Profile set to any valid value

With these informations you can then use the *User basic authentication* API to get a *CTI API* token.

Kerberos token

A kerberos ticket is normally obtained automatically by the browser when accessing a resource requiring access privilge. So in most case a simple HTTP call to the *Single sign-in (SSO authentication)* API will get you a *CTI API* token.

CAS token

CAS token can be obtained by authenticating on a CAS server for a given service. Once you have a token, you can trade it to a *CTI API* token using the *Authentication with CAS* API.

OpenID Token

OpenID token can be obtained by authenticating on an OpenID server for a given client. Once you have a token, you can trade it to a *CTI API* token using the *Authentication with OpenID Connect (OIDC)* API.

CHAPTER

EIGHT

TELEMETRY

In our telemetry system, we utilize two Docker components:

8.1 Usage Writer

The Usage Writer monitors system activities, capturing details such as connection events and XiVO platform configurations. This data is then stored in a XiVO database for analysis. Here's a breakdown of what is collected:

- **Configuration:** - Presence of edge and switchboard - Number of agents, MDS (Managed Data Servers), meeting rooms, switchboards, users, and WebRTC users
- **Events:** - User logins - Software type (web browser, electron, mobile) - Line type (phone, WebRTC, UA)

8.2 Usage Collector

The Usage Collector retrieves data from the Usage Writer, refines it into specific metrics, and sends it to our central database. It's important to note that all data sent is anonymized to protect user privacy.

8.3 System Architecture and Data Flow

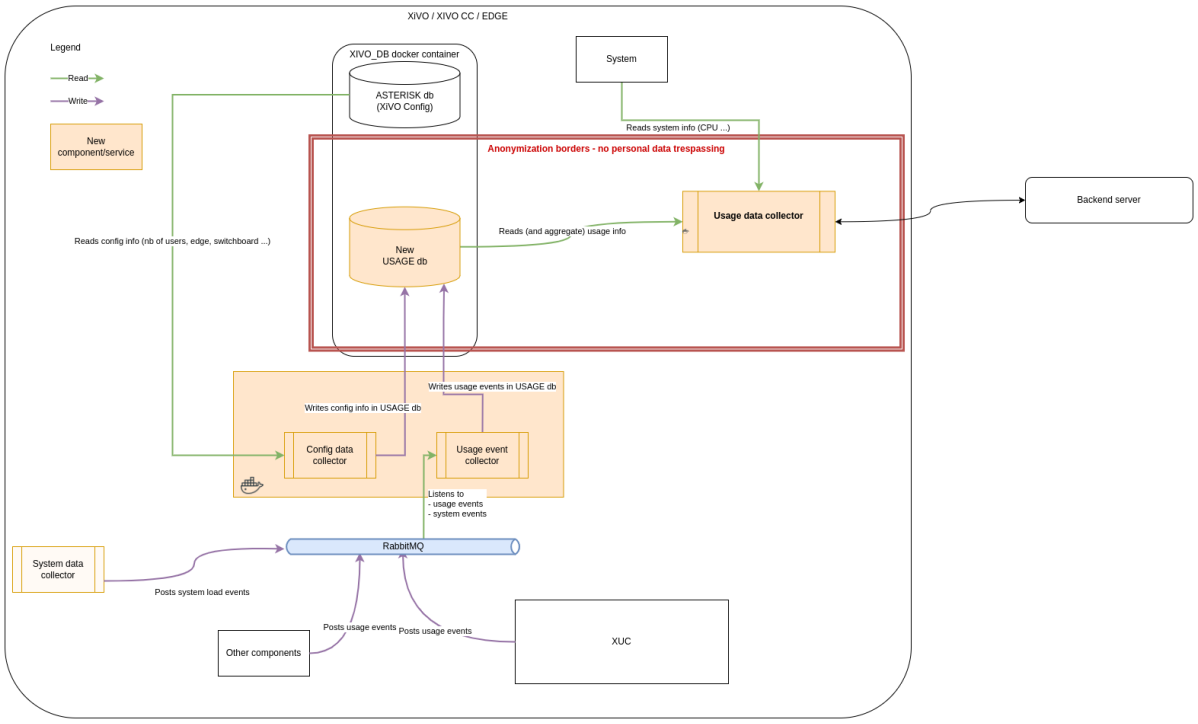
This image illustrates the data flow in our telemetry system, showcasing how data moves from the Usage Writer to the central database, undergoing refinement for meaningful analysis. This streamlined flow ensures accurate insights for informed decision-making while safeguarding user anonymity.

8.4 Data retention

By default, data in the usage (`usm`) database is kept for the last 365 days (1 rolling year). This is configured in the crontab `/etc/cron.d/xivo-purge-db`.

The data purge is run every day at 2:25 a.m. Log of the purge can be seen in the syslog:

```
grep xivo-purge-usm /var/log/syslog
```


XiVO / XiVO CC / EDGE
Legend
Read
Write
New component/service
XIVO_DB docker container
ASTERISK db (XiVO Config)
System
Reads system info (CPU ...)
Anonymization borders - no personal data trespassing
Usage data collector
Backend server
Reads (and aggregate) usage info
New USAGE db
Reads config info (nb of users, edge, switchboard ...)
Writes usage events in USAGE db
Writes config info in USAGE db
Config data collector
Usage event collector
Listens to
- usage events
- system events
RabbitMQ
System data collector
Posts system load events
Posts usage events
Posts usage events
Other components
XUC

CHAPTER

NINE

CONTRIBUTING

General information:

9.1 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

9.1.1 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:

```
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

[1] `easy_install` can be found in the debian package python-setuptools : `sudo apt-get install python-setuptools`

- -, 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
-----------

Foo.
```

Incorrect:

```
Section1
========
```

(continues on next page)

(continued from previous page)

```
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

9.2 Debugging Asterisk

9.2.1 Precondition

To debug asterisk crashes or freezes, you need the following debug packages on your XiVO:

Commands (replace *<VERSION_INSTALLED>* by the version installed on your XiVO - for example *xivo-freya*):

```
xivo-dist <VERSION_INSTALLED>
apt-get update
apt-get install gdb
apt-get install -t <VERSION_INSTALLED> asterisk-dbg
```

9.2.2 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
   ```

9.2.3 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
```

9.2.4 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

9.2.5 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.

9.2.6 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.

9.2.7 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

9.3 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
```

9.3.1 xivo-confgend

```
twistd3 -no -u xivo-confgend -g xivo-confgend --python=/usr/bin/xivo-confgend
```

No debug mode in confgend.

9.3.2 xivo-provd

```
twistd3 -no -u xivo-provd -g xivo-provd -r epoll xivo-provd -s -v
```

- -s for logging to stderr
- -v for verbose

9.3.3 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
```

9.4 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.

9.5 XiVO Guidelines

9.5.1 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.

9.5.2 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.

9.5.3 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.

9.6 Debian packaging for XiVO

9.6.1 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
```

9.7 Profiling Python Programs

9.7.1 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
```

9.7.2 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
```

9.7.3 External Links

- Official python documentation
- PyMOTW
- coverage.py

9.8 Style Guide

9.8.1 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:
```

(continues on next page)

(continued from previous page)

```
    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)

if final_price < money:
    product.pay(money)
```

9.8.2 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)
```

9.8.3 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)
```

9.8.4 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
```

9.9 Translating XiVO

French and English are maintained by Avencall group. Other languages are provided by the community.

9.9.1 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*.

9.9.2 Web Interface

Translations are currently available in French and English. There are no plans to translate the Web interface in other languages.

9.10 XiVO Package File Structure

9.10.1 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
```

(continues on next page)

(continued from previous page)

```
|   `-- 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

9.10.2 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:

9.11 CTI Server

This section describes the informations and tools for CTI Server.

9.11.1 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.

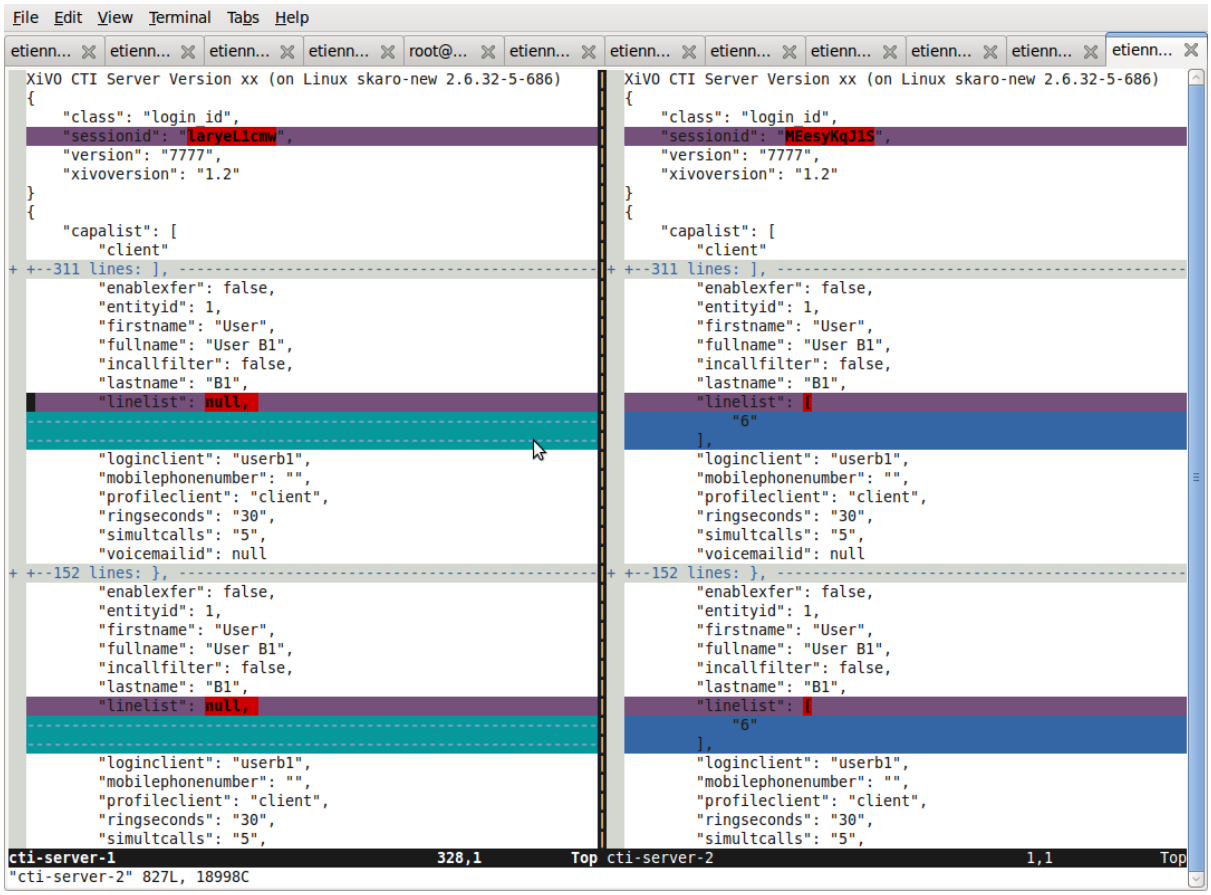
The simplest invocation is:

```
$ 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
```

9.11.2 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},
```

(continues on next page)

(continued from previous page)

```
  "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",
   "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":
      {"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",
```


(continues on next page)

(continued from previous page)

```
↪ "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"
}
```

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
```

(continues on next page)

(continued from previous page)

```
      },
      "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": [
    {
```

(continues on next page)

(continued from previous page)

```
      "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"
    }
  ]
}
```

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",
```

(continues on next page)

(continued from previous page)

```
  "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",
   "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",
    "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",
 "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>
```

(continues on next page)

(continued from previous page)

```
        <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": {
    "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>"],
```

(continues on next page)

(continued from previous page)

```
                    ...,
                    ["<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

Warning: This module is **DEPRECATED** and will be removed in the next version.

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.

9.12 Diagrams

9.12.1 Agent states

Graphs representing states and transitions between agent states. Used in Agent status dashboard and agent list.

`Download (DIA)`

9.12.2 Architecture

See *XiVO PBX Services Links*

9.13 Provisioning

This section describes the informations and tools for xivo-provd.

9.13.1 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.

9.13.2 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-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'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/`
```

9.13.3 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	
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

9.13.4 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:

Where:

- the XiVO is installed inside a virtual machine

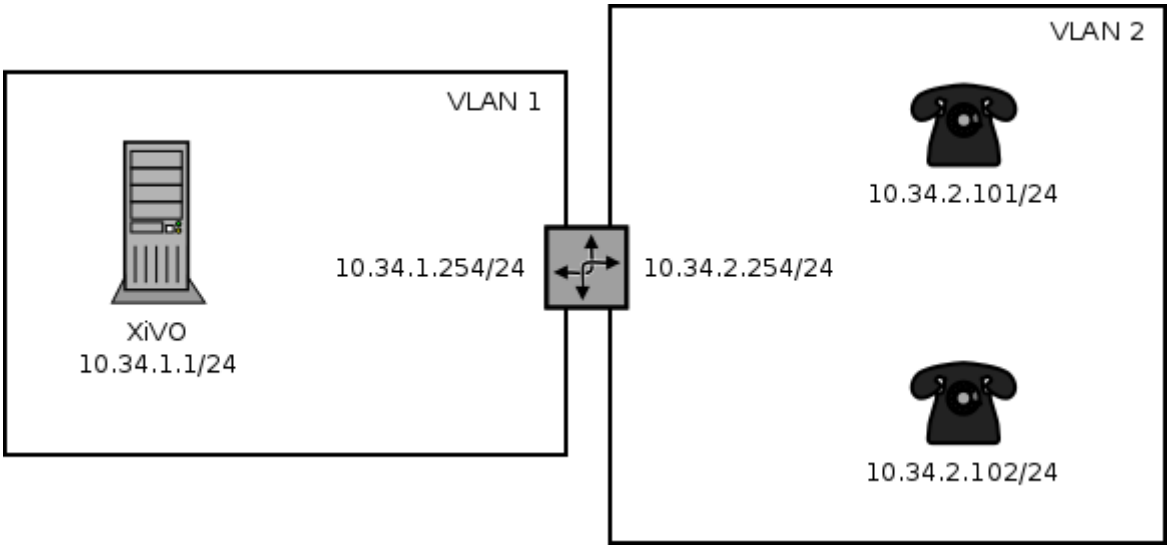


Fig. 1: Phones behind a NAT

- 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
    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.

9.14 Web Interface

9.14.1 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
```

9.14.2 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

9.15 Community Documentation

This page provides links to resources on various topics around XiVO. They have been generously created by people from the community.

9.15.1 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
Installing XiVO on Raspberry Pi (Raspivo)	French	Intermediate	Iris Network	2015
`How to backup XiVO to external FTP` with `backup-ftp.sh`	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	English	Beginner	Nerd Vittles	16.07

9.15.2 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.

Experimental Features

9.16 Experimental Features

> **Warning:** This section lists experimental features. When listed here, it means that they are not complete and not for production. No support will be provided for these features.

9.16.1 Cross-MDS User Group Management

Overview

This feature allows administrators to add users to groups regardless of their MDS location within a XiVO system. It simplifies group management by enabling the creation of diverse groups that span multiple MDS instances.

Benefits

- **Flexible Group Management**: Create groups with users from any MDS, enhancing organizational flexibility.
- **Unified Group Calling**: Ensure that all users in a group receive calls simultaneously, regardless of their MDS location.
- **Simplified Administration**: Manage groups centrally while maintaining MDS associations.

Use Cases

- **Distributed Teams**: Manage teams distributed across different locations by creating groups that include users from multiple MDS.
- **Unified Customer Service**: Ensure consistent customer service by creating groups that handle calls across different MDS instances.

Configuration

To enable this experimental feature, you must configure the XiVO system to allow device state sharing and cross-MDS user group management. This requires updating the relevant configuration files and ensuring all MDS instances are properly connected.

1. **Set the `XDS_MULTI_MDS_CALLGROUP` variable to `TRUE` in the XiVO configuration:**

   Edit the file `/etc/docker/xivo/custom.env` and add the following line:

   ```
   XDS_MULTI_MDS_CALLGROUP=TRUE
   ```

2. **Update the service to take configuration into account:**

```
xivo-dcomp up -d
asterisk -rx "core reload"
```

3. **Repeat this configuration on all MDS instances:**

   Edit the file `/etc/docker/mds/custom.env` on each MDS instance and add the same line:

```
XDS_MULTI_MDS_CALLGROUP=TRUE
```

4. **Update the MDS service to apply the configuration:**

```
xivo-dcomp up -d
asterisk -rx "core reload"
```

Note: This configuration enables device state sharing and cross-MDS user group management across your XiVO and MDS servers. Ensure you apply these changes to every relevant instance for the feature to function correctly.

Deprecated Features

9.17 Deprecated Features

Warning: This section lists deprecated features.

9.17.1 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.

Intended usage and features

Contents

- *Intended usage and features*
  - *Routing*
    - *Implementation of routing*
  - *Centralized user management*
  - *XiVO integration*

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.

Routing

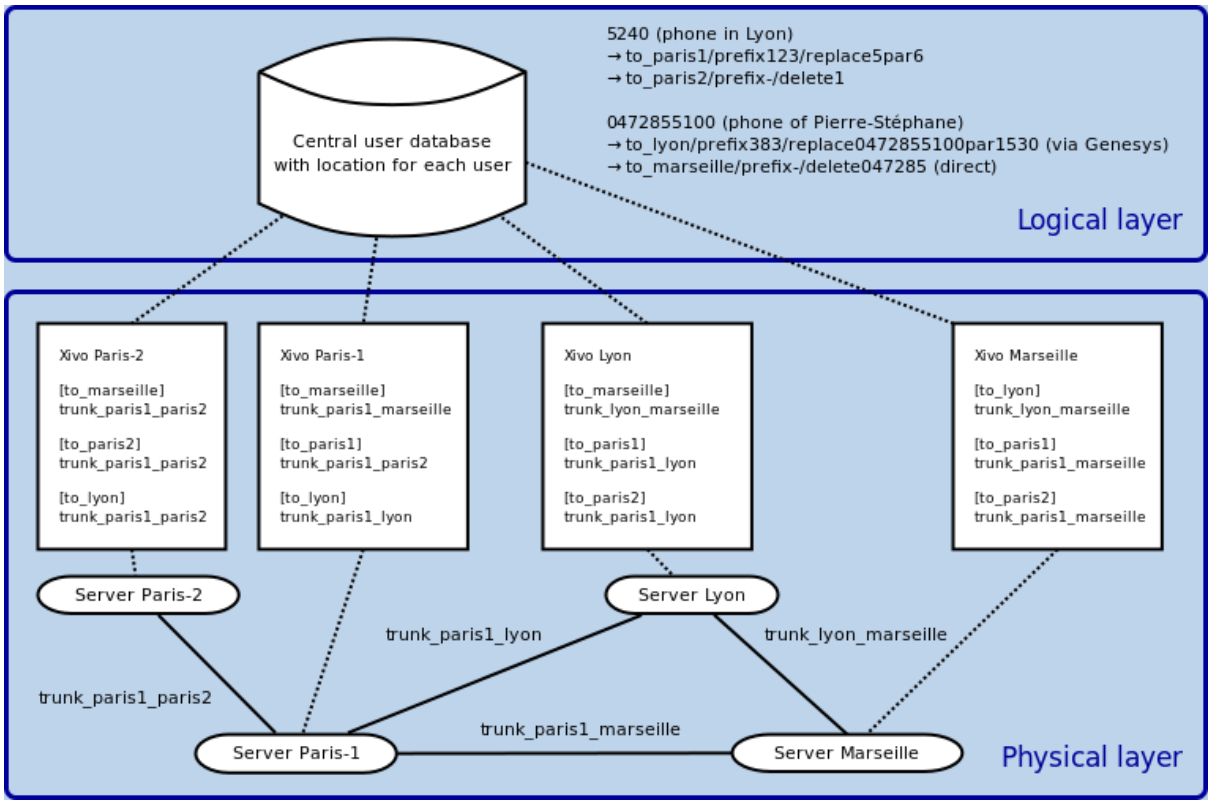
Routing supports heterogeneous numbering plan administration:

- 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.

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.
- 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.

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*.

Installation

Contents

- *Installation*
  - *Requirements & Limitations*
  - *XiVO(s) Requirements & Limitations*
  - *Installation by installer package*
  - *Configuration*
    - *Authentication*
    - *Interval overlapping*
  - *Run the application*
  - *Application logs*

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.

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*.

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`.

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.

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

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">
```

Upgrade

The XiVO Centralized User Management (XCU) upgrade.

Contents

- *Upgrade*
  - *Prepare sources list*
  - *Upgrade steps*

    * *Preparing the upgrade*
    * *Upgrade*
  - *Version specific XCU upgrade procedures*
    * *Upgrade from versions before 2020.18*
    * *Upgrade from versions before 2018.04*
    * *Upgrade from versions before 2017.06*

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`.

You must change the sources list to point towards the version you want to upgrade.

For example if you want to upgrade to Freya GCU:

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

Upgrade steps

Preparing the upgrade

- Read *GCU 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.

Version specific XCU upgrade procedures

Upgrade from versions before 2020.18

Before upgrading to version 2020.18, you must change the `not_routed` mode of all entity's intervals.

For example, change `not_routed` mode to `routed` mode in table `entity_intervals`:

```
UPDATE entity_intervals SET interval_routing_mode = 'routed' WHERE interval_routing_
↪mode = 'not_routed';
```

You can also do it per-entity and per-interval by doing it manually in the GCU web iunterface **before the upgrade**.

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.

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.

Contents

- *Configuration of Xivos for Centralized Routing*
  - *Xivo configuration for centralized routing*
    - *Things to verify on Xivo before manual configuration*
    - *Manually update address of routing server*

    * *Connect Xivos by trunks*
    * *Add routing contexts*
    * *Add outgoing calls*
  - *Debug centralized routing*

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`

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.

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.

Contents

- *Manual installation*
  - *Database setup*
    - *I can't connect to PostgreSQL*
  - *Generate SSH key*

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 `/var/lib/postgresql/17/data/`) and add the following line at the end :

```
local   all     all     trust
```

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
```

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.

Contents

- *Web interface*
  - *Definitions*
  - *Dashboard*
  - *XiVO*
    - *Create XiVO*
    - *View XiVO*
  - *Entity*
    - *Create entity*
    - *View entity*

    * *Edit entity*
  - *Line templates*
    * *List templates*
    * *Create template*
    * *Edit template*
  - *User*
    * *Create user*
    * *Edit user*
  - *Administrators*
    * *List administrators*
    * *Create administrator*
    * *Edit administrator*

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.

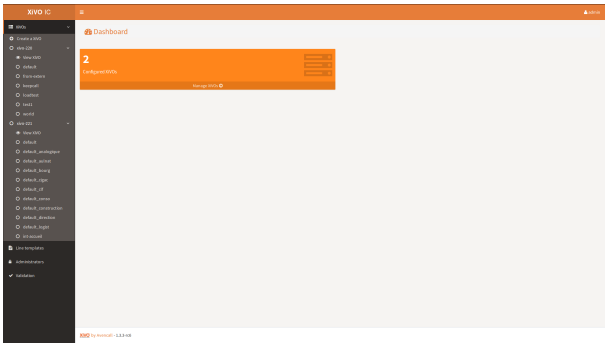
Entity
: Entities, also called Contexts, are the parts of the dialplan. Users are attached to them.

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.

Dashboard

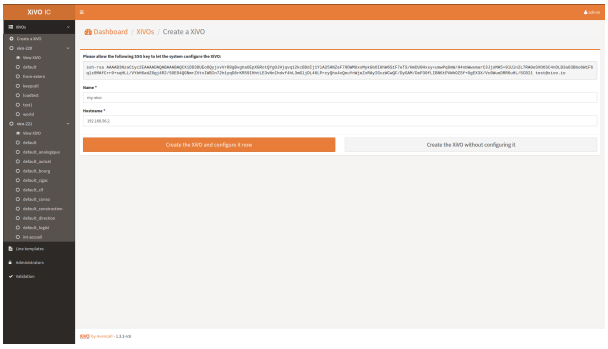


The dashboard provides you some insights about your XiVO systems.

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.

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 :

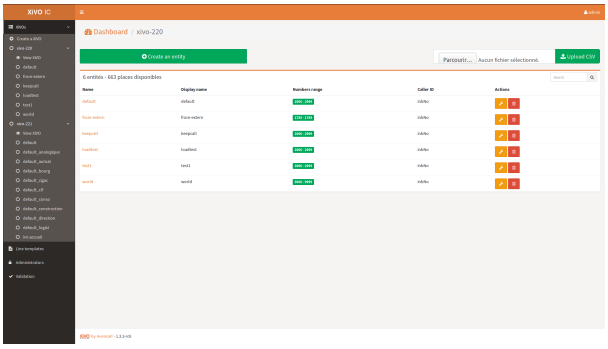
- **Create the XiVO and configure it now** : XCU will save the information, try to connect to the XiVO server and perform the configuration. XiVO services will be unavailable during the operation.

> **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



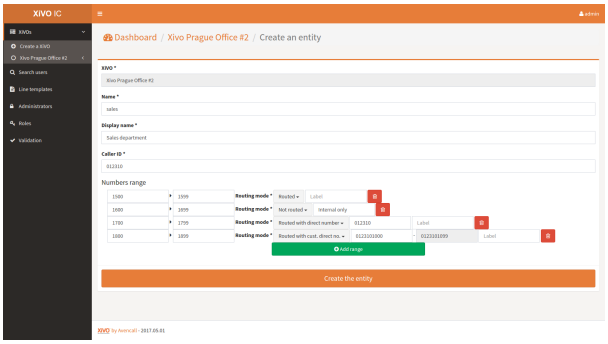
On the sidebar, each XiVO has its own **View XiVO** link. This page allows you to :

- **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

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
- **Media Server**: Media server chosen for this entity. One media server available by entity, the default one is Mds Main.
- **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

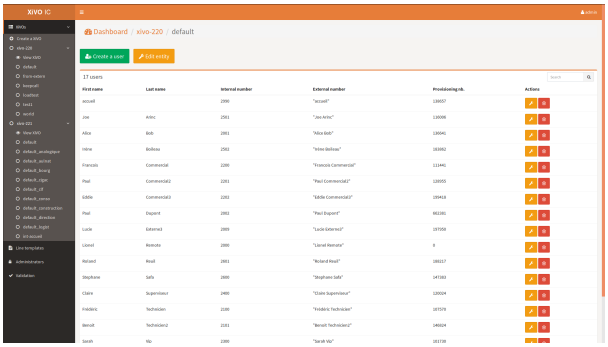
Routing mode affects how numbers from given interval are routed via centralized routing:

Routing mode	centralized routing
Routed	Internal number is used for centralized routing
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

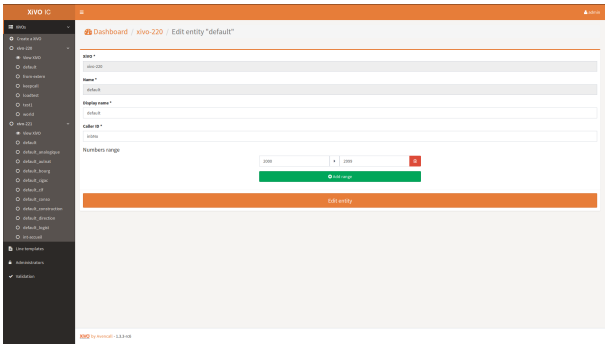
View entity



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.*

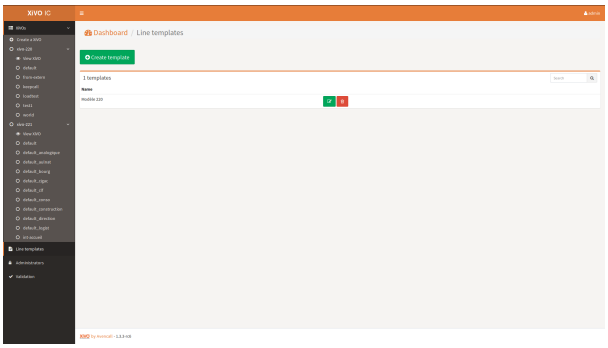
Edit entity



This page allow you to modify an entity. Please refer to the *Create entity* section for fields details.

Line templates

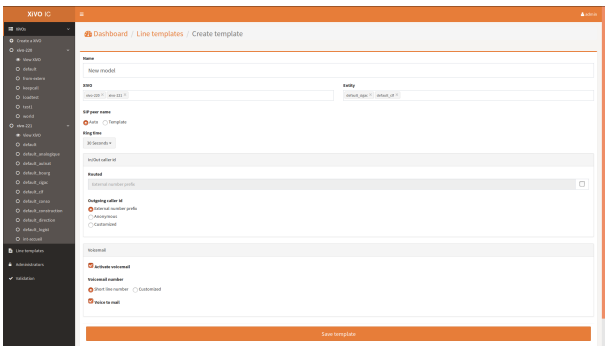
List templates



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

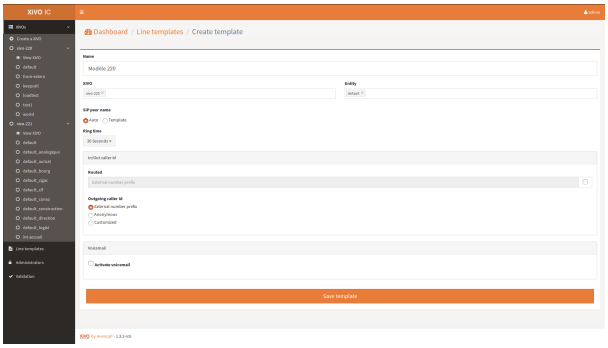


This page allows you to add a new line template. You have to provide the following informations :

- **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*, *Model*, *WebRTC* or *Unique Account*
- **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.

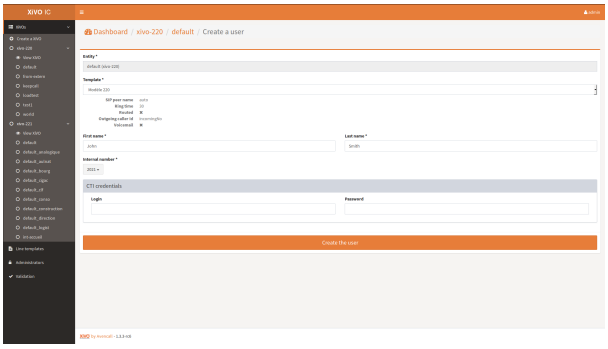
Edit template



This page allows you to modify a template. Please refer to the *Create template* section for fields details.

User

Create user

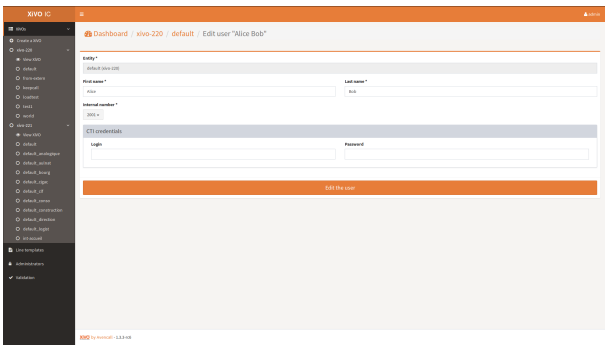


This page allows you to add a new user to an entity. You have to provide the following information :

- **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

When saving the form, a confirmation is displayed on top of the user list with the name and provisionning number of the user newly created.

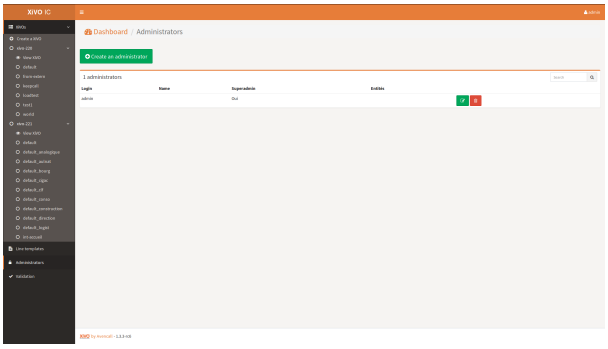
Edit user



This page allows you to modify a user. Please refer to the *Create user* section for fields details.

Administrators

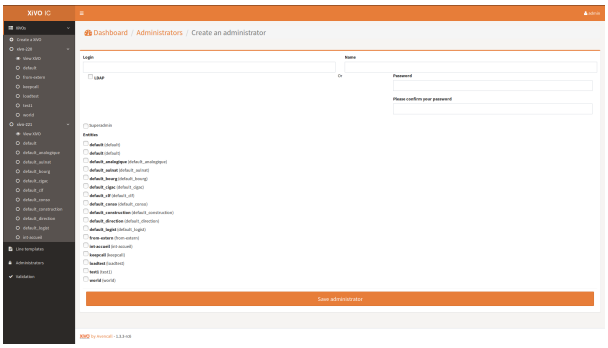
List administrators



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

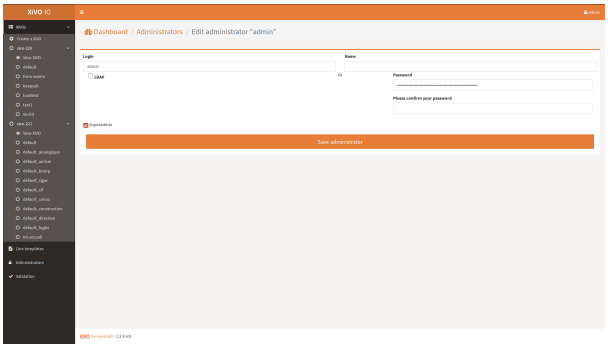
Create administrator



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.

REST API

The XiVO Centralized User Management (XCU) exposes some REST API that you can use to integrate with your tools.

Contents

- *REST API*
  - *General form*
  - *Login*
  - *XiVO*
    - *List*
    - *Get*
    - *Create*
    - *Synchronize configuration files*
  - *Entities*
    - *List*
    - *Get*
    - *Create*
    - *Delete*
    - *Edit*
    - *List users*
    - *List available numbers*
  - *Users*
    - *Get*
    - *Create*
    - *Delete*
    - *Edit*
  - *Templates*

    * *List*
    * *Get*
    * *Create*
    * *Delete*
    * *Edit*
  - *Administrators*
    * *List*
    * *Get*
    * *Create*
    * *Delete*
    * *Edit*
  - *Example (Python 3)*

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

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'
```

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

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,
```


(continues on next page)

(continued from previous page)

```
        "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"
    }
  ]
}
```

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",
```

(continues on next page)

(continued from previous page)


```
          "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"
  ]
}
```

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

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,
```

(continues on next page)

(continued from previous page)

```
    "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

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\/
```

(continues on next page)

(continued from previous page)

```
↪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,
  "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

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"]

        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})
```

(continues on next page)

(continued from previous page)

```
                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,
        "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")
```

GCU Release Notes

Contents

- *GCU Release Notes*
  - *2020.18*
    - *New Features*
    - *Behavior Changes*

2020.18

Below is a list of *New Features* and *Behavior Changes* in the Centralized User Management interface introduced in the 2020.18 version.

New Features

Templates

- It is possible to choose WebRTC or Unique Account as a peer type - see *Create template*

Entities

- When creating an entity we can now choose a Media Server - see *Create entity*

Users

- After creating a user, a confirmation message is displayed - see *User confirmation message*
- Internal number for a user can be now chosen by typing and narrowing the available numbers result. The number is selected by clicking on it or by pressing tab key.

Behavior Changes

Entities

- Removed *not routed* mode - see *routing modes*

CHAPTER

TEN

RELEASE NOTES

10.1 Quantum

Below is a list of *New Features* and *Behavior Changes* compared to the previous LTS version, Pollux (2025.10).

10.1.1 New Features

Assistants (UC, CC)

- Users' Do Not Disturb status now appears in the search results - see *Search for UC App* and *Search for CC Agent App*.

WebRTC

- Audio devices verification feature: test the configured audio device (input/output) inside the configuration - see *Testing Audio Devices*
- Headset call control: added Yealink headset brand added - see *Headset call control (WebHID)*

Reporting

- Metabase BI tool integration (replaces SpagoBI)
- Add group table to be able to do statistics on groups and not only queues

Resilience

- Introducing Automatic or Manual failover mode - see *Failover Mode*
- Adding node type in Web interface - see *Node type and statuses*
- **With Edge**, users can automatically reconnect (or with a refresh) to the active stack

XiVO PBX

- CSV user import: ability to import and update SIP options
- *Provisioning*: Extended device line support up to 256 lines
- (*Since Quantum.03*) **Simultaneous calls limit**: The simultcalls parameter now tracks all active calls (outgoing, incoming via groups/queues, boss-secretary filter etc.) - see to *Simultaneous Calls Limit*

API

- Configmgt API for schedule management in web interface
- Schedule associations endpoints
- Resilience API enhancements

Edge:

- Add timestamp with correct timezone for SIP proxy (kamailio) and TURN server (coturn) logs

Meetingroom

- User joining via the sharing link enters now the Meetingroom **as Guest** (previously it entered as Moderator) - this behavior can be reverted, see *Enable moderator for external participants*.
- A lobby room can now be configured for guest participants - see *Enable lobby for external participants*
- (**from Quantum.03**) Can pre-fill visioconference invitation emails with configurable template using meetingroom parameters and attached data from ongoing call

System

- Debian 13 migration for XiVO PBX, XiVO CC, Edge, and Meeting Rooms
- *Edge*: SIP proxy (Kamailio) version was bumped from 5.5.0 to 6.1.1

10.1.2 Behavior Changes

System

- XiVO PBX, XiVO CC, Edge, and Meeting Rooms now install on **Debian 13 (Trixie)**
- **Timezone**: Docker containers take now there timezone from the environment variable `TZ`. It defaults *Europe/Paris*. It can be **customized** by adding in the `custom.env` file with the `TIMEZONE` env var.
- xuc JVM option is now defined as `JAVA_OPTS=${JAVA_OPTS_XUC:--Xms512m -Xmx2048m}` (instead of `JAVA_OPTS=-Xms512m -Xmx2048m`). Therefore it can be easily overriden in `custom.env`.
- Launching an upgrade now requires an explicit `y` answer (applies to `xivo-upgrade`, `mds-upgrade`, `xivocc-upgrade`)
- New log files for:
  - `xivocc-configure-xivopbx` script in `/var/log/xivocc_configure_xivopbx_*.log`
  - `xivouc-installer` script in `/var/log/xivouc_installer_*.log`
- The `xivocc-recording-config` script now automatically sanitizes IPBX name (removes space and dash (`-`) characters)

API

- Config Mgt `update_ha_config` API now uses **POST** instead of **GET**
- Schedules are now managed with configmgt APIs in the web-interface. It means configmgt must be running to be able to list / create / update / delete schedules and associate / deassociate schedules items in the web-interface

Edge

- The Kamailio component from Edge is now automatically selecting the xivo main (also known as mds0) without needing to pass it as extra_host
- The use of `XIVO_MDS_HOST_DEFAULT` is now deprecated and can safely be removed from .env and from extra_host in file `kamailio-edge.yml`
- OCSP check deactivated by default on Edge Web Proxy

Meetingroom

- User's rights changed when joining a meetingroom: a user joining via the sharing link enters now the Meetingroom **as Guest** (previously it entered as Moderator) - this behavior can be reverted, see *Enable moderator for external participants*.

	Before	After
User joins **via sharing link**	Moderator	**Guest**
User joins via Assistant (UC, CC Agent, Switchboard)	Moderator	Moderator

Provisioning

- When synchronizing a device:
  - if ip and remote_state_sip_username are known, two NOTIFY will be sent to the device: one by uri and the other by peer
  - if the notify by peer fails (for example if user is on MDS), it will fallback to another NOTIFY by uri constructed from ip and remote_state_sip_username

Reporting BI

- Metabase installed has been automated using the package xivocc-metabase only

Resilience

- New visuals for resilience, with a dedicated resilience management page, and some informations alongside the monitoring page (displayed by default after login)

XiVO PBX

- The reverse lookup should only match the fields defined in directories and not Outgoing_caller_id for example
- Recording mode description added in webi
- When creating a queue in webi, recording activation checkbox is no longer displayed
- (*Since Quantum.03*) **Simultaneous calls limit enforcement**: The simultcalls parameter now provides consistent enforcement across all call types. Previously, the limit was only enforced for direct incoming call to the user. The counter now tracks outgoing calls, calls received via groups etc. Below is a table listing the major behavior differences.

  Note: See *Simultaneous Calls Limit* for more details.

  Given a user U1 with simultcalls = 1, G1 and G2[2] groups which U1 is a member of, and Q1 and Q2[Page 995, 2] queues which U1's agent is a member of:

U1 Current Call State	New Call Towards U1	Before (Pre-Quantum)	After (Quantum)
U1 calls[1]	U2 calls U1 directly	U1 **rings**	**Busy**
U1 calls[1]	U2 calls G1 or Q1	U1 not called (InUse)	U1 not called (InUse)
U1 calls[1]	U2 calls group G2 or Q2[2]	U1 **called**	**U1 not called**
U1 on call (via group/queue)	U2 calls U1 directly	U1 **rings**	**Busy**
U1 on call (via Boss/secretary filter)	U2 calls U1 directly	U1 **rings**	**Busy**

10.1.3 Deprecations

This release deprecates:

- LTS Kuma (2023.05) : This version is no longer supported.
- SpagoBi has been deprecated in favor of Metabase.
- On newer Debian13 installs, there is longer the `/etc/timezone`: it should be removed from any docker compose volume definition.

[2] Group G2 (Queue Q2) has the *Call a member already in line* option activated
[1] Call emitted by the user to other users, groups, queues, external numbers, conferences, meeting rooms.

10.1.4 Upgrade

Manual steps for LTS upgrade

> **Warning:** **Don't forget** to read carefully the specific steps to upgrade from another LTS version

Upgrade Jabbah to Kuma

In this section is listed the manual steps to do when migrating from Jabbah to Kuma.

Contents

- *Upgrade Jabbah to Kuma*
  - *Before Upgrade*
    - *On XiVO PBX*
      - *UC Embedded*
    - *On XiVO CC*
  - *After Upgrade*
    - *On XIVO PBX*
      - *UC Embedded*
    - *On XiVO CC*

> **Warning:** Upgrade to Kuma:
> - Postgres database will be updated from 11 to version 15 during upgrade.
> - Asterisk will be updated from 18 to version 20 during upgrade.

Before Upgrade

On XiVO PBX

Important: **Behavior change** If you come from a version lower than 2022.05.10 this upgrade will take more time to REINDEX the asterisk database.

Examples: - with 1 000 000 cel and 75 000 queue_log and 50 000 call_log it takes **7 sec** (on a high performance disk Read:400MB/s, Write:500MB/s) - with 3 600 000 cel and 15 000 000 queue_log and 4 000 000 call_log it takes **22 min** (on a *low* performance disk - Read:80MB/s, Write:80MB/s)

UC Embedded

During upgrade, `pgxivocc` container will be dropped. Data from the following databases will be dropped if not backuped before the upgrade:

- spagobi: you should backup it if you added custom spagobi reports to your UC install
- recording: you should backup it if you added recording to your UC install
- xivo_stats: history will be recomputed after upgrade

On XiVO CC

After Upgrade

On XIVO PBX

- Postgres was upgraded from version 11 to version 15. During the upgrade the following postgres 11 configuration files were restored in the postgres 15 folder:
  - `pg_hba.conf`
  - and `conf.d/*.conf` files
  - The rest of the postgres 11 configuration was saved in this directory `/var/tmp/xivo-migrate-db-11-to-15/postgresql-11-conf-backup`. You might want to compare its content with what is now for postgres 15 (in `/var/lib/postgresql/15/data`).

UC Embedded

- pgxivocc and db_replic containers were removed to finish the migration you MUST run:

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

- Call history was cleaned: xivo_stats will recompute it. You can check the progress with the following command:

```
psql -U asterisk -qc '
SELECT
    x.max_cel_id,
    y.last_cel_id AS last_cel_processed,
    x.max_cel_id - y.last_cel_id
      AS Nb_cel_to_process
FROM
  (SELECT max(id) AS max_cel_id FROM cel) x
  NATURAL FULL JOIN
  (SELECT id AS last_cel_id FROM last_cel_id) y'
```

- Chat history was restored (if it is installed): you can check xivo-upgrade log to verify if everything went well.
- When you checked everything is ok, you should clean pgxivocc data:

```
rm -rf /var/lib/postgresql/xivouc/
```

On XiVO CC

Important: **Database** after the upgrade you MUST upgrade pgxivocc from 11 to version 15.

To do this, launch the script (it will restart the pgxivocc container). On XiVO CC:

```
xivocc-migrate-db-11-to-15
```

Or on XiVO with UC embedded:

```
xivouc-migrate-db-11-to-15
```

If you are installing the chat backend, you must upgrade pgxivocc first and then start the chat backend installation.

- The configuration to enable webrtc on mds has been automated (see *XiVO CC Configuration*). If you have this file */etc/docker/nginx/sip_proxy/sip_proxy.conf*, you can check that its content is the same as the now auto-generated file on the container with :

  ```
  xivocc-dcomp exec nginx cat /etc/nginx/sip_proxy/sip_proxy.conf
  ```

  You should remove it as it is no more mounted.

  ```
  rm /etc/docker/nginx/sip_proxy/sip_proxy.conf
  ```

Upgrade Kuma to Luna

In this section is listed the manual steps to do when migrating from Kuma to Luna.

Contents

- *Upgrade Kuma to Luna*
  - *Before Upgrade*
    - *On XiVO PBX*
      - *UC Embedded*
    - *On XiVO CC*
  - *After Upgrade*
    - *On XIVO PBX*
      - *Agid dockerization: email configuration*
      - *UC Embedded*
    - *On XiVO CC*

Before Upgrade

On XiVO PBX

UC Embedded

On XiVO CC

After Upgrade

On XIVO PBX

Agid dockerization: email configuration

In Luna xivo-agid service was dockerized, which affected email exchange. To make sure that mail exchange will be working, please check differences between base template and customized templated for mail configuration and make sure you include *#XIVO_DOCKER_NET#* in your custom templates (in the section *mynetworks*). In case of XDS installation, you need to do this step also on every MDS.

```
BASE_TEMPLATE_FOR_MAIL_CONFIG=/usr/share/xivo-config/templates/mail/etc/
↪postfix/main.cf
CUSTOMIZED_TEMPLATE_FOR_MAIL_CONFIG=/etc/xivo/custom-templates/mail/etc/
↪postfix/main.cf
vimdiff "$BASE_TEMPLATE_FOR_MAIL_CONFIG" "$CUSTOMIZED_TEMPLATE_FOR_MAIL_
↪CONFIG"

# Make the changes to CUSTOMIZED_TEMPLATE_FOR_MAIL_CONFIG and apply them␣
↪using following script

xivo-update-config
```

UC Embedded

On XiVO CC

Upgrade Luna to Maia

- *Before Upgrade*
  - *On XiVO PBX*
  - *On XiVO CC*
  - *On MDS*
  - *On Edge*
  - *On Meeting Rooms*
- *After Upgrade*
  - *On XiVO PBX*
  - *On MDS*
  - *On XiVO CC*

- *On Edge*
- *On Meeting Rooms*
- *Upgrade to Debian12*

Warning: For Maia Debian was upgraded to Debian 12 (Bookworm).

Therefore:

- the upgrade to Maia will take longer than usual
- upgrade from older version than XiVO Izar (2022.05) are not supported (you need first to upgrade to a version above or equal to Izar before being able to upgrade to Maia).

Please read carefully *Debian 12 (Bookworm) upgrade notes* page.

Note also that upgrade to Debian 12 on MeetingRooms, Edge and XiVO CC is manual (see manual procedure below).

Before Upgrade

On XiVO PBX

N.A.

On XiVO CC

- The XiVO CC upgrade is now handled by the `xivocc-upgrade` script.
  1. You need to install it manually to be able to use it:

     ```
     xivo-dist xivo-maia
     apt update
     apt install xivocc-upgrade
     ```

  2. Then follow the XiVO CC upgrade process as described in *Upgrade to another LTS version*

On MDS

N.A.

On Edge

- Upgrade system to Debian 12 (Bookworm):
  1. Rewrite the apt preferences for docker-ce

     ```
     #set docker-ce preferences
     cat > /etc/apt/preferences.d/docker-ce << EOF
     Package: docker-ce*
     Pin: version 5:25.0.5*
     Pin-Priority: 1000

     EOF
     ```

2. Remove docker-compose

```
rm -f /usr/local/bin/docker-compose
```

3. Install docker compose plugin

```
apt update
apt install docker-compose-plugin
```

4. Update bash alias to launch services:

```
sed -i 's/docker-compose/docker compose/g' ~/.bashrc
source ~/.bashrc
```

5. and then follow the *Debian 12 upgrade manual procedure*:

On Meeting Rooms

- Upgrade system to Debian 12 (Bookworm):
  1. Rewrite the apt preferences for docker-ce

```
#set docker-ce preferences
cat > /etc/apt/preferences.d/docker-ce << EOF
Package: docker-ce*
Pin: version 5:25.0.5*
Pin-Priority: 1000

EOF
```

  2. Remove docker-compose

```
rm -f /usr/local/bin/docker-compose
```

  3. Install docker compose plugin

```
apt update
apt install docker-compose-plugin
```

  4. Update bash alias to launch services:

```
sed -i 's/docker-compose/docker compose/g' ~/.bashrc
source ~/.bashrc
```

  5. and then follow the *Debian 12 upgrade manual procedure*:

After Upgrade

On XiVO PBX

- Upgrade: the upgrade deactivated all apt sources list except those needed for XiVO (xivo, debian, docker, postgres). After the upgrade you might need to reactivate some custom sources list you had. **Please make sure that they are compatible with Debian 12**.
- If you customized your directories, upgrade them so they match the new display filter. Default directories (xivodir, internal) have been automatically upgraded to match the new Display filter variable. Check out *internal* matching the xivo directory (internal xivo users) and *xivodir* for the phonebook contacts. You can reduce the value mapping to what stands as default, and/or update your customization.

On MDS

- Upgrade: the upgrade deactivated all apt sources list except those needed for XiVO (xivo, debian, docker, postgres). After the upgrade you might need to reactivate some custom sources list you had. **Please make sure that they are compatible with Debian 12**.

On XiVO CC

- Upgrade system to Debian 12 (Bookworm) with the *following manual procedure*:

On Edge

Ensure you correctly replaced the alias edge-dcomp *as stated in the documentation*.

On Meeting Rooms

Ensure you correctly replaced the alias meetingrooms-dcomp *as stated in the documentation*.

Upgrade to Debian12

- These steps are to be done before an upgrade from Luna to Maia on **XiVO CC**, **Edge** and **Meeting Rooms**:
  1. Check GRUB **before upgrading**

     ```
     install_device=$(debconf-show grub-pc | grep 'grub-pc/install_devices:' |␣
     ↪cut -b3- | cut -f2 -d' ' | cut -d',' -f1)
     if [ "$install_device" -a ! -e "$install_device" ]; then
       echo -e "\e[1;31mYou must install GRUB BEFORE upgrading\e[0m"
     fi
     ```

     If it's broken you can fix it this way before rechecking

     ```
     apt update
     apt install grub-pc
     dpkg-reconfigure grub-pc
     ```

  2. Check you are in Debian 11 (we prevent upgrade from Debian 10 - see https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=993755#62)

     ```
     if [ "$(cut -d '.' -f 1 /etc/debian_version)" -ne 11 ]; then
       echo -e "\e[1;31mYou are not in Debian 11. Please upgrade first to Debian␣
     ↪11\e[0m"
       echo -e "\e[1;31mSee Helios to Izar Upgrade page.\e[0m"
     fi
     ```

  3. Upgrade to Debian12

     ```
     # Move to bookworm
     sed -i 's/bullseye/bookworm/' /etc/apt/sources.list /etc/apt/sources.list.d/
     ↪*.list
     apt update

     export DEBIAN_FRONTEND=noninteractive
     export APT_LISTCHANGES_FRONTEND=none
     ```

(continues on next page)

(continued from previous page)

```
force_yes="--allow-downgrades --allow-remove-essential --allow-change-held-
↪packages"

echo "Download packages..."
apt full-upgrade -d --yes
echo "Executing full upgrade actions..."
apt full-upgrade --yes ${force_yes} -o Dpkg::Options::="--force-confnew"
apt autoremove --yes
```

4. Reboot

Upgrade Maia to Naos

Contents

- *Upgrade Maia to Naos*
  - *Before Upgrade*
    - *On XiVO PBX*
    - *On XiVO CC*
    - *On MDS*
    - *On Edge*
    - *On Meeting Rooms*
  - *After Upgrade*

Before Upgrade

On XiVO PBX

N.A.

On XiVO CC

N.A

On MDS

N.A

On Edge

N.A

On Meeting Rooms

N.A

After Upgrade

N.A

Upgrade Naos to Orion

Contents

- *Upgrade Naos to Orion*
  - *Before Upgrade*
    - *On XiVO PBX*
    - *On XiVO CC*
    - *On MDS*
    - *On Edge*
    - *On Meeting Rooms*
  - *After Upgrade*

Before Upgrade

On XiVO PBX

- If you have *Recording On Gateway Features* configured you have to update the subroutine in the configuration file, see Gateway Configuration section: *2. Enable recording*

On XiVO CC

N.A

On MDS

N.A

On Edge

N.A

On Meeting Rooms

N.A

After Upgrade

N.A

Upgrade Orion to Pollux

In this section is listed the manual steps to do when migrating from Orion to Pollux.

Contents

- *Upgrade Orion to Pollux*
  - *Before Upgrade*
    - *On XiVO PBX*
    - *On XiVO CC*
      - *XIVOCC_TYPE Variable*
    - *On MDS*
    - *On Edge*
    - *On Meeting Rooms*
  - *After Upgrade*
    - *On XiVO PBX*
      - *PostgreSQL upgrade*
      - *High Availability*
    - *On XiVO CC*
      - *Variable Cleanup and Validation*
      - *Manual Review*
      - *Log Location*
    - *On MDS*
      - *PostgreSQL upgrade*
    - *On Edge*
    - *On Meeting Rooms*

Warning: Upgrade to Pollux:

- Postgres database will be updated from 15 to version 17 during upgrade.

Before Upgrade

On XiVO PBX

When upgrading from a Debian 11 (Bullseye) based version (Izar to Luna) you might need to manually remove the bullseye-backport repository before launching upgrade:

```
sed '/bullseye-backports/d' -i /etc/apt/sources.list
```

On XiVO CC

XIVOCC_TYPE Variable

- Before starting the upgrade, you MUST define the variable `XIVOCC_TYPE` in the `custom.env` file.

  This variable indicates the installation type, allowing the installer to clean up and validate the environment accordingly.

  Accepted values:

  - `full`: Complete installation including all components (XUC, XUC management, reporting, recording, and NGINX).
  - `app`: Application-only installation including **xuc**, **xucmgt**, and **nginx**.
  - `backend`: Reporting and recording modules only.

  If the variable is missing, it will be set to `full` by default, and a warning will be displayed in the log file:

  ```
  WARN: XIVOCC_TYPE variable has been automatically set to 'full' (fully␣
  ↪installed).
  ```

  Important: You should explicitly set the correct value to avoid unnecessary configuration steps or invalid cleanup.

On MDS

When upgrading from a Debian 11 (Bullseye) based version (Izar to Luna) you might need to manually remove the bullseye-backport repository before launching upgrade:

```
sed '/bullseye-backports/d' -i /etc/apt/sources.list
```

On Edge

N.A

On Meeting Rooms

N.A

After Upgrade

On XiVO PBX

PostgreSQL upgrade

If the upgrade went well, it's suggested to remove the old postgres log files:

```
rm -rf /var/log/postgresql/postgresql-15-main*.log
rm -rf /var/log/postgresql/postgresql-11-main*.log
```

High Availability

Important: When upgrading a High Availibility installation from an older LTS (like Maia) you **MUST** follow these additionnals steps. These steps deactivate and then reactivate the High Availability in order to ensure the new Resilience features are available.

Warning: Services will be restarted during this procedure.

1. Disable **XiVO Slave** - see *Disable node*,

   Warning: After this step services may need to be stopped on the slave in order to not disturb telephony operation on the Master.

2. Disable **XiVO Master** - see *Disable node*,
3. Ensure needed steps of *Resilience Standby Stack Installation* are followed depending on the deployment type. It consists mainly in configuring/re-configuring the CC associated to the *XiVO Standby*.
4. And finally reconfigure the resilience as described in *Resilience Configuration Details* to ensure Pollux Resilience features are available

On XiVO CC

Variable Cleanup and Validation

- The `custom.env` file is now automatically **validated and cleaned** depending on the value of `XIVOCC_TYPE`.
- The script performs the following actions:
  - Only variables relevant to the selected type are retained.
  - Deprecated or unused variables are removed.
  - Required variables are added if missing (with default or empty values).
  - Some values are validated (e.g., IP format check for `REPORTING_HOST`).
- Examples of automatic handling:

```
WARN: 'RECORDING_SERVER_HOST' is not an IP address. Resetting its value in␣
↪custom.env.
INFO: Setting 'REPORTING_HOST' from 'XUC_HOST=10.10.10.20' in custom.env.
INFO: custom.env cleaned and validated for type 'backend'.
```

- If optional variables are missing, they are appended as empty keys.

Manual Review

Important: After the upgrade, you MUST review the `custom.env` file manually.

In particular:

- Verify the value of `XIVOCC_TYPE`.
- If the installer cleared or reset any variables, fill in the correct values.
- Ensure required IP addresses or hostnames are properly defined.

Log Location

- All operations performed on `custom.env` are logged in:

```
/var/log/xivocc-installer.log
```

- You should read this log file to confirm that the migration proceeded correctly and that no manual steps remain.

On MDS

PostgreSQL upgrade

If the upgrade went well, it's suggested to remove the old postgres log files:

```
rm -rf /var/log/postgresql/postgresql-15-main*.log
rm -rf /var/log/postgresql/postgresql-11-main*.log
```

On Edge

Note: Specific steps to workaround changes from containerd.io 2.x

1. Ensure containerd is pinned to the right version and configured with the new `LimitNOFILE` value:

```
cat > /etc/apt/preferences.d/containerd <<EOF
Package: containerd.io*
Pin: version 2.*
Pin-Priority: 1000
EOF

mkdir -p /etc/systemd/system/containerd.service.d
cat > /etc/systemd/system/containerd.service.d/override.conf <<EOF
```

(continues on next page)

(continued from previous page)

```
[Service]
LimitNOFILE=100000
EOF

systemctl daemon-reload
systemctl restart containerd.service
```

2. Restart the containers to apply the services

```
edge-dcomp up -d
```

On Meeting Rooms

Note: Specific steps to workaround changes from containerd.io 2.x

1. Ensure containerd is pinned to the right version and configured with the new `LimitNOFILE` value:

```
cat > /etc/apt/preferences.d/containerd <<EOF
Package: containerd.io*
Pin: version 2.*
Pin-Priority: 1000
EOF

mkdir -p /etc/systemd/system/containerd.service.d
cat > /etc/systemd/system/containerd.service.d/override.conf <<EOF
[Service]
LimitNOFILE=100000
EOF

systemctl daemon-reload
systemctl restart containerd.service
```

2. Restart the containers to apply the services

```
meetingrooms-dcomp up -d
```

Upgrade Pollux to Quantum

Contents

- *Upgrade Pollux to Quantum*
  - *Before Upgrade*
    - *On XiVO PBX*
    - *On XiVO CC*
    - *On MDS*
    - *On Edge*
    - *On Meeting Rooms*
  - *After Upgrade*

    * *On XiVO PBX*
    * *On XiVO CC*
    * *On MDS*
    * *On Edge*
    * *On Meeting Rooms*
  - *Upgrade to Debian 13*

Warning: For Quantum Debian was upgraded to Debian 13 (Trixie).

Therefore:

- the upgrade to Quantum will take longer than usual
- upgrade from older version than XiVO Maia (2024.05) are not supported (you need first to upgrade to a version above or equal to Maia before being able to upgrade to Quantum).

Please read carefully *Debian 13 (Trixie) upgrade notes* page and specifcally the *Major changes associated with Debian 13*.

Note also that upgrade to Debian 13 on MeetingRooms and Edge is manual (see manual procedure below).

Before Upgrade

On XiVO PBX

N.A.

On XiVO CC

- SpagoBI container was removed: remove any mention of spagobi service in any configured docker compose override file.

On MDS

N.A

On Edge

- Ensure `XIVO_MDS_HOST...` variables have the correct separator between *hostname* and *IP* for new docker compose:

  Important: **This is a mandatory step**: Edge `.env` file contains `XIVO_MDS_HOST_MDS...` variables used as an `extra_hosts` value in the kamailio compose file.

  They were defined with `hostname: IP` (with *column* followed by a *space* between *hostname* and *IP* which is not supported anymore by docker compose). They **MUST** now be defined with an = separator between *hostname* and *IP* like: `hostname=IP`

  The `XIVO_MDS_HOST_DEFAULT` won't be used anymore, but you must wait *the upgrade end* to remove it.

  The changes needed are illustrated in the tables below:

For .env file:

Before	After

Listing 1: `.env` file extract
`...`
`TURN_EXTERNAL_IP=DMZ_IP1`
`XIVO_MDS_HOST_DEFAULT="default: LAN_IP2"`
`XIVO_MDS_HOST_MDS1="mds1: LAN_IP3"`
	Listing 2: `.env` file extract
	`...`
	`TURN_EXTERNAL_IP=DMZ_IP1`
	`XIVO_MDS_HOST_DEFAULT="default=LAN_IP2"`
	`XIVO_MDS_HOST_MDS1="mds1=LAN_IP3"`

Run the commmand below to change separator in `XIVO_MDS_HOST...` variables:

```
sed -i '/XIVO_MDS_HOST/s/: /=/' /etc/docker/edge/.env
```

- If Edge is used in a *Resilient architecure* adjust Edge configuration as follows:
  - Add the `XIVO_HOST_STANDBY` and `XIVOCC_HOST_STANDBY` variables in Edge `.env` file (replace <..._IP> with their actual values):

    ```
    XIVO_HOST_STANDBY=<XIVO_STANDBY_IP>
    XIVOCC_HOST_STANDBY=<XIVOCC_STANDBY_IP>
    ```

  - Also ensure that XiVO Standby IP is among the allowed peers for TURN configuration in Edge `.env`:

    ```
    TURN_ALLOWED_PEERS="--allowed-peer-ip=<XIVO_MAIN_IP> --allowed-peer-ip=<XIVO_STANDBY_IP> --allowed-peer-ip=... (MDS_IP, MEETINGROOM_IP, etc)"
    ```

- Upgrade system to Debian 13 (Trixie):
  1. Rewrite the apt preferences for docker-ce and containerd.io:

     ```
     #set docker-ce preferences
     cat > /etc/apt/preferences.d/docker-ce << EOF
     Package: docker-ce*
     Pin: version 5:28*
     Pin-Priority: 1000
     EOF

     cat > /etc/apt/preferences.d/containerd <<EOF
     Package: containerd.io*
     Pin: version 2.*
     Pin-Priority: 1000
     EOF
     ```

  2. Ensure override for containerd service is installed:

     ```
     mkdir -p /etc/systemd/system/containerd.service.d
     cat > /etc/systemd/system/containerd.service.d/override.conf <<EOF
     [Service]
     LimitNOFILE=100000
     EOF

     systemctl daemon-reload
     ```

  3. and then follow the *Debian 13 upgrade manual procedure*

On Meeting Rooms

- Upgrade system to Debian 13 (Trixie):
  1. Rewrite the apt preferences for docker-ce and containerd.io:

```
#set docker-ce preferences
cat > /etc/apt/preferences.d/docker-ce << EOF
Package: docker-ce*
Pin: version 5:28*
Pin-Priority: 1000
EOF

cat > /etc/apt/preferences.d/containerd <<EOF
Package: containerd.io*
Pin: version 2.*
Pin-Priority: 1000
EOF
```

  2. Ensure override for containerd service is installed:

```
mkdir -p /etc/systemd/system/containerd.service.d
cat > /etc/systemd/system/containerd.service.d/override.conf <<EOF
[Service]
LimitNOFILE=100000
EOF

systemctl daemon-reload
```

  3. and then follow the *Debian 13 upgrade manual procedure*

After Upgrade

On XiVO PBX

- Reboot (Debian was upgraded)

On XiVO CC

- xuc JVM option is now defined as `JAVA_OPTS=${JAVA_OPTS_XUC:--Xms512m -Xmx2048m}` (instead of `JAVA_OPTS=-Xms512m -Xmx2048m`). Check that after the upgrade the xuc takes the correct parameters as expected on the installation. For example it may be possible to remove the override that defined *JAVA_OPTS=${JAVA_OPTS_XUC}*.
- Reboot (Debian was upgraded)

On MDS

- Reboot (Debian was upgraded)

On Edge

- `XIVO_MDS_HOST_DEFAULT` variable is not needed anymore and should be removed from edge's `.env`.

  For .env file:

Before	After
Listing 3: `.env` file extract	Listing 4: `.env` file extract

```
...
TURN_EXTERNAL_IP=DMZ_IP1
XIVO_MDS_HOST_DEFAULT=
↪"default=LAN_IP2"
XIVO_MDS_HOST_MDS1="mds1=LAN_IP3"
```

```
...
TURN_EXTERNAL_IP=DMZ_IP1
XIVO_MDS_HOST_MDS1="mds1=LAN_IP3"
```

  Then apply the changes to the container:

```
edge-dcomp up -d
```

On Meeting Rooms

N.A

Upgrade to Debian 13

These steps are to be done before an upgrade from Pollux to Quantum on **Edge** and **Meeting Rooms**

- Check GRUB **before upgrading**

```
install_device=$(debconf-show grub-pc | grep 'grub-pc/install_devices:
↪' | cut -b3- | cut -f2 -d' ' | cut -d',' -f1)
if [ "$install_device" -a ! -e "$install_device" ]; then
  echo -e "\e[1;31mYou must install GRUB BEFORE upgrading\e[0m"
fi
```

- If it's broken you can fix it this way before rechecking

```
apt update
apt install grub-pc
dpkg-reconfigure grub-pc
```

- Check you are in Debian 12

```
if [ "$(cut -d '.' -f 1 /etc/debian_version)" -ne 12 ]; then
  echo -e "\e[1;31mSystem is not in Debian 12. Please upgrade first to␣
↪Debian 12\e[0m"
  echo -e "\e[1;31mSee Luna to Maia Upgrade page.\e[0m"
fi
```

- Check if /boot is a separate partition. If so ensure it is at least 768 MB in size, and has about 300 MB free.

```
boot_mount_point=$(df /boot --output=target |tail -n +2)
if [ "${boot_mount_point}" = "/boot" ]; then
  echo -e "\e[1;31m/boot is a separate partition.\e[0m"
  echo "Please check it is at least 768 MB in size, and has about 300␣
↪MB free."
else
  echo "/boot is not a separate partition. Nothing to do."
fi
```

- Install new key and refresh the docker repository

```
# Add Docker's official GPG key:
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/
↪docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: trixie
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
```

- Remove the previous docker source file

```
rm /etc/apt/sources.list.d/docker.list
```

- Upgrade to Debian 13

```
# Move to trixie
sed -i 's/bookworm/trixie/' /etc/apt/sources.list /etc/apt/sources.
↪list.d/*.list
apt update

export DEBIAN_FRONTEND=noninteractive
export APT_LISTCHANGES_FRONTEND=none
force_yes="--allow-downgrades --allow-remove-essential --allow-change-
↪held-packages"

echo "Download packages..."
apt full-upgrade -d --yes
echo "Executing full upgrade actions..."
apt full-upgrade --yes ${force_yes} -o Dpkg::Options::="--force-confnew
↪"
apt autoremove --yes
```

- **Before reboot** check potential network interface name change - see Network interface names may change
  Use this script to know whether the change might happen.

```
echo ""
echo "Checking potential network interface name change:"
iface_rename=0
# Check whether predictable name was deactivated
if grep -q "net.ifnames=0" /proc/cmdline; then
  echo "  So-called 'Predictable Name' is deactivated per kernel␣
↪cmdline (net.ifnames=0)."
  echo "  Skipping this check as unrelevant."
else
  for net_iface in /sys/class/net/*; do
    net_iface_name=$(basename "${net_iface}")
    # Skip virtual interface
    if readlink -f "${net_iface}" | grep -q "virtual"; then
      continue
    fi
    echo -n "  ${net_iface_name} ... "
    net_iface_new_name=$(udevadm test-builtin net_setup_link "${net_
↪iface}" 2>/dev/null| grep -oP 'ID_NET_NAME=\K.*')
    if [ "${net_iface_name}" = "${net_iface_new_name}" ]; then
      echo "OK (won't be renamed)"
    else
      if grep -q "${net_iface_name}" /etc/udev/rules.d/70-persistent-
↪net.rules; then
          iface_rename=1
          echo -e "\e[1;33m WARNING: might be renamed to ${net_iface_
↪new_name} but name should be kept via legacy udev rules\e[0m"
          echo "    Interface ${net_iface_name} is present in /etc/
↪udev/rules.d/70-persistent-net.rules therefore it should not be␣
↪renamed."
          echo -e "    \e[1;31mBEFORE REBOOTING\e[0m ensure this is␣
↪actually true by checking this udev rules file."
          echo "    You might also consider to migrate legacy /etc/
↪udev/rules.d/70-persistent-net.rules to a systemd .link files."
      else
          iface_rename=2
          echo -e "\e[1;31m NOK! ALERT: will be renamed to ${net_iface_
↪new_name}\e[0m"
          echo -e "    \e[1;31mBEFORE REBOOTING\e[0m create a systemd .
↪link file for this interface in order to ensure stable name"
          echo -e "\e[1;31m     If not done it will be impossible to␣
↪reconnect via SSH after reboot\e[0m"
      fi
    fi
  done
  if [ $iface_rename -ne 0 ]; then
    echo "  Find an example of systemd .link file here:"
    echo "    https://wiki.debian.org/NetworkInterfaceNames#CUSTOM_
↪SCHEMES_USING_.LINK_FILES"
  fi
fi
echo ""
```

- Reboot

Generic upgrade procedure

Then, follow the generic upgrade procedures:

- *XiVO PBX upgrade procedure*
- *XiVO CC upgrade procedure*
- *XDS upgrade procedure*
- *Resilience upgrade procedure*

10.2 Quantum Bugfixes Versions

10.2.1 Components version table

Table listing the current version of the components.

Note: Because the **SpagoBI** technology is growing old and does not receive open-source updates, we won't build our SpagoBI image after the one from **Maia: 2024.05.latest**. That image will still work just as before for further releases, starting with Naos.

Component | current ver.	
XiVO	
xivo	2026.05.03
config-mgt	2026.05.03
db	2026.05.02
outcall	2026.05.02
db_replic	2026.05.00
nginx	2026.05.00
webi	2026.05.03
stats-reports-exporter	2026.05.03
usage-writer	2026.05.00
usage-collector	2026.05.00
agid	2026.05.03
confgend	2026.05.03
asterisk	8:22.2.0-1
docker-ce	5:28.5.2
XiVO CC	
mattermost	2026.05.00
nginx	2026.05.00
pack-reporting	2026.05.00
pgxivocc	2026.05.00
recording-rsync	2024.10.04
recording-server	2026.05.02
metabase	2026.05.03
full-stats	2026.05.02
xucserver	2026.05.03
xucmgt	2026.05.04
Edge	
edge	2026.05.02
nginx	2026.05.00
kamailio	2026.05.00
coturn	2026.05.00
Meeting Rooms	
meetingroom	2026.05.00
web-jitsi	2026.05.00

continues on next page

Table 1 – continued from previous page

Component | current ver.	
jicofo-jitsi	2026.05.00
prosody-jitsi	2026.05.00
jvb-jitsi	2026.05.00
jigasi-jitsi	2026.05.00
IVR	
ivr-editor	2026.05.00

10.2.2 2026.05.04 (Quantum.04): June 2026

Consult the 2026.05.04 (Quantum.04) Roadmap.

Components updated:

Docker: xucmgt

UC App

- #9127 - Xucmgt applications should not retry the connexion for every auth error types

10.2.3 2026.05.03 (Quantum.03): June 2026

Consult the 2026.05.03 (Quantum.03) Roadmap.

Components updated:

Docker: config-mgt, metabase_export_import, xivo-agid, xivo-confgend, xivo-metabase, xivo-metabase-reports, xivo-stats-reports-exporter, xivo-web-interface, xucmgt, xucserver

Debian: xivo-config, xivo-service, xivo-upgrade, xivocc-installer, xivo

API

- #9109 - API - handle error cases for schedules

CCAgent

- #9099 - Agents and activities popover may breaks display in desktop assistant

CTI

- #9067 - Resilience - After a fallback on standby stack UC app is stuck
- #9088 - Upgrade to latest JDK bugfix - stats report exporter

Meeting Room

- #9027 - As a user I can send the meetingroom share link via mail with template

Reporting

- #9070 - Agents report - Queues calls counter wrong when a call overflow from one queue to the other

Reporting BI

- #9021 - Add doc for Metabase
- #9022 - Enhance sample reports import

> **Important:** **Behavior change** metabase installed has been automated using the package xivocc-metabase only

System

- #8801 - Resilience (BRP/PRA) - As an admin I want to be able to chose between auto or manual takeover
- #9025 - Resilience - Add resilience management table in web interface monitoring page

> **Important: Behavior change** New visuals for resilience, with a dedicated resilience management page, and some informations alongside the monitoring page (displayed by default after login)

- #9066 - Resilience - Can't login after a fallback on standby stack
- #9089 - Resilience - Fix service order and refresh in resilience management page

UC App

- #9090 - Forward destination not displayed in header when fwd number enter in E.164 format

WebRTC

- #9116 - UA account is not detected as WebRTC if there is another custom sip options set before webrtc

XiVO PBX

- #8643 - Take into account user's simultcall for incoming and outgoing calls

> **Important: Behavior change** The simultcalls parameter now provides consistent enforcement across all call types. Previously, the limit was only enforced for direct incoming call to the user. The counter now tracks outgoing calls, calls received via groups etc.
>
> See *Simultaneous Calls Limit* for details.

- #9032 - Debian 13 - Upgrade fixes

10.2.4 2026.05.02 (Quantum.02): May 2026

Consult the 2026.05.02 (Quantum.02) Roadmap.

Components updated:

Docker: config-mgt, edge-kamailio, edge-nginx, rabbitmq, recording-server, xivo-agid, xivo-confgend, xivo-db, xivo-db-replication, xivo-edge, xivo-full-stats, xivo-outcall, xivo-web-interface, xucmgt, xucserver

Debian: xivo-confd, xivo-provisioning, xivo-sysconfd, xivo-upgrade, xivocc-installer, xivocc-recording, xivo

API

- #9009 - As an admin using XiVO APIs in the swagger is more guided

CTI

- #9057 - Token expiration and renewal problem between xuc and xivo (auth, dird, ctid ...)

Edge

- #9060 - Add time in Edge components log
- #9063 - Edge - Remove schedules API available through edge (revert #9026)

Meeting Room

- #9037 - Meetingrooms - Distinguish between internal users as admin and guests as guest

Provisioning

- #9036 - Webi - Installing plugin does not work

Recording

- #9055 - Empty recording file is not deleted

System

- #9023 - As an admin I want a verification popup when I chose to start/stop services on Main/Standby
- #9024 - Debian 13 - Update db, confgend, agid to Debian13
- #9043 - Enable services when disabling HA

UC App

- #8984 - Fix Mattermost version to 2026.05
- #9046 - Incorrect title for meeting room status

XiVO PBX

- #9013 - It's not possible to order schedules
- #9034 - Webi - Control system menu is not visible if Resilience is not activated
- #9061 - Manual post installation wizard configuration ends with a timezone error
- #9064 - Ensure xivo repo key is updated during upgrade

10.2.5 2026.05.01 (Quantum.01): April 2026

Consult the 2026.05.01 (Quantum.01) Roadmap.

Components updated:

Debian: xivo-dao, xivo

CC Agent

- #9033 - Not possible to log agent

10.2.6 2026.05.00 (Quantum.00): April 2026

Consult the 2026.05.00 (Quantum.00) Roadmap.

Components updated:

Docker: config-mgt, debian-builder, edge-kamailio, edge-nginx, recording-server, xivo-agid, xivo-confgend, xivo-db, xivo-edge, xivo-metabase, xivoxc-nginx, xucmgt, xucserver

Debian: xivo-agentd, xivo-agentd-cli, xivo-agentd-client, xivo-amid, xivo-amid-client, xivo-auth, xivo-auth-client, xivo-auth-keys, xivo-bus, xivo-call-logs, xivo-confd, xivo-confd-client, xivo-config, xivo-dao, xivo-dird, xivo-dird-client, xivo-dird-phoned, xivo-dist, xivo-dxtora, xivo-dxtorc, xivo-fetchfw, xivo-lib-python, xivo-lib-rest-client, xivo-manage-db, xivo-monitoring, xivo-provd-cli, xivo-provd-client, xivo-provisioning, xivo-purge-db, xivo-python-celery-packaging, xivo-python-dumbnet-packaging, xivo-service, xivo-sync, xivo-sysconfd, xivo-upgrade, xivo-utils, xivocc-installer, xivocc-recording, xivo

Others: xivo-desktop-assistant, xivo-install-script, xivo-test-helpers

API

- #8876 - Configmgt - I can remove all dialaction referencing a user
- #9026 - Edge - Add schedules API available through edge

Desktop Assistant

- #8993 - Add Always on top icon in desktop assistant

Edge

- #8699 - Resilience (BCP/PCA) - with Edge - As a User I want to be able to reconnect automatically (or with a refresh) to the Active stack

> **Important:** **Behavior change** The Kamailio component from Edge is now automatically selecting the the xivo main (also known as mds0) without needing to pass it as extra_host. Thus the use of XIVO_MDS_HOST_DEFAULT is now deprecated and can safely be removed from .env and from extra_host in file *kamailio-edge.yml* Check the Edge Configuration on Resilience Installation to see what to change for resilience on a edge deployment.

Reporting BI

- #8943 - âœ¨ Be able to create a dashboard for queue report
- #8999 - Access Metabase on its HTTPS url follow up

System

- #8950 - Debian 13 - Build asterisk and dahdi-dkms
- #8962 - Slave services must be stopped after resilience init
- #8977 - Tell monit to remonitor the services after slave services are started

UC App

- #5212 - As a user, I want to know when my colleague is in Do Not Disturb status
- #9008 - As a user I want to be able to verify my audio devices
- #9017 - PhoneStatusUpdate message is incorrect after asterisk crash

XiVO PBX

- #8820 - XiVO PBX can be installed on Debian 13
- #9029 - Asterisk configuration reload might fail because confgend is not ready

10.3 Quantum Intermediate Versions

10.3.1 XiVO Quantum Intermediate Versions

2026.04.01 (IV Quantum): March 2026

Consult the 2026.04.01 (IV Quantum) Roadmap.

Components updated:

Docker: config-mgt, metabase_export_import, xivo-db-replication, xivo-edge, xivo-metabase, xivo-web-interface, xivoxc-nginx

Debian: xivo-dird, xivo-sysconfd, xivocc-installer, xivo

Directory

- #8965 - Be able to request directory per label

Reporting

- #8952 - Add group table to be able to do statistics on groups and not only queues

Reporting BI

- #8940 - As an admin I want to be able to access Metabase on its HTTPS url
- #8944 - Be able to create a dashboard for agent report

System

- #8909 - Be able to stop Master stack

edge

- #8976 - Deactivate by default the OCSP check in edge

2026.04.00 (IV Quantum): March 2026

Consult the 2026.04.00 (IV Quantum) Roadmap.

Components updated:

Docker: config-mgt, xivo-jicofo-jitsi, xivo-jigasi-jitsi, xivo-jvb-jitsi, xivo-meetingrooms, xivo-metabase, xivo-prosody-jitsi, xivo-web-interface, xivo-webi-nginx, xucmgt, xucserver

Debian: xivo-config, xivo-service, xivo-sysconfd, xivocc-installer, xivo

Others: xivo-desktop-assistant, xivo-doc-devices

Asterisk

- #8975 - Put back xivo-generic-sip-check-peer subr

Desktop Assistant

- #8806 - Screen sharing is not working in desktop application
- #8970 - Desktop App - msi installer - Cannot add application to session startup app

Reporting BI

- #8812 - âœ¨ As an admin I want to be able to install Metabase Addon on an existing CC

System

- #8794 - Handle error messages in webi
- #8907 - Be able to trigger slave services start and stop
- #8908 - Be able to display HA Node Type in Webi header Master / Slave
- #8948 - xivouc installer should take xuc_host variable if available
- #8960 - Chat backend can cause error with specific char
- #8961 - Install CC and then configure PBX afterward is not working

WebRTC

- #8842 - Check headset - Clarify message for compatible but not supported brand
- #8933 - Palliative for no sound on first call with Windows 11 and Chrome

XiVO PBX

- #8861 - Webi - Be able to filter schedule

2026.03.00 (IV Quantum): February 2026

Consult the 2026.03.00 (IV Quantum) Roadmap.

Components updated:

Docker: config-mgt, metabase_export_import, xivo-agid, xivo-full-stats, xivo-metabase, xivo-usage-collector, xivo-usage-writer, xivo-web-interface, xucmgt, xucserver

Debian: xivo-sysconfd, xivo-upgrade, xivocc-installer, xivo

API

- #8733 - API - Dynamic filters schedules

CC Agent

- #8848 - CC Agent - Agent mobile set up does not work
- #8897 - CC Agent - Can't see agent of my group

Provisioning

- #8918 - XiVO Provisioning / Webi - Extend number of available lines for devices to 256

Reporting

- #8530 - Bump scala in xivo-full-stats
- #8929 - xivo_stats can't hanble cel table when id are not correlated with timestamp

Reporting BI

- #8773 - Be able to visualize and import an equivalent of the agent support report

System

- #8885 - Sometimes, the generated mattermost user password generated is invalid (missing symbols)
- #8904 - Be able to chose between automatic or manual failover
- #8923 - Update mirror key at upgrade
- #8937 - Chat backend install fails if password contains '$'

UC App

- #8847 - UC App - ensure all icons are cache busted

Usage statistics

- #8894 - USM metrics to fix

WebRTC

- #8841 - HID - Integrate Yealink brand (BH70)
- #8852 - WebRTC - remove deprecated API calls and add log when SDP creation fails

XiVO PBX

- #8912 - When I create a queue in webi, I should not see checkbox to activate recording
- #8916 - Title: general_5_info field does not display entered value in dird
- #8935 - Do not fail callerid_forphones if XIVO_INCALL_ID is not set

2026.02.00 (IV Quantum): January 2026

Consult the 2026.02.00 (IV Quantum) Roadmap.

Components updated:

Docker: config-mgt, metabase_export_import, xivo-metabase, xivo-web-interface, xucserver

Debian: xivo-confd, xivo-provisioning, xivocc-installer, xivocc-recording, xivo

Others: xivo-install-script

API

- #8869 - Enhance HA API - sysconfd API proxy

> **Important: Behavior change** The Config Mgt *update_ha_config* API now uses POST instead of GET

CTI

- #8755 - PoC - Try to add a reload mechanism in xuc
- #8863 - Introducing remote actor for Conference tracker

Provisioning

- #8879 - The remote_state_sip_username value is never inserted into the device configuration.

> **Important: Behavior change** When synchronizing a device:
>
> – if ip and remote_state_sip_username are known, two NOTIFY will be sent to the device: one by uri and the other by peer
>
> – if the notify by peer fails (for example if user is on MDS), it will fallback to another NOTIFY by uri constructed from ip and remote_state_sip_username

Reporting BI

- #8768 - Be able to visualize an equivalent of the queue support report

System

- #8557 - Resilience (BCP/PCA) - (Doc) As an Admin I want xuc on CC slave to reconnect automatically
- #8850 - As an admin I want to be able to install UC Addon silently

> **Important: Behavior change** New log files for
>
> – xivocc-configure-xivopbx script in `/var/log/xivocc_configure_xivopbx_*.log`
>
> – xivouc-installer script in `/var/log/xivouc_installer_*.log`

- #8853 - As an admin I want to be able to install xivocc-recording silently

> **Important: Behavior change** The `xivocc-recording-config` script now automatically sanitizes IPBX name (removes space and dash (`-`) characters) - ref:*recording_configuration*.

UC App

- #8856 - RabbitMQ messages are duplicated when rabbitMQ connection is restarted

XiVO PBX

- #8800 - CSV user import: Be able to import and update sip options in user csv import
- #8839 - Missing footer in callback list creation

2026.01.00 (IV Quantum): January 2026

Consult the 2026.01.00 (IV Quantum) Roadmap.

Components updated:

Docker: config-mgt, recording-server, xivo-webi-nginx, xucmgt

Debian: xivo-dist, xivo-upgrade, xivocc-installer, xivo

Others: xivo-install-script

Auth Delegation

- #8830 - Auth delegation - When configured on recording server it breaks HTTP API calls

CC Agent

- #8828 - Pause status list is not updated in CC Agent when agent (un)paused from elsewhere

Edge

- #8760 - Edge and meetingrooms can be installed on a Debian 13

Mobile App

- #8805 - MobileApp >= 2.6.4 cannot connect on UC Addon server - missing version file

System

- #8793 - XiVOCC can be installed on Debian 13

> **Important: Behavior change** * XiVOCC installs now on Debian 13 * Launching an upgrade now needs an explicit `y` answer (applies to `xivo-upgrade`, `mds-upgrade`, `xivocc-upgrade`)

XiVO PBX

- #8811 - Configmgt - Bump play & java version

2025.11.00 (IV Quantum): December 2025

Consult the 2025.11.00 (IV Quantum) Roadmap.

Components updated:

Docker: config-mgt, xivo-agid, xivo-web-interface, xucmgt

Debian: xivo-dird, xivo-dist, xivo-upgrade, xivocc-installer, xivo

Others: xivo-desktop-assistant

API

- #8563 - API - CRUD Schedule
- #8732 - API - Schedule associations

CC Manager

- #8797 - CC Manager - Some actions on agents are not displayed

Desktop Assistant

- #8712 - Bump electron version for Quantum

Mobile App

- #8786 - Revert playing silence when calling mobile app

Reporting

- #8683 - âœ¨ Be able to install Metabase as a BI tool instead of SpagoBI
- #8780 - Upgrade fails if /var/log/xivo-switchboard-reports/ is empty
- #8799 - Missing translation for statistics menu in webi

System

- #8779 - UC addon - Upgrade fails because of wrong override definition (mentionning switchboard_report)
- #8782 - Upgrade from below Pollux fails if pgxivocc is stopped
- #8787 - containerd.io upgrade to 2.x creates a problem of Too many open files

UC App

- #8781 - Unique account is not working without Edge
- #8783 - Call management menu is not displayed when a WebRTC user associates a MobileApp and activates Headset integration

XiVO PBX

- #8566 - Schedule - Display linked object in the schedule
- #8731 - Use Configmgt API in webi for schedules

> **Important:** **Behavior change** Schedules are now managed with configmgt APIs in the web-interface. It means configmgt must be running to be able to list / create / update / delete schedules and associate / deassociate schedules items in the web-interface.

- #8735 - Webi - schedule associations api
- #8757 - Reverse lookup on incoming call shouldn't match outgoingcallerid of internal user when not configured

> **Important:** **Behavior change** The reverse lookup should only match the fields defined in directories and not Outgoing_caller_id for exemple.

- #8798 - Describe recording mode in webi

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- search

INDEX

C

ctiserver, 233

D

devices, 240

E

environment variable
    REPORTING_HOST, 1007
    XIVOCC_TYPE, 1006–1008

I

interconnections, 277, 279, 280, 282

M

mail, 131

N

network, 132

R

REPORTING_HOST, 1007

U

users, 347

V

VLAN, 133

W

wizard, 13

X

XIVOCC_TYPE, 1006–1008